Active Training Trajectory Generation for Inverse Dynamics Model Learning with Deep Neural Networks

Siqi Zhou and Angela P. Schoellig

***Abstract*— Inverse dynamics models have been used in robot control algorithms to realize a desired motion or to enhance a robot's performance. As robot dynamics and their operating environments become more complex, there is a growing trend of learning uncertain or unknown dynamics from data. While techniques such as deep neural networks (DNNs) have been successfully used to learn inverse dynamics, it is usually implicitly assumed that the learning modules are trained on sufficiently rich datasets. In practical implementations, this assumption typically results in a trial-and-error training process, which can be inefficient or unsafe for robot applications. In this paper, we present an active trajectory generation framework that allows us to systematically design informative trajectories for training DNN inverse dynamics modules. In particular, we introduce an episode-based algorithm that integrates a spline trajectory optimization approach with DNN active learning for efficient data collection. We consider different DNN uncertainty estimation techniques and active learning heuristics in our work and illustrate the proposed active training trajectory generation approach in simulation. We show that the proposed active training trajectory generation outperforms adhoc, intuitive training approaches.**

I. Introduction

In recent years, learning techniques such as deep neural networks (DNNs) and Gaussian processes (GPs) are increasingly used in robot control to compensate for uncertain and unmodeled dynamics that would otherwise impact a robot's performance. In previous work [1], we studied a DNN-based approach that enhances the trajectory tracking performance of black-box control systems. In particular, a DNN module is trained to approximate the inverse dynamics of the underlying system, and at test time, it is pre-cascaded to the system to enhance the tracking performance (Fig. 1). While we verified the efficacy of our approach with extensive experiments [1], [2], one open question that requires further exploration is a systematic trajectory generation approach for training the DNN inverse dynamics module.

In addtion to our work, various neural network (NN)-based control architectures have been proposed in the control literature. In these works, a common assumption is that the NNs are trained on datasets that sufficiently cover the operational space [3]. In practical applications, this assumption often results in a trial-and-error process of collecting data, training the model, testing in experiment, and repeating this process until satisfactory control performance is achieved based on a representative dataset. This trial-and-error process can lead

The authors are with the Dynamic Systems Lab (http://www.dynsyslab.org), Institute for Aerospace Studies, University of Toronto, Canada. The authors are also affiliated with the Vector Institute for Artificial Intelligence, Toronto. Emails: siqi.zhou@robotics.utias.utoronto.ca, schoellig@utias.utoronto.ca

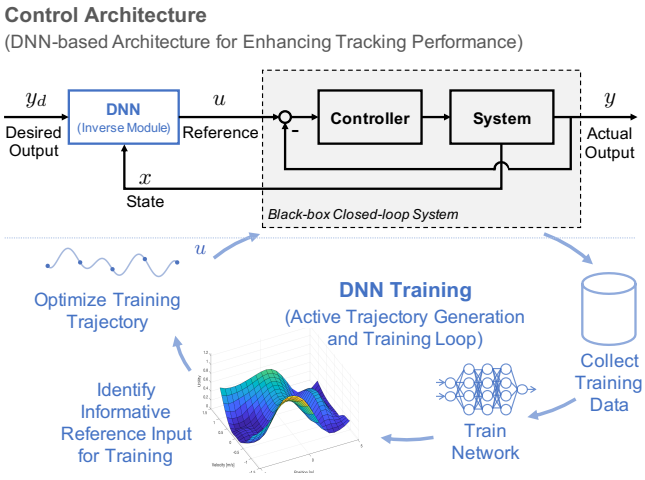

Fig. 1. A deep neural network (DNN)-based control architecture *(top)* was proposed in [1], [2] to enhance the tracking performance of black-box, closed-loop control systems. In this work, we study an episode-based active trajectory generation approach *(bottom)* for systematically training the DNN inverse dynamics module. With the proposed approach, the DNN module is trained in a closed-loop manner, where in each episode, informative points for training the DNN module are identified and a smooth trajectory is generated for collecting the data in the next set of robotics experiments.

to unnecessary training and related costs on physical robots, or safety risks in industrial applications. This fact motivates us to investigate approaches that guide the data collection process towards experiments that are the most informative for (D)NN-based model learning.

From the machine learning literature, a concept that can be adopted for informative DNN training data collection is active learning [4]. With active learning, instead of passively training the learner using a pre-collected dataset, the learner is allowed to 'actively select' the most informative points to query. This concept has been applied to various machine learning problems such as segmentation and image/text classification with the goal of saving the time and cost associated with manually labelling datasets [4]. In this work, we adopt the idea of active learning and propose an optimization framework that allows us to systematically generate feasible and informative trajectories for training DNN inverse dynamics models. With the proposed active trajectory generation framework, we aim to *(i)* improve the efficiency of the data collection and *(ii)* provide a means for monitoring when the training of DNN-based robot model learning is sufficient.

II. Related Work

The concept of active learning is closely tied to the theory of optimal experimental design (OED), which is a branch of statistics that outlines the mathematical foundations and

statistical criterion for automating query selections [5]. The theory of OED has been discussed in a wide range of contexts including neuroscience [6], system identification [7], structural optimization [8], and experimental economics [9]. In these different applications, the shared goal for OED is to maximize the information content gathered about an underlying process of interest with limited experiments.

