TYPE Original Research
PUBLISHED 16 June 2023
DOI 10.3389/fnbot.2023.1166911

Check for updates

OPEN ACCESS

EDITED BY
Miriam Zacksenhouse,
Technion Israel Institute of Technology, Israel

REVIEWED BY
Yongping Pan,
National University of Singapore, Singapore
Önder Tutsoy,
Adana Science and Technology University, Türkiye

*CORRESPONDENCE
Niceto R. Luque
✉ nluque@ugr.es

RECEIVED 15 February 2023
ACCEPTED 25 May 2023
PUBLISHED 16 June 2023

CITATION
Valencia-Vidal B, Ros E, Abadía I and Luque NR (2023) Bidirectional recurrent learning of inverse dynamic models for robots with elastic joints: a real-time real-world implementation. *Front. Neurorobot.* 17:1166911. doi: 10.3389/fnbot.2023.1166911

Bidirectional recurrent learning of inverse dynamic models for robots with elastic joints: a real-time real-world implementation

Brayan Valencia-Vidal[1,2], Eduardo Ros[1], Ignacio Abadía[1] and Niceto R. Luque[1]*

[1]Department of Computer Engineering, Automation and Robotics, Research Centre for Information and Communication Technologies, University of Granada, Granada, Spain, [2]Research Group Osiris & Bioaxis, Faculty of Engineering, El Bosque University, Bogotá, Colombia

Collaborative robots, or cobots, are designed to work alongside humans and to alleviate their physical burdens, such as lifting heavy objects or performing tedious tasks. Ensuring the safety of human–robot interaction (HRI) is paramount for effective collaboration. To achieve this, it is essential to have a reliable dynamic model of the cobot that enables the implementation of torque control strategies. These strategies aim to achieve accurate motion while minimizing the amount of torque exerted by the robot. However, modeling the complex non-linear dynamics of cobots with elastic actuators poses a challenge for traditional analytical modeling techniques. Instead, cobot dynamic modeling needs to be learned through data-driven approaches, rather than analytical equation-driven modeling. In this study, we propose and evaluate three machine learning (ML) approaches based on bidirectional recurrent neural networks (BRNNs) for learning the inverse dynamic model of a cobot equipped with elastic actuators. We also provide our ML approaches with a representative training dataset of the cobot's joint positions, velocities, and corresponding torque values. The first ML approach uses a non-parametric configuration, while the other two implement semi-parametric configurations. All three ML approaches outperform the rigid-bodied dynamic model provided by the cobot's manufacturer in terms of torque precision while maintaining their generalization capabilities and real-time operation due to the optimized sample dataset size and network dimensions. Despite the similarity in torque estimation of these three configurations, the non-parametric configuration was specifically designed for worst-case scenarios where the robot dynamics are completely unknown. Finally, we validate the applicability of our ML approaches by integrating the worst-case non-parametric configuration as a controller within a feedforward loop. We verify the accuracy of the learned inverse dynamic model by comparing it to the actual cobot performance. Our non-parametric architecture outperforms the robot's default factory position controller in terms of accuracy.

KEYWORDS

robot dynamic modeling, gated recurrent units, bidirectional recurrent neural networks, compliant robots, torque control

1. Introduction

The presence of robots among us is becoming increasingly common as technology provides them with new capabilities needed for performing diverse tasks in unstructured, scenarios, i.e., healthcare centers, homes, etc. In fact, the International Federation of Robotics (IFR) reported ∼3 million robots operating in 2020, a 10% increase compared to the previous year (IFR, 2021). This ubiquitous presence entails an increasing human–robot interaction (HRI) in a collaborative manner. Interacting with a human can be regarded as an unstructured task because it is ambiguous, and it is not based on rigid rules. Rather, HRI tasks require general policies incorporating high rates of exception and complexity, in which adaptability and flexibility are mandatory (Bicchi et al., 2008). It is precisely the need for this HRI to be safe that motivates the emergence and design criteria for collaborative robots, also called cobots (Giuliani et al., 2010).

The latest cobot generation integrates elastic actuators that offer passive compliance and minimizes execution forces to comply with safe HRI while performing unstructured tasks (Giuliani et al., 2010). These elastic joints reduce the risk of damage due to the dampening effect over the contact forces between the cobot and a stiff environment, i.e., cobot–cobot, cobot–operator, or cobot–environment contact. However, the integration of the elastic components entails a trade-off: an increase in the complexity of the cobot dynamics and modeling, which, in turn, presents challenges in control (Lee et al., 2017). The dynamic model of any robot relates joint torque values to their resultant joint motion, which implies that position and velocities could be estimated from the applied torque commands (direct dynamic model) and vice versa (inverse dynamic model). The classical methods for identifying a robot dynamic model are based on either the Lagrange equations (analytical dynamic model) or the Newton–Euler equations (numerical dynamic model; Swevers et al., 2007).

Although flexible joint robot dynamic identification is a viable option for modeling compliant cobots, this method has limitations in accurately capturing the model's parameters and the non-linear and time-varying dynamics (Pratt and Williamson, 1995). Simple analytical dynamic models often do not consider joint stiffness and elasticity or, if they do, they approximate linear behavior with coefficients that have a high degree of uncertainty (Kwon and Book, 1994; Ata et al., 1996; Calanca et al., 2011; Lee et al., 2017). Additionally, other non-linear effects such as backlash and frictional torque are often overlooked (Madsen et al., 2020). The dynamics of these effects pose a challenge to the accuracy of the model as they are affected by environmental conditions and the level of maintenance of the robot. Therefore, data-driven dynamic modeling using Machine Learning (ML) approaches, specifically artificial neural networks (ANNs), has gained popularity. ANNs are capable of capturing complex and non-linear dynamics, making them well-suited for this purpose (Xu et al., 2014; Li and Li, 2021; Liu et al., 2021). This is particularly important in tasks that involve close human-robot interaction, where ensuring the safety of human operators is critical. It is worth noting, however, that modeling flexible joints in robot dynamic identification may still be useful in certain scenarios. Nevertheless, the limitations of this approach have motivated the exploration of alternative methods such as data-driven dynamic modeling using ML techniques.

Several research studies (Graves et al., 2005; Rueckert et al., 2017; Liu et al., 2019) have consistently reported better performance of Recurrent Neural Networks (RNNs), specifically bidirectional RNNs (BRNNs), over non-recurrent ANNs in time-series problems (of a moderate number of inputs), such as the one we are dealing with. RNNs are extensively used to model different dynamic systems (Jin et al., 2022). Particularly, in robotics, unidirectional RNNs have been trained to learn the inverse or direct dynamics of rigid robots (Mukhopadhyay et al., 2019; Wang et al., 2020). Hybrid solutions combining an analytical description of the rigid robot dynamics with a data-driven deep learning (DL) model (semi-parametric model) has also been proposed (Liu et al., 2020; Çallar and Böttger, 2023). Conversely, in cobotics, unidirectional and BRNNs have been trained to learn cobot models only relating desired joint positions to actual joint positions, without taking into account the actual torque values that are required to achieve these positions (Chen and Wen, 2019). This limitation can restrict the application of these models to torque control, where knowledge of the actual torque values is crucial. Note that the majority of these studies were focused on the modeling of rigid robots and never on the application of ML to robots with elastic joints that are inherently safer, but difficult to be commanded in torque accurately due to their complex dynamics, i.e., passively compliant cobots. Importantly, these studies do not account for the future states of the joints of the robot, which provide relevant information for appropriate learning of an inverse dynamics model (Jordan and Rumelhart, 1992). Therefore, there is a need for research on (ML)-based inverse dynamic models that can account for the complex dynamics of passively compliant cobots and physical force interactions, ultimately facilitating model-free torque control. We propose and evaluate three machine learning algorithm configurations based on recurrent neural networks (RNNs).

Three approaches can be found in dynamic systems modeling depending on whether the physical parameters of the system are incorporated or not, i.e., parametric, non-parametric, and semi-parametric models. We proposed a configuration that builds a non-parametric model (NID) and two configurations that build semi-parametric models (SID and ESID) incorporating the robot's rigid body parametric model (RBD). Specifically, we wanted to determine whether combining the RBD model with learning models (BRNN) could provide greater robustness in prediction and generalization than when the RBD model was not taken into account. We also wanted to investigate whether the way in which information from the RBD model was incorporated into the semi-parametric models influenced their performance. In the SID configuration, the output of the RBD model is an input to the BRNN, whereas, in the ESID configuration, the BRNN is trained to estimate the torque corresponding to the dynamics not modeled in RBD so that the outputs of the BRNN and RBD can be combined.

Considering all the implications above, we can enumerate the main contributions of this work as follows:

- First, this study proposes an efficient dynamic data robot acquisition method. Using statistical analysis over the dataset, we identify the subset of trajectories that provided the highest dynamic cobot information, i.e., good data vs. extensive big data paradigm. The resulting training dataset is made publicly available for replication and comparison purposes.

- Second, we introduce an ML approach to the inverse dynamic modeling of cobots based on three different BRNN configurations: a non-parametric BRNN, i.e., data-driven model and two semi-parametric BRNNs, i.e., data-driven model + analytical rigid-dynamic model. We compare these data-driven models to the analytical rigid-dynamic model provided by the manufacturer and find that the BRNN-based models systematically outperform the analytical model.
- Finally, we demonstrate that implementing a BRNN-based controller that has learned the inverse dynamics of the robot can improve the performance of a cobot, even under persistent collisions or lack of active feedback. Our approach carries high accuracy while maintaining the cobot's compliance. This was illustrated as a proof-of-concept within the Supplementary video.

2. Materials and methods

2.1. Passive compliant robot

The Baxter robot® was used as our robotic demonstrator. Baxter is a collaborative and compliant robot (cobot) consisting of two arms with seven degrees of freedom (DoF) each, equipped with serial elastic actuators (SEAs) (Fitzgerald, 2013). Unlike rigid actuators, these SEAs incorporate a spring between the motor gear and the actuator end (Pratt and Williamson, 1995), which allows absorbing, to some extent, contact forces, that is, providing passive compliance. In addition to the SEAs, Baxter also holds a passive spring at the S1 joint (see Section 3), which further increases the dynamic complexity of the robot arm.

