Mechatronics 115 (2026) 103451

Contents lists available at ScienceDirect

Mechatronics

journal homepage: www.elsevier.com/locate/mechatronics

On hybrid inverse dynamic modeling for industrial robots ☆

S. Clavel [a,*], M. Alamir [b], J. Faure-Favre [a]

[a] Stäubli company, 74210 Faverges, France
[b] University of Grenoble Alpes, CNRS, Grenoble INP, GIPSA-lab, 38000 Grenoble, France

ARTICLE INFO

Keywords:
Robots manipulators
Dynamic modeling
Gray-box modeling
Machine learning
Deep learning

ABSTRACT

Inverse dynamic modeling plays a crucial role in developing optimal feedforward control laws in robotics. A promising approach to improve its accuracy involves using *hybrid* approaches combining physics-based models with data-driven models. This paper presents an in-depth study of such an approach, based on Gated Recurrent Unit (GRU) neural networks for application on 4-joints and 6-joints `Stäubli` industrial robots. We start with a comprehensive review of the recent literature. Subsequently, we demonstrate that our model significantly surpasses more traditional black-box models in terms of accuracy and extrapolation. Furthermore, we explore the various factors that influence its accuracy with real data.

1. Introduction

The dynamic modeling of industrial robots has been widely explored in the literature, notably due to its critical importance in achieving compliant, precise, and responsive control. Indeed, the inverse dynamic model is commonly used in robots' control laws as a feedforward mechanism, making its accurate identification crucial for attaining high control performances: the more precise the model, the better the control.

The conventional method for modeling the inverse dynamics of a robot relies on the derivation of *white-box* models, generally based on Eulerian, Newtonian, or Lagrangian physics equations. With such white-box approaches, the model's accuracy depends on both the granularity of the mathematical modeling and the precision of the physical parameters identification. In practice, due to these two factors, the accuracy of white-box models often reaches a limit. To avoid time-consuming modeling and identification efforts, only basic physical effects are modeled and physical parameters are frequently inaccurately identified.

To overcome these limitations, the use of machine learning and even deep learning algorithms is now a well established practice with rather remarkable success. Many data-driven algorithms have already demonstrated convincing results for modeling the dynamics of robots, including Locally Linear Regressors, Support Vector Regressors, Gaussian Process Regressors, or Neural Networks. However, Recurrent Neural Networks are considered highly promising due to their design, which enables them to effectively leverage temporal correlations in data, making them ideal for addressing tasks involving time-series. Popular recurrent architectures such as Long Short-Term Memory (LSTM) networks or Gated Recurrent Unit (GRU) networks have been designed to exploit both short-term and long-term temporal dependencies, addressing challenges like vanishing or exploding gradients commonly faced when analyzing long time-series. The use of LSTMs or GRUs for dynamic modeling of robots enables reaching accuracy levels that outperform those achieved through traditional methods.

In fact, GRU cells are a newer and lighter variation of the well-known LSTM cells: whereas LSTMs have complex and heavy internal architectures, GRUs are characterized by their simpler design, reducing computational complexity while often achieving comparable accuracy [1,2]. Thus, their streamlined architecture can be beneficial in embedded systems, such as robots, where speed, as well as memory, and computational efficiency are critical. Furthermore, since GRU structure has fewer parameters compared to deeper architectures, it may help limit overfitting. This could be advantageous for learning the dynamics of industrial robots that have highly variable operating conditions, in a context where data are often limited and noisy.

There are two main approaches: black-box modeling and gray-box modeling. First, black-box models aim at learning the whole robots' dynamics, without any prior information, completely replacing the physical equations. In their research, Liu et al. [3] studied a black-box model based on an LSTM network and reached high accuracies. The same approach was used by Rueckert et al. [4] with real data, who demonstrated its superiority in terms of accuracy and training time compared to a more conventional method based on Gaussian Processes (GPs). Similar conclusions were drawn by Mukhopadhyay et al. [2] where the authors compared different black-box models, based on

☆ This article is part of a Special issue entitled: 'TC 4.3 Robotics (IFAC WC 2026)' published in Mechatronics.

