Design of Deep Neural Networks as Add-on Blocks for Improving Impromptu Trajectory Tracking

Siqi Zhou, Mohamed K. Helwa, and Angela P. Schoellig

Abstract— This paper introduces deep neural networks (DNNs) as add-on blocks to baseline feedback control systems to enhance tracking performance of arbitrary desired trajectories. The DNNs are trained to adapt the reference signals to the feedback control loop. The goal is to achieve a unity map between the desired and the actual outputs. In previous work, the efficacy of this approach was demonstrated on quadrotors; on 30 unseen test trajectories, the proposed DNN approach achieved an average impromptu tracking error reduction of 43% as compared to the baseline feedback controller. Motivated by these results, this work aims to provide platform-independent design guidelines for the proposed DNN-enhanced control architecture. In particular, we provide specific guidelines for the DNN feature selection, derive conditions for when the proposed approach is effective, and show in which cases the training efficiency can be further increased.

I. INTRODUCTION

High-accuracy trajectory tracking is an essential requirement for many industrial applications including robot-aided inspection and advanced manufacturing [1], [2]. Classical control methods for trajectory tracking, such as model predictive control (MPC) or PID control, either require a sufficiently accurate dynamic model of the plant [3], which may not be available, or rely on manual tuning of the system controller parameters, which can be difficult and time-consuming, and may result in overly conservative behavior [4]. Learning methods such as iterative learning control (ILC) have been successfully applied to many robotic applications to improve the tracking performance through repeated trials [5], [6]. However, this requires repeated training of the known, desired trajectory. In this paper, we consider a more challenging problem: impromptu tracking; that is, to accurately track an arbitrary reference in a single attempt.

Given the ability of deep neural networks (DNNs) to generalize knowledge, a DNN-enhanced control architecture has been proposed in [7] to improve the tracking performance of traditional feedback controllers for any given desired trajectory. In this proposed architecture (illustrated in Fig. 1), a DNN module is pre-cascaded to a baseline feedback control system to adjust the reference inputs to the feedback control system with the goal of achieving perfect output tracking. On 30 test trajectories, the DNN-based approach led to an average of 43% tracking error
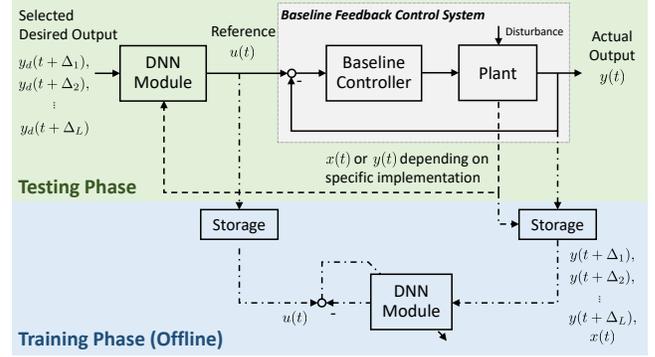
Fig. 1. An illustration of the proposed control architecture and training process considered in this work. During the testing phase, at a particular time step t, the selected desired output $\{y_d(t+\Delta_1), y_d(t+\Delta_2),..., y_d(t+\Delta_L)\}$ and the current state $x(t)$ (or the current output $y(t)$) are inputs to the DNN module to generate reference signals $u(t)$ for the baseline system, where $t \in \mathbb{Z}_{\geq 0}$ is the discrete-time index and $\Delta_i \in \mathbb{Z}_{\geq 0}$ for all i. During the training phase, a set of training trajectories are performed on the baseline system, and $u(t)$, $y(t)$, and $x(t)$ are stored for training the DNNs offline.

reduction as compared to the baseline controller [7]. Though the effectiveness of the architecture was well-illustrated on quadrotor vehicles, general DNN design guidelines were not provided in [7]. In particular, the inputs and outputs of the DNN module were determined through experimental trial-and-error. Similar to the fundamental work on feature selection for image classification and speech recognition, in this work we derive rules for feature selection in the context of tracking control. In addition, we derive conditions under which the proposed approach is effective and identify cases for which the training efficiency can be further improved. These contributions together provide a platform-independent formulation of the approach utilized in [7].

Below we first provide a brief review of the relevant literature (Section II). We then define our problem (Section III), derive theoretical insights on the DNN module design (Section IV), and present simulation and experimental results (Sections V and VI).

II. RELATED WORK

In the literature, there are many examples where machine learning techniques have been successfully integrated with control system designs to improve the tracking performance in uncertain environments [5], [8]–[10]. As discussed in [7], examples of these machine learning techniques include, but are not limited to, Gaussian Processes (GPs), ILC, and DNNs. In comparison, DNNs are flexible modeling frameworks capable of approximating highly nonlinear functions

and generalizing learned experience to unseen situations [11]. As compared to GPs specifically, DNNs have the advantage that their computation time and memory requirement do not increase with the size of the training dataset [7]; as compared to ILC, DNN-based approaches can be more conveniently generalized to untrained tasks [9].