While OED can encompass experimental conditions in a broader sense, active learning focuses on input design for black-box models. As shown in [10], common active learning heuristics such as uncertainty sampling can be connected mathematically to OED optimality criteria. Examples of active learning can be found in classification tasks, where unlabelled images or text are selected heuristically to minimize prediction errors [11], [12], or in regression tasks [13]–[15], where regions lacking data are identified for further exploration. As discussed in [16], in addition to improved data efficiency, with controlled input selection, active learning is also a way to enhance generalization of black-box models. Despite its advantages, it should be noted that typical frameworks of active learning provide a set of informative points but do not account for issues such as continuity or feasibility constraints that are critical to collecting data on physical robots for model learning.

In the literature, one approach to account for feasibility constraints is to parametrize an input trajectory and formulate an optimization problem for trajectory generation [17]. This approach has been utilized to generate excitation trajectories for identifying dynamic parameters of manipulators [18]–[20]. In particular, the identification of the manipulator dynamics is written as a linear regression problem. The joint trajectories are parametrized as splines [19] or harmonic functions [18], [20], and the trajectory parameters are optimized to minimize the model parameter uncertainty or the condition number of the linear regression model. In this paper, we similarly formulate an optimization problem for generating input trajectories. However, we present a method that incorporates an active learning objective for training generic inverse dynamics models parametrized as DNNs.

III. Problem Statement

We consider the DNN-enhanced control architecture shown in Fig. 1. In this learning-based approach, a DNN module is trained offline to approximate the inverse dynamics of a baseline closed-loop system. The trained DNN is then pre-cascaded to the baseline system at test time to enhance its tracking performance. The goal of this work is to derive a framework that allows us to design informative trajectories for systematically training the DNN inverse module.

A. Preliminaries

We consider a baseline closed-loop system represented by

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

where k is the discrete time index, $x \in \mathbb{R}^n$ is the system state, $u \in \mathbb{R}$ is the reference of the closed-loop baseline system, $y \in \mathbb{R}$ is the output, and f, g, and h are smooth functions with consistent dimensions. System (1) is said to have a relative degree r around an operating point (x_0, u_0) if *(i)* $\frac{\partial}{\partial u} h \circ f^p(f(x(k)) + g(x(k))u(k)) = 0, \forall p = 0, ..., r-2$ for each point in the neighbourhood of (x_0, u_0), and *(ii)* $\frac{\partial}{\partial u} h \circ f^{r-1}\big(f(x(k)) + g(x(k))u(k)\big) \neq 0$ at (x_0, u_0), where $h \circ f$ is the composition of the functions h and f, and $f^p(\cdot)$ denotes the pth composition of f, with $f^0(x(t)) = x(t)$ [21]. As shown in [1], if a system has a well-defined relative degree r, and $y(k+r)$ is affine in $u(k)$, we can derive the reference for exact tracking (i.e., $y(k+r) = y_d(k+r)$):

$$u(k) = \frac{1}{\mathcal{G}\left(x(k)\right)} \left(y_d(k+r) - \mathcal{F}\left(x(k)\right)\right), \quad (2)$$

where $\mathcal{F}\left(x(k)\right) = h \circ f^r(x(k))$ and $\mathcal{G}\left(x(k)\right) = \frac{\partial}{\partial u} h \circ f^{r-1}(f(x(k)) + g(x(k))u(k))$. When the exact dynamics of the baseline system (1) is unknown but it is minimum phase and has a well-defined relative degree, then we can train a DNN module to approximate Eqn. (2) to effectively enhance the tracking performance of the baseline system [1], [2]:

$$u(k) = \eta_\theta\left(x(k), y_d(k+r)\right), \quad (3)$$

where $\eta_\theta(\cdot)$ denotes the nonlinear function represented by the DNN module, and θ denotes the DNN module parameters.

Algorithm 1 Episode-based Training Trajectory Optimization for DNN Inverse Dynamics Model Learning

1: Initialize with some trajectory $\{u(k)\}$ for $k = 1, ..., K$
2: **while** not converged **do**
3: Run the training trajectory on the baseline system and record the input-output response data $\{x(k), y(k), u(k)\}$ for $k = 0, ..., K$
4: Extend DNN training dataset with input $\xi = [x(k), y_d(k+r)]$ and output $\gamma = u(k)$
5: Train DNN inverse dynamics model on the dataset
6: Select training points that maximize an active learning utility function
7: Form a quadratic program (QP) to obtain a feasible training trajectory $\{u(k)\}$ for $k = 0, ..., K$
8: **end while**

B. Episode-based Active Learning Framework

We consider an episode-based training trajectory generation framework outlined in Alg. 1 for systematically training the DNN inverse module. In each episode, we first identify the informative points for training the DNN module and formulate an optimization problem to generate a training trajectory $\{u(k)\}$, $k = 0, ..., K$, where K is the predefined training trajectory length. The generated training trajectory is sent to the baseline system, and the input-output response data $\{x(k), y(k), u(k)\}$ is recorded. The training dataset for the DNN model is extended using the obtained system response data, where the paired input and output of the training dataset are $\xi = [x(k), y(k+r)]$ and $\gamma = u(k)$, respectively. Given this dataset, the DNN module is trained with standard stochastic gradient descent (SGD) algorithms. In the next sections, we derive the details of the proposed active training trajectory generation in Ln. 6-7 of Alg. 1.

IV. Background on Active Learning for DNNs

In this section, we provide a brief summary of the active learning literature for DNNs to facilitate our discussion.