* Corresponding author.
E-mail addresses: s.clavel@staubli.com (S. Clavel), mazen.alamir@grenoble-inp.fr (M. Alamir).

https://doi.org/10.1016/j.mechatronics.2025.103451
Received 16 October 2025; Received in revised form 25 November 2025; Accepted 23 December 2025

GPs, simple RNNs, LSTMs, GRUs or basic Multi-Layer Perceptrons and showed that the use of LSTMs or GRUs generally leads to better results.

The second approach is the use of gray-box models, also known as *hybrid* models, which exploit the strengths of white-box and black-box models by combining physical equations with data-driven algorithms. A first way to do so is to build a residual model whose goal is to learn the errors of a prior physical model. This approach was used by Liu et al. [5] for example, where the authors built a residual LSTM model that led to significantly better accuracy than their previous black-box approach. The same method was used by Wang et al. [6], who trained a residual LSTM model with attention mechanisms for a 7-axis robot. Similarly, in [7], the authors learned residual bidirectional GRUs for a collaborative robot with elastic joints but did not observe a clear superiority of their hybrid models compared to their black-box version. In their research, Çallar and Böttger [8] tried different models, including residual ones, to learn the dynamics of a 7-axis robot performing specific movements with highly non-linear dynamics. The same type of work was conducted by Tao et al. [9], who have developed a residual LSTM model combined with bagging techniques, leading to higher accuracy than GPs. A second way to implement a gray-box model is by using physics-informed deep learning, which consists of integrating physics laws in the model architecture and the learning process. This type of work has been done by Lutter et al. [10,11] who integrated Lagrangian and Newtonian mechanics in deep learning models, by constraining their architecture and loss function based on physics equations. In their study, Reuss et al. [12] combined this method with a residual model based on LSTMs, resulting in a significant improvement compared to their LSTM black-box model. As a matter of fact, the study of hybrid models based on physics and data-driven algorithms has a growing interest in many fields of robotics research such as hydraulic manipulator control [13,14], soft robot modeling [15], motion planning [16].

In the present contribution, we developed a hybrid model based on GRUs, to derive the inverse dynamics of a 4-axis `TS2-80` and a 6-axis `TX2-90 Stäubli` robots. We explored a wide range of hyperparameters to tune its accuracy and compared it with a black-box approach. To give the reader an idea of the comprehensiveness of this paper, we have listed below the different research items that we have covered and Table 1 synthesizes, for each research item, which paper has addressed it. The research items are:

(a) Comparison of hybrid models with black-box models
(b) Impact of the temporal span of input time-series
(c) Impact of the input features
(d) Impact of the model's size
(e) Impact of the training set's size

Table 1
Research items addressed in the literature.

	(a)	(b)	(c)	(d)	(e)
Liu et al. [3]	✗	✗	✓	✓	✗
Liu et al. [5]	✓	✗	✗	✗	✗
Mukhopadhyay et al. [2]	✗	✗	✗	✓	✓
Reuss et al. [12]	✓	✗	✗	✗	✗
Rueckert et al. [4]	✗	✗	✗	✓	✓
Tao et al. [9]	✗	✗	✗	✗	✗
Valencia-Vidal et al. [7]	✓	✓	✗	✓	✓
Wang et al. [6]	✗	✓	✓	✓	✗
Çallar and Böttger [8]	✓	✓	✓	✓	✗
This paper	✓	✓	✓	✓	✓

The paper is organized as follows. First, Section 2 recalls the problem formulation and presents the architecture of the proposed model, then some aspects of its behavior are analyzed in Section 3. Subsequently, Section 4 highlights the benefits of using a hybrid model towards a black-box model. Finally, Section 5 presents an extensive study of the main hyperparameters that affect the accuracy of the hybrid model before a conclusion is given.

2. Methodology

In this section, we introduce our hybrid model M_H, which is composed of a physics-based model M_P and a residual data-driven model M_θ. Here, we first formulate the problem M_H should solve, and then describe its architecture.