As summarized in early review papers [11], [12], DNNs have been combined with various control techniques such as *predictive control* and *adaptive control* to directly or indirectly account for uncertainties and non-idealities in the overall system. In the recent literature, there are many advanced works illustrating the efficacy of utilizing DNNs in different control applications. For instance, in [13], it is illustrated that DNNs can approximate the unmodeled dynamics of quadrotors to assist linear quadratic regulators (LQR) in trajectory tracking control. In [14] and the references therein, DNNs are used to compensate for the model uncertainties in impedance control of manipulators. Moreover, in [10], for single and double pendulums, DNN models are utilized in nonlinear model predictive control (NMPC) for determining the optimal control input. In [15], simulations on several aerospace systems illustrate that DNNs can also be used to approximate solutions of the Hamilton-Jacobi-Bellman (HJB) equations to save the online computation time.

Adding to this body of work, [7] proposes pre-cascading a DNN module to a feedback control system of a quadrotor to achieve a unity mapping from the desired output to the actual output. By aiming for a unity map, this methodology is similar to *direct inverse control* and *adaptive inverse control* [11], [16]–[19]. However, there are several major differences. One of the major differences is that the DNN models in [7] are not directly applied to the open-loop plant. Instead, they modify the reference of a stable closed-loop system and can run at a lower rate, as compared to the underlying baseline controller, which makes the approach less prone to stability issues. Moreover, in adaptive inverse control, the weights of the DNN models must be updated online to ensure the stability of the overall system [17], and the convergence of the weights requires good initializations [19]. In contrast, with the proposed architecture in [7], improving tracking performance is decoupled from achieving stability. Stability is guaranteed by appropriately designing the controller of the baseline closed-loop system. Improvements to the tracking performance are achieved by the DNNs. Furthermore, as opposed to [18], where recurrent neural networks (RNNs) are used, in the proposed architecture, feedforward neural networks (FNNs) are used for model learning. This makes the proposed architecture less prone to instability issues and simplifies practical implementations [20]. Given these advantages and further motivated by the successful results in [7], in this work, we aim to further analyze the DNN-enhanced control architecture of [7], and identify conditions where the proposed approach is most effective and efficient.

III. PROBLEM STATEMENT

We consider the control architecture shown in Fig. 1. A DNN is introduced as an add-on module to a closed-loop, stable system in order to improve the tracking control performance. We aim to derive general guidelines for the design of the DNN module. This includes:

1) identification of conditions under which the add-on DNN module improves the system's tracking performance,
2) rules for the DNN feature selection, and
3) characterization of conditions under which the efficiency of the training can be further improved.

In order to address the aspects outlined above, we first assume that the underlying closed-loop system can be described by a linear time-invariant (LTI), single-input-single-output (SISO) system. This discussion is then extended to nonlinear systems. In particular, we assume the discrete-time dynamics of the baseline system can be represented by

$$x(t+1) = Ax(t) + bu(t), \ \ y(t) = cx(t) \quad (1)$$

in the linear scenario, and by

$$x(t+1) = f\left(x(t)\right) + g\left(x(t)\right)u(t), \ \ y(t) = h\left(x(t)\right) \quad (2)$$

in the nonlinear scenario, where $t \in \mathbb{Z}_{\geq 0}$ is the discrete-time index, $x \in \mathbb{R}^n$ is the state, $y \in \mathbb{R}$ is the output, $u \in \mathbb{R}$ is the reference, A, b, and c are constant matrices of appropriate dimensions, and $f(\cdot)$, $g(\cdot)$, and $h(\cdot)$ are smooth functions.

IV. MAIN RESULTS

In this section, given the system representations (1) and (2), we first identify the function that the DNN would need to represent. We then use this result to derive *(i)* necessary conditions for the add-on DNN module to be effective (Section IV-A) and *(ii)* the DNN input features necessary to model the underlying function (Section IV-B). Moreover, we derive necessary conditions enabling further improvements of the efficiency of the DNN training (Section IV-C). Even though these discussions start from known dynamics of the baseline feedback loop, in practice only minimal knowledge of the closed-loop system is needed. As will be discussed in detail, this knowledge can be either obtained from simple identification experiments such as step response tests or from basic knowledge about the system.

A. Underlying Function Modeled by the DNNs

The DNN add-on module aims to establish an identity mapping from the desired output y_d to the actual output y. In this part, we will show that the function approximated by the DNN module is the output equation of the inverse dynamics of the feedback control loop. Stable zero dynamics of the baseline system is consequently a necessary condition for the proposed approach to be effective.

For the convenience of discussion, we first state the definition of the relative degree of a dynamical system. For the linear SISO system (1), the *relative degree* is the smallest integer r for which $cA^{r-1}b \neq 0$ [21]. Using this definition, it can be shown that the input and output of (1) are related by

$$y(t+r) = cA^r x(t) + cA^{r-1}bu(t). \quad (3)$$

Then, at time step t, by selecting

$$u(t) = \frac{1}{cA^{r-1}b}\left(-cA^r x(t) + y_d(t+r)\right), \qquad (4)$$

$y(t+r) = y_d(t+r)$ is satisfied as desired. By considering $y_d(t+r)$ as the input and $u(t)$ as the output, Eqn. (4) is in fact the output equation of the inverse dynamics of the baseline system (1). Thus, recalling the architecture in Fig. 1, by training the DNN module to approximate Eqn. (4), exact tracking can be achieved in theory.

Note that there is an inherent delay of r time steps from the input to the output in Eqn. (3), which is a known fact for discrete-time systems with relative degrees r. From Eqn. (4), at a particular time step t, the computation of $u(t)$ depends on $y_d(t+r)$ to compensate for the inherent delay. In practice, for off-line or on-line trajectory generation algorithms, a preview of r steps of the desired trajectory y_d is typically available; hence, the non-causality in Eqn. (4) is *not* an issue in practice.