A. DNN Model Preliminaries

We consider a L-layer fully-connected DNN denoted by $\gamma = \eta_\theta(\xi)$, where ξ is the input to the network, γ is its output, and $\theta = (w_1, ..., w_L, b_1, ..., b_L)$ is an augmented vector of weights and biases parametrizing the network. We can express an L-layer network η_θ as

$$\begin{aligned} &\zeta_0 = \xi, \\ &\zeta_l = \sigma\left(w_l \zeta_{l-1} + b_l\right), \quad \forall l = 1, ..., L-1, \\ &\gamma = w_L \zeta_{L-1} + b_L, \end{aligned} \tag{4}$$

where $w_l \in \mathbb{R}^{N_l \times N_{l-1}}$ and $b_l \in \mathbb{R}^{N_l}$, N_l is the number of neurons in a hidden layer, $\sigma : \mathbb{R}^{N_l} \mapsto \mathbb{R}^{N_l}$ is the element-wise activation operation applied in a hidden layer, and ζ_l for $l = 1, ..., L-1$ is the output of a hidden layer. Given a training dataset with D paired input-output points $\mathcal{D} = \{\xi_d, \gamma_d\}_{d=1}^D$, SGD algorithms can be used to find a set of parameters θ that minimizes the network prediction error.

B. Predictive Uncertainty Estimation for DNNs

Common active learning heuristics are based on the uncertainty of the learner's output predictions (i.e., the predictive uncertainty). For each input, there is a corresponding prediction of mean and variance of the output. Intuitively, with active learning, we wish to collect data at inputs which the learner is uncertain about. We review three common predictive uncertainty estimation techniques for DNNs.

1) Fisher Information: In the Fisher information approach, we assume that the conditional probabilities $p(\gamma|\xi, \theta)$ at distinct inputs ξ are independent Gaussian distributions with expectations $\mathbb{E}_{p(\gamma \mid \xi, \theta)}[\gamma \mid \xi, \theta] = \eta_\theta(\xi)$, where $\mathbb{E}_{p(\gamma \mid \xi, \theta)}[\cdot]$ denotes the expectation over $p(\gamma \mid \xi, \theta)$. We can approximate the predictive variance at a given input ξ as

$$\text{Var}(\gamma \mid \xi, \theta) = \nabla_\theta \eta(\xi)^T M^{-1} \nabla_\theta \eta(\xi), \tag{5}$$

where ∇_θ denotes the gradient with respect to the parameters θ, and $M = \frac{1}{S^2} \sum_{d=1}^D \nabla_\theta \eta(\xi_d) \nabla_\theta \eta(\xi_d)^T$ with $S^2 = \frac{1}{2D} \sum_{d=1}^D ||\eta_\theta(\xi_d) - \gamma_d||^2$ approximates the Fisher information matrix [22].

2) Bagging: Bagging is one of the common methods for empirically estimating the predictive variance of DNNs. In typical implementations, an ensemble of DNN models is trained, and randomization is introduced to the DNN ensemble via randomized batch samples and/or initial weights [23]. The predictive variance at a given input is estimated based on the empirical variance of the DNN outputs.

3) Dropout Approximate Inference: An alternative ensemble uncertainty estimation technique is based on dropout approximate inference, which can be interpreted as a variational approximation of the Bayesian inference performed with deep Gaussian processes [13]. In this approach, a single DNN is trained with stochastic dropout [24]. The predictive variance of the DNN at test time is estimated based on multiple forward passes with independently sampled dropout units:

$$\text{Var}(\gamma \mid \xi, \theta) = \hat{S}^2 + \tau^{-1} I, \tag{6}$$

where $\hat{S}^2$ is an empirical estimate of the predictive variance found via multiple forward passes, $\tau = \frac{p_{\text{keep}} l^2}{2K\lambda}$ is the model precision, p_{keep} is the dropout Bernoulli distribution parameter, K is the data size, and I is the identity matrix [13].

C. Measures of Informativeness

With active learning, our goal is to collect data $\mathcal{D}$ that maximizes the information we gain about the underlying process. We denote a generic active learning problem as

$$\xi^* = \arg\max_{\xi} \quad U(\xi, \theta), \tag{7}$$

where U denotes an utility function that measures the amount of information provided by querying ξ. We briefly review the common active learning heuristics for DNNs. More thorough discussions can be found in review papers such as [4].

1) Uncertainty Sampling: Uncertainty sampling is an active learning heuristic that encourages selecting an input ξ^* which the model is currently the most uncertain about. The utility function can be written as $U_{\text{US}} = \mathbb{H}(\gamma \mid \xi, \theta)$, where $\mathbb{H}(\cdot)$ denotes the entropy of a random variable and is a monotonic function of variance for a Gaussian distribution.

2) Ensemble-based Approach: In an ensemble-based approach, multiple DNN models are trained for making predictions, and the extent of disagreement is used as the measure of information. This approach can be thought as a variation of uncertainty sampling, where the uncertainty is estimated by the empirical variance of the DNN ensemble [4].

3) Information Gain: Information gain is another common utility function used for DNN regression problems. It characterizes the expected regression error reduction for an unbiased learner [25] and is defined as $U_{IG} = \mathbb{E}_{p(\xi')}\left[\mathbb{H}(\gamma \mid \xi', \theta) - \mathbb{E}_{p(\gamma \mid \xi, \theta)}[\mathbb{H}(\gamma \mid \xi', \theta^+)]\right]$, where θ^+ is the parameter if the DNN is trained on a candidate input ξ.

