Augmentation of a Lyapunov-based Deep Neural Network Controller with Concurrent Learning for Control-Affine Nonlinear Systems

Sujata Basyal*, Jonathan Ting*, Kislaya Mishra*, Brendon C. Allen*

***Abstract*—A deep neural network (DNN)-based adaptive controller with a real-time and concurrent learning (CL)-based adaptive update law is developed for a class of uncertain, nonlinear dynamic systems. The DNN in the control law is used to approximate the uncertain nonlinear dynamic model. The inner-layer weights of the DNN are updated offline using data collected in real-time; whereas, the output-layer DNN weights are updated online (i.e., in real-time) using the Lyapunov- and CL-based adaptation law. Specifically, the inner-layer weights of the DNN are trained offline (concurrent to real-time execution) after a sufficient amount of data is collected in real-time to improve the performance of the system, and after training is completed the inner-layer DNN weights are updated in batch-updates. The key development in this work is that the output-layer DNN update law is augmented with CL-based terms to ensure that the output-layer DNN weight estimates converge to within a ball of their optimal values. A Lyapunov-based stability analysis is performed to ensure semi-global exponential convergence to an ultimate bound for the trajectory tracking errors and the output-layer DNN weight estimation errors.**

***Index Terms*—Nonlinear control, Deep Neural Networks (DNNs), Concurrent Learning (CL), Lyapunov methods.**

I. Introduction

Many real dynamical systems are highly complex, nonlinear, and challenging to model. Developing a controller that achieves a control objective, such as trajectory tracking, is often difficult. These real-world controllers must take into account uncertainties in the dynamics to ensure safe functioning in an uncertain environment. In order to develop an efficient controller for a system with complex dynamics, researchers are exploring various model approximation techniques [1], [2].

Adaptive control has been explored over the past several decades as an approach to deal with uncertain dynamical systems [3]–[5]. Adaptive control architectures enable the design of control policies without needing to know the specifics of the dynamic model [6], [7]. Adaptive terms in an adaptive controller attempt to estimate the uncertain parameters to ensure that the control objectives are efficiently achieved [8], [9]. Traditional gradient-based adaptive controllers are model-based adaptive approaches that can estimate uncertain functions provided that the function is linearly parametrizable in the unknown parameters and that the unknown parameters are constant [10]–[14]. Traditional adaptive controllers are, therefore, limited to estimating certain classes of functions [2].

To deal with the limitations of traditional adaptive controllers, non-model based adaptive techniques have been developed over time. For example, function approximators such as deep neural networks (DNNs) or neural networks (NNs) have previously been included in control systems to approximate a wide-class of uncertain dynamical systems within some residual error [15], [16]. Both NNs and DNNs are capable of approximating continuous functions over a compact domain [17]. Although NNs have been shown to provide good function approximation performance [18], [19], they are limited to only a single hidden-layer. On the other hand, a multi-hidden-layer DNN structure is capable of improving the function approximation accuracy compared to NNs, particularly for complicated functions, which can lead to improved system performance [20]–[22]. Therefore, DNNs are capable of outperforming NNs since they can better approximate complex dynamical models.

To improve a DNN's function approximation performance, a DNN can be trained like any other machine learning algorithm. Often DNN training methods utilize an optimization method that minimizes a loss function over a given set of input/output data to achieve statistical guarantees of the DNN approximation error [23], [24]. However, due to the extensive computing requirements and slow training speeds associated with training numerous layers of DNN weights and biases, DNN-based controllers have traditionally been unable to be updated in real-time. The inability to update a DNN in real-time and the fact that DNNs can only achieve probabilistic outputs has limited the adoption of DNNs in safety-critical applications. Recent breakthroughs have overcome the limitations of DNNs by developing Lyapunov-based update laws that adaptively update the output-layer DNN weights in real-time, which has been shown to guarantee system performance and to provide system responsiveness [21], [25]–[29]. In fact, the aforementioned results invoke a multi-timescale approach since they also collect data in real-time to iteratively update the inner-layer DNN weights using conventional offline DNN training methods. After each offline training is complete, the inner-layer DNN weights are updated instantaneously. The iterative updates of the inner-layer DNN weights, in turn, improves the output-layer weight estimation performance and yields real-time learning [25]. An advantage of the multi-timescale approach is that performance is guaranteed even if the inner-layer DNN weights are initially randomized [25].

Although the DNN results in guarantee system performance (i.e., trajectory tracking) [21], [25]–[29], they are unable to ensure that the output-layer DNN weights converge to a small neighborhood around their ideal values. A potential solution is to augment the real-time DNN update law with concurrent learning (CL) inspired terms. CL is a data-based approach that modifies an estimation update law using recorded input and output data from numerical simulations or experiments to ensure estimation convergence in a finite time (i.e., persistence of excitation is not required) [30]–[34]. Traditionally, CL-based update laws have been implemented for traditional gradient-based adaptive controllers and the extension to Lyapunov- and DNN-based controllers is an open problem [30]–[34].

