Deep Neural Networks as Add-on Modules for High-Accuracy Impromptu Trajectory Tracking

Supplementary Materials

Introduction

This document provides supporting materials for the submission entitled "Deep Neural Networks as Add-on Modules for High-Accuracy impromptu Trajectory Tracking". In this work, we aim to provide theoretical insights on a DNN-based control architecture (Fig. 1a of main document) for achieving high-accuracy tracking for arbitrary, feasible desired trajectories.

For ease of discussions below, the assumed forms of the baseline system state-space representations from the main document are replicated below. For a linear time invariant (LTI), single-input-single-output (SISO) system, the following state-space representation is assumed:

$$\begin{aligned} x(t+1) &= Ax(t) + bu(t) \\ y(t) &= cx(t) \end{aligned} \tag{1}$$

where $t \in \mathbb{Z}_{\geq 0}$ is the discrete-time index, $x \in \mathbb{R}^n$ is the system state, $y \in \mathbb{R}$ is the output, $u \in \mathbb{R}$ is the reference signal, A, b, and c are constant matrices. For SISO nonlinear systems, the following representation is used:

$$\begin{aligned} x(t+1) &= f\left(x(t)\right) + g\left(x(t)\right) u(t) \\ y(t) &= h\left(x(t)\right) \end{aligned} \tag{2}$$

where $f(\cdot)$, $g(\cdot)$, and $h(\cdot)$ are smooth functions.

The materials below are organized based on the order of reference in the main document.

1 Input Selection for SISO Nonlinear System

For the nonlinear system (2), as derived in [1], by applying the definition of the relative degree[1], at a particular timestep t, $y(t+r)$ can be related to $u(t)$ by

$$y(t+r) = h \circ f^{r-1}(f(x(t)) + g(x(t))u(t)). \tag{3}$$

By assuming $y(t+r)$ is affine in $u(t)$, the above expression can be simplified to

$$y(t+r) = \hat{h}(x(t)) + D(x(t))u(t), \tag{4}$$

where $\hat{h}(x(t)) = h \circ f^r(x(t))$ and $D(x(t)) = \frac{\partial}{\partial u} h \circ f^{r-1}(f(x(t)) + g(x(t))u(t))$ [1, 2]. For this simplified scenario, by choosing

$$u(t) = \frac{1}{D(x(t))} \left(-\hat{h}(x(t)) + y_d(t+r) \right), \tag{5}$$

[1] As defined in [1] and stated in the main document, for the nonlinear system (2), the relative degree r of the system is the smallest integer such that $\frac{\partial}{\partial u} h \circ f^{r-1}(f(x(t)) + g(x(t))u(t)) \neq 0$ for each x in the neighborhood of the operating point x_0. In this definition, $h \circ f$ denotes the composite function of h and f and f^{r-1} denotes the $(r-1)^{\text{th}}$ composition of f with $f^i(x(t)) = f^{i-1} \circ f(x(t))$ and $f^0(x(t)) = x(t)$.

Submitted to the 1st Conference on Robot Learning (CoRL 2017). Do not distribute.

exact tracking is again achieved. For the general nonlinear scenario represented by Eqn. (3), it can be inferred that the control law for achieving exact tracking is a nonlinear function of $x(t)$ and $y_d(t+r)$, i.e.,

$$u(t) = F(x(t), y_d(t+r)). \tag{6}$$

Thus, similar to the linear case, in the proposed architecture, by training the DNN module to model the control laws represented by Eqn. (5) or (6), exact tracking condition can be achieved. These underlying functions to be approximated by the DNN module are again associated with the inverse of the original nonlinear system dynamics.