The above analysis can be extended to nonlinear systems. Following the discussion outlined in [22], we use $h \circ f$ to denote the composition function of f and h, and f^i to denote the ith composition of the function f with $f^0(x(t)) = x(t)$ and $f^i(x(t)) = f^{i-1} \circ (f(x(t)))$. Around an operating point (x_0, u_0), the *relative degree* of system (2) is defined as the smallest integer r such that $\frac{\partial}{\partial u} h \circ f^{r-1}(f(x(t)) + g(x(t))u(t)) \neq 0$ for each point in the neighborhood of the operating point. We assume that system (2) has a well-defined relative degree r in the operating region. Then, it can be shown that the input and the output of system (2) are related by

$$y(t+r) = h \circ f^{r-1}\big(f(x(t)) + g(x(t))u(t)\big). \qquad (5)$$

Assuming $y(t+r)$ is affine in $u(t)$, Eqn. (5) can be simplified to

$$y(t+r) = \hat{h}(x(t)) + D(x(t))u(t), \qquad (6)$$

where $\hat{h}(x(t)) = h \circ f^r(x(t))$ and $D(x(t)) = \frac{\partial}{\partial u} h \circ f^{r-1}(f(x(t))+g(x(t))u(t))$ [22], [23]. Note that in Eqn. (6), $D(x(t)) \neq 0$ by the definition of the relative degree. Following the same argument as for linear systems, the control law for achieving $y(t+r) = y_d(t+r)$ is

$$u(t) = \frac{1}{D(x(t))}\left(-\hat{h}(x(t)) + y_d(t+r)\right) \qquad (7)$$

for the affine case in Eqn. (6), and it is reasonable to assume that

$$u(t) = F\left(x(t), y_d(t+r)\right) \qquad (8)$$

for the general case in Eqn. (5), where F is a nonlinear function to be approximated by the DNN module.

From Eqn. (7) and Eqn. (8), the required information for computing $u(t)$ remains the same as that for linear systems. In particular, the function depends on the current state $x(t)$ and the desired output r steps in the future $y_d(t+r)$, where r is the relative degree of the baseline system.

Insight 1a. Underlying Function: In order to achieve an identity mapping from the desired output to the actual output, the DNN module should approximate the output equation of the inverse dynamics of the baseline system, which corresponds to Eqn. (4) for the linear scenario and Eqn. (8) for the nonlinear scenario.

Insight 1b. Necessary Conditions for the Effectiveness of the Approach: The effectiveness of our proposed DNN architecture depends on two conditions: *(i)* the baseline system has a well-defined relative degree r (within the defined operating region) and *(ii)* the inverse dynamics of the system are stable.

As a result, for implementing the proposed DNN-enhanced architecture, we only need to identify the relative degree of the feedback system rather than its exact dynamics. For linear and nonlinear discrete-time systems, the relative degree is the number of time steps of delay between a change in the input and the resulting change in the output, which can be easily determined from the system's step response in practice. In [7], it is proposed to run the DNN module at a lower sampling frequency as compared to the baseline system to avoid instability problems; in this case, relative degrees should be determined with respect to the DNN sampling frequency. For nonlinear systems, since relative degrees are locally defined [21], having multiple DNN modules for different operating regions may be required.

For linear systems, the stability of the inverse dynamics is equivalent to the stability of the system's zero dynamics, which is characterized by the zeros of the system's transfer function; this can be also checked experimentally using the system's step response [21]. For nonlinear systems, a necessary condition for the stability of the inverse dynamics is the stability of the system's zero dynamics, which are the invariant dynamics of (2) when the input $u(t)$ is selected such that the output $y(t)$ is forced to be 0 at all time steps. Note that for nonlinear systems, this condition is, however, not sufficient [24], [25]. *Therefore, for both systems (1) and (2), a necessary condition for the stability of the inverse dynamics, and hence for the effectiveness of the DNN-based approach, is the stability of the zero dynamics of the baseline system.*

B. Feature Selection

In this subsection, we discuss the correct selection of input features for the DNN module to achieve high tracking performance. Based on the state space representation in the previous section, and from Eqn. (4) and Eqn. (8), the information for the DNN module to correctly generate the reference $u(t)$ consists of the current state $x(t)$ and the future desired output $y_d(t+r)$, where r is the system's relative degree. For linear systems, an alternative feature selection can be obtained using the transfer function of the system (1). In particular, assuming zero initial conditions, the equivalent z-transform of system (1) is

$$\frac{Y(z)}{U(z)} = \frac{\beta_{n-r}z^{n-r} + \beta_{n-r-1}z^{n-r-1} + \cdots + \beta_0}{z^n + \alpha_{n-1}z^{n-1} + \cdots + \alpha_0}, \qquad (9)$$

where $Y(z)$ and $U(z)$ are the z-transforms of the output and reference of the system, n is the system's order and r is its relative degree, and α_i and β_i are constants. Assuming both

the dynamics and the zero dynamics of (9) are stable, it can be easily verified that by choosing the following control law

$$u(t) = \frac{1}{\beta_{n-r}} y_d(t+r) + \frac{\alpha_{n-1}}{\beta_{n-r}} y_d(t+r-1) + \cdots$$
$$+ \frac{\alpha_0}{\beta_{n-r}} y_d(t-n+r) - \frac{\beta_{n-r-1}}{\beta_{n-r}} u(t-1)$$
$$- \frac{\beta_{n-r-2}}{\beta_{n-r}} u(t-2) - \cdots - \frac{\beta_0}{\beta_{n-r}} u(t-n+r),$$
(10)

exact tracking is achieved. Based on Eqn. (10), the input features of the DNN can be selected as $\{y_d(t+r), y_d(t+r-1),..., y_d(t-n+r), u(t-1), u(t-2),..., u(t-n+r)\}$.