In this study, a Lyapunov- and DNN-based control framework is developed for an uncertain control-affine nonlinear dynamic systems. The output-layer DNN weights are updated using a combination of real-time state feedback and CL-inspired terms. CL-based feedback is incorporated into the update law to ensure that the output-layer weights of the DNN converge to their optimum values. The inner-layer weights are iteratively updated using collected data and an offline function. The DNN is used to approximate the uncertain terms in the dynamic model. A Lyapunov-like stability analysis is performed to ensure semi-global exponential convergence to an ultimate bound for the the trajectory tracking and output-layer DNN weight estimation errors.

II. System Dynamics

Consider a control-affine nonlinear system modeled as

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

where $x(t) : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ represents the states, $f(x(t)) : \mathbb{R}^n \to \mathbb{R}^n$ represents the uncertain, nonlinear drift dynamics, $g(x(t), t) : \mathbb{R}^n \times \mathbb{R}_{\geq 0} \to \mathbb{R}^{n \times m}$ represents the known control effectiveness matrix, and $u(t) : \mathbb{R}_{\geq 0} \to \mathbb{R}^m$ represents the control input. To facilitate the control design for the dynamics in (1), the following assumptions are made.

Assumption 1. The function $f(x(t))$ is continuous for all $x(t) \in \Omega$, where $\Omega \subset \mathbb{R}^n$ is a compact simply connected set.

Assumption 2. The control effectiveness matrix, $g(x(t), t)$, is a full-row rank matrix $\forall t \geq t_0$, where $t_0 \in \mathbb{R}_{\geq 0}$ is the initial time, and the right pseudo inverse of $g(x(t), t)$ exists and is denoted by $g^+(x(t), t) : \mathbb{R}^n \to \mathbb{R}^{m \times n}$, where $g^+(\cdot) \triangleq g^T(\cdot)\left(g(\cdot)\, g^T(\cdot)\right)^{-1}$. Furthermore, $g^+(x(t), t)$ is bounded such that $g^+(x(t), t) \in \mathcal{L}_\infty$.

III. Control Objective

The control objective is for the generalized states in (1) to track a user-defined time-varying smooth trajectory, $x_d(t) : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$, despite the unknown drift dynamics in the dynamic model.

Assumption 3. The time derivative of the desired trajectory and the trajectory itself are continuous and bounded such that $x_d,\ \dot{x}_d \in \mathcal{L}_\infty$.

The tracking objective for the system is quantified using a measurable tracking error denoted by $e : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$, and is defined as

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

Taking the time derivative of (2) and using (1) yields the open-loop error system as

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

A major objective of this work is to approximate the drift dynamics, $f(x(t))$, using a DNN.[1]

A. DNN Approximation and Update Law

DNN-based control architectures are well-suited for approximating continuous dynamic functions that lie in a compact set.[2] Based on universal function approximation property, there exists a DNN with ideal weights, biases, and activation functions for the drift dynamics $f(x)$ such that

$$f(x) = W^T \sigma(\Phi(x)) + \epsilon(x), \tag{4}$$

$\forall x \in \Omega$, where $W \in \mathbb{R}^{k \times n}$ represents an unknown ideal output-layer weight matrix of the DNN, $\sigma : \mathbb{R}^p \to \mathbb{R}^k$ represents a vector of the unknown ideal activation functions associated with the output-layer of the DNN, $\Phi : \mathbb{R}^n \to \mathbb{R}^p$ represents the inner-layers of the DNN and contains their unknown ideal weight matrices and activation functions, and $\epsilon : \mathbb{R}^n \to \mathbb{R}^n$ represents the unknown function reconstruction error. The inner portion of the ideal DNN, Φ, can be expressed as

$$\Phi \triangleq V_I^T \phi_I(V_{I-1}^T \phi_{I-1}(V_{I-2}^T \phi_{I-2}(...V_1^T \phi_1(x)), \tag{5}$$

where $I \in \mathbb{N}$ denotes the number of the user-defined input- and hidden-layers associated with the ideal DNN, V_i, $\forall i \in [1, I]$ represents the input- and hidden-layer weight matrices, and ϕ_i, $\forall i \in [1, I]$ represents vectors of the input- and hidden-layer ideal activation functions.

The DNN is updated using a multiple timescale approach, where the weights and biases of the input- and hidden-layers are trained offline using a traditional training method, and the output-layer weights are updated in real-time using a Lyapunov- and CL-based update law. This timescale approach uses data that was collected from previous experiments or simulations to pre-train all of the DNN weights. If no prior data is available, the DNN can be initialized with random weights. Furthermore, data is collected concurrent to real-time execution and is used to iteratively update the inner-layer DNN weights. Using (4), the DNN estimation

[1] For notational brevity, all functional dependencies are hereafter suppressed unless required for clarity of exposition.