2.2. Cobot dynamic modeling and the parametric issue

Any robot dynamic model describes the relationship between the torque values applied on the robot joints and the resulting motion. This relationship is mathematically expressed using the analytic Lagrange formulation as follows:

$$\tau = M(q)\ddot{q} + H(q,\dot{q}) + G(q) + \xi(q,\dot{q},\ddot{q}), \tag{1}$$

where q, $\dot{q}$, and $\ddot{q}$, are joint positions, velocities, and accelerations, respectively. τ stands for the vector containing the joint torque values. $M(q) \in R^{nxn}$ defines the robot inertia matrix. The $H(q,\dot{q}) \in R^n$ computes the inertia and Coriolis effects, whereas $G(q) \in R^n$ computes the gravitational effects onto the robot. Finally, $\xi(q,\dot{q},\ddot{q}) \in R^n$ stands for the torque/force effects of those robot elements that were not considered elsewhere in the dynamic model, i.e., viscous friction, or the non-linear effects of the SEA springs.

Most rigid robots, i.e., equipped with high ratio gearboxes, conveniently assume $\xi = 0$ in dynamic modeling since $M(q)\ddot{q}$ and $H(q,\dot{q})$ torque contributions to the final τ are significantly larger than $\xi(q,\dot{q},\ddot{q})$. In these cases, the torques governed by rigid body dynamics (τ_{RBD}) can be defined as follows:

$$\tau_{\text{RBD}} = M(q)\ddot{q} + H(q,\dot{q}) + G(q). \tag{2}$$

However, this is no longer the case for non-rigid robots (cobots), such as Baxter. Modeling $\xi(q,\dot{q},\ddot{q})$ becomes key in associating accurately the applied cobot torque values and the subsequent motion. ξ related cobot parameter demands for accurate identification methods, which are usually mathematically intractable (Lee et al., 2017). Mathematically intractability motivates the employment of non-parametric methods for cobot dynamic modeling (Polydoros et al., 2015). Consequently, modeling an elastic robot is a challenging task that requires not only understanding or learning the dynamics of the rigid components (Equation 2) but also the additional complex dynamics inherent in its elastic joints [$\xi(q,\dot{q},\ddot{q})$]. Accurately capturing these additional dynamics requires sophisticated techniques and algorithms, as they involve frictional and elastic behavior that cannot be captured just by rigid body dynamics. Dynamic modeling that performs well for rigid robots may not be suitable for elastic ones due to the increased complexity involved in modeling their additional complex dynamics. Therefore, modeling elastic robots requires careful consideration of their specific requirements and characteristics (Madsen et al., 2020).

Note that the Baxter rigid analytical dynamic model was used for comparative purposes and the proposed semi-parametric model. We use the Unified Robot Description Format (URDF) file provided by the manufacturer of the Baxter robot. This file contains physical parameters of the robot links such as masses, inertia tensors, and relative centers of masses. *Pybullet* library (Coumans and Bai, 2016-2021) was used to build a model of the Baxter robot with these parameters. Pybullet uses the method of Newton–Euler formulation proposed by Luh et al. (1980) for the calculation of the inverse dynamics.

2.3. Cobot dynamics learning: the recurrent neural network

Inverse dynamic modeling estimates those torque values needed to generate a certain cobot movement. This problem is identified in ML terms as a regression problem (Nguyen-Tuong and Peters, 2008). To solve this regression problem, i.e., estimating the relationship between a dependent variable (torque) and independent variables (joint positions, velocities, and accelerations), we used an RNN. Note that Equation (1) describes the relationship between the robot's joint torque values and its kinematic state variables [$x = (q,\dot{q},\ddot{q})$]. The recursive Newton–Euler formulation is often used to calculate the torque values based on the robot's state at a specific instant (Luh et al., 1980). To determine the torque sequence [$Y = (y_1, ..., y_T)$] along a given trajectory, a possible approach is to first discretize the trajectory, then determine the kinematic state at each discrete instant [$X = (x_1, ..., x_T)$], and finally solve Equation (1) for each kinematic state. This approach enables us to treat the inverse dynamic problem as a time-series regression problem. By predicting the output Y based on the input sequence X [$Y = F(X)$, where F is a non-linear function], we can determine the torque sequence along a given trajectory (Rueckert et al., 2017). Interestingly, recent research has shown a relationship between the state space of a model and the hidden states of RNNs (Schüssler et al., 2019; Ljung et al., 2020; Uribarri and Mindlin, 2022), leading to investigations into the potential of RNNs to predict time series of complex non-linear dynamical systems in robotics (Mohajerin and Waslander, 2019; Çallar and Böttger, 2023).

The neural network was only fed with position and velocity values as inputs, whereas the acceleration values were inferred from those position and velocity data. When velocity sensors are available, incorporating this information directly into the network is generally preferable to estimating it from joint positions. However, in cases where velocity sensors are not available or are unreliable, estimating velocity from position data can be an alternative approach, although it may introduce errors due to noise, drift over time or, more importantly, the discretization error or truncation error (Brown et al., 1992). Regarding acceleration, our robot's sensors do not provide for acceleration. Therefore, we chose to infer acceleration internally through the RNN. Specifically, the RNN can use the time-serial information from position and velocity data to estimate acceleration. While this approach may also introduce some errors, it has been shown to work well in previous studies (Liu et al., 2019).

Recurrent neural networks are traditionally implemented using cell solutions, the most notable being the Long Short Term Memory (LSTM) cell and the Gated Recurrent Unit (GRU) cell (Chung et al., 2014). LSTM cell consists of two recurrent inputs called forged gate (c_{t-1}) and the input gate (h_{t-1}), and the output h_t is defined as follows:

$$\begin{aligned} i_t &= \sigma(W_i x_t + U_i h_{t-1} + b_i), \\ f_t &= \sigma(W_f x_t + U_f h_{t-1} + b_f), \\ r_t &= \sigma(W_r x_t + U_r h_{t-1} + b_r), \\ o_t &= \sigma(W_o x_t + U_o h_{t-1} + b_o), \\ \widetilde{c}_t &= tanh(W_{\widetilde{c}} x_t + U_{\widetilde{c}} h_{t-1} + b_{\widetilde{c}}), \\ c_t &= \sigma(f_t * c_{t-1} + i_t * \widetilde{c}_t), and \\ h_t &= o_t * tanh(c_t). \end{aligned} \tag{3}$$

Conversely, GRU is defined by one only gate, the update gate, which reduces mathematical complexity and, therefore, computational cost (Yang et al., 2020). The equations governing the GRU behavior are as follows:

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z), \tag{4}$$

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r), \tag{5}$$

$$\widetilde{h}_t = tanh(W_{\widetilde{h}} x_t + U_{\widetilde{h}}(h_{t-1} * r_t) + b_{\widetilde{h}}), and \tag{6}$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \widetilde{h}_t. \tag{7}$$

The operator $*$ stands for the Handamard (i.e., element-wise) product of two vectors. $\sigma(\cdot)$ and $tanh(\cdot)$ represent the sigmoid and the hyperbolic tangent activation functions, respectively. W_z, W_r, and $W_{\widetilde{h}}$ are the weight matrices of the input (x_t), whereas U_z, U_r, and $U_{\widetilde{h}}$ are the weight matrices of the recurrent input gate (h_{t-1}) within the cell. b_z, b_r, and $b_{\widetilde{h}}$ refer to the bias vectors. r and z are the reset and update gates, respectively. $\widetilde{h}_t$ is the candidate output, whereas h_t is the output of the cell unit at time t. h_t also stores the information of the previous state. Note that we selected the GRU cells for the RNN due to their lower computing cost (Yang et al., 2020; see Section 3.3).

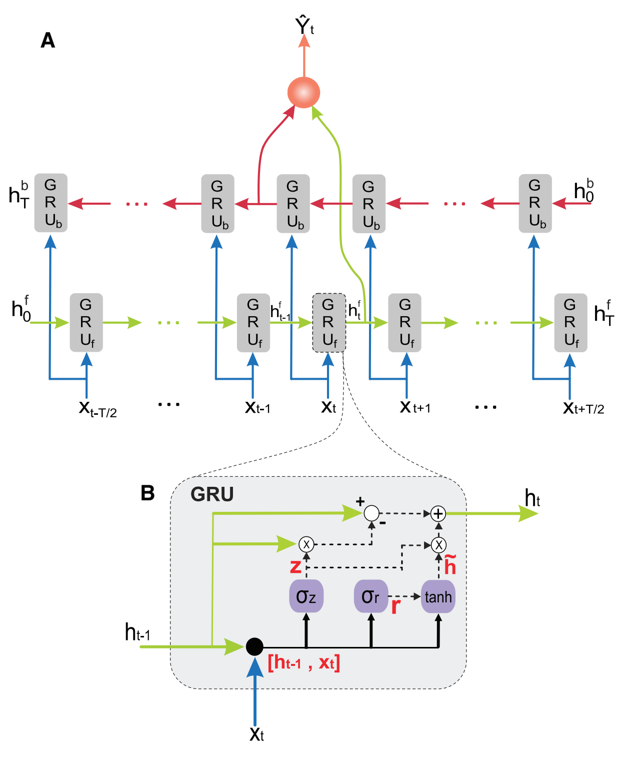

FIGURE 1
Structure of the proposed neural network. **(A)** Bidirectional recurrent neural network (BRNN) consisting of a backward (red) and a forward (green) recurrent neural network (RNN). **(B)** RNN cell solution; the gated recurrent unit (GRU).