Insight 2. Feature Selection: By associating the DNN module with the inverse dynamics of the baseline feedback control loop, the state space and transfer function representations provide two approaches for selecting the input features of the DNN module to achieve exact tracking. In particular, for the state space representations of linear and nonlinear systems, the input features to the DNN module are $x(t)$ and $y_d(t+r)$. For the transfer function representation of linear systems, the input features are $\{y_d(t+r), y_d(t+r-1),..., y_d(t-n+r), u(t-1), u(t-2),..., u(t-n+r)\}$.

The transfer function approach does not require the knowledge of the internal state $x(t)$ of the system, but is limited to linear systems. In practice, this approach may be applied to cases where the baseline system can be approximated by linear dynamics (e.g., [26], [27]). For general nonlinear control problems, one may utilize the state space approach to implement the DNN module together with standard state estimation techniques. The inclusion of the state in the DNN inputs also allows this approach to partially compensate for initial errors or disturbances in the system. Furthermore, the dimension of the input to the DNN is $(n+1)$ for the state space approach and $(2n-r+1)$ for the transfer function approach. For high-dimensional systems with low relative degrees, the state space approach may be preferable because of the relatively smaller DNN input dimension.

C. Improvement of Training Efficiency

In the implementation of [7], the position components of the input and output of the DNN module were taken relative to the current position (referred to as the *difference learning scheme*) to simplify the training process. By using relative terms, the function learned by the DNN module is translationally invariant with respect to position. Consequently, the amount of data needed for the training is reduced, and the training process becomes more efficient. In this subsection, we prove that there is an implicit assumption for the difference learning scheme to be valid. In particular, we consider the linear system (1) using both transfer function and state space formulations. For the state space formulation, we assume that the output $y = x_1$, the first element of the state vector x, and that for step inputs, the steady state values of the remaining states $x_2, \cdots, x_n$ are all zeros. This assumption is valid, for instance, for mechanical systems with position-velocity state space.

Lemma 1. A Necessary Condition for Difference Learning: Consider system (1) and the DNN-enhanced architecture in Fig. 1. Then, the DNN-enhanced approach with a difference learning scheme is able to achieve exact tracking only if the baseline system has a unity DC gain.

Proof. We first consider the transfer function formulation in Eqn. (10), which corresponds to the exact inverse of system (1). By defining $\Delta_u(t+k) := u(t+k) - y_d(t)$ and $\Delta_{y_d}(t+k) := y_d(t+k) - y_d(t)$ for $k \in \mathbb{Z}$, it can be shown that Eqn. (10) can be rewritten as

$$\Delta_u(t) = \frac{1}{\beta_{n-r}} \Delta_{y_d}(t+r) + \frac{\alpha_{n-1}}{\beta_{n-r}} \Delta_{y_d}(t+r-1) + \cdots$$
$$+ \frac{\alpha_0}{\beta_{n-r}} \Delta_{y_d}(t-n+r) - \frac{\beta_{n-r-1}}{\beta_{n-r}} \Delta_u(t-1)$$
$$- \frac{\beta_{n-r-2}}{\beta_{n-r}} \Delta_u(t-2) - \cdots - \frac{\beta_0}{\beta_{n-r}} \Delta_u(t-n+r)$$
$$+ \underbrace{\frac{1}{\beta_{n-r}} \left(1 - \sum_{i=0}^{n-r} \beta_i + \sum_{i=0}^{n-1} \alpha_i \right) y_d(t)}_{\triangleq s(y_d(t))}.$$
(11)

On the right-hand side of Eqn. (11), the only term containing the non-difference, time-dependent variable $y_d(t)$ is $s(y_d(t))$. It is possible to express $\Delta_u(t)$ as a function of the remaining Δ_{y_d} and Δ_u terms and hence utilize the difference learning scheme if and only if $s(y_d(t)) = 0$. For arbitrary $y_d(t)$, this condition implies $\sum_{i=0}^{n-r} \beta_i = 1 + \sum_{i=0}^{n-1} \alpha_i$. By examining the transfer function in Eqn. (9), this is equivalent to the condition of having a baseline system with a unity DC gain.