[2] The subsequent stability analysis in Theorem 1 proves that if $x(t_0)$ is initialized within a compact set, then it will remain in a compact set (i.e., $x \in \Omega$) for all time.

of the drift dynamics is denoted as $\hat{f}_{d,i} : \Omega \to \mathbb{R}^n$, and is defined as

$$\hat{f}_{d,i} \triangleq \widehat{W}^T \hat{\sigma}_i \left(\hat{\Phi}_i(x) \right), \tag{6}$$

where $\widehat{W} : \mathbb{R}_{\geq 0} \to \mathbb{R}^{k \times n}$ is the estimate of the output-layer DNN weights, $\hat{\sigma}_i : \mathbb{R}^p \to \mathbb{R}^k$ is the i^{th} user-defined estimation of the activation functions of the DNN output-layer, $\hat{\Phi}_i : \mathbb{R}^n \to \mathbb{R}^p$ is the i^{th} estimate of the inner-layers of the DNN, and $i \in \mathbb{N}$ represents the index for each update of the DNN.[3] The mismatch between the ideal output-layer DNN weights and the weight estimates is denoted as $\widetilde{W} : \mathbb{R}_{\geq 0} \to \mathbb{R}^{k \times n}$, and is defined as

$$\widetilde{W}(t) \triangleq W - \widehat{W}(t). \tag{7}$$

Assumption 4. Based on the universal function approximation property, there exists known constants $\overline{W}$, $\overline{\sigma}$, $\overline{\hat{\sigma}}$, $\overline{\epsilon} \in \mathbb{R}_{>0}$ that upper bound the unknown ideal output-layer DNN weights W, the unknown vector of ideal activation functions of the DNN output-layer $\sigma(\cdot)$, the user-defined estimates of the output-layer activation functions $\hat{\sigma}(\cdot)$, and the function approximation error $\epsilon(\cdot)$ respectively, such that $\sup_{x(t) \in \Omega} \|W\| \leq \overline{W}$, $\sup_{x(t) \in \Omega} \|\sigma(\cdot)\| \leq \overline{\sigma}$, $\sup_{x(t) \in \Omega} \|\hat{\sigma}_i(\cdot)\| \leq \overline{\hat{\sigma}}, \forall i \in \mathbb{N}$, and $\sup_{x(t) \in \Omega} \|\epsilon(\cdot)\| \leq \overline{\epsilon}$. Furthermore, the DNN estimate is designed such that $\sup_{x(t) \in \Omega} \left\| \sigma(\Phi(x)) - \widehat{\sigma}_i \left(\widehat{\Phi}_i(x) \right) \right\| \leq \overline{\tilde{\sigma}}, \forall i \in \mathbb{N}$, where $\overline{\tilde{\sigma}} : \mathbb{R}_{\geq 0}$ denotes a known constant.

To aid the subsequent analysis, the DNN approximation in (4) can be combined with the dynamic model in (1) to yield

$$W^T \sigma(\Phi(x)) + \epsilon(x) = \dot{x}(t) - g(x(t), t) u(t). \tag{8}$$

B. Control Development

Substituting (4) into (3) yields a modified open-loop error system as

$$\dot{e}(t) = g(x(t), t) u(t) - \dot{x}_d(t) + W^T \sigma(\Phi(x)) + \epsilon(x). \tag{9}$$

Based on (9) and the subsequent stability analysis, the control input is designed as

$$u \triangleq g^+(x(t), t) \left[-ke - k_s \text{sgn}(e) + \dot{x}_d(t) - \hat{f}_{d,i} \right], \tag{10}$$

where $k,\ k_s \in \mathbb{R}_{>0}$ represents positive, user-defined control gains, and $\text{sgn}(\cdot)$ represents the signum function. Taking the open-loop error system in (9), and substituting in (6) and (10) yields the closed-loop error system as

$$\begin{aligned} \dot{e}(t) &= -ke - k_s \text{sgn}(e) - \widehat{W}^T \hat{\sigma}_i \left(\hat{\Phi}_i(x) \right) \\ &\quad + W^T \sigma(\Phi(x)) + \epsilon(x). \end{aligned} \tag{11}$$

[3]The subscript i on $\hat{\sigma}_i$ and $\hat{\Phi}_i$ represents the i^{th} training iteration of the DNN. The expression $\hat{\Phi}_i(x)$ is explicitly expressed as $\hat{\Phi}_i(x) = \hat{V}_{i,I}^T \hat{\phi}_{i,I} \left(\hat{V}_{i,I-1}^T \hat{\phi}_{i,I-1} (\hat{V}_{i,I-2}^T \hat{\phi}_{i,I-2} (\ldots \hat{V}_{i,1}^T \hat{\phi}_{i,1}(x)) \right)$, where $\hat{V}$ represents estimates of the ideal inner-layer weights and $\hat{\phi}(.)$ represents estimates of the ideal inner-layer activation functions for the corresponding training iterations.

