Reinforcement Learning Tracking Control of Nonlinear Strict-Feedback Systems Using Multilayer Neural Networks

Insha Yaqoob[1,1,*], and S.Sabbir[1,1]

[a]*Dept. of Electronics and Communication Engineering NIT Jaipur, India*
Dept. of Electrical and Computer Engineering, University of Rhode Island,USA
,

Abstract

This paper presents a novel safe integral reinforcement learning (IRL)-based optimal trajectory tracking scheme for nonlinear systems with uncertain dynamics that is subject to constraints. We leverage multilayer neural networks (MNNs) for actor-critic MNNs along with an NN identifier in the backstepping process for minimizing a discounted value function. A time-varying barrier Lyapunov function (TVBLF) is utilized for handling constraints and to provide safety assurances. Online weight update laws for the actor and critic MNNs are derived using singular value decomposition (SVD)-based method that is driven by Bellman error. We introduce an online lifelong learning (LL) method, utilizing the Fisher Information Matrix (FIM) based on Bellman error in MNNs to address catastrophic forgetting. The method's effectiveness is demonstrated through simulations on mobile robot multitask tracking. The paper concludes with a stability analysis of the closed-loop system.

Keywords: Optimal Control, Strict-Feedback, Neural Networks, Lifelong Learning, Barrier Constraints.

---

*Corresponding author
[1]Emails: Insha Yaqoob - $\text{irv}_1@uri.edu$

Final version submitted to Control Engineering Practice

1. Introduction

Optimal tracking control in nonlinear systems with constraints is pivotal due to its widespread applications in industrial and robotic systems. The Hamilton-Jacobi-Bellman (HJB) equation, central to optimal control, find its online solutions in actor-critic designs with approximate dynamic programming (ADP) [1, 2, 3, 4, 5]. However, they mainly cater to nonlinear systems in affine form.

Research into adaptive NN-based tracking using backstepping for strict-feedback systems exists [6, 7, 8, 9]. Yet, when combined with ADP's policy/value iterations [7], the optimal adaptive control (OAC) methods use neural networks (NNs) and need numerous iterations to find a solution to HJB equation. Recent literature uses a positive function to determine weight updates in a single layer NNs and relaxed PE conditions [6, 10].

Current OAC methods rely on single-layer NNs with basis functions as activation functions to approximate both value function and control inputs, but choosing appropriate basis functions for unknown systems is not well understood. While the MNN's universal function approximation enables a single and two-layer NN control scheme for adaptive CT systems [11, 12, 13], recent discussions on MNN-based adaptive control only present offline inner-layer weight training and online output-layer weight tuning [14]. Despite the evolution of NN control schemes [15], the potential of deep MNNs with online tuning in optimal trajectory tracking for uncertain nonlinear strict-feedback CT systems is not yet been explored. Challenges include online MNN weight adjustments without facing the vanishing gradient issue [16] and ensuring system stability.

The significance of safety considerations in domains such as autonomous driving and industrial robotics is not explored in optimal control framework for ADP or OAC-based methods[6, 10]. On the other hand, researchers have investigated safety methods for control systems with constant constraints by utilizing barrier Lyapunov functions (BLFs) and control barrier functions (CBFs) [17] [18] [19]. CBFs and BLFs have been demonstrated to be powerful methods in the control domain for designing the controllers offering better performance and intervene only when safety is at risk of being compromised. Methods utilizing state transformation techniques [20] are primarily suitable for rectangular state constraints, i.e., the safe set is a hyperrectangle, which does not encompass the complex safety specifications. Studies have also examined control designs using time-varying BLF (TVBLF) for adaptive systems [21]. However, incorporating ADP-based optimal control in nonlinear CT systems with constraints presents further difficulties while solving the HJB equation.

Besides optimality and safety, industrial or robotic systems operate in multitasking situations, requiring the NNs to undergo intermittent retraining for each task due to changes in dynamics. However, when control schemes use online learning instead of offline training, catastrophic forgetting [22] is quite common. To address this issue, online lifelong learning (LL) techniques over [22], [23] are needed for safety-critical systems. Moreover, offline LL elastic weight consolidation (EWC) [22] technique has been shown to be effective in mitigating forgetting, whereas it can lead to an exploding gradient[23]. The works have effectively handled the LL issues in non-optimal adaptive control systems using online EWC utilizing the tracking error [24, 25], however solving the optimal control methods requires a completely different approach because of the actor-critic framework and the challenges of solving the HJB equation in each step of backstepping.

Therefore, we propose a novel trajectory-tracking framework based on ADP and lifelong integral reinforcement learning (LIRL) for a multi-tasking environment. This framework uses a novel singular value decomposition (SVD)-based actor-critic MNN with an activation function to formulate optimal control policies differing from [6],[10]. The TVBLF is augmented into the cost function to ensure the error vector stays within predefined constraints. A new online LL MNN method is introduced using Bellman error to prevent catastrophic forgetting while avoiding gradient explosion in contrast to offline methods [22] and online methods which utilizes the tracking error and online EWC [24, 25] with partially known dynamics.

The key contributions are;

1) Introducing a novel optimal tracking control method that distinctively separates the safety components from the nominal control part, leveraging the ADP and Bellman optimality principle subject to both static and dynamics constraints in contrast to the existing literature [18, 20].

2) Development of an SVD-based MNN tuning approach for the actor-critic MNNs to maintain optimality and safety using the safety-based Bellman error and control input error while providing stability in contrast to single-layer-based optimal control methods using basis function [6, 10]and multilayer-based non-optimal methods utilizing the tracking errors [25].

3) An online LL technique that uses Bellman error and a regularization-based approach to assess the importance of NN weights for the critic MNN in

Final version submitted to Control Engineering Practice

contrast to non-optimal offline based EWC methods [22], and online EWC-based methods [24, 25], within a learning framework to prevent catastrophic forgetting in the multi-tasking environment.

4) Validating the efficiency and stability of the tracking system by establishing Lyapunov stability, ensuring safety, and demonstrating effectiveness on a mobile robot system, taking into account both kinematics and dynamics.

For the sake of simplicity, the time variable t will be omitted unless otherwise mentioned.

2. Problem Formulation and Optimal Control

This section presents the problem setup and introduces a safe lifelong reinforcement learning (SLRL) control approach.

2.1. System description

Consider the following strict feedback system

$$\begin{aligned} \dot{x}_1(t) &= f_1\left(x_1\right) + g_1(x_1)x_2, \\ \dot{x}_2(t) &= f_2\left(\bar{x}_2\right) + g_2(\bar{x}_2)u, \end{aligned} \tag{1}$$

where $\bar{x}_2(t) = [x_1(t), x_2(t)]^\top \in \mathbb{R}^2$ is the set of system states, x_1, x_2, are the system states, $u \in \mathbb{R}$ is the control input, and $f_i\left(.\right), g_i(.), i = 1,2$ are unknown yet Lipschitz continuous functions and satisfying $f_i(0) = 0$. The standard following assumptions are stated to proceed.

Assumption 1 ([1])**.** *The nonlinear CT system is controllable and observable. In addition, the control coefficient matrix satisfies $\|g_i(.)\| \leq g_{iM}$, where g_{iM} is an unknown positive constant.*

Assumption 2 ([1, 26])**.** *The state vector is assumed to be measurable. In addition, the desired trajectory $r_d(t)$ in $\mathbb{R}$, and all of its higher derivatives, are assumed to be bounded such that $\|r_d\| \leq k_{di} < K_{bi}$ where k_{di}, K_{bi} are the bounded constraints on the reference trajectory and system states respectively and the reference trajectory is obtained using the generator function as $\dot{r}_d \triangleq h_d(r_d)$.*

Provided assumption 2 the state constraints can be transformed into the error constraints as $\|\zeta_i\| < \kappa_{ei}$, where $\kappa_{ei} = K_{bi} - k_{di}$ where K_{bi} is the constraint on system states x_i, $i = 1,2$ and $\zeta_1 = x_1 - r_d(t), \zeta_2 = x_2 - \alpha$

denote tracking errors, α being the virtual controller and r_d be the reference trajectory.

In this paper, we aim to develop an optimal tracking control, u^*, for (1), which ensures that the tracking errors should remain in a safe set for all $t \geq 0$ such that $\|\zeta_i(t)\| < \kappa_{ei}(t)$ where $\kappa_{ei} : \mathbb{R}^+ \to \mathbb{R}$, $i = 1, 2$ are the constraints on the tracking errors ζ_i, generated using a generator function [26] as $\dot{\kappa}_{ei} \triangleq \psi(\kappa_{ei})$ while minimizing the performance index functions $\mathcal{V}_1, \mathcal{V}_2$ to be defined later and mitigating the catastrophic forgetting in multitasking scenarios. Next, the TVBLF is presented.

2.2. Barrier Lyapunov function

To perform the objective, we initially partition the state space $\mathbb{R}$ into subsets: a safe set $\mathcal{S} \subset \mathbb{R}$ and an unsafe set $\mathcal{D} = \mathbb{R} \backslash \mathcal{S} \subset \mathbb{R}$. We define the safe set $\mathcal{S}$ as an open set, represented as the exact 0-superlevel set of a C^1 continuously differentiable function $b : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$.

$$\begin{aligned} \mathcal{S} &= \{\zeta_i \in \mathbb{R} \mid b(\zeta_i, \kappa_{ei}(t)) > 0\}, \\ \partial\mathcal{S} &= \{\zeta_i \in \mathbb{R} \mid b(\zeta_i, \kappa_{ei}(t)) = 0\}. \end{aligned} \tag{2}$$

where $\zeta_i, i = 1, 2$ is the error variable.