V. Formulation of an Active Training Trajectory Generation Framework

In this section, we formulate the active training trajectory generation framework for DNN inverse dynamics learning.

A. Spline Trajectory Generation

We adopt a trajectory generation framework similar to [17]. In particular, we consider reference trajectories parametrized by Nth-order polynomial splines $T_s(t) = \sum_{n=0}^N p_{s,n} t^n$ joined at prescribed times $\{t_1, ..., t_{S-1}\}$, where T_s is the sth polynomial of the spline, t is the continuous time, and $p_{s,n}$ are the coefficients of the sth polynomial. The smoothness of the reference trajectories is enforced by penalizing the mth derivative of the spline:

$$\begin{aligned} \min_p \quad & \int_{t_0}^{t_S} \left|\left|T^{(m)}(t)\right|\right|^2 dt \\ \text{s.t.} \quad & T_s(\tau) = \bar{T}(\tau), \forall s = 1, ..., S, \tau = \{t_{s-1}, t_s\}, \\ & T_s^{(l)}(t_s) = T_{s+1}^{(l)}(t_s), \forall s = 1, ..., S-1, \\ & \qquad\qquad l = 1, ..., l_{\max}, \end{aligned} \tag{8}$$

where $T(t)$ denotes the spline trajectory, $\bar{T}(t)$ is the predefined waypoint at t, $||\cdot||$ denotes the Euclidean norm, $p = (p_1, ..., p_S)$ is an augmented vector with p_s containing the coefficients of the sth polynomial segment (in ascending order), $t_0, ..., t_S$ are the times corresponding to the interior points of the spline trajectory, and $l_{\max}$ is the order of continuity enforced at the interior points.

By substituting the definition of $T_s(t)$ into Eqn. (8) and computing the derivatives, one can show that the spline trajectory generation in Eqn. (8) can be formulated as

$$\begin{aligned} \min_{p_1,...,p_S} \quad & \sum_{s=1}^{S} p_s^T Q_s p_s \\ \text{s.t.} \quad & A_s p_s - b_s = 0, \quad \forall s = 1, ..., S, \\ & Ap = 0, \end{aligned} \tag{9}$$

where Q_s is a matrix enforcing smoothness of the sth polynomial segment, and the pairs (A_s, b_s) and the matrix A enforce continuity constraints at the interior points of the spline. The matrix Q_s for the sth polynomial is

$$Q_s = \left[\begin{array}{c|c} 0_{m\times m} & 0_{m\times(N-m+1)} \\ \hline 0_{(N-m+1)\times m} & \widetilde{C}^T \widetilde{Q}_s \widetilde{C} \end{array}\right], \tag{10}$$

where $\widetilde{C} = \text{diag}(c(m), ..., c(N))$, $c(q) = \prod_{i=0}^{l-1} q - i$, and

$$\widetilde{Q}_s = \begin{bmatrix} t_s - t_{s-1} & \cdots & \frac{t_s^{N-m+1} - t_{s-1}^{N-m+1}}{N-m+1} \\ \vdots & \iddots & \vdots \\ \frac{t_s^{N-m+1} - t_{s-1}^{N-m+1}}{N-m+1} & \cdots & \frac{t_s^{2(N-m)+1} - t_{s-1}^{2(N-m)+1}}{2(N-m)+1} \end{bmatrix}. \tag{11}$$

For the continuity constraints, we note that the lth derivative of T_s evaluated at time t is $T_s^{(l)}(t) = A_{slt}\, p_s$, where $A_{slt} = \begin{bmatrix} 0_l & c(l) & c(l+1)t & \cdots & c(N)t^{N-l} \end{bmatrix}$, where 0_l is a zero row vector with dimension indicated by the subscript. The zero-order continuity constraint can be enforced by setting

$$A_s = \begin{bmatrix} A_{s0t_{s-1}} \\ A_{s0t_s} \end{bmatrix} \text{ and } b_s = \begin{bmatrix} \bar{T}(t_{s-1}) \\ \bar{T}(t_s) \end{bmatrix}. \tag{12}$$

The higher-order continuity constraints at the interior points are introduced with A. For instance, the lth-order continuity enforced at t between the sth and $(s+1)$th polynomials can be introduced by augmenting the following row to the matrix

$$[A]_i = \begin{bmatrix} 0_{(s-1)(N+1)} & A_{slt} & -A_{(s+1)lt} & 0_{(S-s-1)(N+1)} \end{bmatrix}, \tag{13}$$

where $[A]_i$ denotes the ith row of A.

B. Integrating Active Learning and Trajectory Optimization

In this subsection, we integrate the active learning concept discussed in Sec. IV and the spline trajectory generation approach presented in Sec. V-A.

Based on the approaches discussed in Sec. IV, in each episode, we estimate the DNN module predictive uncertainty and calculate the utility over the DNN input space. This allows us to identify a set of points in the DNN input space that are the most informative for training the DNN module:

$$\xi^* = \arg\max_{\xi} \quad U(\xi, \theta), \tag{14}$$

where $\xi = [x(k), y(k+r)]$ and $\gamma = u(k)$ for our problem.