Based on the subsequent stability analysis, the output-layer weight estimate law with CL-inspired terms is denoted by $\dot{\widehat{W}} : \mathbb{R}_{\geq 0} \to \mathbb{R}^{k \times n}$ and is designed as

$$\begin{aligned} \dot{\widehat{W}} &= \text{proj} \Bigg(\Gamma \hat{\sigma}_i \left(\hat{\Phi}_i(x) \right) e^T + k_{cl} \Gamma \sum_{j=1}^{m} \hat{\sigma}_i \left(\hat{\Phi}_i(x_j) \right) \cdot \\ &\quad \left(\dot{\hat{x}}_j - g_j u_j - \widehat{W}^T \hat{\sigma}_i \left(\hat{\Phi}_i(x_j) \right) \right)^T \Bigg), \end{aligned} \tag{12}$$

where $\text{proj}(\cdot)$ represents a smooth projection (cf. [35]) that ensures that the estimate of the output-layer weights, $\widehat{W}$, is constrained within user-selected bounds, $\Gamma \in \mathbb{R}^{k \times k}$ represents a user-defined positive definite diagonal gain matrix that enables adjustment of the rate at which $\widehat{W}$ is learned, and $k_{cl} \in \mathbb{R}_{>0}$ is a control gain. Based on Assumption 4, (7), and (12), it can be determined that the error between the ideal and estimated output-layer DNN weights, $\widetilde{W}$, is upper bounded by a constant, $\overline{\widetilde{W}} \in \mathbb{R}_{>0}$, such that $\left\| \text{vec} \left(\widetilde{W} \right) \right\| \leq \overline{\widetilde{W}}$.

Assumption 5. Let $\{x_j, u_j, \dot{\hat{x}}_j\}_{j=1}^h$ denote a history stack of previous states, control inputs, and numerically-computed state-derivative estimates corresponding to times $t_j \in \mathbb{R}_{>0}, \forall j$, where $h \in \mathbb{N}$ is the size of the history stack. It is assumed that the errors between the state-derivative estimates and the true state-derivatives are bounded such that $\|\dot{\hat{x}}_j - \dot{x}_j\| \leq \overline{d_x}, \forall j$, where $\overline{d_x} \in \mathbb{R}_{>0}$ is a known constant. Furthermore, it is assumed that the system is sufficiently excited over a finite duration of time such that $\lambda_{\min}(S_{cl}) > \lambda_s > 0, \forall t \geq T$, where $T \in \mathbb{R}_{>0}$ is a finite time, $\lambda_{\min}(\cdot)$ is the minimum eigenvalues of $(\cdot)$, $\lambda_s \in \mathbb{R}_{>0}$ is a user-selected constant, and

$$S_{cl} \triangleq \sum_{j=1}^{h} \hat{\sigma}_i \left(\hat{\Phi}_i(x_j) \right) \hat{\sigma}_i \left(\hat{\Phi}_i(x_j) \right)^T. \tag{13}$$

Note that the pre-training of the DNN is done using previous experimental or simulation data. The pre-trained DNN provides an initial estimate of the inner-layer DNN , $\widehat{\Phi}_1(\cdot)$, and an initial estimate of the output-layer weights $\widehat{W}(t_0)$. Furthermore, the output-layer weights are updated in real-time using the proposed update law in (12). The summation terms (i.e., the CL-inspired terms) are included in the update law to ensure that the output-layer DNN weights converge to a small neighborhood around their optimal values. State and input information are collected in real-time and used in the adaptive update law to train the output-layer DNN weights in real-time and to train the inner-layer DNN weights offline. After the inner-layer DNN weights are sufficiently trained, they will be updated instantaneously to generate the updated DNN estimate, $\widehat{\Phi}_{i+1}(\cdot)$.

IV. Stability Analysis

To facilitate the stability analysis for the closed-loop error system in (11), a positive definite and continuously differentiable common Lyapunov-like function candidate, denoted

as $V_L : \mathbb{R}^{n(1+k)} \times \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$, is defined as

$$V_L(z,t) \triangleq \frac{1}{2} e^T e + \frac{1}{2} \text{trace}\left(\widetilde{W}^T \Gamma^{-1} \widetilde{W}\right), \tag{14}$$

where trace $(\cdot)$ is the trace operator and $z : \mathbb{R}_{\geq 0} \to \mathbb{R}^{n(1+k)}$ is defined as

$$z \triangleq \left[e^T, \ \text{vec}\left(\widetilde{W}\right)^T\right]^T, \tag{15}$$