Assumption 3. *[17] The initial condition of the state variable in equation* (2) *belongs to the interior of $\mathcal{S}$, i.e., $\zeta_0 \in int(\mathcal{S})$.*

Definition 1 ([17])**.** *For a set $\mathcal{S} \subseteq \mathbb{R}$ and an initial condition $\zeta(0) = \zeta_0$, the set $\mathcal{S}$ is considered forward invariant for system (1) if $\zeta_0 \in \mathcal{S} \implies \zeta_i(t) \in \mathcal{S}, \forall t \geq 0$.*

Consider the TVBLF for the control design as

$$b(\zeta_i, \kappa_{ei}(t)) = \log\left(\frac{\kappa_{ei}^2}{(\kappa_{ei}^2 - \zeta_i^2(t))}\right), \tag{3}$$

The TVBLF (3) is chosen because it satisfies the following properties that are important for the control design, $b_{\zeta_i,\kappa_{ei}} : \mathrm{Int}(\mathcal{S}) \to \mathbb{R}_{\geq 0}$ that satisfies $\inf_{\zeta_i \in \mathrm{Int}(\mathcal{S})} b(\zeta_i, \kappa_{ei}) \geq 0, \quad \lim_{\zeta_i \to \partial\mathcal{S}} b(\zeta_i, \kappa_{ei}) = \infty, \quad b(0, \kappa_{ei}(t)) = 0$. In this optimal framework, we propose to augment the instantaneous cost with TVBLF, whose minimization implies satisfaction of the original constraints. Next, the SLRL-based control design is introduced.

2.3. Optimal backstepping control

To develop an optimal control using the backstepping technique, define the tracking error as $\zeta_1(t) = x_1(t) - r_d(t)$. Taking the time derivative to obtain

$$\dot{\zeta}_1(t) = f_1(x_1) + g_1(x_1)x_2 - \dot{r}_d(t), \tag{4}$$

where $x_2(t)$ is the virtual control and $\dot{r}_d(t) \triangleq h_d(r_d)$. In this paper, we will explore the concept of two-step backstepping. Nevertheless, the same methodology can be applied to n-step backstepping as well.

Step 1: For the first backstepping step, the augmented barrier based optimal performance index function $\mathcal{V}_1^*(\bar{\zeta}_1) = \min_{\alpha\in\mathcal{U}} \mathcal{V}_1(\bar{\zeta}_1)$ for the optimal tracking problem leading to optimal controller with safety is defined as

$$\begin{aligned} \mathcal{V}_1^*\left(\bar{\zeta}_1\right) &= \min_{\alpha\in\mathcal{U}} \left(\int_t^\infty e^{-\gamma_1(\tau-t)} (\zeta_1(\tau)^\top Q_1 \zeta_1(\tau) + \alpha^\top R_1 \alpha + \lambda_1 \frac{db_{r1}}{dt}) d\tau \right) \\ &= \left(\int_t^\infty e^{-\gamma_1(\tau-t)} (\zeta_1(\tau)^\top Q_1 \zeta_1(\tau) + \alpha^{*\top} R_1 \alpha^* + \lambda_1 \frac{db_{r1}}{dt}) d\tau \right), \end{aligned} \tag{5}$$

where $\mathcal{U}$ denotes the set of admissible control policies, $\bar{\zeta}_1 = [\zeta_1^\top, r_d^\top, \kappa_{e1}^\top]^\top$, $\mathcal{V}_1(\bar{\zeta}_1) = \left(\int_t^\infty e^{-\gamma_1(\tau-t)} \bar{h}_{1m}\left(\zeta_1, \alpha\right) d\tau\right)$, and $\bar{h}_{1m} = \zeta_1(\tau)^\top Q_1 \zeta_1(\tau) + \alpha^\top R_1 \alpha + \lambda_1 \frac{db_{r1}}{dt}$, and κ_{e1} is a bounded constraint generated dynamically using $\dot{\kappa}_{e1} = \psi(\kappa_{e1})$, $b_{ri} = \ln\left(\frac{\kappa_{ei}^2}{\kappa_{ei}^2 - \zeta_i^\top \zeta_i}\right)$ satisfying $b_{ri}(0) = 0, b_{ri}(\zeta_i) > 0 \quad \forall \zeta_i \in \mathcal{S}_i/\{0\}$, and $\lim_{\zeta_i \to \partial\mathcal{S}_i} b_{ri}(\zeta_i) = \infty$, $\partial\mathcal{S}_i$ and $\text{Int}(\mathcal{S}_i)$ to be the boundary and the interior of the set $\mathcal{S}_i$ respectively, Q_1, R_1 are positive, γ_1 is the discount factor, $\gamma_1 > 0, \lambda_1$ is the trade off factor between optimality and safety, α and α^* be the virtual and optimal virtual control inputs respectively.

Remark 1. *To ensure that $\mathcal{V}_1$ remains finite, and the control input is admissible, as will be demonstrated in the subsequent theorem, we adopt a discounted cost function as given in* (5)*. Additionally, the incorporation of the barrier terms for safety into the instantaneous cost introduces additional challenges in deriving the optimal backstepping-based approach.*

Next, the definition of a safe input set is stated.

Definition 2 (Safe Input Set)**.** *Given a system state ζ_i, the safe input set $\mathcal{U}_s$ is defined as the set of all control inputs u_i such that the application of u_i keeps the state vector of* (4) *within a predefined safe set $\mathcal{S}$ for all future times, i.e $\mathcal{U}_s = \{u_i \in \mathcal{U} | \zeta_i(t) \in \mathcal{S}, \forall t \geq 0$ under the control $u_i\}$*

Final version submitted to Control Engineering Practice

Assumption 4 ([17]). *We assume that for any initial condition $\zeta_0 \in \mathcal{S}$, the set of admissible control policies is non-empty, i.e.,$\mathcal{U} \cap \mathcal{U}_s \neq \emptyset$.*

For the trajectory tracking, by using the Bellman optimality principle, which is in line with the ([27]), the Hamiltonian-based equation can be written as

$$\begin{aligned} H_1^* = &\gamma_1 \mathcal{V}_1^* - \frac{\partial \mathcal{V}_1^{*\top}}{\partial \zeta_1} \left[f_1(x_1) + g_1(x_1)\alpha^*(t) - h_d \right] \\ &- \zeta_1^\top Q_1 \zeta_1 - U(\alpha^*) - \nabla b_{r1}^\top \left[f_1(x_1(t)) - h_d + g_1(x_1(t))\alpha^*(t) \right] - \Psi_1 = 0. \end{aligned} \tag{6}$$

where $\Psi_1 = \frac{\partial b_{r1}^\top}{\partial \kappa_{e1}} \psi_1(\kappa_{e1}) + \frac{\partial \mathcal{V}_1^{*\top}}{\partial r_d} h_d(r_d) + \frac{\partial \mathcal{V}_1^{*\top}}{\partial \kappa_{e1}} \psi_1(\kappa_{e1})$, $U(\alpha^*) = \alpha^{*\top} R_1 \alpha^*$. The additional terms in the Hamiltonian arise because of the barrier function and the reference trajectory which leads to the design of safe optimal tracking controller as will be shown next.

By taking the derivative of (6) with respect to α^*, and using the stationary conditions we have, $\partial H_1 / \partial \alpha^* = 0$, the optimal control policy α^*, is given by

$$\alpha^*(\bar{\zeta}_1) = -\frac{1}{2} R_1^{-1} g_1^\top(x_1) \nabla \mathcal{V}_1^*(\bar{\zeta}_1) - \frac{\lambda_1}{2} R_1^{-1} g_1^\top(x_1) \nabla b_{r1} \tag{7}$$

where $\bar{\zeta}_1 = [\zeta_1^\top, r_d^\top, \kappa_{e1}^\top]^\top$, $\nabla \mathcal{V}_1^*, \nabla b_{r1}$ are the partial derivatives of the optimal value function $\mathcal{V}_1^*$ and BLF b_{r1} with respect to ζ_1 respectively.

Given the value function (5) and $\nabla \mathcal{V}_1^*$ being unknown yet continuous on a compact set Ω, they can be represented by MNNs characterized by weights and activation functions with reference trajectory as one of the input. Hence, we can approximate it as follows

$$\mathcal{V}_1^*(\bar{\zeta}_1) = W_{c1}^\top \sigma(V_{c1}^\top \sigma(\bar{\zeta}_1)) + \epsilon_{c1}(\bar{\zeta}_1), \tag{8}$$

where σ is the activation function, W_{c1} and V_{c1} are the output and hidden layer target weights, respectively, which are unknown but bounded on a compact set, $\sigma_{11} = \sigma(V_{c1}^\top \sigma(\bar{\zeta}_1)), \sigma_{21} = \sigma(\bar{\zeta}_1)$ where σ is the activation function with reference trajectory as one of the input. Similarly, an actor NN will approximate the optimal virtual control input (7) on a compact set as

$$\alpha^* = W_{a1}^\top \sigma_a \left(V_{a1}^\top \sigma_a(\bar{\zeta}_1) \right) + \varepsilon_1(\bar{\zeta}_1) - \frac{\lambda_1}{2} R_1^{-1} g_1^\top(x_1) \nabla b_{r1}, \tag{9}$$

where W_{a1}, V_{a1} are the target unknown weights bounded on a compact set, and ε_1 denotes the function reconstruction error, σ_a is the activation function.