In contrast to typical active learning applications, where the input ξ^* is directly used as the next point to query, for our inverse learning problem, we need to identify the informative DNN outputs γ^* (i.e., the informative references) to be sent to the baseline system. To this end, given the informative input(s) ξ^*, we evaluate the corresponding outputs of the DNN module $\gamma^* = \eta_\theta(\xi^*)$ and the associated standard deviations $\Delta\gamma$ at these inputs. A set of candidate DNN outputs γ_s^* are sampled from $\mathcal{N}(\gamma^*, \alpha\Delta\gamma)$, where $\mathcal{N}(\gamma^*, \alpha\Delta\gamma)$ denotes a Gaussian distribution with a mean of γ^* and a standard deviation of $\alpha\Delta\gamma$, and $\alpha \geq 1$ is a parameter that controls the range of exploration. Intuitively, if the uncertainty estimates of the DNN module are sufficiently accurate, the samples γ_s^* encourage explorations over regions where the DNN module is currently uncertain, where the range of exploration is proportional to the extent of uncertainty. We treat γ_s^* as the informative reference points that are used in the generation of the training trajectory for the DNN inverse module.

The samples γ_s^* are incorporated into the spline trajectory generation algorithm as constraints at the interior points. In particular, we formulate the following problem:

$$\begin{aligned} \min_{p_1,...,p_S} \quad & \sum_{s=1}^{S} p_s^T Q_s p_s \\ \text{s.t.} \quad & A_s p_s = \gamma_s^*, \quad \forall s = 1, ..., S, \\ & Ap = 0, \\ & A_{slt_i} p_s \leq b_{sl,\max}, \quad \forall t_i \in \mathcal{T}, \end{aligned} \tag{15}$$

where $\mathcal{T}$ is a set of sample times along the trajectory, and the inequality constraints introduced with the pairs $(A_{slt_i}, b_{sl,\max})$ can be used to account for additional feasibility constraints (e.g., bounds on velocities and accelerations).

Note that we optimize a continuous-time trajectory $T(t)$. The trajectory is discretized at a defined sampling interval Δt to obtain the sequence of references $\{u(k)\}$ for training.

VI. Simulation Results

In this section, we use a numerical example to illustrate the proposed active training trajectory generation. We consider a minimum phase baseline system represented by

$$\begin{aligned} x(k+1) &= \begin{bmatrix} 0 & 1 \\ -0.15 & 0.8 \end{bmatrix} x(k) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(k) \\ y(k) &= \begin{bmatrix} -0.2 & 0.5 \end{bmatrix} x(k). \end{aligned} \tag{16}$$

Following Eqn. (3), we can train a DNN module with input $\xi = [x(k), y_d(k+1)]$ and output $\gamma = u(k)$ to approximate the exact inverse of the baseline system (Eqn. (2)). In [1], with a similar system, we showed that the DNN inverse learning approach can reduce the tracking error of the baseline system to approximately zero.

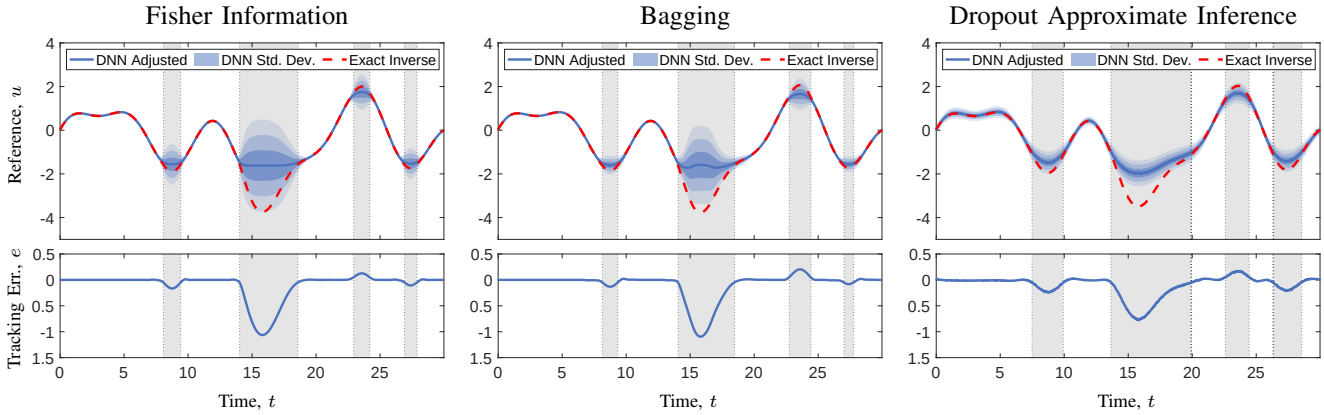

Fig. 2. A comparison of the references u and the tracking errors, $e = y_d - y$, of three DNN-enhanced systems implemented based on different uncertainty estimation techniques: *(i)* Fisher information, *(ii)* bagging, and *(iii)* dropout approximate inverse. The results correspond to a test trajectory $y_d(t) = \frac{3}{5}\cos\left(\frac{\pi}{3}t\right) + \sin\left(\frac{\pi}{5}t\right) + \sin\left(\frac{\pi}{10}t\right) - \frac{3}{5}$. From the plots, it can be seen that at time steps where the DNNs are uncertain (blue shadings in the top row), we correspondingly observe relatively large tracking errors (regions shaded in grey); conversely, at time steps where the DNNs are certain, the tracking errors of the DNN-enhanced systems are close to zero (unshaded regions). Given the correlation between the DNN uncertainty and the tracking error, we can then exploit the uncertainty information to efficiently design trajectories for training the DNN inverse modules.