The same condition can be derived for the state space formulation in (4). For this case, in order to introduce translational invariance with respect to y_d, the inputs to the DNN module are defined as $\Delta_{y_d}(t+r) := y_d(t+r) - y_d(t)$ and $\Delta_x(t) := x(t) - \begin{bmatrix} y_d(t) & 0 \cdots & 0 \end{bmatrix}^\mathsf{T}$, and the output is $\Delta_u(t) = u(t) - y_d(t)$. For proving Lemma 1 in this case, we show that if the DC gain of the feedback system (1) is not unity, then the DNN trained based on the difference learning scheme cannot correct constant errors (offsets) and the overall DNN-enhanced system will not be able to achieve zero steady state errors for constant references. Consider a baseline feedback system with a DC gain $K_0 \neq 1$. Suppose, by contradiction, that we have zero steady state error for the desired step reference $y_d(t) = K$, for all $t \geq 0$ and arbitrary constants K. For this case, for all $t \geq 0$, $y_d(t+r) = y_d(t) = K$ and $\Delta_{y_d} = 0$ uniformly. Also, since the steady state error is zero by assumption, $\Delta_x = 0$ at the steady state. Thus, at the steady state, the inputs to the DNN are all zeros, and $\Delta_u = \bar{b}$, where $\bar{b}$ is the network constant bias, which is preselected in the training phase. This implies that the input to the baseline feedback system at the steady state is $K + \bar{b}$, and consequently, the steady state output of the system is $y_{ss} = K_0(K + \bar{b})$. Since $y_{ss} = K$ by assumption, we have $\bar{b} = (K/K_0) - K$. Since $K_0 \neq 1$ by assumption, $\bar{b}$ is dependent on the arbitrary value K, a contradiction to the fact that $\bar{b}$ is a constant value. $\square$

From the proof above, the correct output of the DNN module at steady state to achieve zero steady state error for a step reference K is $\Delta_u = (K/K_0) - K$. Since K is arbitrary, then, except for the case that $K_0 = 1$, the mapping from $\Delta_x(t) = 0$ and $\Delta_{y_d}(t + r) = 0$ to $\Delta_u(t)$ is one-to-many, which cannot be effectively learned by DNNs [28].

In the above discussion, we used $y_d(t)$ as the reference point in the difference learning scheme. In practice, during the testing phase, $y_d(t)$ on either or both input and output sides of the DNN module can be replaced with $y(t)$ to additionally compensate for initial errors and disturbances.

Insight 3. A Necessary Condition for Difference Learning: Based on the above theoretical study for linear systems, for the difference learning scheme utilized in [7] to be valid, the baseline system must have a unity DC gain. If one plans to use the difference learning scheme to improve the DNN training efficiency, then the baseline system should be designed to achieve zero/small steady state errors for step references.

In Section VI, we verify the necessity of this condition for nonlinear systems through experiments.

V. SIMULATION RESULTS

In order to illustrate *Insight 1*, we consider two LTI, SISO systems, representing the baseline, closed-loop dynamics in Fig. 1:

$$x(t+1) = \begin{bmatrix} 0 & 1 \\ -0.15 & 0.8 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t)$$
$$y(t) = \begin{bmatrix} -0.2 & 1 \end{bmatrix} x(t), \tag{12}$$

$$x(t+1) = \begin{bmatrix} 0 & 1 \\ -0.15 & 0.8 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t)$$
$$y(t) = \begin{bmatrix} -450.9 & 450 \end{bmatrix} x(t). \tag{13}$$

The two systems have the same poles but different zeros. System (12) has a stable (minimum phase) zero at 0.2, and (13) has an unstable (non-minimum phase) zero at 1.002. For the purpose of illustrating *Insight 1*, the state space approach is used for selecting the features of the DNN module. At a particular time step t, the input to the DNN module is $\mathcal{I} = \{x(t), y_d(t+r)\}$, and the output is $\mathcal{O} = \{u(t)\}$. The equation to be approximated by the DNN module is Eqn. (4). For this simulation study, with known system matrices (A, b, c), Eqn. (4) can be computed at each time step and utilized to examine *Insight 1*.

A. DNN Architecture and Training

For this simulation example, Matlab's Neural Network Toolbox is used for constructing and training the DNN models. For both systems, the DNN models are fully-connected FNNs with 2 hidden layers of 20 hyperbolic tangent neurons. The training data is generated from the baseline system response to sinusoidal trajectories with 25 different combinations of amplitudes $\{1, 2, 3, 4, 5\}$ and frequencies $\{0.024, 0.032, 0.048, 0.091, 1.000\}$ Hz. To avoid overfitting a particular training trajectory, the training dataset is constructed using a balanced number of randomly chosen
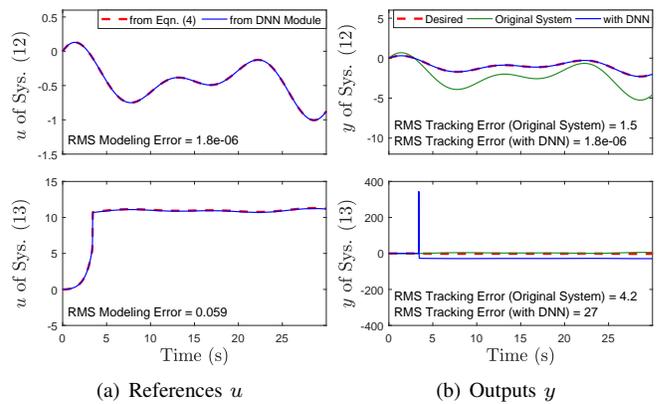

(a) References u (b) Outputs y

Fig. 2. The references and outputs of the closed-loop systems (12) and (13) for the test trajectory $y_d(t) = \sin\left(\frac{2\pi}{15}t\right) + \cos\left(\frac{2\pi}{12}t\right) - 1$. From (a), the DNN accurately learns the output equation of the inverse dynamics of the baseline systems. From (b), for system (12), the DNN-enhanced approach is able to approximately achieve exact tracking; however, for system (13), which has an unstable zero, the inherent instability causes numerical issues and prevents the DNN-enhanced approach from being effective.

data points from each trajectory. The Levenberg-Marquardt algorithm is used for DNN model parameter training to minimize the mean squared error between the DNN's output and the target from the training dataset. For both systems, the DNN training converges within 1000 iterations. We test the overall DNN-enhanced systems on a set of untrained trajectories.