Now, to use the IRL formulation in order to avert the need for drift dynamics, it is necessary to integrate the infinitesimal representation of (5) throughout the interval $[t-T, t]$ as shown in [1], where $T > 0$ is a time interval to get

$$\begin{aligned} \mathcal{V}_1^*(\bar{\zeta}_1(t-T)) = \int_{t-T}^{t} e^{-\gamma_1(\tau-t+T)}[Q(\zeta_1(\tau)) \\ + U(\alpha^*) + \lambda_2 \frac{db_{r1}}{dt}]d\tau + e^{-\gamma_1 T}\mathcal{V}_1^*(\bar{\zeta}_1(t)). \end{aligned} \tag{10}$$

Substituting (8) in (10), the HJB approximation error can be written as

$$\varepsilon_{B1} \equiv \int_{t-T}^{t} e^{-\gamma_1(\tau-t+T)} \left[Q(\zeta_1) + U(\alpha^*) + \lambda_2 \frac{db_{r2}}{dt}\right] d\tau + \varphi_1, \tag{11}$$

where, $\varphi_1 = e^{-\gamma_1 T} W_{c1}^\top \sigma_{11}(\bar{\zeta}_1(t)) - W_{c1}^\top \sigma_{01}(\bar{\zeta}_1(t-T))$, $\varepsilon_{B1} = \varepsilon_1(\bar{\zeta}_1(t-T)) - e^{-\gamma_1 T}\varepsilon_1(\bar{\zeta}_1(t))$. Since the target weights, W_{c1} and V_{c1}, are unknown, approximate (8) on a compact set as

$$\hat{\mathcal{V}}_1(\bar{\zeta}_1(t)) = \hat{W}_{c1}^\top \sigma(\hat{V}_{c1}^\top \sigma(\bar{\zeta}_1)), \tag{12}$$

where $\hat{V}_{c1}, \hat{W}_{c1}$ are the estimated weights of MNN, σ is the nonlinear activation function and $\hat{\sigma}_{11} = \sigma(\hat{V}_{c1}^\top \sigma(\bar{\zeta}_1))$. Since the target weights are unknown, the actor NN will be designed to estimate the optimal virtual control given by

$$\hat{\alpha} = \hat{W}_{a1}^\top \sigma_a(\hat{V}_{a1}^\top \sigma_a(\bar{\zeta}_1)) - \frac{\lambda_1}{2} R_1^{-1} g_1^\top(x_1) \nabla b_{r1}, \tag{13}$$

where $\hat{W}_{a1}, \hat{V}_{a1}, \sigma_a$ are the estimated weights and the activation function of actor NN respectively.

Remark 2. *The admissibility of the initial control policy, u_0, is a necessary condition for ensuring that the value function, $\hat{\mathcal{V}}_j$, is finite. Additionally, the stability of the nonlinear system* (1) *must be maintained when the admissible control policy, u_0, is applied. Next, the following assumption is needed.*

Assumption 5 ([1])**.** *The target weights and the reconstruction errors for both actor and critic NN are bounded on a compact set.*

To move on, using (12) and (13) in (11), the instantaneous HJB or Bellman tracking error can be represented as

$$\hat{\varepsilon}_{B1} \equiv p_1 + \hat{W}_{c1}^\top \Delta\hat{\sigma}_{11}, \tag{14}$$

where $p_1 = \int_{t-T}^{t} e^{-\gamma_1(\tau-t+T)} \left[Q(\zeta_1) + U(\hat{\alpha}) + \lambda_1 \frac{db_{r1}}{dt}\right] d\tau$ denotes the integral reward function, $\Delta\hat{\sigma}_{11} = e^{-\gamma_1 T}\hat{\sigma}_{11} - \hat{\sigma}_{01}$, with T must be chosen as small as possible [1], $\hat{\sigma}_{11} = \sigma(\hat{V}_{c1}^\top \sigma(\bar{\zeta}_1(t))), \hat{\sigma}_{01} = \sigma(\hat{V}_{c1}^\top \sigma(\bar{\zeta}_1(t-T)))$. It is important to note that the HJB error, as defined in equation (14), will serve as the criterion for adjusting the weights as it is computationally tractable.

Remark 3. *The barrier term is integrated into the IRL reward component of the Bellman error, emphasizing an implicit prioritization of safety. This error will be employed in the following section to adjust the critic MNN weights. Moreover, the MNN uses the activation function with a reference trajectory as one of the inputs to approximate the value function and the control input.*

STEP 2: Define the tracking error as $\zeta_2(t) = x_2(t) - \hat{\alpha}(t)$. Taking the time derivative to obtain

$$\dot{\zeta}_2(t) = f_2(\bar{x}_2) + g_2(\bar{x}_2)u - \dot{\hat{\alpha}}(t), \tag{15}$$

where $\bar{x}_2 = [x_1, x_2]^\top$ are the set of system states, $u(t)$ is the actual control. The optimal performance index function $\mathcal{V}_2^*(\bar{\zeta}_2)$ where $\bar{\zeta}_2 = [\zeta_2^\top, \hat{\alpha}^\top, \kappa_{e2}^\top]^\top$ leading to optimal controller as

$$\begin{aligned} \mathcal{V}_2^*\left(\bar{\zeta}_2\right) &= \min_{u\in\mathcal{U}} \left(\int_t^\infty e^{-\gamma_2(\tau-t)} (\zeta_2(\tau)^\top Q_2 \zeta_2(\tau) + u^\top R_2 u + \lambda_2 \frac{db_{r2}}{dt}) d\tau \right) \\ &= \int_t^\infty e^{-\gamma_2(\tau-t)} (\zeta_2(\tau)^\top Q_2 \zeta_2(\tau) + u^{*\top} R_2 u^* + \lambda_2 \frac{db_{r2}}{dt}) d\tau, \end{aligned} \tag{16}$$

where $\mathcal{U}$ denotes the set of admissible control policies, $\mathcal{V}_2(\bar{\zeta}_2) = \left(\int_t^\infty e^{-\gamma_2(\tau-t)} \bar{h}_{2m}\left(\zeta_2, u\right) d\tau\right)$, and $\bar{h}_{2m}(\zeta_2, u) = \zeta_2(\tau)^\top Q_2 \zeta_2(\tau) + u^\top R_2 u + \lambda_2 \frac{db_{r2}}{dt}$, and κ_{e2} is a bounded constraint generated dynamically using $\dot{\kappa}_{e2} = \psi(\kappa_{e2})$, γ_2 is the discount factor, $\gamma_2 > 0$.

To find the optimal control input that minimizes the cost function, we apply the Bellman optimality principle to get

$$\begin{aligned} H_2^* &= \gamma_2 \mathcal{V}_2^* - \frac{\partial \mathcal{V}_2^{*\top}}{\partial \zeta_2} \dot{\zeta}_2 - \zeta_2^\top Q_2 \zeta_2 - U(u^*) \\ &- \nabla b_{r2}^\top \left[f_2(\bar{x}_2(t)) - \dot{\hat{\alpha}} + g_2(\bar{x}_2(t)) u(t) \right] - \Psi_2 = 0 \end{aligned} \tag{17}$$

where $\Psi_2 = \frac{\partial b_{r2}^\top}{\partial \kappa_{e2}} \psi_2(\kappa_{e2}) + \frac{\partial \mathcal{V}_2^*}{\partial \hat{\alpha}} \frac{\partial \hat{\alpha}}{\partial t} + \frac{\partial \mathcal{V}_2^{*\top}}{\partial \kappa_{e2}} \psi_2(\kappa_{e2})$, where $U(u^*) = u^{*\top} R_2 u^*$ and $\mathcal{V}_2^* \triangleq \mathcal{V}_2^*(\bar{\zeta}_2)$. By taking the derivative of (17) with respect to u^*, and applying stationary conditions using Bellman optimality principle, i.e. $\partial H_2/\partial u^* = 0$, the optimal control policy, u^*, is given by

$$u^*(\bar{\zeta}_2) = -\frac{1}{2}R_2^{-1}g_2^\top \nabla \mathcal{V}_2^*(\bar{\zeta}_2) - \frac{\lambda_2}{2}R_2^{-1}g_2^\top \nabla b_{r2}, \tag{18}$$

where $\nabla \mathcal{V}_2^*, \nabla b_{r2}$ are the partial derivatives of the optimal value function $\mathcal{V}_2^*$ and BLF b_{r2} with respect to ζ_2.