Although being effective at test time, the DNN module in our previous work was trained on hand-designed sinusoidal trajectories; the quality of the training could only be validated when the DNN module is tested on the baseline system. In the following simulation study, we illustrate the active-learning-based framework for systematically designing DNN training trajectories. This framework allows us to *(i)* infer training quality prior to testing the DNN module on the system and *(ii)* efficiently collect data that is needed for good generalization. In the following subsections, we first present a set of simulations to examine the correlation between the predictive uncertainty of the DNN module and the system's tracking performance (Sec. VI-A) and then utilize the uncertainty estimates in the proposed active learning framework for DNN training trajectory design (Sec. VI-B).

A. DNN Predictive Uncertainty Estimation

In this subsection, we examine three techniques for estimating the DNN inverse dynamics module's predictive uncertainty: *(i)* Fisher-information-based estimation, *(ii)* bagging, and *(iii)* dropout approximate inference (see Sec. IV-B).

1) Architecture of the DNN Inverse Module: For the three techniques we implement in this simulation study, the DNNs are three-layer feedforward networks, and each hidden layer of the networks has ten hyperbolic tangent neurons. There are overall 161 weight and bias parameters in each network. The DNN modules are initially trained on a sinusoidal trajectory $y_d(t) = \sin\left(\frac{\pi}{10}t\right)$ sampled at $\Delta t = 0.015$. The training dataset is constructed from the system response data with paired input and output $\mathcal{D} = \{x(k), y(k+1); u(k)\}_{k=1}^{K}$, where K is the size of the training dataset. Standard SGD algorithms are used for optimizing the network parameters.

2) Implementation of the Different Uncertainty Estimation Techniques: The Fisher-information-based approach is implemented for a DNN module with a single network. The predictive uncertainty of the DNN module at test time is computed based on Eqn. (5), where the module input and output are $\xi = [x(k), y_d(k+1)]$ and $\gamma = u(k)$, respectively. The inverse of the Fisher information matrix M is independent of the input ξ, and we pre-compute M^{-1} from the training data to minimize the computational load at test time. For the bagging approach, we use a committee of 20 DNNs trained with different randomly sampled initial parameters [23]. The uncertainty of the DNN module at test time is estimated based on the empirical variance of the committee outputs. For the dropout approximate inference approach, we train a DNN with stochastic dropout [13], [24]. The uncertainty of the DNN at test time is estimated using Eqn. (6). Here, we use 300 forward passes with independent weight dropout samples, and the dropout probability is 0.05.

3) Simulation Results: We show the results of the DNN predictive uncertainty estimation techniques on a test trajectory $y_d(t) = \frac{3}{5}\cos\left(\frac{\pi}{3}t\right) + \sin\left(\frac{\pi}{5}t\right) + \sin\left(\frac{\pi}{10}t\right) - \frac{3}{5}$. This test trajectory differs from the training trajectory, and the DNN inputs encountered at test time are only partially covered by the training dataset. Fig. 2 shows the predictions of the DNN modules (top row) and the resulting tracking errors of the corresponding DNN-enhanced systems (bottom row). The time intervals where the systems have relatively larger tracking errors are shaded in grey. From the unshaded regions of the plots, we see that when the uncertainty of the DNN module is small, the reference computed by the DNN module coincides with the reference computed based on the exact inverse of the system (Eqn. (2)), and the tracking error of the DNN-enhanced system is close to zero. From the shaded regions, for each technique, we see a correlation between the uncertainties of the DNN modules and the tracking errors of the DNN-enhanced systems. Given this correlation, we can then exploit the uncertainty information to efficiently design trajectories for training the DNN inverse module.

B. Active Training Trajectory Generation

In previous work [1], [2], we trained the DNN inverse modules based on hand-designed trajectories. In this subsection, we illustrate the proposed active training trajectory generation with the system considered in Sec. VI-A.

1) Simulation Setup: In this simulation study, we examine four approaches for training trajectory generation:

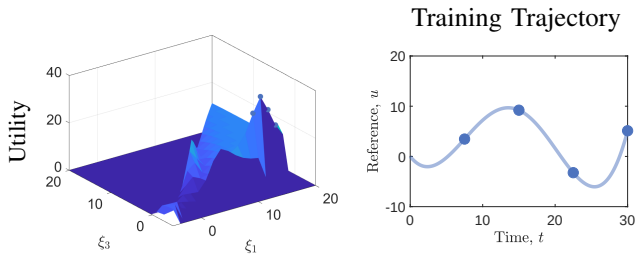

Fig. 3. Illustration of the proposed active trajectory generation approach with Fisher-information-based uncertainty estimation. The plots shown above correspond to the first training episode. The informative points for training the DNN module are identified based on the utility function *(left)*. The informative points are then passed to the trajectory generation algorithm (Eqn. (15)) to generate a smooth training trajectory *(right)*.

(M1) *Baseline approach:* We consider a baseline training trajectory generation approach that resembles previous work [1]. In particular, in consecutive episodes, the DNN module is trained on sinusoidal trajectories with increasing amplitudes and frequencies. For the results in this subsection, the amplitudes of the sinuosoidal trajectories range from 1 to 5.5, and the frequencies range from 0.02 Hz to 0.2 Hz.

(M2) *Fisher-information-based approach:* In each episode, the uncertainty of the DNN module is estimated based on Eqn. (5) and is used in the active trajectory generation framework proposed in Sec. V-B. The DNN module is a three-layer feedforward network with ten hyperbolic neurons in each hidden layer.