B. Testing Results

In order to illustrate *Insight 1*, the baseline and DNN-enhanced performance of system (12) and system (13) are tested for the trajectory $y_d(t) = \sin\left(\frac{2\pi}{15}t\right) + \cos\left(\frac{2\pi}{12}t\right) - 1$. From Fig. 2(a), it can be seen that for both systems, the reference signal outputted by the DNN module (blue) and that computed based on Eqn. (4) (red) almost coincide. This verifies our theoretical insight that the underlying function learned by the DNN module is the output equation of the inverse dynamics of the baseline feedback control system. This also shows that a well-trained DNN module can accurately approximate the output equation of the inverse dynamics of the baseline feedback control system. Along this particular test trajectory, the root-mean-square (RMS) modeling errors of the DNN for systems (12) and (13) are approximately 2×10^{-6} and 0.06, respectively. In spite of having good modeling accuracy, one can see from Fig. 2(b) that for the non-minimum phase system (13), when the DNN module is pre-cascaded to the system, the output response of the system (blue) suffers from numerical issues. Indeed, even when Eqn. (4) is used to calculate the reference and propagate the system forward in time, due to the inherent instability, the precision of the computation quickly falls and leads to unbounded output. Thus, as reflected in this example, even with an unstable zero very close to the unit circle, the associated numerical issues prevent the DNN module from effectively improving the performance of the closed-loop system. In contrast, for the minimum phase system (12), the DNN module approximates the output equation of the inverse

dynamics, and the corrected reference signal generated by the DNN module approximately achieves an identity mapping from the desired output (red) to the actual output (blue). The RMS tracking error for this DNN-enhanced system is of the order of 10^{-6} m, which is only limited by the modeling accuracies and numerical precision in this simulation setting.

VI. QUADROTOR EXPERIMENTS

In this section, *Insight 2* and *Insight 3* are illustrated using experiments on a quadrotor vehicle.

A. Experiments Setup

The full state of the quadrotor consists of the translational positions $\mathbf{p} = (x, y, z)$ and velocities $\mathbf{v} = (\dot{x}, \dot{y}, \dot{z})$, the roll-pitch-yaw Euler angles $\boldsymbol{\theta} = (\phi, \theta, \psi)$, as well as the rotational velocities $\boldsymbol{\omega} = (p, q, r)$. The control problem is to design a controller such that the translational position of the center of mass of the quadrotor precisely tracks the desired trajectories $x_d(t)$, $y_d(t)$, $z_d(t)$.

The baseline controller (grey box in Fig. 1) is a standard nonlinear controller consisting of a nonlinear transformation and a PD controller (see [7] for more details). In the DNN-enhanced scenario, a DNN module is used to correct the position and velocity reference signals sent to the base-line controller. For the experiments, the DNNs are fully-connected FNNs with 4 hidden layers of 128 rectified linear units (ReLU). The inputs to the DNN module are selected based on *Insight 2* and compared to the selection in [7]. The outputs of the DNN module are the translational position and velocity references ($\mathbf{p}_r$ and $\mathbf{v}_r$) given to the baseline controller.

B. DNN Feature Selection

In [7], through experimental trial-and-error, the DNN module input selection that led to efficacious performance was found to be $\mathcal{I} = \{\mathbf{p}_d(t + 4) - \mathbf{p_a}(t), \mathbf{p}_d(t + 6) - \mathbf{p_a}(t), \mathbf{v}_a(t), \mathbf{v}_d(t + 4), \mathbf{v}_d(t + 6), \boldsymbol{\theta}_a(t), \boldsymbol{\theta}_d(t + 4), \boldsymbol{\theta}_d(t + 6), \boldsymbol{\omega}_a(t), \boldsymbol{\omega}_d(t + 4), \boldsymbol{\omega}_d(t + 6), \ddot{z}_a(t), \ddot{z}_d(t + 4), \ddot{z}_d(t + 6)\}$, where the subscripts a and d denote the actual and desired values, respectively. In this subsection, we show that by following *Insight 2*, a similar performance as in [7] can be obtained with significantly less DNN inputs. For the fairness of the comparison, the baseline controller, the DNN architecture, and the training process are identical to [7]; the only difference is the selected inputs to the DNN module.