We also incorporated an ANN readout layer to each GRU cell since cell and hidden states are the same (Figure 1). We assembled these GRU cells following a BRNN model (Graves et al., 2005). In our BRNN implementation, we used two independent RNNs to process the input sequence in both forward and backward directions. This approach allowed us to take into account both past and future contexts when predicting the output values. By using two separate RNNs, we could achieve improved performance compared to a unidirectional RNN. One RNN handled positive time direction (GRU$_f$), i.e., information from the past toward the current state, whereas the other RNN handled negative time direction (GRU$_b$), i.e., information from the future toward the current state (Schuster and Paliwal, 1997). We fed the model with target future states of the robot, i.e., following desired joint position and velocity values.

This BRNN model was later incorporated into three different configurations (Figure 2). (i) A non-parametric inverse dynamic configuration (NID) in which the BRNN received joint positions and velocities (seven positions and seven velocity values) as inputs, and it predicted the torque values for each joint. (ii) A semi-parametric inverse dynamic configuration (SID) in which the

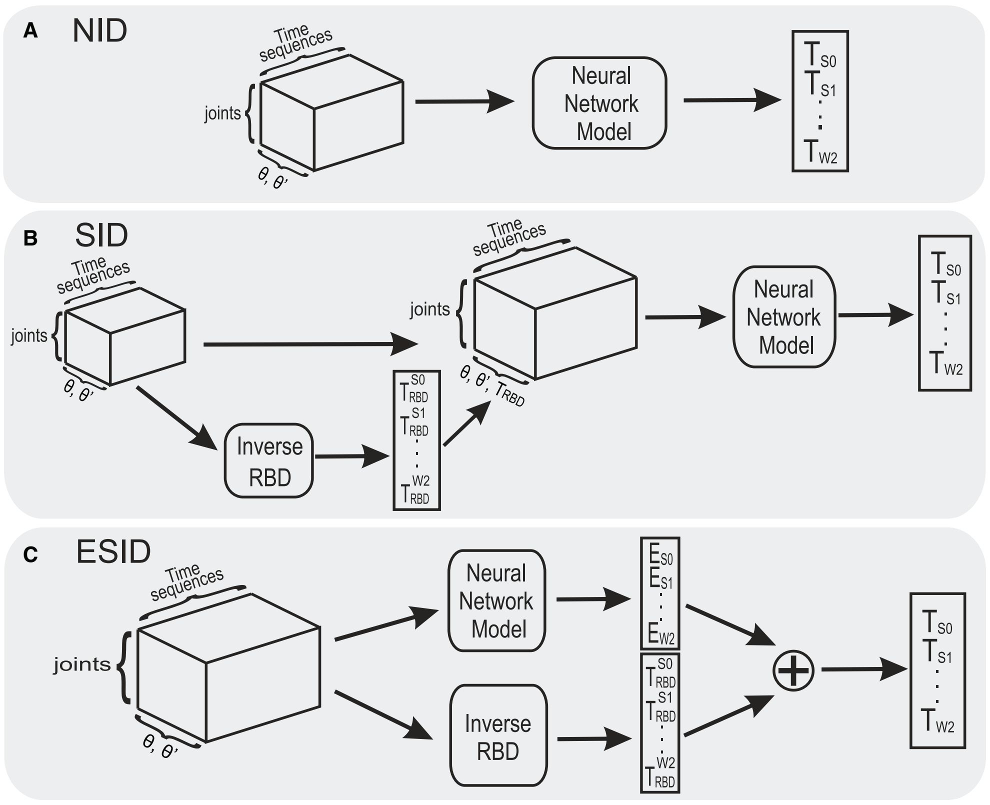

FIGURE 2
Three proposed inverse dynamic modeling configurations using a BRNN network. **(A)** Non-parametric configuration (NID). **(B)** Semi-parametric configuration (SID) receiving joint positions, velocities, and estimated torques from the inverse RBD model. **(C)** Error compensator inverse dynamic (ESID) configuration compensates for the dynamics that are not modeled by the inverse RBD model.

BRNN received joint positions, velocities, and the torque estimated by the rigid body dynamic (RBD) model (described by Equation 2) as inputs, and it predicted the torque values for each joint. (iii) An error compensator semi-parametric inverse dynamic configuration (ESID) in which the BRNN received joint positions and velocities as inputs and whose outputs $[\xi_{\text{ESID}}(q, \dot{q})]$ were added to the rigid-body dynamic model to calculate the total torque values of each joint (τ_{total}), as defined in

$$\tau_{\text{total}} = \tau_{\text{RBD}} + \xi_{\text{ESID}}(q, \dot{q}). \tag{8}$$

In Equation (8), τ_{RBD} represents the estimated rigid body dynamics of the robot, as described in Equation (2). The BRNN is trained to approximate the ξ_{ESID} term, which is equivalent to the ξ term in Equation (1) and encompasses all unmodeled dynamics not captured by Equation (2). The difference between the torque commanded and the torque estimated by the τ_{RBD} model is used as the ground truth for ξ_{ESID}.

The BRNN was implemented in Python using the Keras module running on TensorFlow. BRNN consisted of 64 hidden units that were trained with 32 batch sizes during 200 epochs. Adam Solver (Kingma and Ba, 2014) configured with a learning rate of 0.001 did optimize the loss function. The mean square error (MSE) metric was used as loss function for the actual and predicted torque values. The MSE metric is suitable for quantifying the difference between two sets of values when a small range of values is considered. Conversely, the mean absolute error (MAE) metric was used to compare the torque value predictions made by each proposed configuration at each joint. The MAE metric computes the average of the absolute difference between the actual and the predicted torque values (the actual values being those obtained from the dataset control loop, see Sections 2.4, 2.5 and Figure 3). The MAE and MSE metrics are described by

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(\tau_i - \hat{\tau}_i)^2 \text{ and} \tag{9}$$

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|\tau_i - \hat{\tau}_i|. \tag{10}$$

In which, τ_i and $\hat{\tau}_i$ are the actual torque value and predicted torque value, respectively. N is the total number of data.

We also used the coefficient of determination, r^2, which is defined as

$$r^2 = 1 - \frac{\sum_{i=1}^{N}(\tau_i - \hat{\tau}_i)^2}{\sum_{i=1}^{N}(\tau_i - \overline{\tau})^2} \tag{11}$$

where

$$\overline{\tau} = \frac{1}{N}\sum_{i=1}^{N}\tau_i. \tag{12}$$

r^2 is a single value ranging from 0 to 1. This metric was used to evaluate and compare the overall performance of the

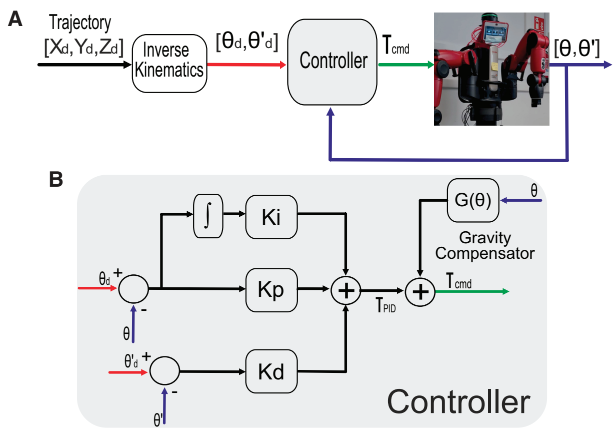

FIGURE 3
Schematic of the control system to obtain the dataset. **(A)** The general control system and **(B)** detailed PID controller plus gravity compensator.

three proposed configurations in predicting torque values for the seven joints.

2.4. Building the dataset: the torque control loop

Our BRNN demanded related torque vs. position/velocity data to be trained/tested. To that aim, we implemented a Baxter torque control scheme that allowed obtaining the joint commanded torque values. The torque-sensed values (τ_s) represent the measured torque values in the joints, while the torque-commanded values (τ_c) represent the desired torque values that the motors should exert on the joints. In the absence of collisions, τ_s and τ_c are correlated, but not identical. Since the robot's internal controller affects the actual torque applied to the joint, a dynamic behavior occurs between the commanded torque and the actual torque applied to the joint. This dynamic behavior is even more significant in cobots with a series of actuators (SEAs). Incorporating τ_c into the dynamic cobot model allows us to account for the dynamics of the cobot's internal controller, resulting in a more accurate representation of the cobot's behavior. In contrast, many data-based studies derive the dynamic cobot model from τ_s when they lack access to τ_c or only have position control, which may not capture the full dynamic behavior of the cobot. Torque-commanded values vs. robot state data provided more comprehensive robot information than the torque-sensed values used in other approaches (Liu et al., 2019). A more comprehensive robot dynamic dataset ultimately helps implement a more realistic inverse dynamic model.

The Robot Operating System (ROS) middleware was used to implement the torque control loop. A classical PID torque control plus gravity compensation was at the core of the control loop for trajectory tracking (Figure 3), thus obtaining actual positions, velocities, and commanded torque values per each desired trajectory (see Supplementary material). A 500-Hz sampling frequency was used for controlling, sensing, and data storing.

2.5. Building the dataset: trajectories tracked

The Baxter trajectory benchmark consisted of a set of random point-to-point movements for 1 min together with a set of closed periodic trajectories described by

$$\begin{aligned} x &= r \sin\left(\frac{2\pi}{T}t\right) + 0.56, \\ y &= r \cos\left(\frac{2\pi}{T}t\right) + 0.06, \textit{and} \\ z &= -0.22 \cos\left(\frac{2\pi}{T_z}t\right) + 0.18, \\ &\text{for } 0\text{ s} \leq t \leq 60\text{ s} \end{aligned} \tag{13}$$

and

$$\begin{aligned} x &= \left(r - \frac{0.01}{T_z}t\right) \sin\left(\frac{2\pi}{T}t\right) + 0.56, \\ y &= \left(r - \frac{0.01}{T_z}t\right) \cos\left(\frac{2\pi}{T}t\right) + 0.06, \textit{and} \\ z &= -0.22 \cos\left(\frac{2\pi}{T_z}t\right) + 0.18, \\ &\text{for } 0\text{ s} \leq t \leq 8T_z\text{ s}, \end{aligned} \tag{14}$$

instead of exclusively using a single periodic trajectory (Liu et al., 2019; Wang et al., 2020). These trajectories were selected to provide the greatest variety of information to the neural network (Kappler et al., 2017), i.e., covering as much workspace as possible at different speeds. To that aim, parameters r, T, and T_z varied accordingly (see Table 1). Each trajectory defined by (13) was executed for 1 min, whereas the three trajectories defined by (14) were executed during 80, 100, and 120 s, respectively, depending on the T_z value (see Table 1). The desired Cartesian space trajectories were converted to joint space using inverse kinematics (Coumans and Bai, 2016-2021); desired joint position and velocities were then fed to the PID controller, which generated the joint torque commands that resulted in actual joint positions and velocities (Figure 3). A continuous benchmark execution lasted 18 min (at a 500-Hz sampling frequency), which provided 540,000 data samples. Each data sample comprised position, velocity, and torque values per each Baxter joint. Note that the joint collisions, positions, and velocities constraints were accounted for.