(M3) *Bagging-based approach:* A committee of 20 DNNs is used to estimate the predictive uncertainty of the DNN module in each training episode, and the proposed active trajectory generation framework is similarly applied. The architectures of the DNNs are identical to that used in (M2). To reduce the computation load in the testing phase, we use a subset of five DNNs for reference computation.

(M4) *Dropout-inference-based approach:* A single network is trained with stochastic dropout, and the uncertainty of the DNN module is estimated based on the dropout approximate inference technique [13]. The DNN architecture is identical to that used in (M2), and the proposed active training trajectory generation approach is similarly applied.

For approaches (M2)-(M4), we use a grid search to identify informative points for training the DNN modules. For the system we consider (Eqn. (16)), the input space of the DNN module is $\xi = [x(k), y_d(k+1)]$. In order for the approach to be efficient, we restrict the grid search to a subset of the DNN input space that excludes regions with implausible combinations of $x(k)$ and $y_d(k+1)$.

2) Simulation Results: We illustrate the proposed active DNN training trajectory generation approach in Fig. 3. As shown in the left panel, in each episode, the utility function is evaluated over a set of points of the DNN input space. The four points with the highest utilities are selected. The DNN output mean and uncertainty are evaluated at these selected points and are sent to the training trajectory generation algorithm formulated in Eqn. (15). As shown in the right panel, the generated reference trajectory passes through the informative points and is sufficiently smooth for the baseline system to be able to execute.

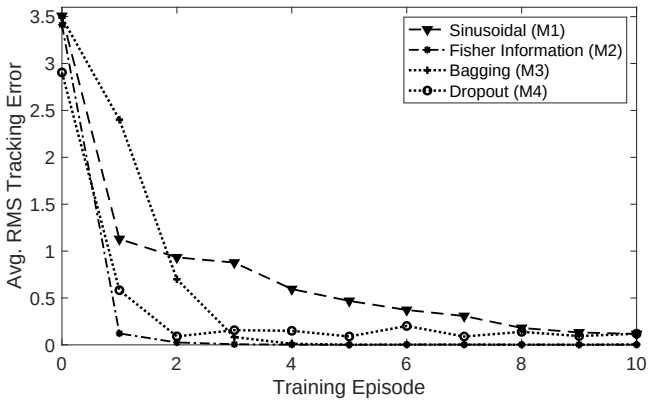

Fig. 4. The average RMS tracking errors of the DNN-enhanced systems on ten test trajectories for an increasing number of the DNN training episodes.

We test the generalizability of the DNN modules trained with the different training trajectory generation approaches on ten test trajectories. The test trajectories have the form of $y_d(t) = \alpha_1 \sin\left(\frac{2\pi}{\beta_1}t\right) - \alpha_2 \cos\left(\frac{2\pi}{\beta_2}t\right) + \alpha_2$, and the parameters (α_i, β_i) are randomly generated from uniform distributions $\alpha_i \sim \mathcal{U}(0,5)$ and $\beta_i \sim \mathcal{U}(5,50)$ for $i = \{1,2\}$, where $\mathcal{U}$ denotes a uniform distribution. We initialize the training with a sinusoidal trajectory $y_d(t) = \sin\left(\frac{\pi}{10}t\right)$. Fig. 4 shows the average root-mean-square (RMS) tracking error of the DNN-enhanced systems on the ten test trajectories after the algorithm in Alg. 1 is ran for an increasing number of training episodes. From the plot, it can be seen that all the training trajectory generation approaches are effective in the sense that the corresponding DNN-enhanced systems converge to small RMS tracking errors. In comparison, the DNN modules trained with the active training trajectory generation approaches (M2)-(M4) generally have faster convergence as compared to the baseline sinusoidal trajectory generation approach (M1). This means that a lower number of training trajectories is needed to learn an effective inverse model.

It should be noted that even though, with (M1), the designed sinusoidal trajectories in the training dataset approximately cover the frequencies and amplitudes of the test trajectories, the DNN module has relatively high generalization errors in the last episodes. The imperfect result of (M1) is partially due to the fact that the references corresponding to the exact inverse of the system do not necessarily lie within the desired output space which the training trajectories are initially designed to cover. This mismatch between the exact references and desired outputs makes designing training trajectories based on strategies such as (M1) non-intuitive. The active trajectory generation approaches circumvent this issue by incorporating the learning module in the loop and actively identifying the uncertain references (the DNN outputs) that are required to cover the operational space of interest.

In addition to the improved learning efficiency, it should be noted that, with the active trajectory generation approaches, the evaluation of the utility function over the DNN input space in each episode (Fig. 3) provides us with a means to monitor or infer the quality of training prior to testing the DNN module on the baseline system. On the contrary,

with typical hand-designed training trajectory generation approaches, the DNN training process is open-loop, and the quality of training may incorrectly be assumed to be good, which is unsafe for practical applications.

VII. Discussion

We proposed and illustrated a DNN training trajectory generation framework that integrates active learning and a spline trajectory optimization algorithm for smooth reference trajectory generation. This guided training trajectory generation approach circumvents the need to hand design training trajectories, which often involves intuition-based decision making (e.g., choosing parameters of sinusoidal training trajectories) and results in inefficient data collection and DNN training. While showing great promise, there are two limitations with the current approach to be aware of.