where vec $(\cdot)$ denotes the vectorization operator. Moreover, V_L can be bounded as

$$\beta_1 \|z\|^2 \leq V_L \leq \beta_2 \|z\|^2, \tag{16}$$

where $\beta_1 \triangleq \min\left(\frac{1}{2}, \frac{1}{2}\lambda_{\min}\left(\Gamma^{-1}\right)\right)$ and $\beta_2 \triangleq \max\left(\frac{1}{2}, \frac{1}{2}\lambda_{\max}\left(\Gamma^{-1}\right)\right)$, where $\lambda_{\max}(\cdot)$ is the the maximum eigenvalue of $(\cdot)$. Furthermore, V_L can also be bounded as

$$\frac{1}{2}\|e\|^2 \leq V_L \leq \frac{1}{2}\|e\|^2 + v_a, \tag{17}$$

where $v_a \in \mathbb{R}_{>0}$ is defined as

$$v_a \triangleq \frac{1}{2}\lambda_{\max}\left(\Gamma^{-1}\right)\overline{\widetilde{W}}^2. \tag{18}$$

Theorem 1. *For the nonlinear and uncertain dynamic model in (1) that satisfies Assumptions 1-5, the control input in (10) and the adaptation law in (12) ensure that the closed-loop error system in (11) yields a bounded result $\forall t \in [t_0, T)$ in the sense that*

$$\|z(t)\| \leq max\left(\sqrt{\frac{\beta_2}{\beta_1}}\|z(t_0)\|, \sqrt{\frac{v_1}{2k\beta_1}}\right), \tag{19}$$

and semi-global exponential convergence to an ultimate bound $\forall t \in [T, \infty)$ in the sense that

$$\begin{aligned}\|z(t)\|^2 \quad \leq \quad & \tfrac{\beta_2}{\beta_1}\|z(T)\|^2 e^{-\frac{\delta}{\beta_2}(t-T)} \\ & + \tfrac{v_2\beta_2}{\delta\beta_1}\left(1 - e^{-\frac{\delta}{\beta_2}(t-T)}\right)\end{aligned}, \tag{20}$$

provided that the following gain condition is satisfied

$$k_s \geq \overline{W}\overline{\sigma} + \overline{\epsilon}, \tag{21}$$

where $v_1 \triangleq 2kv_a + C\overline{\widetilde{W}}$, $C \in \mathbb{R}_{>0}$ is a known constant, $\delta \triangleq min\left(k, \frac{1}{2}k_{cl}\lambda_s\right)$, and $v_2 \triangleq \frac{C^2}{2k_{cl}\lambda_s}$.

Proof: Let $z(t)$ be a Filippov solution to the differential inclusion $\dot{z} \in K[h](z)$ for $t \in [t_0, \infty)$, where $h : \mathbb{R}^{n(1+k)} \to \mathbb{R}^{n(1+k)}$ is defined as $h(z) \triangleq \left[\begin{array}{cc} \dot{e}^T, & \text{vec}(\dot{\widetilde{W}})^T \end{array}\right]^T$, and $K[\cdot]$ is Filippov's differential inclusion operator defined in [36]. Due to the iterative updates of the DNN and the control law having a sliding mode term, the controller is discontinuous. Thus, the time-derivative of V_L exists almost everywhere (a.e.) within $t \in [t_0, \infty)$ such that $\dot{V}_L(z) \overset{\text{a.e.}}{\in} \dot{\tilde{V}}_L(z)$, where $\dot{\tilde{V}}_L$ denotes the generalized time-derivative of (14). The generalized time derivative of V_L in (14) is defined as $\dot{\tilde{V}}_L \subseteq \bigcap_{\xi \in \partial V_L(z)} \xi^T \left[K[h]^T(z), 1\right]^T$, where $\partial V_L(z)$ denotes the Clark's generalized gradient of $V_L(z)$. Since $V_L(z)$ is continuously differentiable in z, $\partial V_L(z) = \nabla V_L(z)$, where ∇ is the gradient operator.

Taking the generalized time derivative of (14), using properties of the trace operator and the matrix vectorization operator, and substituting the output-layer DNN weight estimation error in (7), the closed-loop error system in (11), and the update law in (12), yields

$$\begin{aligned}\dot{\tilde{V}}_L \quad \subseteq \quad & -ke^Te - k_s e^T K[\text{sgn}(e)] + e^T\epsilon(x) \\ & -e^T\widehat{W}^T K\left[\hat{\sigma}_i\left(\hat{\Phi}_i(x)\right)\right] + e^T W^T \sigma(\Phi(x)) \\ & -e^T\widetilde{W}^T K\left[\hat{\sigma}_i\left(\hat{\Phi}_i(x)\right)\right] \\ & -\text{trace}\left(k_{cl}\widetilde{W}^T \sum_{j=1}^{h} K\left[\hat{\sigma}_i\left(\hat{\Phi}_i(x_j)\right)\right] \cdot \right. \\ & \left. \left(\dot{\hat{x}}_j - g_j u_j - \widehat{W}^T K\left[\hat{\sigma}_i\left(\hat{\Phi}_i(x_j)\right)\right]\right)^T\right).\end{aligned} \tag{22}$$