2 Alternative Input Selection for SISO Linear System

For linear systems, system (1) can be equivalently represented by

$$\frac{Y(z)}{U(z)} = c(zI - A)^{-1}b = \frac{\beta_{n-r}z^{n-r} + \beta_{n-r-1}z^{n-r-1} + \cdots + \beta_0}{z^n + \alpha_{n-1}z^{n-1} + \cdots + \alpha_0}, \tag{7}$$

where $U(z)$ and $Y(z)$ are the z-transforms of the input and output of system (1), and α_i and β_i are scalar constants. For achieving exact tracking, the control law to be approximated by the DNN module is

$$\begin{aligned} u(t) = \frac{1}{\beta_{n-r}}y_d(t+r) + \frac{\alpha_{n-1}}{\beta_{n-r}}y_d(t+r-1) + \cdots + \frac{\alpha_0}{\beta_{n-r}}y_d(t-n+r) \\ - \frac{\beta_{n-r-1}}{\beta_{n-r}}u(t-1) - \frac{\beta_{n-r-2}}{\beta_{n-r}}u(t-2) - \cdots - \frac{\beta_0}{\beta_{n-r}}u(t-n+r), \end{aligned} \tag{8}$$

and the necessary input features to the DNN module are thus $\{y_d(t+r), y_d(t+r-1), \cdots, y_d(t-n+r), u(t-1), u(t-2), \ldots, u(t-n+r)\}$.

3 Necessary Condition for Applying Difference Learning Scheme

The proofs for Insight 4 in the main document are included below. The proof for the transfer function formulation is provided in Section 3.1 and that for the state-space formulation is given in Section 3.2.

3.1 Proof for Transfer Function Formulation

For ease of illustrating Insight 4 in the main document, the transfer function representation of the SISO, LTI system is first considered. It can be shown that the representation in Eqn. (8) can be equivalently written as

$$\begin{aligned} \Delta_u(t) = \frac{1}{\beta_{n-r}}\Delta_{y_d}(t+r) + \frac{\alpha_{n-1}}{\beta_{n-r}}\Delta_{y_d}(t+r-1) + \cdots + \frac{\alpha_0}{\beta_{n-r}}\Delta_{y_d}(t-n+r) \\ - \frac{\beta_{n-r-1}}{\beta_{n-r}}\Delta_u(t-1) - \frac{\beta_{n-r-2}}{\beta_{n-r}}\Delta_u(t-2) - \cdots - \frac{\beta_0}{\beta_{n-r}}\Delta_u(t-n+r) \\ \underbrace{+ \frac{1}{\beta_{n-r}}\left(1 - \sum_{i=0}^{n-r}\beta_i + \sum_{i=0}^{n-1}\alpha_i\right)y_d(t)}_{\triangleq s(y_d(t))}, \end{aligned} \tag{9}$$

where $\Delta_u(t+k) := u(t+k) - y_d(t)$ and $\Delta_{y_d}(t+k) := y_d(t+k) - y_d(k)$ for $k \in \mathbb{Z}$. In this setting, the possibility of using the difference learning scheme (i.e., the possibility of expressing $\Delta_u(t)$ as a function only of the relative time-varying terms) requires $s(y_d(t)) = 0$, or $\sum_{i=0}^{n-r}\beta_i = 1 + \sum_{i=0}^{n-1}\alpha_i$. From Eqn. (7), it can be readily seen that this condition for applicability of the difference learning scheme is equivalent to the condition of achieving zero steady state inputs for step responses. In specific, for a unity step input $U(z) = \frac{z}{z-1}$, the steady state value of the output y_{ss} is

$$y_{\text{ss}} = \underbrace{\lim_{z\to 1}\left[(z-1)\frac{\beta_{n-r}z^{n-r} + \beta_{n-r-1}z^{n-r-1} + \cdots + \beta_0}{z^n + \alpha_{n-1}z^{n-1} + \cdots + \alpha_0}\frac{z}{z-1}\right]}_{\text{Final Value Theorem}} = \frac{\sum_{i=0}^{n-r}\beta_i}{1 + \sum_{i=0}^{n-1}\alpha_i}, \tag{10}$$

which by imposing zero steady state error requirement (i.e., $y_{\text{ss}} = 1$), the same condition is obtained:

$$\sum_{i=0}^{n-r} \beta_i = 1 + \sum_{i=0}^{n-1} \alpha_i. \tag{11}$$

For the state-space formulation, for the special case of position-velocity-like systems, a contradiction statement can be used to show that if zero steady state error (i.e., $y_d(t) - y_{ss} = 0$) is not achieved for step inputs, then the underlying function to be modelled by the DNN model is one-to-many, which at the best, an averaged solution is learned by the DNN module [3]. Details of the proof are included in the following subsection.