For implementing the state space approach of *Insight 2*, we first examined the step responses of the baseline feedback system, and the relative degrees of the system are determined to be 4, 4, and 2 in the x-, y-, and z-directions, respectively. Moreover, following *Insight 3*, since the step responses of the baseline system have zero steady state errors, we apply the difference learning scheme as in [7]. Based on *Insight 2* with the difference learning scheme and by assuming that exact tracking is approximately achieved using the DNN-enhanced architecture, the DNN module inputs are selected to be $\mathcal{I} = \{x_d(t+4) - x_d(t), y_d(t+4) - y_d(t), z_d(t+2) - z_d(t), \dot{x}_d(t+3) - \dot{x}_d(t), \dot{y}_d(t+3) - \dot{y}_d(t), \dot{z}_d(t+1) - \dot{z}_d(t), \boldsymbol{\theta}_a(t), \boldsymbol{\omega}_a(t)\}$.
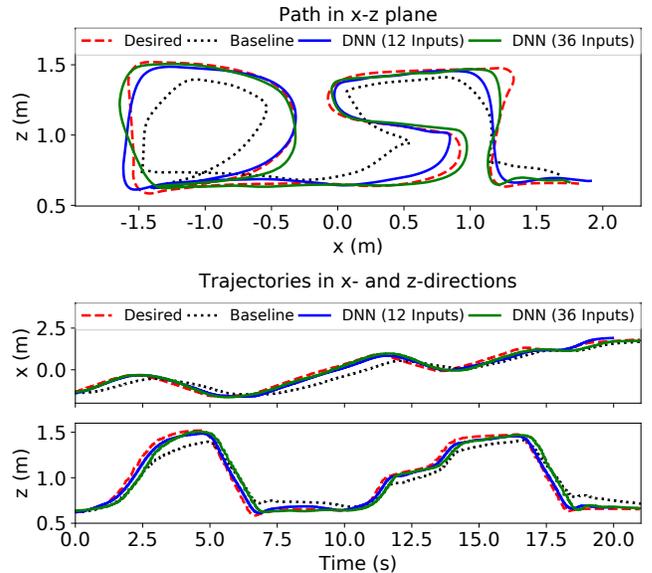

Fig. 3. Comparison between the DNN-enhanced systems based on the feature selection in [7] (DNN with 36 inputs) and the one using *Insight 2* (DNN with 12 inputs). From the path *(top)* and the trajectory *(bottom)* plots, despite having only $1/3$ of the inputs, the DNN module designed based on *Insight 2* leads to similar tracking performance improvements.

TABLE I
PERCENTAGE REDUCTION IN RMS TRACKING ERROR

Traj. ID	'DSL'	2	3	4	5	Avg.
12 Inputs	52.4%	48.4%	42.0%	46.6%	36.9%	45.2%
36 Inputs	55.6%	61.4%	39.7%	43.9%	26.6%	45.5%

In comparison, the DNN module designed based on *Insight 2* with the difference learning has 12 inputs, while that in [7] has 36 inputs. Fig. 3 shows the performance of the two DNN modules on a 2D test trajectory. Similar comparisons are carried out on four additional test trajectories; Table I summarizes the RMS tracking error reduction achieved by the two DNN-enhanced systems. From these results, it can be seen that despite having significantly less input features, the DNN trained with features selected based on *Insight 2* leads to comparable performance to that in [7]. From the trajectory plots in Fig. 3, it can be seen that the delay in the z-direction is reduced with the new feature selection, where relative degrees are properly identified.

C. Difference Learning

In Section VI-B, we showed that when the baseline system satisfies the condition specified in *Insight 3*, the DNN module trained with the difference learning scheme effectively improved the tracking performance. In this subsection, in order to illustrate the necessity of the condition in *Insight 3*, the same input features, DNN architecture, and training process are applied to a modified feedback control system, namely the same baseline system with a factor of 0.5 multiplied to the reference input z_r. For this modified baseline system, the steady state value of the output in the z-direction, denoted z_{ss}, is approximately 0.5 m for a unit step reference. Fig. 4 shows the performance of the DNN-enhanced system
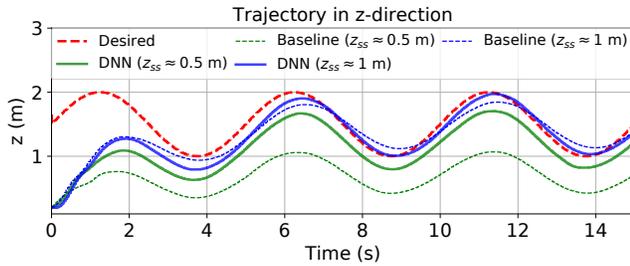
Fig. 4. Comparison of applying the difference learning scheme to the DNN for the two baseline systems – one achieves zero steady state error for step references, which is the necessary condition specified in *Insight 3*, and the other one does not. When this necessary condition is not achieved, the application of the difference learning scheme prevents the DNN module from properly compensating for biases in the response.

for the original and modified baseline system scenarios. For the original scenario (blue), the DNN module trained with the difference learning scheme results in good tracking performance; however, for the modified scenario (green), the DNN module cannot fully compensate for the bias in z. This result is consistent with the discussion in Section IV-C.

VII. CONCLUSIONS AND FUTURE WORK

We provided theoretical insights into a DNN-enhanced control architecture for achieving high-accuracy, impromptu tracking. These insights represent general design guidelines for applying the DNN-enhanced architecture to any practical system. Through theoretical derivations, simulations and experiments, we showed that the DNN module in the proposed architecture is an approximation of the output equation of the inverse dynamics of the baseline system. Due to this association, we illustrated that the DNN-enhanced control architecture (initially proposed in [7]) may not be effective for closed-loop systems with unstable zero dynamics. We also provided guidelines for efficiently selecting the DNN features, which led to a performance similar to previous trial-and-error techniques [7] but had a significantly lower DNN input dimension. Moreover, it is shown through theory and experiments that the applicability of the difference learning scheme in [7] relies on a necessary condition: the baseline system achieves zero steady state errors for step references.

Potential extensions of this work include the exploration of approaches for adapting the proposed DNN-enhanced architecture to non-minimum phase systems, and the incorporation of uncertainty estimations in the DNN-based learning approach.