Consider an arbitrary $i \in \mathbb{N}$ such that $\hat{\sigma}_i$ and $\hat{\Phi}_i$ are continuous. Adding and subtracting $e^T W^T \hat{\sigma}_i\left(\hat{\Phi}_i(x)\right)$ into (22) and using the definition of $\widetilde{W}$ from (7), yields

$$\begin{aligned}\dot{\tilde{V}}_L \quad \subseteq \quad & -ke^Te - k_s e^T K[\text{sgn}(e)] + e^T\epsilon(x) \\ & +e^T W^T\left[\sigma(\Phi(x)) - \hat{\sigma}_i\left(\hat{\Phi}_i(x)\right)\right] \\ & -\text{trace}\left(k_{cl}\widetilde{W}^T \sum_{j=1}^{h} \hat{\sigma}_i\left(\hat{\Phi}_i(x_j)\right) \cdot \right. \\ & \left. \left(\dot{\hat{x}}_j - g_j u_j - \widehat{W}^T \hat{\sigma}_i\left(\hat{\Phi}_i(x_j)\right)\right)^T\right).\end{aligned} \tag{23}$$

Note that Filippov's differential inclusion operator, $K[\cdot]$, is defined such that $-eK[\text{sgn}(e)] \leq -\|e\|$. Using Assumption 4, the gain condition in (21), and the fact that $-eK[\text{sgn}(e)] \leq -\|e\|$ yields

$$\begin{aligned}\dot{V}_L \quad \overset{\text{a.e.}}{\leq} \quad & -k\|e\|^2 - \text{trace}\left(k_{cl}\widetilde{W}^T \sum_{j=1}^{h} \hat{\sigma}_i\left(\hat{\Phi}_i(x_j)\right) \cdot \right. \\ & \left. \left(\dot{\hat{x}}_j - g_j u_j - \widehat{W}^T \hat{\sigma}_i\left(\hat{\Phi}_i(x_j)\right)\right)^T\right).\end{aligned} \tag{24}$$

Adding and subtracting terms into (24) and using Assumptions 4 and 5, (7), (8), (13), and properties of the trace operator yields

$$\begin{aligned}\dot{V}_L \quad \overset{\text{a.e.}}{\leq} \quad & -k\|e\|^2 + C\left\|\text{vec}\left(\widetilde{W}\right)\right\| \\ & -\text{trace}\left(k_{cl}\widetilde{W}^T S_{cl}\widetilde{W}\right).\end{aligned} \tag{25}$$

Notice that (25) holds for any arbitrary $i \in \mathbb{N}$, such that (25) holds for all $t \in \left[t_0, \infty\right)$. During the time interval $t \in \left[t_0, T\right)$, sufficient excitation has not yet been achieved according to Assumption 5. Thus, it can conservatively be assumed that the term S_{cl} is only positive semidefinite $\forall t \in \left[t_0, T\right)$, and (25) can be upper bounded as

$$\dot{V}_L \overset{\text{a.e.}}{\leq} -k\|e\|^2 + C\overline{\widetilde{W}}, \quad \forall t \in \left[t_0, T\right). \tag{26}$$

Solving the differential inequality in (26), and using (16) and (17) yields

$$\begin{aligned} \|z(t)\|^2 &\leq \tfrac{\beta_2}{\beta_1}\|z(t_0)\|^2 e^{-2k(t-t_0)} \\ &\quad + \tfrac{v_1}{2k\beta_1}\left[1-e^{-2k(t-t_0)}\right], \forall t \in \left[t_0, T\right). \end{aligned} \tag{27}$$

Inspection of (27) leads to the result in (19).

During the time interval $t \in [T, \infty)$, sufficient learning has occurred according to Assumption 5. Therefore, the term S_{cl} is positive definite $\forall t \in [T, \infty)$ and

$$-\text{trace}\left(k_{cl}\widetilde{W}^T S_{cl}\widetilde{W}\right) \leq -k_{cl}\lambda_s \left\|\text{vec}\left(\widetilde{W}\right)\right\|^2. \tag{28}$$

Using (15) and (28), and completing the squares yields an upper bound for (25) as

$$\dot{V}_L \overset{\text{a.e.}}{\leq} -\delta\|z\|^2 + \frac{C^2}{2k_{cl}\lambda_s}, \quad \forall t \in [T, \infty). \tag{29}$$

Solving the differential inequality in (29), and using (16) yields (20) in the theorem statement.