Joint positions, velocities, and torque-commanded values were sampled at a frequency of 500 Hz and then stored for analysis. To create the learning sets, a cross-validation technique known as shuffle-split was used. This involved partitioning the total dataset into sub-sequences and shuffling them to create a training set (80% of samples) and a validation set (20% of samples). Normalization was performed on both the training and validation sets using the mean and standard deviation computed from the training set.

To facilitate the result replication and comparisons without needing to access the cobot, we made the dataset available at

TABLE 1 Trajectories tracked.

Trajectory	T (s)	r (m)	T_z (s)
Helical (13)	2.5	0.18	15.0
			12.5
			10.0
		0.15	15.0
			12.5
			10
	2.0	0.10	12.0
			10.0
			8.0
		0.05	12.0
			10.0
			8.0
Spiral (14)	2.5	0.18	10.0
			12.5
			15.0

T is the period on the x and y axes; $2r$ is the motion amplitude in the xy plane; T_z is the period on the z axis.

(see Supplementary material) https://github.com/EduardoRosLab/Baxter_Dynamic_Model.git.

2.6. Analyzing the GRU stability

Evaluating the stability of a system is crucial in any control problem. Input-State Stability (ISS) property of a GRU network (i.e., bounded inputs result in bounded NN states regardless of the initial condition; Limon et al., 2009) can be evaluated using a mathematical method proposed by Bonassi et al. (2021). For a GRU network to comply with ISS, the following condition must be satisfied:

$$v_{sb} < 1, \tag{15}$$

where

$$v_{sb} := \frac{1}{4}\left(||U_r||_\infty (||U_{\tilde{h}}||_\infty + \tilde{\sigma}_{\tilde{h}}) + \frac{1+\tilde{\phi}_r}{1-\tilde{\sigma}_z} ||U_z||_\infty \right), \tag{16}$$

and

$$\begin{aligned}
\tilde{\sigma}_{\tilde{h}} &= \sigma(||W_{\tilde{h}} \quad U_{\tilde{h}} \quad b_{\tilde{h}}||_\infty), \\
\tilde{\sigma}_z &= \sigma(||W_z \quad U_z \quad b_z||_\infty), and \\
\tilde{\phi}_r &= tanh(||W_r \quad U_r \quad b_r||_\infty).
\end{aligned}$$

We apply (15) as a constraint during the training procedure to make our BRNN with GRU cells comply with the ISS property. The learning stability imposition to our BRNN shall decrease the torque command precision but ensure learning stability during the BRNN learning descent curve (Bonassi et al., 2021). By imposing learning stability during the BRNN learning descent curve, the analysis shall result in more robust weight values that could mitigate online learning problems, such as improper updating of network weights or/and instability in the learning process, that could negatively impact the performance of the BRNN and cause damage to the robot.

3. Results

3.1. Optimal temporal window and network size for the BRNN

The BRNN required selecting the number of forward and backward steps used to make the torque value predictions, i.e., temporal window size. To find the optimal trade-off between computational cost (time-window size) and BRNN resolution, we run the NID configuration with different time periods ranging from 6 to 70 ms. We performed the learning three times per time-window size. Cross-validation (shuffle-split) was used to create each learning set: 80% of the samples for the training set and 20% of samples for the validation set.

The results (Figure 4) depicted a precision (MSE) vs. time-window length (ms) exponential behavior. We found that the MSE average and variance did not substantially differ from time-window sizes larger than 46 ms. Accordingly, we used a 50-ms time-window size to maintain MSE values low at a minimum computational cost.

To determine the optimal GRU number of the BRNN, we explored values of 32, 64, and 128 cells. The MSE values achieved in training, validation, and prediction for non-trained circular (Test 1) and a square (Test 2) trajectory were taken into account (see Figure 4). According to these results a dimension of 64 GRU units was selected, as it provides a higher generalization of the data.

3.2. Trajectory benchmark analysis

We trained the NID configuration using five different data subsets of trajectories defined within the benchmark (see Table 1) to quantify their capability of learning the inverse dynamic model from a minimum amount of data. We independently trained NID with (i) random trajectories only, (ii) helical trajectories only, (iii) helical and random trajectories, (iv) spiral trajectories only, and (v) spiral and random trajectories (Figure 5A).

We then used a non-trained circular and a square trajectory to test performance after training with each different subsets (i–v), and we also ran a one-way ANOVA with *post-hoc* Tukey analysis to compare the obtained results, i.e., the amount of information that each trajectory data subset brings to the inverse dynamic model.

Interestingly, we found non-significant statistical differences when performing the circular test trajectory for subsets (iii), (iv), and (v) compared to the full dataset. However, when performing the square test trajectory, whose shape substantially differed from any trajectory used during training, only subsets (iii) and (v) carried non-significant statistical differences compared to the full dataset (Figure 5B). We found that the random trajectories were instrumental in providing NID with generalization abilities for

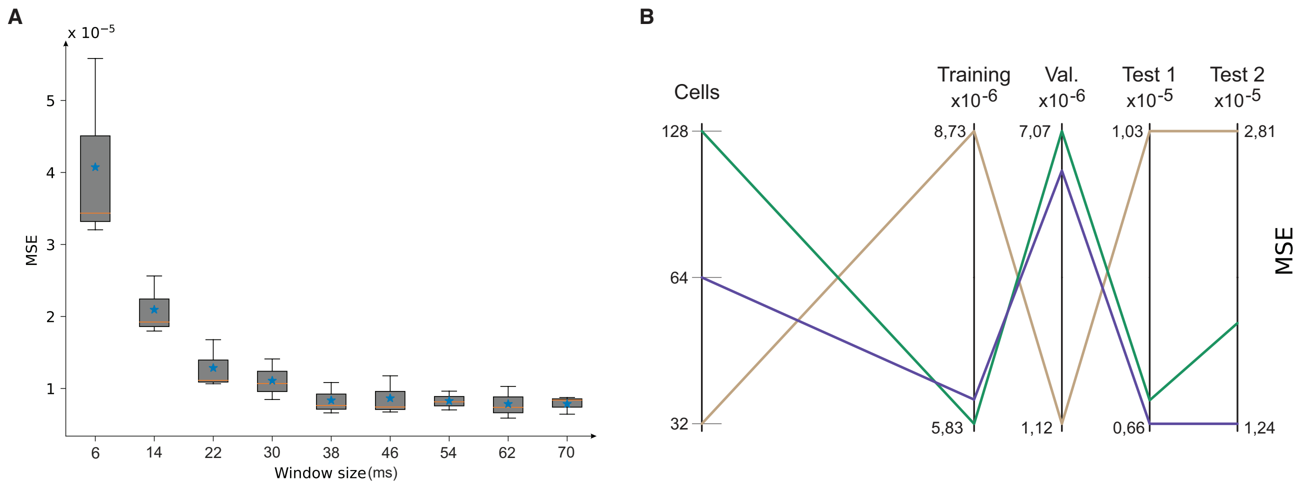

FIGURE 4
(A) Mean Squared Error (MSE) evolution after the NID configuration underwent training using different BRNN time-window sizes. **(B)** Network dimensionality analysis; the number of GRUs within the BRNN. The MSE indicates that for a number of 64 GRUs, our BRNN reaches lesser MSE values during the test phase; Test 1 was a circle trajectory whereas Test 2 was a squared trajectory. This dimensionality analysis indicates that a higher number of GRUs (128) leads to a lack of generalization, while a lower number (32) leads to a lack of precision in learning.

obtaining a precise inverse dynamic model, i.e., MAE significantly decreased (Figure 5B). The trajectory data subset (v) was chosen as the training dataset since it matched performance with the full dataset but using a lesser amount of data (180,000 vs. 540,000; see red bars in Figure 5B).

3.3. Precision of NID, SID, and ESID configurations

Each configuration, namely NID, SID, and ESID (as described in Section 2), underwent independent training for three runs. Cross-validation was employed to generate training/validation sets for each run. The data set used for this purpose was obtained from the trajectory subset (v) mentioned earlier (Figure 6). We found that, in terms of error accuracy, these three configurations behaved similarly during the training phase. SID and ESID started the training from lower MSE values than NID due to the inverse RBD model information provided to the BRNN. However, they all converged to the same MSE value in 50 iterations (Figure 6). We stopped learning after 200 epochs to prevent the three configurations from overfitting the training dataset, thus preserving the configuration capacity for generalization.

We then used the MAE metric to compare the performance per cobot joint of the aforementioned configurations, i.e., NID (see Supplementary Figure 1), SID, and ESID, compared to the performance of the rigid body dynamic model of the cobot described (2). We found that any of the three configurations (Figure 7) largely outperformed the manufacturer RBD model (MAE values: NID 0.15 $\pm$ 0.02 Nm, SID 0.12 $\pm$ 0.02 Nm, and ESID 0.12 $\pm$ 0.02 Nm, RBD 3.83 Nm). The performance improvement was particularly remarkable at the S1 joint (shoulder). Baxter holds an external spring (elastic passive component) that facilitates the shoulder to fix position (Fitzgerald, 2013). This external spring is not even considered within the RBD model, which would explain much of this significant difference.