Similar to step 1, we can express (16) using MNN as follows

$$\mathcal{V}_2^*(\bar{\zeta}_2) = W_{c2}^\top \sigma(V_{c2}^\top \sigma(\bar{\zeta}_2)) + \epsilon_{c2}(\bar{\zeta}_2), \tag{19}$$

where σ is the activation function, W_{c2} and V_{c2} are the output and hidden layer target weights, respectively, which are unknown but bounded on a compact set, $\sigma_{12} = \sigma(V_{c2}^\top \sigma(\bar{\zeta}_2)), \sigma_{22} = \sigma(\bar{\zeta}_2)$. Now to use the IRL formulation, similar to the step-1 we have

$$\begin{aligned} \mathcal{V}_2^*(\bar{\zeta}_2(t-T)) = \int_{t-T}^{t} e^{-\gamma_2(\tau - t + T)} \\ [Q(\zeta_2(\tau)) + U(u^*) + \lambda_2 \frac{db_{r2}}{dt}]d\tau + e^{-\gamma_2 T}\mathcal{V}_2^*(\bar{\zeta}_2(t)). \end{aligned} \tag{20}$$

Similarly, an actor MNN will approximate the optimal virtual controller (18) on a compact set as

$$u^* = W_{a2}^\top \sigma_a\left(V_{a2}^\top \sigma_a(\bar{\zeta}_2)\right) + \varepsilon_2\left(\bar{\zeta}_2\right) - \frac{\lambda_2}{2}R_2^{-1}g_2^\top \nabla b_{r2}, , \tag{21}$$

where W_{a2}, V_{a2} are the target bounded unknown weights and ε_2 denotes the function reconstruction error, $\sigma_a(\bar{\zeta}_2(t))$ is the activation function. Substituting (19) in (20), the HJB approximation error can be written as

$$\varepsilon_{B2} \equiv \int_{t-T}^{t} e^{-\gamma_2(\tau - t + T)}\left[Q(\zeta_2) + U(u^*) + \lambda_2 \frac{db_{r2}}{dt}\right]d\tau + \varphi_2, \tag{22}$$

where, $\varphi_2 = e^{-\gamma_2 T}W_{c2}^\top \sigma_{12}(\bar{\zeta}_2(t)) - W_{c2}^\top \sigma_{02}(\bar{\zeta}_2(t-T))$, $\varepsilon_{B2} = \varepsilon_2(\bar{\zeta}_2(t-T)) - e^{-\gamma_2 T}\varepsilon_2(\bar{\zeta}_2(t))$. Since the target weights, W_{c2} and V_{c2}, are unknown, approximate (19) on a compact set as

$$\hat{\mathcal{V}}_2(\bar{\zeta}_2) = \hat{W}_{c2}^\top \sigma(\hat{V}_{c2}^\top \sigma(\bar{\zeta}_2)), \tag{23}$$

where $\hat{V}_{c2}, \hat{W}_{c2}$ are the estimated weights of critic NN, and σ is the nonlinear activation function and $\hat{\sigma}_{12} = \sigma(\hat{V}_{c2}^\top \sigma(\bar{\zeta}_2))$. Since the target weights are unknown, the actor NN will be designed to estimate the optimal control input given by

$$\hat{u} = \hat{W}_{a2}^{\top}\sigma_a(\hat{V}_{a2}^{\top}\sigma_a(\bar{\zeta}_2)) - \frac{\lambda_2}{2}R_2^{-1}g_2^{\top}\nabla b_{r2}, , \tag{24}$$

where $\hat{W}_{a2}, \hat{V}_{a2}, \sigma_a(\bar{\zeta}_2)$ are the estimated weights and the activation function of an actor NN. To move on, using (23) in (22), the instantaneous HJB error can be represented as

$$\hat{\varepsilon}_{B2} \equiv p_2 + \hat{W}_{c2}^{\top}\Delta\hat{\sigma}_{12}, \tag{25}$$

where, $p_2 = \int_{t-T}^{t} e^{-\gamma_2(\tau-t+T)}\left[Q(\zeta_2) + U(\hat{u}) + \lambda_2\frac{db_{r2}}{dt}\right]d\tau$ is the integral reward function, $\Delta\hat{\sigma}_{12} = e^{-\gamma_2 T}\hat{\sigma}_{12} - \hat{\sigma}_{02}$, with T must be chosen as small as possible [1], $\hat{\sigma}_{12} = \sigma(\hat{V}_{c2}^{\top}\sigma(\bar{\zeta}_2(t))), \hat{\sigma}_{02} = \sigma(\hat{V}_{c2}^{\top}\sigma(\bar{\zeta}_2(t-T)))$.

Remark 4. *The value function is determined by the weights of the critic neural network. The control policy is composed of the weights of the actor neural network from all the layers and a separate term based on safety considerations. This term ensures that the system state remains within the safe set at all times. Furthermore, the IRL-based technique just necessitates knowledge of the control input matrix g_j to compute the control input error, without the necessity for internal dynamics. Given the unknown nature of the dynamics, a NN identifier will be implemented.*

2.4. NN Identifier

In this section, for approximating the unknown dynamics $f(x_1)$ and $g(x_1)$ over a compact domain, a single-layer NN identifier is employed. Utilizing $\hat{g}(x_1)$, derived from the NN identifier, the control strategy is then implemented on the nonlinear system by substituting the actual matrix g_1 with the estimated matrix to obtain the estimated control coefficient matrix $\hat{g}_1$. The approximations are represented as $f(x_1) = {V_{f_1}}^{\top}\sigma_{f_1}(x_1) + \varepsilon_{f_1}(x_1)$ and $g(x_1) = V_{g_1}^{\top}\sigma_{g_1}(x_1) + \varepsilon_{g_1}(x_1)$, where V_{f_1} and V_{g_1} are the NN weight matrices, $\sigma_{f_1}(x_1,)$ and $\sigma_{g_1}(x_1)$ are the activation functions, and $\varepsilon_{f_1}(x_1)$ and $\varepsilon_{g_1}(x_1)$ are the NN reconstruction errors respectively. Since the target identifier weights for online learning are not known, therefore, the estimated identifier weights can be used to estimate the unknown dynamics as $\hat{f}_1(x_1) = \hat{V}_{f_1}^{\top}\sigma_{f_1}(x_1)$ and $\hat{g}_1(x_1) = \hat{V}_{g_1}^{\top}\sigma_{g_1}(x_1)$, where $\hat{V}_{f_1}$ and $\hat{V}_{g_1}$ represent the estimated tunable weights of the NN identifier. Define

$$\dot{x_1}(t) = Z_1^{\top}\boldsymbol{\sigma}(\boldsymbol{\xi_1})\bar{\alpha} + \varepsilon_I(x_1), \tag{26}$$

where $\bar{\alpha} = [1, \alpha]^\top$, is the augmented control input, α is given by (18), $Z_1 = \begin{bmatrix} V_{f_1}^\top & V_{g_1}^\top \end{bmatrix}^\top \in \mathbb{R}^{2l \times n}$ represents the augmented NN identifier weights, and $\boldsymbol{\sigma}(\boldsymbol{\xi_1}) = \text{diag}\{\sigma_{f_1}(x_1), \sigma_{g_1}(x_1)\}$ denotes the augmented activation function for the NN identifier. The reconstruction error of the NN identifier, denoted by $\varepsilon_I(x_1)$, is defined as $\varepsilon_I(x_1) = (\varepsilon_{f_1}(x_1) + \varepsilon_{g_1}(x_1)\alpha)$. Next, we state the following Assumption.

Assumption 6 ([28])**.** *The NN identifier's reconstruction error is bounded above on a compact set such that* $\|\varepsilon_I\|^2 \leq b_0 \|x_1\|^2$ *and* $\|Z_j\| \leq Z_{m_j}$ *where* Z_{m_j}, b_0 *are constants.*

Moving on, define the dynamics of the NN identifier as

$$\dot{\hat{x}}_1(t) = \hat{Z}_1^\top \boldsymbol{\sigma}(\boldsymbol{\xi}_1)\bar{\alpha} + K(x_1 - \hat{x}_1), \tag{27}$$

In this case, K represents a constant identifier gain matrix, and $\hat{x}_1(t)$ indicates the estimated state vector. The weights associated with the enhanced NN identifier are expressed as follows: $\hat{Z}_1 = \begin{bmatrix} \hat{V}_{f_1}^\top & \hat{V}_{g_1}^\top \end{bmatrix}^\top$, $e_{i_1} = x_1 - \hat{x}_1$ is the definition of the state estimation error. The weight update laws for the single layer NN identifier are obtained using Lyapunov analysis as

$$\dot{\hat{Z}}_1 = -\alpha_{v1}\hat{Z}_1 + \sigma(\boldsymbol{\xi}_1)\bar{\alpha}e_{i_1}^\top, \tag{28}$$

where $\alpha_{v1} > 0$ is a tuning parameter, the sigma modification term is added to relax the persistence of excitation (PE) condition. Similarly, the identifier will be designed for the second step. The following subsection presents the derivation of the novel weight update law for safe RL-based actor-critic NN.

2.5. SVD-based weight tuning laws

In the context of Backpropagation or SGD, the error between actual and target NN output is propagated backward to adjust the weights of the MNN. This process faces issues like vanishing gradients when using sigmoidal or tanh activation functions, leading to ineffective learning.