3.2 Proof for the State-Space Formulation

This subsection provides the proof on the necessity of achieving zero steady state error for applying the difference learning scheme in a special case of the state-space formulations. In particular, a position-velocity-like system, where $y(t)$ is the first element of $x(t)$ and $x(t) = [y(t) \quad 0 \cdots \quad 0]^\intercal$ at the steady state for step inputs, is considered. With a proof by contradiction, we will show that if the baseline system does not achieve zero steady state error for step inputs, constant (i.e., translational) errors cannot be corrected for by the proposed DNN-enhanced controller.

For proof by contradiction, first consider a system such that for arbitrary step inputs $u(t) = a$, zero steady state errors are not achieved (i.e., $y_{ss} = \kappa a$ with $\kappa \neq 1$). By the way of contradiction, assume that a DNN module trained with the difference learning scheme can lead to exact tracking $y_{ss} = y_d(t)$ for arbitrary step inputs. For step inputs with desired amplitude $y_d(t) = b$, with the difference learning scheme and at the steady state, the inputs to the DNN module are $\Delta_y = y_d(t+r) - y_d(t) = 0$ and $\Delta_x := [\Delta_y \quad 0 \cdots \quad 0]^\intercal = \mathbf{0}$, and the output is $\Delta_u = c$, where c is the bias of the DNN model (i.e., the output of the DNN when the input is $\mathbf{0}$). On the other hand, at the steady state, the input to the feedback system is

$$u(t) = c + y_d(t) = c + b \tag{12}$$

and the output is

$$y_{ss} = \kappa(c + b). \tag{13}$$

By the exact tracking assumption in the contradiction argument, $y_{ss} = b$ and thus

$$\kappa(c + b) = b. \tag{14}$$

By rearranging Eqn. (14), the bias of the DNN module is then

$$c = b\left(\frac{1}{\kappa} - 1\right). \tag{15}$$

For the considered system, since zero steady state error is not achieved for step inputs, $\left(\frac{1}{\kappa} - 1\right) \neq 0$, and the constant bias of the DNN module is dependent on the amplitude of the desired step input b. As the constant bias of the trained DNN can have only a single value, this leads to a contradiction. Therefore, a DNN module trained with the difference learning scheme cannot achieve exact tracking for a system where zero steady state error is not achieved for step inputs.

From another perspective, in these cases where zero steady error is not achieved for step inputs, the mapping to be learned by the DNN module is one-to-many. As discussed in [3], for one-to-mappings, the DNN trained with the current approach, at best, learns an average of the corresponding outputs contained in the training dataset; for non-convex problems, the average learned by the DNN may not in fact be a valid solution. Therefore, based on this theoretical discussion, it is expected that the DNN trained with the difference learning scheme cannot effectively compensate for biases of the original system response if the original system has non-zero steady state errors for step inputs.

References

[1] M. Sun and D. Wang. Analysis of nonlinear discrete-time systems with higher-order iterative learning control. *Dynamics and Control*, 11(1):81–96, 2001.

[2] T.-J. Jang, H.-S. Ahn, and C.-H. Choi. Iterative learning control for discrete-time nonlinear systems. *Intl. Journal of Systems Science*, 25(7):1179–1189, 1994.

[3] M. I. Jordan and D. E. Rumelhart. Forward models: Supervised learning with a distal teacher. *Cognitive Science*, 16(3):307–354, 1992.