Importantly, we found no statistical difference among the three configurations (NID, SID, and ESID) when comparing performance (MAE) by ANOVA tests. We must, however, also highlight that the NID configuration did not use prior information provided by the RBD model, which is often not available (Smith and Mistry, 2020; Huang et al., 2021).

For comparative purposes, our NID configuration was also implemented using LSTM cell units. We found similar validation errors for LSTM and GRU cell implementations (i.e., MSE = $6.40 * 10^{-6}$ vs. MSE = $6.27 * 10^{-6}$, respectively), but a faster processing speed using the GRU cell implementation (i.e., 5,600 sample set computation time takes 640 ms LSTM vs. 230 ms GRU). In light of these results, and being RT computing a limiting factor in robotics, we finally chose to use the GRU cell implementation. To compare the performance of our BRNN architecture, we also tested two other architectures: a multilayer perceptron (MLP) with three hidden layers and 64-128-64 neurons and a unidirectional RNN with 64 GRU cells. Both RNNs performed better than MLP. This suggests that using information from both past and future time steps is important for learning the inverse cobot dynamics. The results are summarized in Table 2.

3.4. Evaluating NID, SID, and ESID configurations: generalization of the inverse dynamic model

To assess the generalization capabilities of the NID, SID, and ESID configurations, their inverse dynamics models faced a series of trajectories (test set) neither used during training nor validation: (i) circular trajectories on the XY and XZ plane, (ii) a helical trajectory, and (iii) a square trajectory (see Supplementary material). An inverse dynamic model, which can generalize, shall predict precise torque values for any trajectory regardless the training/validation trajectory used.

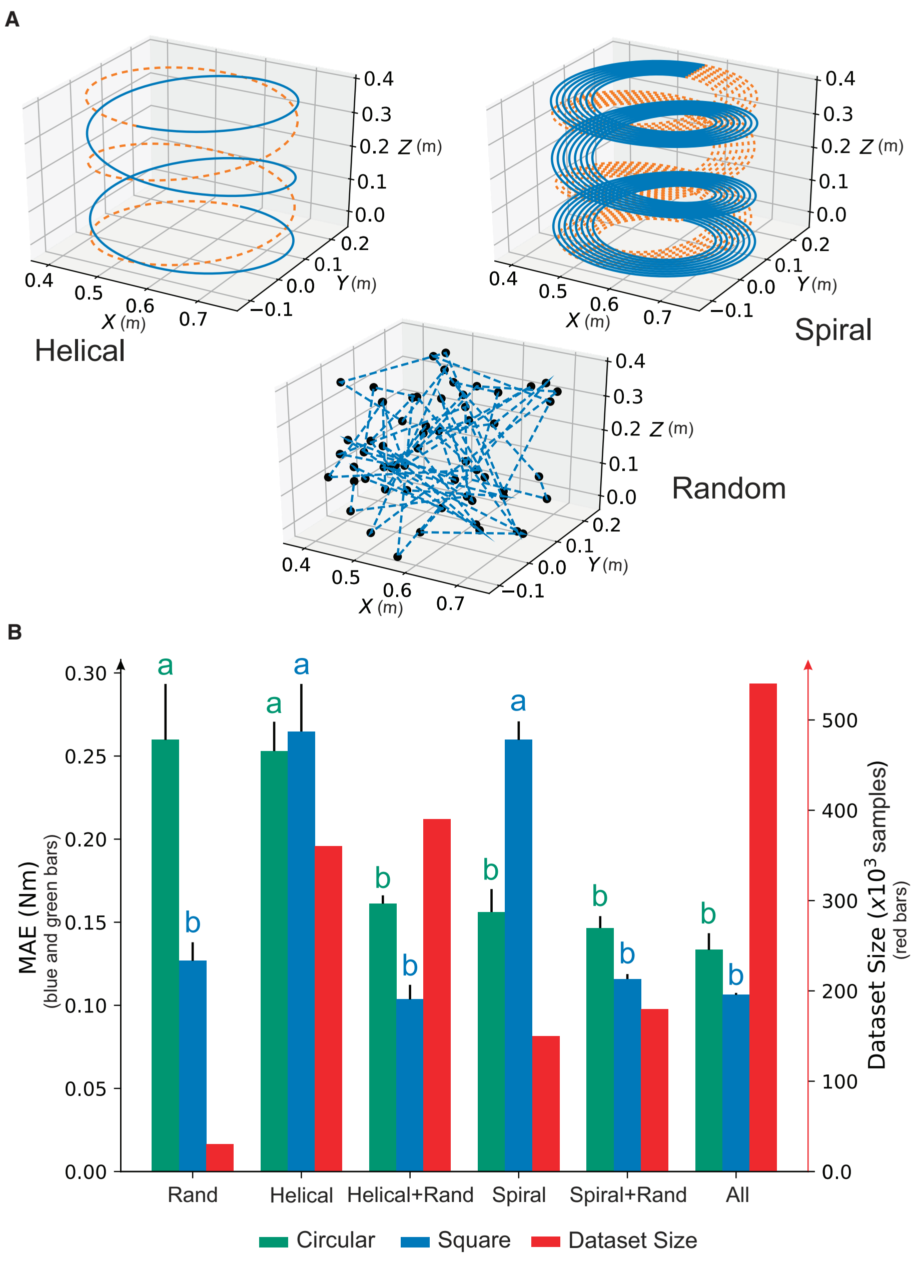

FIGURE 5
Selecting the training data subset. **(A)** Illustration of the trajectory set to be used **(B)**. A trade-off between precision and data subset size using the NID configuration following either circular or square non-trained trajectories. A different letter represents statistically significant differences found (ρ <0.05) among the performance of the trajectory data subsets.

To analyze and quantify how good each inverse dynamic model was at learning from the given dataset and applying the learned information to new trajectories neither used during training nor validation, we obtained their MAE values per joint and their r^2 values. MAE allowed us to compare individual joint precision (see Supplementary material), whereas r^2 provided for the correlation between the predicted and actual torque values in a single value measurement.

We found that the generalization ability for the NID, SID, and ESID configurations remained similar. Importantly, the more alike the new non-trained trajectories (test set) were to the trained trajectories, the higher r^2 is obtained (0.92 and 0.90, see Table 2) and vice versa (0.82 and 0.76, see Table 2). Despite NID, SID, and ESID comparable performance, no prior knowledge of analytical dynamics for NID configuration is needed.

3.5. Case study: real-time, real-world, feed-forward control

Using the inverse dynamic robot model as a controller is of common use (Jordan and Rumelhart, 1992; Stogiannos et al., 2018). Here, we propose our NID configuration as a feedforward

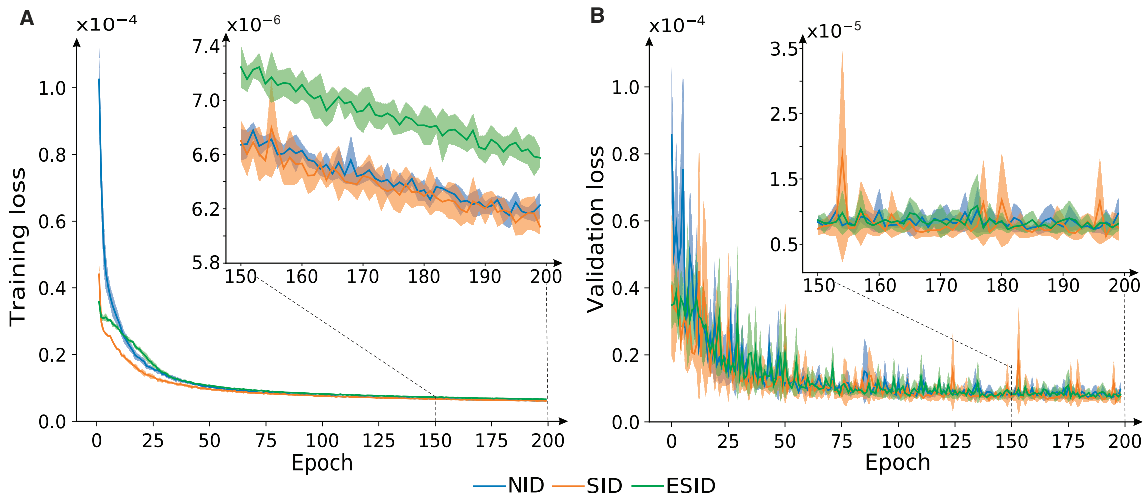

FIGURE 6
Evolution of the loss function for NID, SID, and ESID configurations. **(A)** Training and **(B)** test datasets. **(A, B)** Zoom-in depicts how overfitting is prevented.

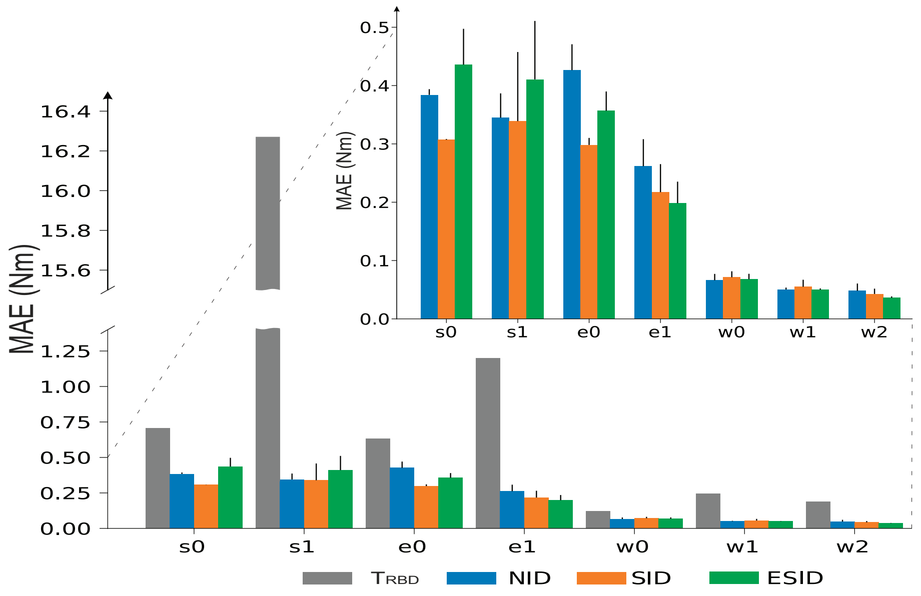

FIGURE 7
Comparison among the NID, SID, and ESID proposed configurations and the analytical rigid body dynamics model.