A solution to this is the proposed SVD-based technique with an exploration feature, which modifies the singular values of the gradient by adding a small random noise, thereby avoiding instability. The SVD of the NN activation function gradient is represented as $\nabla\sigma_j(X_j) = \mathcal{U}_j\Sigma_j\mathcal{W}_j^\top = A_j$, with the modified SVD being $\bar{A}_j = \mathcal{U}_j\Sigma_j\mathcal{W}_j^\top + \mathcal{U}_j\mathfrak{e}_{0j}I\mathcal{W}_j^\top$. Next, the following Lemma is stated

Simplifying for each subsystem we have

Lemma 1. *Using the SVD approach, the update laws for the proposed actor-critic MNN are derived as:*

$$\begin{aligned}\dot{\hat{W}}_{c1}(t) &= \beta_{cw1}\left(S_{cw1}\hat{\varepsilon}_{B1}\right) - c_{w1}\hat{W}_{c1},\\ \dot{\hat{V}}_{c1}(t) &= \beta_{cv1}\left(\bar{S}_{cv1}\hat{\varepsilon}_{B1}\right) - c_{v1}\hat{V}_{c1},\\ \dot{\hat{W}}_{a1}(t) &= \beta_{aw1}\left(S_{aw1}u_{e1}\right) - \gamma_{aw1}\hat{W}_{a1},\\ \dot{\hat{V}}_{a1}(t) &= \beta_{av1}\left(\bar{S}_{av1}u_{e1}\right) - \gamma_{av1}\hat{V}_{a1}.\end{aligned}$$

where $\beta_{cw1}, \beta_{cv1}, \beta_{aw1}, \beta_{av1}$ are the tuning parameters chosen by the designer, $c_{w1}, c_{v1}, \gamma_{aw1}, \gamma_{av1}$ are the design parameters and are chosen smaller to not let weights go zero, $S_{cw1} = \frac{\Delta\hat{\sigma}_{11}}{1+\Delta\hat{\sigma}_{11}^\top\Delta\hat{\sigma}_{11}}$, where $\Delta\hat{\sigma}_{11} = e^{-\gamma_1 T}\sigma(\hat{V}_{c1}^\top\sigma(\bar{\zeta}_1(t))) - \sigma(\hat{V}_{c1}^\top\sigma(\bar{\zeta}_1(t-T)))$, $\bar{S}_{cv1} = \frac{\bar{\kappa}_1}{1+\bar{\kappa}_1^\top\bar{\kappa}_1}$, with $\bar{\kappa}_1 = (\bar{A}_{11}(\sigma(\bar{\zeta}_1))^\top)^\top - (\bar{A}_{21}(\sigma(\bar{\zeta}_1(t-T)))^\top)^\top$. Here, $\bar{A}_{1j}$ and $\bar{A}_{2j}$ are defined as:

$$\begin{aligned}\bar{A}_{11} &= \mathcal{U}_{11}\Sigma_{11}\mathcal{W}_{11} + \mathcal{U}_{11}(\delta_{11}\Sigma_{11})\mathcal{W}_{11},\\ \bar{A}_{21} &= \mathcal{U}_{21}\Sigma_{21}\mathcal{W}_{21} + \mathcal{U}_{21}(\delta_{21}\Sigma_{21})\mathcal{W}_{21},\end{aligned}$$

with $\mathcal{U}_{11}\Sigma_{11}\mathcal{W}_{11} = svd(\nabla(\sigma(V_{c1}^\top\sigma(\zeta_1))))$ and $\mathcal{U}_{21}\Sigma_{21}\mathcal{W}_{21} = svd(\nabla(\sigma(V_{c1}^\top\sigma(\zeta_1(t-T)))))$ where $\nabla\sigma(x)$ is the partial derivative of σ with respect $x = V_{c1}^\top\sigma(\bar{\zeta}_1)$ and $\nabla\sigma(x(t-T))$ is the partial derivative with respect to $x(t-T) = \hat{V}_{c1}^\top\sigma(\bar{\zeta}_1(t-T))$.

Similarly, $S_{aw1} = \frac{\kappa_{aw1}}{1+\kappa_{aw1}^\top\kappa_{aw1}}$ and $\bar{S}_{av1} = \frac{\bar{k}_{av1}}{1+\bar{k}_{av1}^\top\bar{k}_{av1}}$, where $\kappa_{aw1} = \hat{\sigma}_a(\hat{V}_{a1}^\top\sigma_a(\bar{\zeta}_1))$, $\bar{k}_{av1} = (\bar{A}_{31}(\sigma_a(\zeta_1))^\top)^\top$. The $\bar{A}_{31}$ is given by

$$\bar{A}_{31} = \mathcal{U}_{31}\Sigma_{31}\mathcal{W}_{31} + \mathcal{U}_{31}(\delta_{31}\Sigma_{31})\mathcal{W}_{31},$$

representing the SVD of $\nabla(\sigma_{a1}(\hat{V}_{a1}^\top\sigma_a(\bar{\zeta}_1)))$, where $\nabla(\sigma(x))$ means the gradient of σ with respect to input $x = \hat{V}_{a1}^\top\sigma(\bar{\zeta}_1)$, δ_{ij} is the random noise added to singular values. The error u_{ej} between estimated and actual control for each step is defined as

$$u_{e1} = \hat{W}_{a1}^\top\sigma_{a1}(\hat{V}_{a1}^\top\sigma_{a1}(\bar{\zeta}_1(t))) + \frac{1}{2}R_1^{-1}\hat{g}_1^\top\nabla\mathcal{V}_1$$

where $\nabla\mathcal{V}_1 = \nabla(\hat{\sigma}(\hat{V}_{c1}^\top\sigma(\bar{\zeta}_1)))^\top\hat{W}_{c1}$ and $\hat{e}_{Bj}$ is the estimated Bellman error for each step given by eqs. (14), (25) and $\hat{\mathcal{V}}_j$ is the estimated value function obtained using the critic MNN. In a similar way the weight update laws can be obtained for the second step of backstepping.

Proof. The update laws for critic MNN are derived by defining the performance measure first.

$$E_j = \frac{1}{2}\mathrm{tr}(\hat{\varepsilon}_{Bj}^{\top}\hat{\varepsilon}_{Bj}). \tag{29}$$

To perform online updates for both layers, we use the normalized gradient descent method. The derivatives of E_j with respect to $\hat{W}_{cj}$ and $\hat{V}_{cj}$ are computed and then used to update the weights as $\dot{\hat{W}}_{cj} = -\beta_{cwj}\frac{\partial E_j}{\partial \hat{W}_{cj}}, \dot{\hat{V}}_{cj} = -\beta_{cvj}\frac{\partial E_j}{\partial \hat{V}_{cj}}$. The modified SVD is then used to replace the gradients, leading to the critic weight update laws given in Lemma 1. Similarly, the weight update laws for the actor MNN are derived by defining the performance function as $E_{aj} = \frac{1}{2}u_{ej}^2$ By using the gradient descent approach as $\dot{\hat{W}}_{aj}(t) = -\beta_{awj}\frac{\partial E_{aj}(t)}{\partial \hat{W}_{aj}}, \dot{\hat{V}}_{aj}(t) = -\beta_{avj}\frac{\partial E_{aj}(t)}{\partial \hat{V}_{aj}}$. These derivatives are used to update the NN weights with additional terms added for stability. The gradients are again replaced with the modified SVDs, resulting in the actor weight update laws as given in Lemma 1. □

Remark 5. *To ensure that the gradient does not decrease too rapidly during learning and to promote exploration of the parameter space, it is beneficial to keep the singular vectors unchanged while introducing noise to the singular values of the gradient can help prevent a decrease of gradient during learning and address the issue of vanishing gradient ultimately enhancing robustness and efficiency.*

Remark 6. *Though the sigma modification term in the update laws relaxes the persistence of the excitation (PE) condition, preventing parameter estimate divergence and ensuring weight estimate error boundedness, it does not overcome the catastrophic forgetting problem; an online LL is needed.*

Next, LL for MNNs is discussed.

2.6. Lifelong Learning-based Control

Catastrophic forgetting is a challenge faced by online NN systems operating in multitasking settings, where previously learned information can be lost when new tasks are introduced. On the side "LL" signifies the ability to adapt to tasks while retaining past knowledge.

To address forgetting in multitasking scenarios a proposed online learning and adaptation technique is utilized. This method, integrated into a trajectory tracking control system employing MNN-methodology helps prevent forgetting by identifying and preserving parameters during optimization. The performance function to be used for the proposed LL is given as

$$L(\hat{W}_{cj}, \hat{V}_{cj}) = L_{Bj} + \frac{\lambda_{wj}}{2}\bar{\Omega}_{wj}\left(\hat{W}_{cj} - \hat{W}^*_{Acj}\right)^2 + \frac{\lambda_{vj}}{2}\bar{\Omega}_{vj}\left(\hat{V}_{cj} - \hat{V}^*_{Acj}\right)^2, \quad (30)$$

where L_{Bj} is the loss function, $\bar{\Omega}_{wj}$ and $\bar{\Omega}_{vj}$ are the Fisher information matrices (FIM) that are utilized to obtain the significance of i-th weight of the MNN, and $\hat{W}^*_{Acj}$, $\hat{V}^*_{Acj}$ are the optimal MNN weights from the previous task. When calculating the FIM, for each task it's important to compute the gradient of the Bellman error to the weights denoted as $\nabla_{\hat{W}_{cij}}\hat{\varepsilon}_{Bj}$, where $\hat{\varepsilon}_{Bj}$ represents the HJB error at each stage. This evaluation of gradients is essential for understanding how the NN weights are related to the Bellman error from tasks. Moreover examining the Jacobian matrix of this function provides insights, into which model parameters are most influential and critical in capturing the Bellman error and ensuring optimal performance continuity from earlier tasks. The Jacobian for the weights of each layer is computed as $J_{\hat{W}_{cij}} = \frac{\partial \nabla_{\hat{W}_{cij}}\hat{\varepsilon}_{Bj}}{\partial \hat{W}_{cij}}$. The FIM is then approximated as

$$\Omega_{wij} = \mathbb{E}(J_{\hat{W}_{cij}} J^\top_{\hat{W}_{cij}}) \quad (31)$$

where $\mathbb{E}$ denotes the expectation of the product of the Jacobian matrix and its transpose. In a similar way, the diagonal FIM can be calculated for each NN weight. The elements of the FIM matrix contain the data regarding the HJB error associated with each weight. The FIM for task 1 is equal to zero, while the FIM for task 2, which is estimated based on task 1, is bounded due to the bounded nature of the closed-loop system for task 1 as shown in stability proof. The proposed method can be scaled to accommodate N tasks by maintaining the MNN weights close to those learned during the first two tasks, either as one penalty term or as two distinct penalties.