REFERENCES

[1] J. Nikolic, M. Burri, J. Rehder, S. Leutenegger, C. Huerzeler, and R. Siegwart, "A UAV system for inspection of industrial facilities," in *IEEE Aerospace Conf.*, 2013, pp. 1–8.

[2] Y. X. Su, B. Y. Duan, C. H. Zheng, Y. Zhang, G. Chen, and J. Mi, "Disturbance-rejection high-precision motion control of a stewart platform," *IEEE Trans. on Control Systems Technology*, vol. 12(3), pp. 364–374, 2004.

[3] J. B. Rawlings and D. Mayne, *Model Predictive Control: Theory and Design*. Nob Hill Publishing, 2015.

[4] K. Åström and T. Hägglund, "Revisiting the Ziegler–Nichols step response method for PID control," *Journal of Process Control*, vol. 14(6), pp. 635–650, 2004.

[5] A. P. Schoellig, F. L. Mueller, and R. D'Andrea, "Optimization-based iterative learning for precise quadrocopter trajectory tracking," *Autonomous Robots*, vol. 33(1-2), pp. 103–127, 2012.

[6] C. J. Ostafew, A. P. Schoellig, and T. D. Barfoot, "Visual teach and repeat, repeat, repeat: Iterative learning control to improve mobile robot path tracking in challenging outdoor environments," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2013, pp. 176–181.

[7] Q. Li, J. Qian, Z. Zhu, X. Bao, M. K. Helwa, and A. P. Schoellig, "Deep neural networks for improved, impromptu trajectory tracking of quadrotors," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2017, pp. 5183–5189.

[8] F. Berkenkamp, A. P. Schoellig, and A. Krause, "Safe controller optimization for quadrotors with gaussian processes," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2016, pp. 493–496.

[9] D. A. Bristow, M. Tharayil, and A. G. Alleyne, "A survey of iterative learning control," *IEEE Control Systems Magazine*, vol. 26(3), pp. 96–114, 2006.

[10] J.-A. M. Assael, N. Wahlström, T. B. Schön, and M. P. Deisenroth, "Data-efficient learning of feedback policies from image pixels using deep dynamical models," *arXiv preprint:1510.02173v2*, 2015.

[11] K. J. Hunt, D. Sbarbaro, R. Żbikowski, and P. J. Gawthrop, "Neural networks for control systems a survey," *Automatica*, vol. 28(6), pp. 1083–1112, 1992.

[12] S. Balakrishnan and R. Weil, "Neurocontrol: A literature survey," *Mathematical and Computer Modelling*, vol. 23(1-2), pp. 101–117, 1996.

[13] S. Bansal, A. K. Akametalu, F. J. Jiang, F. Laine, and C. J. Tomlin, "Learning quadrotor dynamics with neural nets for flight control," *arXiv preprint:1610.05863*, 2016.

[14] W. He, Y. Dong, and C. Sun, "Adaptive neural impedance control of a robotic manipulator with input saturation," *IEEE Trans. on Systems, Man, and Cybernetics: Systems*, vol. 46(3), pp. 334–344, 2016.

[15] C. Sánchez-Sánchez and D. Izzo, "Real-time optimal control via deep neural networks: study on landing problems," *arXiv preprint arXiv:1610.08668*, 2016.

[16] M. T. Frye and R. S. Provence, "Direct inverse control using an artificial neural network for the autonomous hover of a helicopter," in *IEEE Systems, Man and Cybernetics Intl. Conf.*, 2014, pp. 4121–4122.

[17] S. S. Ge and J. Zhang, "Neural-network control of nonaffine nonlinear system with zero dynamics by state and output feedback," *IEEE Trans. on Neural Networks*, vol. 14(4), pp. 900–918, 2003.

[18] Y. Zhang, G. Tao, and M. Chen, "Adaptive neural network based control of noncanonical nonlinear systems," *IEEE Trans. on Neural Networks and Learning Systems*, vol. 27(9), pp. 1864–1877, 2016.

[19] F.-C. Chen and H. K. Khalil, "Adaptive control of a class of non-linear discrete-time systems using neural networks," *IEEE Trans. on Automatic Control*, vol. 40(5), pp. 791–801, 1995.

[20] H. Jaeger, *Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the "echo state network" approach*. GMD-Forschungszentrum Informationstechnik, 2002, vol. 5.

[21] F. J. Doyle and M. A. Henson, "Nonlinear systems theory," in *Nonlinear Process Control*, M. A. Henson and D. E. Seborg, Eds. Prentice Hall PTR Upper Saddle River, New Jersey, 1997.

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

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

[24] H. Sussmann, "Limitations on the stabilizability of globally-minimum-phase systems," *IEEE Trans. on Automatic Control*, vol. 35(1), pp. 117–119, 1990.

[25] A. Isidori, *Nonlinear Control Systems, 3rd edition*. Springer, 1995.

[26] M. K. Helwa and A. P. Schoellig, "On the construction of safe controllable regions for affine systems with applications to robotics," in *IEEE Conf. on Decision and Control (CDC)*, 2016, pp. 3000–3005, Available on ArXiv: 1610.01243.

[27] J. L. Giesbrecht, H. K. Goi, T. D. Barfoot, and B. A. Francis, "A vision-based robotic follower vehicle," in *SPIE Unmanned Systems Technology XI*, vol. 7332, 2009, pp. 73 321O1–73 321O12.

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