2.1. Problem formulation

Our method aims at deriving the inverse dynamics of any n-joint robot. Such dynamics is classically described as a sum of physics effects:

$$\tau = \underbrace{I(q)\ddot{q} + C(q,\dot{q}) + g(q) + F(q,\dot{q},\ddot{q},\ldots)}_{\tau_p} + \tau_\epsilon \tag{1}$$

where $\tau \in \mathbb{R}^n$ represents the motor torques, q, $\dot{q}$ and $\ddot{q} \in \mathbb{R}^n$ are the joint positions, velocities and accelerations, $I(q) \in \mathbb{R}^{n\times n}$ stands for the symmetric positive-definite inertia matrix, C denotes the Coriolis and centrifugal effects term, g refers to the gravitational term, and $F(q,\dot{q},\ddot{q},\ldots)$ concentrates additional *easy-to-model* physics effects (e.g. basic friction, springs, motor shafts inertia or external forces, etc.). The first four terms on the right side of (1) can be gathered in one term τ_p that represents all the *easy-to-model* torques, that is to say, all the torques that can simply be modeled with a white-box approach. Consequently, the term $\tau_\epsilon \in \mathbb{R}^n$ should be a small residual term which only includes really complex and *hard-to-model* physics phenomena (e.g., backlashes, joints' elasticities, complex friction effects, etc.).

To model this sum of effects, we first build the physics-based model M_P to represent the terms gathered in τ_p with the classic physics equations, so that $M_P(q,\dot{q},\ddot{q})$ provides $\hat{\tau}_p$ which is an estimate of τ_p. The P of M_P represents all the identified physics parameters such as mass, gravity centers, inertia, or friction coefficients. Then, the data-driven model M_θ is built to learn the residual effects that are not modeled by M_P, so that $M_\theta(q,\dot{q},\ddot{q},\ldots)$ provides $\hat{\tau}_\epsilon$ which is an estimation of τ_ϵ, or more exactly of $\tau_e := \tau - \hat{\tau}_p$. Here, the θ of M_θ represents all the trainable parameters of the learning model. Please note that, in our work, the model is intended to be used in a pure feedforward, or open-loop, manner. Consequently, q, $\dot{q}$ and $\ddot{q}$ represent here the desired joint positions, velocities, and accelerations, and τ is the desired torque required by the joint motors and computed by the robot's PID controller. This detail is of great importance as this approach is underexplored in the literature, as it can be seen in Table 2 where a comparison with related hybrid dynamics studies is shown.

After having built M_P and M_θ, the hybrid model M_H is simply defined as $M_H := M_P + M_\theta$, which predicts $\hat{\tau}$ which should be an accurate estimation of the torques τ. This model should reach high accuracy levels, on one hand, since M_θ has been trained to specifically fit the inaccuracies of M_P. On the other hand, it should maintain good generalization and extrapolation capabilities, given that M_P is a white-box model. Indeed, considering that M_P is purely physics-based, its mathematical equations remain valid and invariant for a wide variety of operating conditions regardless of payload, position space, or trajectory profile. This contrasts with data-driven models that are prone to overfitting on specific operating conditions since they learn directly and blindly from data. Thus, the two models complement each other: where M_θ is generally fragile in terms of generalization and extrapolation, M_P compensates. Conversely, where M_P struggles to capture very complex physical phenomena, M_θ addresses these challenges.

In this paper, the models' accuracies are evaluated thanks to two dimensionless metrics that are the normalized Mean Absolute Error (nMAE) and the normalized Mean Squared Error (nMSE), defined as follows.

$$\text{nMAE}(\hat{\tau},\tau) = \frac{\sum_{i=0}^{T} |\hat{\tau}_i - \tau_i|}{\sum_{i=0}^{T} |\tau_i|} \tag{2}$$

Table 2
Comparison to related hybrid dynamics studies made with real robots. The subscript letter m denotes measured signals while d denotes desired signals. Papers whose application have been labeled "Identification" do not specify explicitly applications but often mention control.