The additional term in the critic weight update law is derived using the performance function (30) and the normalized gradient descent shown in Lemma 1. Next, the following theorem is presented

Theorem 1. *Consider the system* (1)*, the error dynamics* (4)*,* (15)*, the value function* (5)*,* (16) *and the NN identifier dynamics* (27)*, with the identifier NN weight update laws* (28)*, the estimated control* (13) *and* (24)*. Let Assumptions 1 through 5 hold for each task, and let the initial value u_0 be* *admissible in each task with SVD-based actor-MNN weight update laws be given by Lemma 1 and the LL-based critic MNN tuning laws given by*

$$\dot{\hat{W}}_{cj} = \beta_{cwj} S_{cwj} \hat{\varepsilon}_{Bj}^{\top} - c_{wj} \hat{W}_{cj} - \mathcal{O}_{wj}(\hat{W}_{cj} - \hat{W}^{*}_{Acj}),$$
$$\dot{\hat{V}}_{cj} = \beta_{cvj} (\bar{S}_{cvj} \hat{\varepsilon}_{Bj}^{\top})^{\top} - c_{vj} \hat{V}_{cj} - \mathcal{O}_{vj}(\hat{V}_{cj} - \hat{V}^{*}_{Acj}).$$

where $\mathcal{O}_{wj} = \alpha_{wj}\lambda_{wj}\bar{\Omega}_{wj}, \mathcal{O}_{vj} = \alpha_{vj}\lambda_{vj}\bar{\Omega}_{vj}$, $\lambda_{wj}, \lambda_{vj}$ are the design parameters, α_{wj}, α_{vj} are the NN learning rates, Ω_{wj}, Ω_{vj} are the significance of weights after each task, for weights $\hat{W}_{cj}$ and $\hat{V}_{cj}$ respectively, $\hat{W}_{cj}, \hat{V}_{cj}$ are the weights to be optimized, and $\hat{W}^{*}_{Acj}, \hat{V}^{*}_{Acj}$ are the optimized weights from the previously learned task, Then, the overall closed-loop system will remain UUB, and the estimated control policy is close and bounded to the optimal value.

Proof. See Appendix. □

Next, the propositions on admissibility and safety are stated.

Proposition 1. *A control policy obtained from the optimal performance index $\mathcal{V}_i^*$, subject to the system* (1)*, is admissible if and only if $u_i \in \mathcal{U} \cap \mathcal{U}_s$.*

Proof. The virtual and optimal control policy u_i is admissible when it is bounded and finite given performance index $\mathcal{V}_i^*, i = 1, 2$, for all t. This implies that both the BF $b_{ri}(\zeta_i(t), \kappa_{ei})$ and the cost to go $h_{im}(\zeta_i, u_i)$ have to be bounded and finite, $\forall t$. Taking into account that $h_{im}(\zeta_i, u_i)$ for $u_i \in \mathcal{U}$ and $b_{ri}(\zeta_i(t), \kappa_{ei})$ for $u_i \in \mathcal{U}_s$ (where $\mathcal{U}_s$ is the safe input set) are both bounded, we can deduce that, if $u_i \in \mathcal{U} \cap \mathcal{U}_s$, and given the discounting factor $e^{-\gamma_i(\tau - t)}$, we confirm that $\mathcal{V}_i^* < \infty, \forall t$. This verifies that $\mathcal{V}_i^*$ is bounded and remains finite $\forall t$. Moreover, since $u_i \in \mathcal{U}$, u_i stabilizes the system (1) as proven in Theorem 1. Thus, a control policy $u_i \in \mathcal{U} \cap \mathcal{U}_s$ is admissible. □

Proposition 2. *Consider the dynamical system represented by* (1)*. Assume the origin is contained in $\mathcal{S}$, and that $\zeta_0 \in \mathrm{Int}(\mathcal{S})$. Further assume that there exists a smooth function $\mathcal{V}_i^* \geq 0, i = 1, 2$ which satisfies* (6) *and* (17)*. Then the closed-loop system* (1) *with the control input* (13) *and* (24) *will keep the system safe while minimizing* (5) *and* (16) *respectively.*

Proof. We can prove it by contradiction. Based on Assumption 4, there exists at least one admissible control, since $\mathcal{U} \cap \mathcal{U}_s \neq \emptyset$. Let $u_i \in \mathcal{U} \cap \mathcal{U}_s$

be admissible, thus the modified cost functions (5),(16) are bounded. Now suppose $\zeta_0 \in \mathcal{S}$ and $\mathcal{S}$ is not forward invariant. Then $\exists \bar{t} \geq 0$ such that $\zeta_i(\bar{t}) \to \partial\mathcal{S} \implies b_{ri}(\zeta_i(\bar{t}), \kappa_{ei}) \to \infty \implies \mathcal{V}_i^*(\zeta_i(\bar{t})) \to \infty$, which contradicts. Thus, $\nexists t \geq 0$ for which $\zeta_i \to \partial\mathcal{S}$, so $\zeta_0 \in \mathcal{S} \implies \zeta_i \in \mathcal{S}, \forall t \geq 0$ and by Definitions 1, 2, and Assumption 2, 4 $\mathcal{S}$ is forward invariant and (1) is safe. □

Remark 7. *This optimal framework ensures that the error vector stays within a safe set by using the safety based controller which includes the nominal control input and the additional BLF-based control input obtained using the Bellman optimality principle and the augmented value function in each subsystem.*

Remark 8. *The LL terms based on regularization, when integrated with Lemma 1 MNN update laws, cause an increase in the weight estimation error bounds, whereas stability is unaffected. The bounds in* (**??**) *can be reduced through the selection of a number of neurons or by adjusting the learning rates. While the efficacy of the proposed LL method depends on the overlap of information between tasks, but without overlap, LL still helps prevent forgetting previous tasks.*

3. Simulation Results

The kinematic and dynamic equation of the mobile robot (MR) can be expressed in a strict feedback form as [29]

$$\begin{aligned} \dot{q} &= f_q(q) + g_q(q)\nu, \\ \dot{\nu} &= f_\nu(q, \nu) + g_v(q_j, \nu_j)\tau, \end{aligned} \tag{32}$$

where $q = [x, y, \Psi]^\top$, x, y, and Ψ are actual Cartesian position and orientation of the physical robot, $\nu = [v, \omega]^\top$and v, ω are linear and angular velocities, respectively, $\tau = [\tau_u, \tau_r]^\top$ denotes the control inputs, $f_q(q) = 0_{n\times 1}$, $g_q(q) = S(q)$, $f_\nu(q, \nu) = -M^{-1}C\nu$-$M^{-1}F$, $g_\nu(q, \nu) = M^{-1}B(q)$.

$$\begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} = \begin{bmatrix} \cos\Psi & \sin\Psi & 0 \\ -\sin\Psi & \cos\Psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_r - x \\ y_r - y \\ \Psi_r - \Psi \end{bmatrix} \tag{33}$$

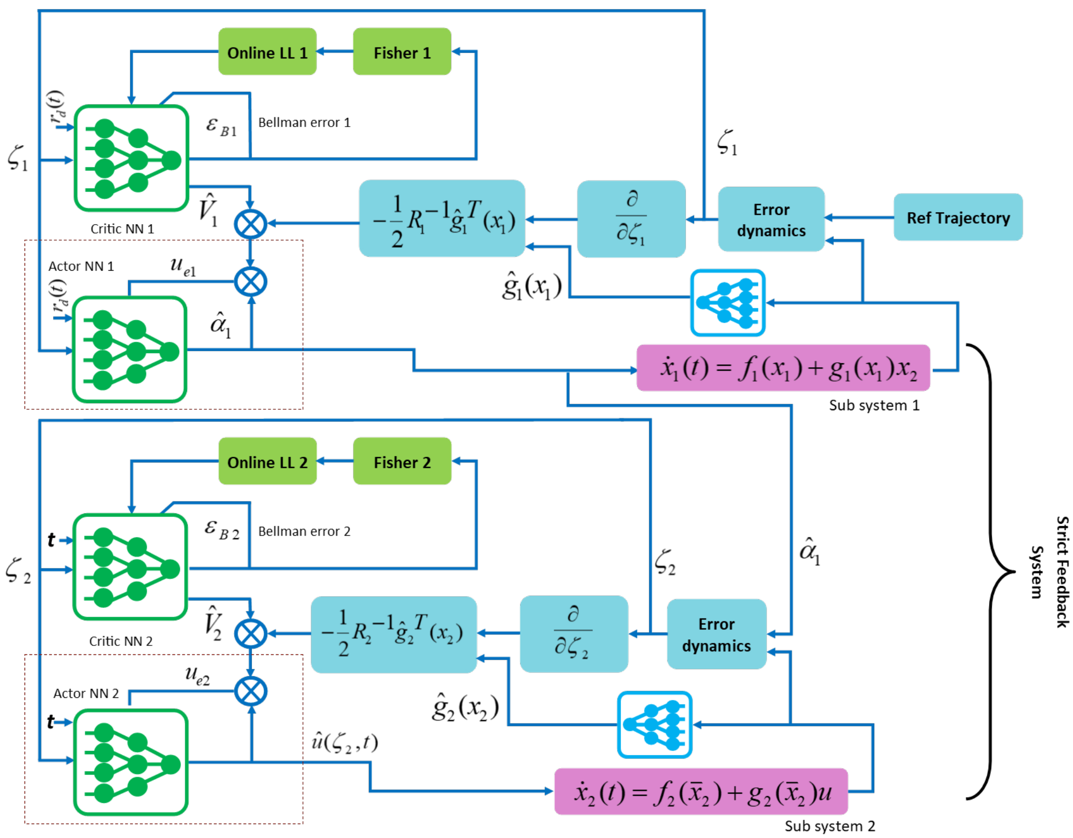


Figure 1: Lifelong learning-based OAT control using MNNs.