One limitation is related to the predictive uncertainty estimation for DNNs. Typical DNNs do not possess parameters to characterize their uncertainties. Although it is shown in Sec. VI-A that common predictive uncertainty estimation techniques can be used to identify regions with insufficient training data, these techniques can lead to non-conservative uncertainty estimates. In future work, we aim to further investigate alternative uncertainty estimation techniques for DNNs, which is an active research area in machine learning.

Another limitation of the current approach is the search over the DNN input space used to determine informative points. For system (1), the DNN inverse module input is $\xi = [x(k), y_d(k+r)]$. Constrained by the system dynamics and any actuation limits, the desired DNN input space encompasses regions with plausible combinations of $x(k)$ and $y_d(k+r)$. In this work, the region of interest excludes the implausible DNN inputs. A potential improvement of the current approach is to automate the search of the DNN input space by simultaneously training a forward model to account for the dynamic constraints.

VIII. Conclusion and Future Work

In this paper, we proposed an active trajectory generation framework for systematically training DNNs that represent inverse dynamics modules deployed to enhance the tracking performance of black-box baseline systems. In simulation, we showed that, by using an active trajectory generation and training approach (Fig. 1), we can significantly improve the data efficiency for training the DNN inverse module. Moreover, the proposed active training trajectory generation framework allows us to infer the training quality of the DNN module prior to testing on the physical system, which can be important for safe operation in practical applications.

Future work includes automating the search of plausible DNN inputs for identifying informative references, exploring non-conservative uncertainty estimates, as well as testing the proposed approach in experiments.

References

[1] S. Zhou, M. K. Helwa, and A. P. Schoellig, "Design of deep neural networks as add-on blocks for improving impromptu trajectory tracking," in *Proc. of the IEEE Conf. on Decision and Control (CDC)*, 2017, pp. 5201–5207.

[2] 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 *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2017, pp. 5183–5189.

[3] 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.

[4] B. Settles, "Active learning literature survey," *Technical Report, University of Wisconsin, Madison*, vol. 52(55-66), 2010.

[5] V. V. Fedorov, *Theory of Optimal Experiments*. Elsevier, 1972.

[6] R. Lorenz, R. P. Monti, I. R. Violante, C. Anagnostopoulos, A. A. Faisal, G. Montana, and R. Leech, "The automatic neuroscientist: a framework for optimizing experimental design with closed-loop real-time fMRI," *NeuroImage*, vol. 129, pp. 320–334, 2016.

[7] L. Ljung, "System identification," in *Signal Analysis and Prediction*. Springer, 1998.

[8] A. J. G. Schoofs, "Experimental design and structural optimization," in *Structural Optimization*. Springer, 1988, pp. 307–314.

[9] J. A. List, S. Sadoff, and M. Wagner, "So you want to run an experiment, now what? some simple rules of thumb for optimal experimental design," *Experimental Economics*, vol. 14(4), pp. 439–457, 2011.

[10] B. Settles, "Active learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 6(1), pp. 1–114, 2012.

[11] Y. Gal, R. Islam, and Z. Ghahramani, "Deep Bayesian Active Learning with Image Data," in *Proc. of the Intl. Conf. on Machine Learning (ICML)*, 2017.

[12] S. Tong and D. Koller, "Support vector machine active learning with applications to text classification," *Journal of Machine Learning Research (JMLR)*, vol. 2, pp. 45–66, 2001.

[13] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *Proc. of the Intl. Conf. on Machine Learning (ICML)*, 2016, pp. 1050–1059.

[14] D. A. Cohn, Z. Ghahramani, and M. I. Jordan, "Active learning with statistical models," *Journal of Artificial Intelligence Research (JAIR)*, vol. 4, pp. 129–145, 1996.

[15] R. Burbidge, J. J. Rowland, and R. D. King, "Active learning for regression based on query by committee," in *Proc. of the Intl. Conf. on Intelligent Data Engineering and Automated Learning (IDEAL)*. Springer, 2007, pp. 209–218.

[16] D. Cohn, L. Atlas, and R. Ladner, "Improving generalization with active learning," *Machine Learning*, vol. 15(2), pp. 201–221, 1994.

[17] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2011, pp. 2520–2525.

[18] G. Calafiore, M. Indri, and B. Bona, "Robot dynamic calibration: Optimal excitation trajectories and experimental parameter estimation," *Journal of Robotic Systems*, vol. 18(2), pp. 55–68, 2001.

[19] W. Rackl, R. Lampariello, and G. Hirzinger, "Robot excitation trajectories for dynamic parameter estimation using optimized b-splines," in *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2012, pp. 2042–2047.

[20] J. Swevers, C. Ganseman, D. Bilgin, J. De Schutter, and H. Van Brussel, "Optimal robot excitation and identification," *IEEE Trans. on Robotics and Automation*, vol. 13(5), pp. 730–740, 1997.

[21] 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.

[22] D. A. Cohn, "Neural network exploration using optimal experiment design," *Neural Networks*, vol. 9(6), pp. 1071–1083, 1996.

[23] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," in *Proc. of Advances in Neural Information Processing Systems (NIPS)*, 2017, pp. 6402–6413.

[24] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research (JMLR)*, vol. 15, pp. 1929–1958, 2014.

[25] S. Geman, E. Bienenstock, and R. Doursat, "Neural networks and the bias/variance dilemma," *Neural Computation*, vol. 4(1), pp. 1–58, 1992.