TABLE 2 Inverse dynamic model torque prediction over different test trajectories.

	MAE						r^2		
Path	RBD	MLP	RNN	NID	SID	ESID	NID	SID	ESID
Circular XY	3.83	1.73	0.48	**0.15**	**0.15**	**0.15**	**0.92**	**0.92**	**0.92**
Helical	3.27	1.62	0.43	0.22	**0.19**	0.22	0.89	0.87	**0.90**
Circular XZ	3.17	1.91	0.32	0.17	**0.15**	0.16	**0.82**	0.79	0.80
Square	4.37	2.25	0.51	0.27	**0.21**	0.23	0.72	**0.76**	**0.76**

The bold values indicate the best results.

controller working conjointly with a feedback PD controller (Figure 8). We developed this case study under the assumption that a better dynamic cobot model shall require less action from the feedback control loop (PD). With this implementation, we could indirectly verify how closely the learned inverse dynamic model matched the real robot. It is important to note that this case study's main aim was not outperforming a PD control but to demonstrate one of the many potential applications of the inverse dynamic model of a collaborative robot. NID configuration is based on a worst-case control scenario compared to SID and ESID since the RBD model is unknown and thus the entire robot dynamics shall be learned. The NID configuration represents not just a theoretical RNN implementation but rather it aims at a robotic real-world RNN control application. Once we trained our NID configuration, we used it as a real feedforward controller (Figure 8). NID estimated the torque values to be provided per joint to follow a desired trajectory (circular trajectory used as reference). The input data to the NID were actual and desired robot states (positions and velocities) of the robot joints.

Four scenarios were proposed to test the capability of our control system to track a circular trajectory as reference:

1. *Stability analysis:* Our NID configuration was trained excluding/including the ISS constraint (15) within the loss function as described by Bonassi et al. (2021). Predictably, the NID accuracy performance decreased when imposing the ISS stability criteria (see Table 3, NID vs. NIDsb). In this case study example, we prioritized NID accuracy and accompanied the NID controller in a feedforward loop with a PD in a feedback loop to compensate for small NID torque value estimation errors and to ensure closed-loop system stability (Hu et al., 2021). Note that the stability depends on the PD closed-loop characteristics rather than on the feedforward GRU term that only provides for precomputed torque values.
2. *Torque estimation accuracy:* We performed a circular trajectory of radius 0.15 m in the XY plane using our proposed controller (NID + PD), as shown in Figures 9A, B. The resulting mean absolute error (MAE) of the end-effector position for NID + PD (7.0 ∗ 10−3 m) was significantly lower than the MAE of the end-effector position achieved by the robot's default factory position controller (79.0∗10−3 m; see Table 4). Our analysis revealed that the NID controller executed the majority of the control action, while the contribution of the PD controller remained residual (see Figure 9C). This suggests that the torque estimated by the NID was closely aligned with the torque required to track the reference trajectory (see Supplementary material).
3. *Compliance and response to disturbances:* We programmed a total locking of all joints during 250 ms at time Δt_1 (Figure 9D). During the locking period, the NID + PD control did not execute high torque values to compensate for the error, as a PD controller without the feedforward component would do. Thus, compliance was improved due to the lower energy at stake (Figure 9D). We also found the NID + PD control to better deal with the disturbances. After the locking period, NID + PD control almost instantaneously resumed tracking the reference trajectory without a significant transient converging stage (Figure 9D; see Supplementary material).
4. *NID Resilience in feedforward control:* We also tested the capability of the NID to track the trajectory in open-loop torque control. To do so, we switched off the PD feedback control action ($\tau_{PD} = 0$) during 500 ms (Δt_2) and confirmed that NID could seamlessly track the trajectory (see Figure 9E) in the absence of active feedback control (see Supplementary Figure 2).

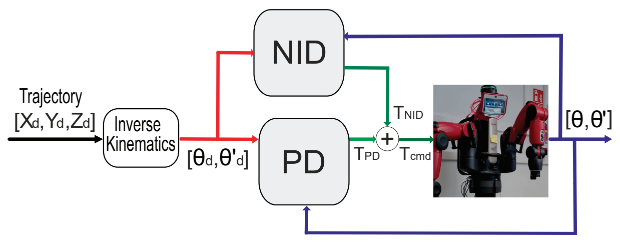

FIGURE 8
NID implemented as a feedforward controller together with a PD feedback controller.

4. Discussion

The presented ML approach identified the inverse dynamic model of a cobot (Baxter) using a BRNN with GRUs at the core. These BRNNs, in turn, were adapted and integrated into NID, SID, and ESID configurations. It is noteworthy that, the GRUs

TABLE 4 Performance on a circular trajectory of the default factory controller and the NID + PD controller.

		Default factory	NID + PD
Joint MAE (rad)	s0	0.112	**0.033**
	s1	0.034	**0.005**
	e0	0.027	**0.007**
	e1	0.151	0. **024**
	w0	**0.014**	0.021
	w1	0.042	**0.035**
	w2	**0.023**	0.034
End-effector MAE (m)		0.079	**0.007**

The bold values indicate the best results.

TABLE 3 Comparison of performance between NID trained with and without considering the ISS property.

	ν_{sb} [GRU$_b$, GRU$_f$] (16)	ISS condition (15)	MSE training	MSE validation
NID	[inf, inf]	No	$6.17 * 10^{-6}$	$6.27 * 10^{-6}$
NID$_{sd}$	[0.15, 0.17]	Yes	$1.83 * 10^{-5}$	$2.26 * 10^{-5}$

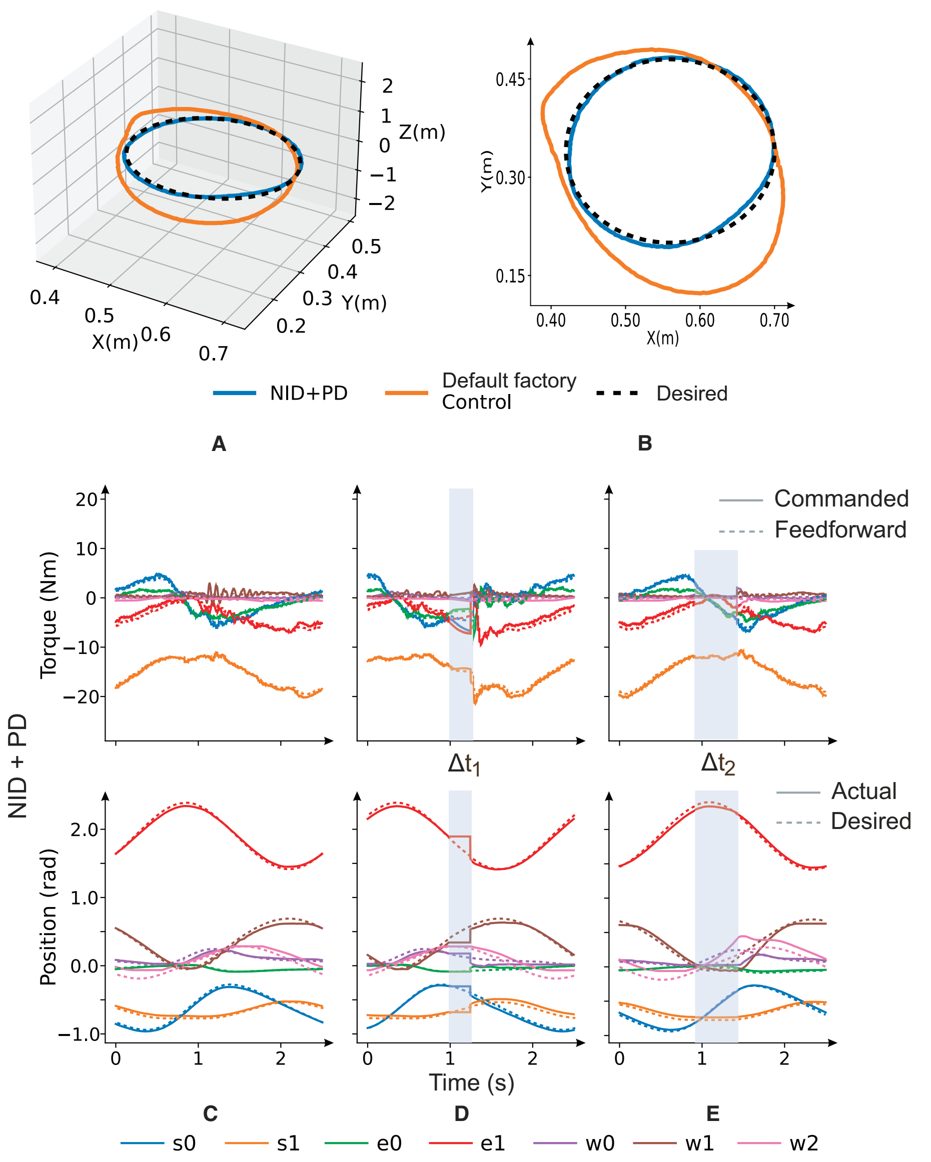

FIGURE 9
Testing the NID + PD control performance using a circular trajectory as reference. Default factory control vs. feedforward NID + PD torque control **(A, B)**. Testing torque value estimation accuracy and PD control contribution are negligible **(C)**. Testing compliance and response to disturbances **(D)**, NID + PD torque response during a 250-ms total locking of all robot joints $\Delta t_1 = 250$ ms. Testing NID resilience **(E)**, feedforward control temporally without feedback control during $\Delta t_2 = 500$ ms.

eliminated the vanishing gradient problem in the temporal input values by keeping the relevant temporal information and passing it down to the next time steps of the BRNN network (Yang et al., 2020). We found that the larger the sliding time window, which was partially containing and feeding the GRUs with temporal input values, the better the performance of Baxter's inverse dynamic model. However, the computational load also increased with the size of the time window, which ultimately jeopardized Baxter's Real Time (RT) operation. The temporal window size trade-off found at 50 ms ensured Baxter's RT operation with no significant loss of accuracy with respect to longer time windows (Francis, 1995). This is remarkable, in contrast to other studies with large temporal windows (Liu et al., 2019; Wang et al., 2020) accounting for precision only and sidestepping RT robot operation. The proposed BRNN algorithms are designed to identify the inverse dynamics of a robot, treating the robot as a time-invariant system. To account for real-time performance effects, such as backlash or non-linearities from friction or elastic components, offline learning of the inverse dynamics is performed using data recorded during the execution of trajectories on a real robot. This incorporation of real-time effects leads to more accurate learned dynamics. Selecting the BRNN sliding time window length was essential, but also the data contained in it. We considered the question of selecting those representative trajectory data that would contribute most effectively to the BRNN's generalization capabilities.