Final version submitted to Control Engineering Practice

Algorithm 1 Lifelong Learning-based Optimal Control for Strict-Feedback Systems

Given: System dynamics, desired trajectory, constraints
Require: Optimal control inputs, updated NN weights
Initialize: Learning rates α, discount factors γ, regularization parameter λ_{reg},design parameters.
Initialize multi-layer actor NN weights $W_{\text{actor}}, V_{\text{actor}}$
Initialize multi-layer critic NN weights $W_{\text{critic}}, V_{\text{critic}}$ with lifelong learning
Initialize single-layer identifier NN weights $W_{\text{identifier}}$
Initialize Fisher information matrix $F \leftarrow \emptyset$
for each task T in set of tasks **do**
    if T is the first task **then**
        No lifelong learning adjustments for the critic
    else
        Compute Fisher information matrix $F[T]$ from previous task
        Update $W_{\text{critic}}, V_{\text{critic}}$ with LL using $F[T]$
    end if
    for each time step within task T **do**
        for each subsystem S **do**
            Calculate tracking error $e_{\text{track}}[S]$
            Calculate virtual control $u_{\text{virtual}}[S]$
            Compute control input $u[S]$ using actor NN with current $W_{\text{actor}}, V_{\text{actor}}$
            Apply control input $u[S]$ to subsystem S
            Propagate dynamics and observe new state
            Calculate cost and Bellman error for subsystem S
            Update identifier NN weights $W_{\text{identifier}}$ using identifier errror
            Update actor NN weights $W_{\text{actor}}, V_{\text{actor}}$ using SVD based update laws
            if not the first task **then**
                Update critic NN weights $W_{\text{critic}}, V_{\text{critic}}$ using Bellman error and $F[T]$ with the proposed SVD based update laws.
            end if
            Enforce safety using TVBLF for subsystem S
        end for
    end for
    if not the first task **then**
        Store optimized critic weights for LL of next task
    end if
end for

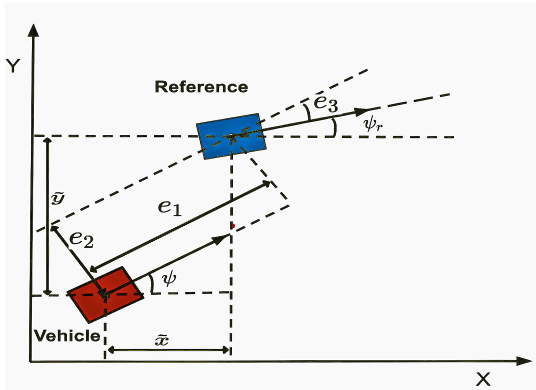
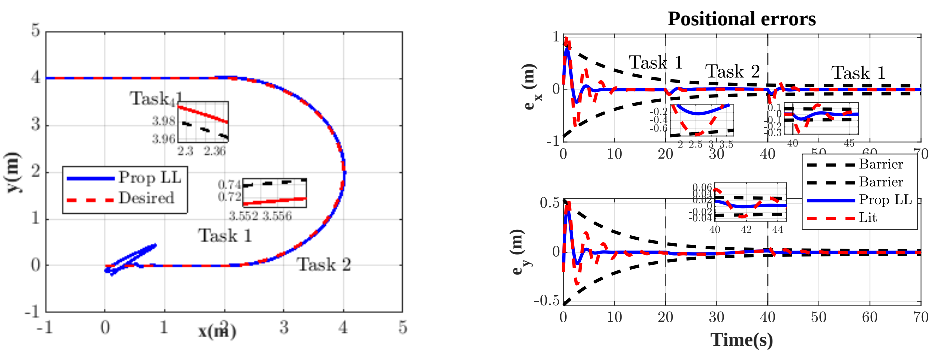


Figure 2: Mobile robot tracking a reference.

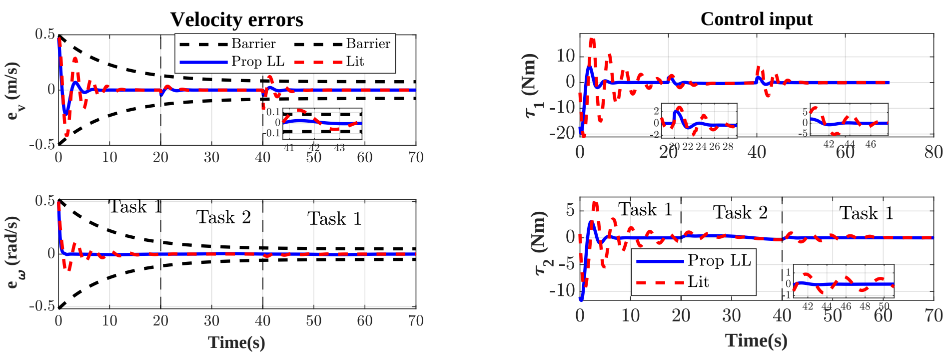
The virtual reference position and orientation are given by x_r, y_r, and Ψ_r. The leader tracks a reference trajectory in various tasks, obtaining the virtual control input α. Velocity tracking error $e_c = \nu - \alpha$ is calculated, and using Lyapunov analysis and system dynamics (32), the actual control input τ is determined. For a single leader robot tracking, the simulation is run for two different tasks where trajectories are changed for each of them. Task one is run for a duration of 0 to 20 secs with trajectories and dynamics parameters as , $m = 10kg, I = 5kg - m^2, R = 0.5m$. Task two is run for a duration of 20 to 30 secs with different desired trajectory , after 30 secs, task one is executed again, $\kappa_{e1} = (0.9-0.08)e^{-0.1t}+0.075, \kappa_{e2} = (0.6-0.08)e^{-0.1t}+0.02$, Given the time-varying constraints κ_{e1} and κ_{e2}, with the generator dynamics as $\dot{\kappa}_{e1} = -0.1(0.9-0.08)e^{-0.1t}, \quad \kappa_{e1}(0) = 0.9-0.08+0.075$ $\dot{\kappa}_{e2} = -0.1(0.6-0.08)e^{-0.1t}, \quad \kappa_{e2}(0) = 0.6-0.08+0.02$, $q_0 = [1, 2, \pi/8]$, $\alpha_i = 1, \beta_{cj} = 2, \beta_{aj} = 1, Q_{1i} = 5,$ $R_{ij} = 0.4, \lambda = 0.13, \delta_{0i} = 0.06$. $\lambda_{wj} = 1.5, \lambda_{vj} = 1$, and $\gamma_j = 0.95$, $T = 0.05$.

All the NNs for actor-critic were designed to have two layers. The critic NN was designed with 22 neurons and actor NN with 20 neurons in the hidden layer; the identifier NN, on the other hand, is a single layer, configured with 12 neurons within its hidden layer, sigmoid is used as an activation function, and the NN weights are chosen in the interval [-1,1] for identifier and for actor critic frameowrk the initial admissible control which is chosen as a nominal PD control in conjunction with the discounted cost function helps to find an appropriate set of weights for initialization. The study presents mobile robot trajectory tracking and a comparative analysis with recent literature



(a) (b)

Figure 3: Mobile robot tracking in Cartesian coordinates and tracking errors using proposed method and literature[6].

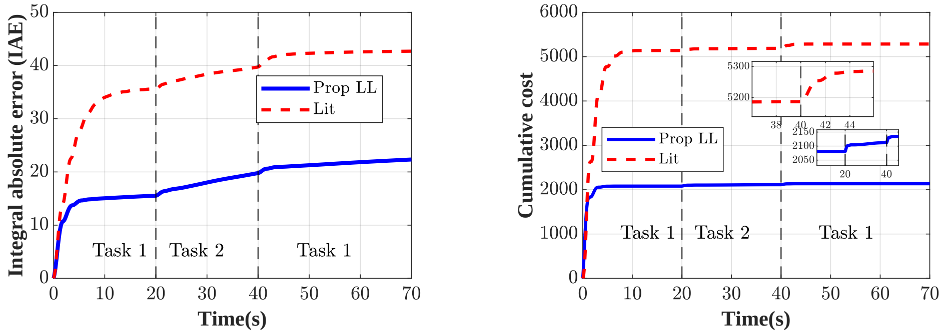


(a) (b)

Figure 4: Velocity errors and torque inputs using proposed safe LL-based MNN and recent literature[6].

[6]. The abbreviations used include 'prop LL', which represents the proposed multilayer LL method, and 'Lit' represents the literature [6].