From (19) and (20), it can be seen that $z \in \mathcal{L}_\infty$. From the definition of z in (15), the terms e, $\widetilde{W} \in \mathcal{L}_\infty$. From the definition of e in (2) and Assumption 3, it is clear that $x \in \mathcal{L}_\infty$. From Assumptions 2 and 4, it is stated that $g^+(x,t), \hat{\sigma}(\cdot) \in \mathcal{L}_\infty$. Furthermore, using (7) with the fact that $\widetilde{W} \in \mathcal{L}_\infty$ ensures that $\widehat{W} \in \mathcal{L}_\infty$ (along with the design of $\dot{\widehat{W}}$ in (12)). Therefore, it can be proven that the DNN estimate defined in (6) and the controller defined in (10) are bounded. Lastly, recall that the DNN in (6) requires x to be bounded for all time. Based on (19) and (20), if $x(t_0) \in \mathcal{L}_\infty$ (i.e., a semi-global result), then $x(t) \in \mathcal{L}_\infty, \forall t \in [t_0, \infty)$. ■

V. Conclusion

A DNN-based controller with a CL-inspired DNN weight update law was developed for an uncertain, control-affine nonlinear dynamic system. In this preliminary work, the control effectiveness matrix was assumed to be known. The inner-layers of DNN are updated offline (concurrent to real-time execution) and the output-layer DNN weights are updated online using a Lyapunov- and CL-based DNN update law. The CL policy in the adaptive controller helps to ensure exponential convergence of the output-layer weights to a small neighborhood containing the ideal DNN weights. A Lyapunov-like stability analysis was performed to ensure semi-global exponential convergence to an ultimate bound for the trajectory tracking errors and the DNN output-layer estimation errors. Future efforts will perform simulations to ensure the effectiveness of the developed control law. Additionally, the controller can be extended to consider systems with an uncertain control effectiveness matrix.

References

[1] A. Boulkroune, M. Tadjine, M. MSaad, and M. Farza. How to design a fuzzy adaptive controller based on observers for uncertain affine nonlinear systems. *Fuzzy Set. Syst.*, 159:926–948, 2008.

[2] S. Sastry and M. Bodson. *Adaptive Control: Stability, Convergence, and Robustness*. Prentice-Hall, Upper Saddle River, NJ, 1989.

[3] Naji A Alibeji, Nicholas A Kirsch, and Nitin Sharma. A muscle synergy-inspired adaptive control scheme for a hybrid walking neuroprosthesis. *Frontiers in bioengineering and biotechnology*, Vol. 3, 2015.

[4] Brian Anderson, Thomas Brinsmead, Daniel Liberzon, and A Stephen Morse. Multiple model adaptive control with safe switching. *Int. J. Adapt. Control Signal Process.*, 15(5):445–470, 2001.

[5] B. Mirkin, P.-O. Gutman, and Y. Shtessel. Continuous model reference adaptive control with sliding mode for a class of nonlinear plants with unknown state delay. In *Proc. Am. Control Conf.*, pages 574–579, 10-12 2009.

[6] Miroslav Krstic, Ioannis Kanellakopoulos, and Peter V. Kokotovic. *Nonlinear and Adaptive Control Design*. John Wiley & Sons, New York, NY, USA, 1995.

[7] Miroslav Krstic and Petar V. Kokotovic. Control Lyapunov functions for adaptive nonlinear stabilization. *Syst. Control Lett.*, 26(1):17–23, 1995.

[8] Z. Hussain, M. A. Zaidan, M. O. Tokhi, and R. Jailani. The adaptive control of FES-assisted indoor rowing exercise. *Proc. Int. Automatic Control Conference (CACS)*, 2009.

[9] Linh Vu, Debasish Chatterjee, and Daniel Liberzon. Input-to-state stability of switched systems and switching adaptive control. *Automatica*, 43(4):639–646, 2007.

[10] V. H. Duenas, C. A. Cousin, A. Parikh, P. Freeborn, E. J. Fox, and W. E. Dixon. Motorized and functional electrical stimulation induced cycling via switched repetitive learning control. *IEEE Trans. Control Syst. Tech.*, 27(4):1468–1479, 2019.

[11] V. Duenas, C. A. Cousin, V. Ghanbari, E. J. Fox, and W. E. Dixon. Torque and cadence tracking in functional electrical stimulation induced cycling using passivity-based spatial repetitive learning control. *Automatica*, 115:1–9, 2020.

[12] G. Bastin and G. Campion. Indirect adaptive control of linearly parametrized nonlinear systems. *IFAC Proceedings Volumes*, 23(1):153–158, 1990. 3rd IFAC Symposium on Adaptive Systems in control and signal Processing 1989, Glasgow, UK, 19-21 April.