Needless to say, there are endless possibilities to explore Baxter's workspace, hence selecting those trajectories that better reveal its non-linear dynamics was pivotal. We found that combining random trajectories together with cyclic ones (modulating their magnitude and frequency, see Section 2) while exploring Baxter's workspace revealed better its non-linear dynamics rather than using cyclic or random exploration only, i.e., data diversity vs. data quantity. The use of this representative training/validation dataset endowed the NID, SID, and ESID configurations with similar generalization capabilities when predicting torque values from their corresponding Baxter's inverse dynamic models. As assumed by Kappler et al. (2017), we verified the superior performance of those configurations using rigid-bodied analytic dynamic modeling supporting BRNN (SID and ESID) when predicting torque values under dissimilar trajectories to the original training set. However, the differences found were not statistically significant among the three configurations during validation. Importantly, NID, unlike SID and ESID configurations, operated with no prior knowledge of Baxter's dynamics, which makes the NID configuration more suitable for robotic applications in which the analytic dynamic model of the robotic agent is either mathematically intractable or not available, i.e., soft robots and elastic robots. Conversely, SID and ESID could benefit from employing the analytical dynamic model when a cobot faces an unexplored working space. Finally, we tested NID configuration as a proof of concept in a real-time, real-control scenario, i.e., NID as a feedforward controller together with a PD controller that helped compensate for small torque estimation errors. Interestingly, model-free control approaches can also be used to control robots with complex modeling dynamics (Bian et al., 2020). These approaches can provide a controller with excellent performance for a specific task, even without an accurate model of the robot (Tutsoy et al., 2023). However, in these methods, the controller and the dynamic model usually operate as a whole, which might limit the independent use of the obtained dynamic model.

The NID configuration, as demonstrated in Sections 3.3 and 3.4, has been shown to have high torque estimation performance. This makes it a good candidate for real-time control applications where the inverse dynamics of the robot are the main actions, as implemented in Section 3.5. In real-time scenarios, there is a risk of joint locking, collisions, errors, or delays in sensing the robot's state. However, the proposed implementation deals with these situations by maintaining the real robot close to the desired trajectory with compliant features, as described in Section 3.5. It should be noted, though, that caution is required when working with trajectories outside the workspace of the training-validation dataset. Since NID is a non-parametric algorithm, its performance in data extrapolation cannot be guaranteed (Mozian et al., 2020). If the real robot is going to work outside the explored space, it would be more advisable to use a semi-parametric inverse dynamic configuration, such as those presented in Section 2.5. We confirmed that the NID + PD controller outperformed the robot's default factory position controller in terms of path tracking accuracy. The NID also helped improve the PD controller in terms of (i) compliance and response to disturbances and (ii) resilience to open-loop control at short time intervals. Note that, as aforementioned, we implemented NID as a worst-case scenario feedforward robot controller. Given a RBD robot model, SID and ESID could also be implemented. Improved torque estimation is one of the main control goals when operating cobots in unstructured scenarios involving human-robot interaction (HRI). Not only the energy at stake is diminished preventing damage in case of collision but also the improved precision and response to disturbances help avoid those collisions (Mohajerin and Waslander, 2019). We also verified the NID resilience by depriving temporally the control architecture of any active feedback information during 500 ms. NID controller was able to keep track of the reference trajectory with precision. The NID capacity of operating with no feedback could be useful in remote control, cloud, or fog computing scenarios, presenting significant non-deterministic latencies in the transference of sensed information. NID could indeed predict the commanded torque values while temporally blocking the sensorimotor feedback, i.e., during open loop operation.

It is important to note that the mechanical and electrical components of a robot may undergo changes in their parameters over time due to factors, such as environmental conditions or wear, resulting in dynamic changes in the cobot (Camoriano et al., 2016). Significant differences between the torque estimated by the learned inverse dynamic model and the torque sensed at the joints may be indicative of such changes in the parameters of the real robot. In such cases, the learned offline dynamic cobot model would no longer be effective for controlling the cobot, and retraining would be necessary after cobot maintenance or repair. There are online learning strategies available for training a BRNN in real-time applications, if needed. However, training the Bidirectional Recurrent Neural Network (BRNN) in real time can be challenging due to multiple factors. These factors include the risk of fatal errors that may occur if the network weights are updated improperly, as well as the potential for a delayed control loop if the computational cost of training the model on real-time data is significant. Furthermore, if the goal is to identify sudden changes in the robot's dynamics caused by external factors, it may be more appropriate to leave the dynamic model invariant during the task execution to prevent any potential issues. Since data-driven approaches have no analytical guarantee of performance, they may be difficult to certify for safety critical applications. This can be mitigated by integrating a PD controller into the loop to account for unexpected uncertainties. Furthermore, if deviations from the expected performance are encountered (for instance detectable by measuring the error or the PD output activity), safety actions, such as interlock or alarm signals, shall be integrated. In future studies, the performance of the PD controller used in the feedforward loop in this study could be compared with other adaptive control strategies for cobots, such as the ones proposed by Pan et al. (2018). Additionally, the possibility of integrating our NID configuration with more robust control architectures or adaptive control strategies could be explored to assess how the balance between identification techniques and control techniques would be.

Data availability statement

The original contributions presented in the study are included in the article/Supplementary material, further inquiries can be directed to the corresponding author.

Author contributions

BV-V and NL conceived the article's initial idea, designed, and modeled and implemented the set-up experimentation. They also prepared the figures, drafted the manuscript, reviewed the manuscript, and approved the final version. IA drafted the manuscript, reviewed the manuscript, and approved the final version. ER conceived the article's initial idea, drafted the manuscript, reviewed the manuscript, and approved the final version.

Funding

This study was supported by the EU with the IMOCOe4.0 [EU H2020RIA-101007311] project and by Spanish national funding [PCI2021-121925]. It was also funded by the INTSENSO [MICINN-FEDER-PID2019-109991GB-I00] and INTARE (TED2021-131466B-I00) projects funded by MCIN/AEI/10.13039/501100011033 and EU NextGenerationEU/PRTR to ER. This study was also supported by the SPIKEAGE [MICINN629PID2020-113422GAI00] and DLROB [TED2021 131294B-I00] projects funded by the Spanish Ministry of Science and Innovation MCIN/AEI/10.13039/501100011033 and European Union NextGenerationEU/PRTR to NL. It was also supported by the Colombian Ministry of Science, Technology, and Innovation by the Grant [call 860] to BV-V.

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

Supplementary material

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/fnbot.2023.1166911/full#supplementary-material

References

Ata, A., Elkhoga, S., Shalaby, M., and Asfour, S. (1996). Causal inverse dynamics of a flexible hub-arm system through Liapunov's second method. *Robotica* 14, 381–389. doi: 10.1017/S0263574700019779

Bian, T., Wolpert, D. M., and Jiang, Z.-P. (2020). Model-free robust optimal feedback mechanisms of biological motor control. *Neural Comput.* 32, 562–595. doi: 10.1162/neco_a_01260

Bicchi, A., Peshkin, M. A., and Colgate, J. E. (2008). "Safety for physical human-robot interaction," in *Springer Handbook of Robotics*, eds B. Siciliano and O. Khatib (Berlin; Heidelberg: Springer Berlin Heidelberg), 1335–1348. doi: 10.1007/978-3-540-30301-5_58

Bonassi, F., Farina, M., and Scattolini, R. (2021). On the stability properties of gated recurrent units neural networks. *Syst. Control Lett.* 157:105049. doi: 10.1016/j.sysconle.2021.105049

Brown, R., Schneider, S., and Mulligan, M. (1992). Analysis of algorithms for velocity estimation from discrete position versus time data. *IEEE Trans. Indus. Electron.* 39, 11–19. doi: 10.1109/41.121906

Calanca, A., Capisani, L. M., Ferrara, A., and Magnani, L. (2011). MIMO closed loop identification of an industrial robot. *IEEE Trans. Control Syst. Technol.* 19, 1214–1224. doi: 10.1109/TCST.2010.2077294

Çallar, T.-C., and Böttger, S. (2023). Hybrid learning of time-series inverse dynamics models for locally isotropic robot motion. *IEEE Robot. Autom. Lett.* 8, 1061–1068. doi: 10.1109/LRA.2022.3222951

Camoriano, R., Traversaro, S., Rosasco, L., Metta, G., and Nori, F. (2016). "Incremental semiparametric inverse dynamics learning," in *2016 IEEE International Conference on Robotics and Automation (ICRA)* (Stockholm), 544–550. doi: 10.1109/ICRA.2016.7487177

Chen, S., and Wen, J. T. (2019). "Neural-learning trajectory tracking control of flexible-joint robot manipulators with unknown dynamics," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (Macau), 128–135. doi: 10.1109/IROS40897.2019.8968608

Chung, J., Gulcehre, C., and Cho, K. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv Preprint. arXiv:1412.3555.*

Coumans, E., and Bai, Y. (2016-2021). *Pybullet, A Python Module for Physics Simulation for Games, Robotics and Machine Learning.* Available online at: http://pybullet.org

Fitzgerald, C. (2013). "Developing baxter," in *IEEE Conference on Technologies for Practical Robot Applications, TePRA* (Woburn, MA). doi: 10.1109/TePRA.2013.6556344

Francis, T. C. A. (1995). *Optimal Sampled-Data Control Systems.* London: Springer.