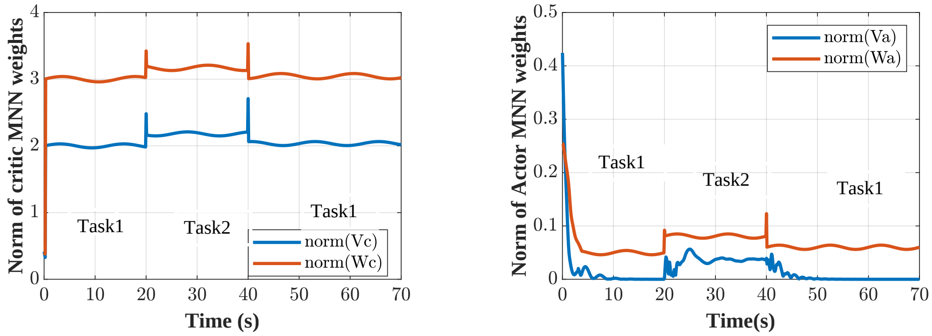
Analyzing Figures 3 and 4, it is evident that the errors consistently remain low in the cases of prop MNN-based LL method, as shown in Fig 3b and Fig 4a. Conversely, the 'Lit' method encounters instances where the errors are higher, particularly initially and during task changes. This is primarily because Tasks 1 and 2 have some overlaps, with only the tracking trajectories being different, while maintaining the same parameters and dynamics as Task 1; therefore, safe LL method shows an improvement when executing Task 2. The control efforts are lower and bounded using safe LL method in Fig 4b. The TVBLF is important for immediately constraining



(a) (b)

Figure 5: Integral absolute error and cumulative cost using proposed safe LL-based MNN and recent literature[6]

.



(a) (b)

Figure 6: Norm of actor-critic MNN weights in multitasking environment.

errors within predefined boundaries during the initial stages of task execution, dynamically adapting to ensure the minimization of error propagation. Concurrently, LL is vital for preserving and utilizing accumulated knowledge effectively, thereby mitigating forgetting, particularly as the system experiences continuous modifications or advancements during the change of tasks.

Figures 5a and 5b illustrate the lower integral absolute error (IAE) and cumulative costs achieved using the proposed safe LL method, as indicated by the plot. In contrast, the Lit method shown by the red plot shows an increase in these metrics initially as well as during the task change due to the constraint violation and catastrophic forgetting when revisiting task 1. The actor-critic weights are shown in Fig 6 for the multi-tasking scenario as the tasks change, there is a slight spike observed, which is because of the sudden task change however, because of overlapping between task 2 and task 1, since only the reference trajectory is changed in tasks therefore, the weights do not change much and remain close to task 1, when moving back to task 1 again at 40 s the weights remain close to the task 1 (0 to 20s) because the LL term uses the knowledge gained from task 1 again when revisiting it hence mitigating the issues of forgetting. In the actor NN the LL is not explicitly added to actor weights since the weights are obtained using control input error obtained using the LL-based critic weights; therefore, LL also affects the actor weights implicitly.

4. Conclusion

This work presented a safe, optimal LL trajectory tracking control strategy for nonlinear CT strict-feedback systems based on MNNs. By combining SVD-based MNNs with LL, an efficient solution to the HJB equations with uncertain dynamics is obtained. This approach improves multitasking performance while reducing the catastrophic forgetting problem in MNNs by using the LL-based penalty term in the weight update laws. Efficient constraint handling and safety are ensured by the addition of a barrier Lyapunov function which gives an explicit safety control term in the control law to handle safety. Our method is effective, as demonstrated by simulations, which also show significant cost reduction over previous research methods and better trajectory tracking control.

References

[1] H. Modares, F. L. Lewis, Optimal tracking control of nonlinear partially-unknown constrained-input systems using integral reinforcement learning, Automatica 50 (7) (2014) 1780–1792.

[2] Q. Wei, Z. Liao, G. Shi, Generalized actor-critic learning optimal control in smart home energy management, IEEE Transactions on Industrial Informatics 17 (10) (2021) 6614–6623.

[3] M. Li, D. Wang, J. Qiao, Neural critic learning for tracking control design of constrained nonlinear multi-person zero-sum games, Neurocomputing 512 (2022) 456–465.

[4] H. Zou, G. Zhang, Dynamic event-triggered-based single-network adp optimal tracking control for the unknown nonlinear system with constrained input, Neurocomputing 518 (2023) 294–307.

[5] W. Zhong, M. Wang, Q. Wei, J. Lu, A new neuro-optimal nonlinear tracking control method via integral reinforcement learning with applications to nuclear systems, Neurocomputing 483 (2022) 361–369.

[6] G. Wen, C. L. P. Chen, S. S. Ge, Simplified optimized backstepping control for a class of nonlinear strict-feedback systems with unknown dynamic functions, IEEE TCYB 51 (9) (2021) 4567–4580.

[7] W. Gao, Z.-P. Jiang, Learning-based adaptive optimal tracking control of strict-feedback nonlinear systems, IEEE TNNLS 29 (6) (2018) 2614–2624.

[8] H. Zargarzadeh, T. Dierks, S. Jagannathan, Optimal control of nonlinear continuous-time systems in strict-feedback form, IEEE TNNLS 26 (10) (2015) 2535–2549.

[9] G. Liu, Y. Yang, D.-W. Ding, Q. Li, Adaptive event-based fixed-time tracking design for strict-feedback nonlinear systems with unknown control coefficients, Neurocomputing 541 (2023) 126278.

[10] G. Wen, W. Hao, W. Feng, K. Gao, Optimized backstepping tracking control using reinforcement learning for quadrotor unmanned aerial vehicle system, IEEE Trans on SMC: Systems (2021) 1–12.

[11] B. Cao, X. Nie, J. Cao, Event-triggered adaptive neural networks tracking control for incommensurate fractional-order nonlinear systems with external disturbance, Neurocomputing 554 (2023) 126586.

[12] F. Lewis, A. Yesildirak, S. Jagannathan, Neural network control of robot manipulators and nonlinear systems, Taylor & Francis, 1998.

[13] I. Ganie, S. Jagannathan, Adaptive control of robotic manipulators using deep neural networks 55 (15) (2022) 148–153, 6th IFAC Conference on Intelligent Control and Automation Sciences ICONS 2022.

[14] R. Sun, M. L. Greene, D. M. Le, Z. I. Bell, G. Chowdhary, W. E. Dixon, Lyapunov-based real-time and iterative adjustment of deep neural networks, IEEE Control Systems Letters 6 (2022) 193–198.

[15] H. Gao, W. He, C. Zhou, C. Sun, Neural network control of a two-link flexible robotic manipulator using assumed mode method, IEEE Transactions on Industrial Informatics 15 (2) (2019) 755–765.

[16] M. Roodschild, J. Gotay Sardiñas, A. Will, A new approach for the vanishing gradient problem on sigmoid activation, Progress in Artificial Intelligence 9 (2020) 351–360.

[17] M. H. Cohen, C. A. Belta, Approximate optimal control for safety-critical systems with control barrier functions, 59th IEEE Conference on Decision and Control (CDC) (2020) 2062–2067.

[18] K. Zhao, Y. Song, Removing the feasibility conditions imposed on tracking control designs for state-constrained strict-feedback systems, IEEE Trans. on Automatic Control 64 (3) (2019) 1265–1272.

[19] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, P. Tabuada, Control barrier functions: Theory and applications, in: 2019 18th European Control Conference (ECC), 2019, pp. 3420–3431.

[20] Y. Yang, K. G. Vamvoudakis, H. Modares, Safe reinforcement learning for dynamical games, International Journal of Robust and Nonlinear Control 30 (9) (2020) 3706–3726.

[21] W. Meng, Q. Yang, Y. Sun, Adaptive neural control of nonlinear mimo systems with time-varying output constraints, IEEE TNNLS 26 (5) (2015) 1074–1085.

[22] J. Kirkpatrick, R. Pascanu, N. C. Rabinowitz, J. Veness, D.Kumaran, Overcoming catastrophic forgetting in neural networks, Proceedings of the National Academy of Sciences 114 (2017) 3521 – 3526.

[23] A. Kutalev, A. Lapina, Stabilizing elastic weight consolidation method in practical ml tasks and using weight importances for neural network pruning, ArXiv abs/2109.10021 (2021).

[24] I. Ganie, S. Jagannathan, Lifelong learning control of nonlinear systems with constraints using multilayer neural networks with application to mobile robot tracking, in: 2023 IEEE Conference on Control Technology and Applications (CCTA), 2023, pp. 727–732.

[25] I. A. Ganie, S. Jagannathan, Lifelong learning-based multilayer neural network control of nonlinear continuous-time strict-feedback systems, International Journal of Robust and Nonlinear Control 34 (2) (2024) 1397–1416.

[26] K. Zhao, Y. Song, C. P. Chen, L. Chen, Control of nonlinear systems under dynamic constraints: A unified barrier function-based approach, Automatica 119 (2020) 109102.

[27] A. Bryson, Y. C. Ho, Applied optimal control: Optimization, estimation and control., Taylor and Francis 51 (2017) 496.

[28] R. Moghadam, S. Jagannathan, Optimal adaptive control of uncertain nonlinear continuous-time systems with input and state delays, IEEE TNNLS (2021) 1–10.

[29] T. Dierks, S. Jagannathan, Neural network control of mobile robot formations using RISE feedback, IEEE Transactions on SMC, Part B (Cybernetics) 39 (2) (2009) 332–347.