[13] I. Kanellakopoulos, P. V. Kokotovic, and A. S. Morse. Systematic design of adaptive controllers for feedback linearizable systems. In *1991 American Control Conference*, pages 649–654, 1991.

[14] S.S. Sastry and A. Isidori. Adaptive control of linearizable systems. *IEEE Trans. Autom. Control*, 34(11):1123–1131, November 1989.

[15] Ding Wang, Derong Liu, Hongliang Li, and Hongwen Ma. Neural-network-based robust optimal control design for a class of uncertain nonlinear systems via adaptive dynamic programming. *Inf. Sci.*, 282:167–179, 2014.

[16] F. L. Lewis, S. Jagannathan, and A. Yesildirak. *Neural network control of robot manipulators and nonlinear systems*. CRC Press, Philadelphia, PA, 1998.

[17] K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Netw.*, 4:251–257, 1991.

[18] Thomas Parisini and Riccardo Zoppoli. A receding-horizon regulator for nonlinear systems and a neural approximation. *Automatica*, 31(10):1443–1451, 1995.

[19] R.M. Sanner and J.J.E. Slotine. Stable adaptive control of robot manipulators using neural networks. *Neural Comput.*, 7(4):753–790, 1995.

[20] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[21] G. Joshi and G. Chowdhary. Deep model reference adaptive control. In *IEEE Conf. Decis. Control*, pages 4601–4608. IEEE, 2019.

[22] Girish Joshi, Jasvir Virdi, and Girish Chowdhary. Asynchronous deep model reference adaptive control. In *Conf. on Robot Learn.*, 2020.

[23] Benjamin Karg and Sergio Lucia. Efficient representation and approximation of model predictive control laws via deep learning. *IEEE Transactions on Cybernetics*, 50(9):3866–3878, 2020.

[24] Michael Hertneck, Johannes Köhler, Sebastian Trimpe, and Frank Allgöwer. Learning an approximate model predictive controller with guarantees. *IEEE Control Systems Letters*, 2(3):543–548, 2018.

[25] 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.*, 6:193–198, 2022.

[26] J. Ting, S. Basyal, and B. C. Allen. Deep neural network based saturated adaptive control of muscles in a lower-limb hybrid exoskeleton. In *ASME Int. Mech. Eng. Congr. Expo. (IMECE)*, 2023.

[27] H. M. Sweatland, B. C. Allen, M. L. Greene, and W. E. Dixon. Deep neural network real-time control of a motorized FES cycle with an uncertain time-varying electromechanical delay. In *ASME Int. Mech. Eng. Congr. Expo. (IMECE)*, 2021.

[28] Girish Joshi, Jasvir Virdi, and Girish Chowdhary. Design and flight evaluation of deep model reference adaptive controller. In *AIAA Scitech 2020 Forum*, page 1336, 2020.

[29] E. Griffis, A. Isaly, D. M. Le, and W. E. Dixon. Deep neural network-based adaptive fes-cycling control: Part ii, a hybrid systems approach. In *Proc. IEEE Conf. Decis. Control*, 2022.

[30] Girish Chowdhary, Tansel Yucelen, Maximillian Mühlegg, and Eric N. Johnson. Concurrent learning adaptive control of linear systems with exponentially convergent bounds. *Int. J. Adapt. Control Signal Process.*, 27(4):280–301, 2013.

[31] Girish V. Chowdhary and Eric N. Johnson. Theory and flight-test validation of a concurrent-learning adaptive controller. *J. Guid. Control Dynam.*, 34(2):592–607, March 2011.

[32] Girish Chowdhary, Maximilian Mühlegg, JonathanP. How, and Florian Holzapfel. Concurrent learning adaptive model predictive control. In Qiping Chu, Bob Mulder, Daniel Choukroun, Erik-Jan van Kampen, Coen de Visser, and Gertjan Looye, editors, *Advances in Aerospace Guidance, Navigation and Control*, pages 29–47. Springer Berlin Heidelberg, 2013.

[33] B. C. Allen, K. J. Stubbs, and W. E. Dixon. Adaptive trajectory tracking during motorized and FES-induced biceps curls via integral concurrent learning. In *Proc. ASME Dyn. Syst. Control Conf.*, 2020.

[34] B. Allen, K. Stubbs, and W. E. Dixon. Data-based and opportunistic integral concurrent learning for adaptive trajectory tracking during switched fes-induced biceps curls. *IEEE Trans. Neural Syst. Rehabil. Eng.*, 30:2557–2566, 2022.

[35] Z. Cai, M. S. de Queiroz, and D. M. Dawson. A sufficiently smooth projection operator. *IEEE Trans. Autom. Control*, 51(1):135–139, January 2006.

[36] A. F. Filippov. Differential equations with discontinuous right-hand side. In *Fifteen papers on differential equations*, volume 42 of *American Mathematical Society Translations - Series 2*, pages 199–231. American Mathematical Society, 1964.