Giuliani, M., Lenz, C., Müller, T., Rickert, M., and Knoll, A. (2010). Design principles for safety in human-robot interaction. *Int. J. Soc. Robot.* 2, 253–274. doi: 10.1007/s12369-010-0052-0

Graves, A., Fernández, S., and Schmidhuber, J. (2005). "Bidirectional LSTM networks for improved phoneme classification and recognition," in *Artificial Neural Networks: Formal Models and Their Applications-ICANN 2005*, eds W. Duch, J. Kacprzyk, E. Oja, and S. Zadrożny (Berlin; Heidelberg: Springer Berlin Heidelberg), 799–804. doi: 10.1007/11550907_126

Hu, C., Ou, T., Zhu, Y., and Zhu, L. (2021). Gru-type larc strategy for precision motion control with accurate tracking error prediction. *IEEE Trans. Indus. Electron.* 68, 812–820. doi: 10.1109/TIE.2020.2991997

Huang, S., Cheng, J., Zhang, J., Zhu, Z., Zhou, H., Li, F., and Zhou, X. (2021). Robust estimation for an extended dynamic parameter set of serial manipulators and unmodeled dynamics compensation. *IEEE/ASME Trans. Mechatron.* 4435, 1–11. doi: 10.1109/TMECH.2021.3076519

IFR (2021). *IFR Presents World Robotics 2021 Reports.* Technical report, International Federation of Robotics, Frankfurt.

Jin, J., Zhao, L., Chen, L., and Chen, W. (2022). A robust zeroing neural network and its applications to dynamic complex matrix equation solving and robotic manipulator trajectory tracking. *Front. Neurorobot.* 16:1065256. doi: 10.3389/fnbot.2022.1065256

Jordan, M. I., and Rumelhart, D. E. (1992). Forward models: supervised learning with a distal teacher. *Cogn. Sci.* 16, 307–354. doi: 10.1207/s15516709cog1603_1

Kappler, D., Meier, F., Ratliff, N., and Schaal, S. (2017). "A new data source for inverse dynamics learning," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (Vancouver, BC), 4723–4730. doi: 10.1109/IROS.2017.8206345

Kingma, D. P., and Ba, J. (2014). "Adam: A Method for Stochastic Optimization," in *Proceedings of the 3rd International Conference on Learning Representations, San Diego* (San Diego, CA).

Kwon, D.-S., and Book, W. J. (1994). A time-domain inverse dynamic tracking control of a single-link flexible manipulator. *J. Dyn. Syst. Measure. Control* 116, 193–200. doi: 10.1115/1.2899210

Lee, C., Kwak, S., Kwak, J., and Oh, S. (2017). Generalization of Series Elastic Actuator configurations and dynamic behavior comparison. *Actuators* 6:26. doi: 10.3390/act6030026

Li, Z., and Li, S. (2021). Neural network model-based control for manipulator: An autoencoder perspective. *IEEE Trans. Neur. Netw. Learn. Sys.* 34, 2854–2868. doi: 10.1109/TNNLS.2021.3109953

Limon, D., Alamo, T., Raimondo, D. M., de la Pe na, D. M., Bravo, J. M., Ferramosca, A., et al. (2009). *Input-to-State Stability: A Unifying Framework for Robust Model Predictive Control*. Berlin; Heidelberg: Springer Berlin Heidelberg, 1–26. doi: 10.1007/978-3-642-01094-1_1

Liu, C., Wen, G., Zhao, Z., and Sedaghati, R. (2021). Neural-network-based sliding-mode control of an uncertain robot using dynamic model approximated switching gain. *IEEE Trans. Cybern.* 51, 2339–2346. doi: 10.1109/TCYB.2020.2978003

Liu, N., Li, L., Hao, B., Yang, L., Hu, T., Xue, T., et al. (2019). Modeling and simulation of robot inverse dynamics using LSTM-based deep learning algorithm for smart cities and factories. *IEEE Access* 7, 173989–173998. doi: 10.1109/ACCESS.2019.2957019

Liu, N., Li, L., Hao, B., Yang, L., Hu, T., Xue, T., et al. (2020). Semiparametric deep learning manipulator inverse dynamics modeling method for smart city and industrial applications. *Complexity* 2020:9053715. doi: 10.1155/2020/9053715

Ljung, L., Andersson, C., Tiels, K., and Schön, T. B. (2020). Deep learning and system identification. *IFAC-PapersOnLine* 53, 1175–1181. doi: 10.1016/j.ifacol.2020.12.1329

Luh, J. Y. S., Walker, M. W., and Paul, R. P. C. (1980). On-line computational scheme for mechanical manipulators. *J. Dyn. Syst. Measure. Control* 102, 69–76. doi: 10.1115/1.3149599

Madsen, E., Rosenlund, O. S., Brandt, D., and Zhang, X. (2020). Comprehensive modeling and identification of nonlinear joint dynamics for collaborative industrial robot manipulators. *Control Eng. Pract.* 101:104462. doi: 10.1016/j.conengprac.2020.104462

Mohajerin, N., and Waslander, S. L. (2019). Multistep prediction of dynamic systems with recurrent neural networks. *IEEE Trans. Neural Netw. Learn. Syst.* 30, 3370–3383. doi: 10.1109/TNNLS.2019.2891257

Mozian, M., Higuera, J. C. G., Meger, D., and Dudek, G. (2020). Learning domain randomization distributions for training robust locomotion policies. 6112–6117. doi: 10.1109/IROS45743.2020.9341019

Mukhopadhyay, R., Chaki, R., Sutradhar, A., and Chattopadhyay, P. (2019). "Model learning for robotic manipulators Usingrecurrent neural networks," in *Proceedings of the TENCON 2019: Technology, Knowledge, and Society* (Kochi), 17–20. doi: 10.1109/TENCON.2019.8929622

Nguyen-Tuong, D., and Peters, J. (2008). "Local Gaussian process regression for real-time model-based robot control," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems* (Nice), 380–385. doi: 10.1109/IROS.2008.4650850

Pan, Y., Wang, H., Li, X., and Yu, H. (2018). Adaptive command-filtered backstepping control of robot arms with compliant actuators. *IEEE Trans. Control Syst. Technol.* 26, 1149–1156. doi: 10.1109/TCST.2017.2695600

Polydoros, A. S., Nalpantidis, L., and Kruger, V. (2015). *Real-Time Deep Learning of Robotic Manipulator Inverse Dynamics*. Institute of Electrical and Electronics Engineers Inc., 3442–3448. doi: 10.1109/IROS.2015.7353857

Pratt, G., and Williamson, M. (1995). "Series elastic actuators," in *Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots, Vol. 1* (Pittsburgh, PA), 399–406. doi: 10.1109/IROS.1995.525827

Rueckert, E., Nakatenus, M., Tosatto, S., and Peters, J. (2017). Learning inverse dynamics models in o(n) time with LSTM networks. 811–816. doi: 10.1109/HUMANOIDS.2017.8246965

Schüssler, M., Münker, T., and Nelles, O. (2019). Deep recurrent neural networks for nonlinear system identification, in *2019 IEEE Symposium Series on Computational Intelligence (SSCI)* (Xiamen, China), 448–454. doi: 10.1109/SSCI44817.2019.9003133

Schuster, M., and Paliwal, K. (1997). Bidirectional recurrent neural networks. *IEEE Trans. Signal Process.* 45, 2673–2681. doi: 10.1109/78.650093

Smith, J., and Mistry, M. (2020). Online simultaneous semi-parametric dynamics model learning. *IEEE Robot. Autom. Lett.* 5, 2039–2046. doi: 10.1109/LRA.2020.2970987

Stogiannos, M., Alexandridis, A., and Sarimveis, H. (2018). Model predictive control for systems with fast dynamics using inverse neural models. *ISA Trans.* 72, 161–177. doi: 10.1016/j.isatra.2017.09.016

Swevers, J., Verdonck, W., and De Schutter, J. (2007). Dynamic model identification for industrial robots. *IEEE Control Syst. Mag.* 27, 58–71. doi: 10.1109/MCS.2007.904659

Tutsoy, O., Barkana, D. E., and Balikci, K. (2023). A novel exploration-exploitation-based adaptive law for intelligent model-free control approaches. *IEEE Trans. Cybern.* 53, 329–337. doi: 10.1109/TCYB.2021.3091680

Uribarri, G., and Mindlin, G. B. (2022). Dynamical time series embeddings in recurrent neural networks. *Chaos Solitons Fract.* 154:111612. doi: 10.1016/j.chaos.2021.111612

Wang, S., Shao, X., Yang, L., and Liu, N. (2020). Deep learning aided dynamic parameter identification of 6-DOF robot manipulators. *IEEE Access* 8, 138102–138116. doi: 10.1109/ACCESS.2020.3012196

Xu, B., Shi, Z., Yang, C., and Sun, F. (2014). Composite neural dynamic surface control of a class of uncertain nonlinear systems in strict-feedback form. *IEEE Trans. Cybern.* 44, 2626–2634. doi: 10.1109/TCYB.2014.2311824

Yang, S., Yu, X., and Zhou, Y. (2020). "LSTM and GRU neural network performance comparison study: taking yelp review dataset as an example," in *Proceedings - 2020 International Workshop on Electronic Communication and Artificial Intelligence, IWECAI 2020* (Shanghai: Institute of Electrical and Electronics Engineers Inc.), 98–101. doi: 10.1109/IWECAI50956.2020.00027