	Input features	Target variable	Approach	Application
Liu et al. [5]	q_m, $\dot{q}_m$, $\ddot{q}_m$	τ_m	Residual LSTM	Identification
Reuss et al. [12]	q_m, $\dot{q}_m$, $\ddot{q}_d$	τ_d	Physics inspired with residual LSTM	Feedforward control
Lutter et al. [10]	q_m, $\dot{q}_m$, $\ddot{q}_m$	τ_d	Physics-informed NN	Feedforward control
Valencia-Vidal et al. [7]	q_m, $\dot{q}_m$	τ_d	Residual bidirectional GRU	Feedforward control
Wang et al. [6]	q_m, $\dot{q}_m$, $\ddot{q}_m$	τ_m	Residual LSTMs with attention	Identification
Çallar and Böttger [8]	q_m, $\dot{q}_m$, $\ddot{q}_m$, r_m	τ_m	Residual LSTMs or transformer	Identification
This paper	q_d, $\dot{q}_d$, $\ddot{q}_d$	τ_d	Residual GRU	Feedforward control and fault detection

$$\text{nMSE}(\hat{\tau}, \tau) = \frac{\sum_{i=0}^{T} \left(\hat{\tau}_i - \tau_i\right)^2}{\sum_{i=0}^{T} \tau_i^2} \tag{3}$$

The next sections provide an in-depth exploration of the architecture of each component of M_H.

2.2. Physics-based component M_P

The general architecture of the physics-based model is described in Fig. 1. In the first layer of M_P, we use the Newton-Euler algorithm to compute τ_{jnt} which represents the joint torques induced by the inertial effects of the motions of each axis. Since each axis of the robot is connected to another, their dynamics have a strong coupling, which can be difficult to model as the number of axes increases. The Recursive Newton-Euler algorithm is a well-known method that allows us to model these complex dynamics in a computationally efficient way [17].

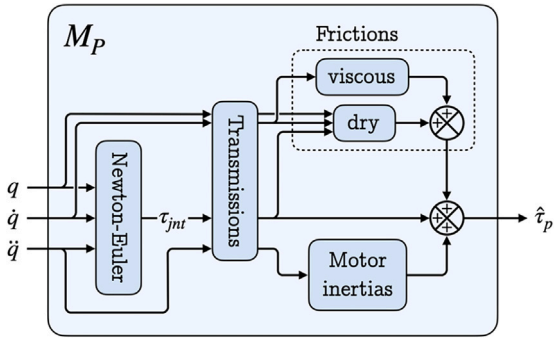

Fig. 1. General architecture of M_P.

The second layer of M_P involves a simple base change that brings the joint torques τ_{jnt}, along with the joint positions q, velocities $\dot{q}$ and accelerations $\ddot{q}$ into the motor reference frame, to obtain the motor torques with motor positions, velocities and accelerations.

Following the transmission layer, friction layers are introduced. First, a component models viscous friction basically as a power of the motor velocities, for each axis i:

$$\mathcal{V}_i(t) = k_i^{(1)} \times \text{sign}\left(\dot{q}_i(t)\right) \times \left|\dot{q}_i(t)\right|^{k_i^{(2)}} \tag{4}$$

where $k_i^{(1)}, k_i^{(2)} \in \mathbb{R}$ represent the usual viscous friction parameters of the axis i. In parallel, a dry friction component is added which is somewhat more complex. First, if the axis i is moving (i.e. $\dot{q}_i(t) \neq 0$), the dry friction torque is calculated based on a modified Coulomb model which is then filtered using a distance relaxation mechanism, as described in (5), (6) and (7). Conversely, if the axis i is not moving (i.e. $\dot{q}_i(t) = 0$),[1] the dry friction torque is progressively reduced to 0 following a time relaxation law as described in (8).

$$\mathcal{D}_i(t)|_{\dot{q}\neq 0} = \widetilde{D}_i(t) + \left(\mathcal{D}_i(t)(t-\delta t) - \widetilde{D}_i(t)\right) \times e^{d_i(t)} \tag{5}$$

$$\text{with } \widetilde{D}_i(t) = \text{sign}\left(\dot{q}_i(t)\right) \times \left(k_i^{(3)} + k_i^{(4)} \left|\tau_{jnt_i}(t)\right|\right) \tag{6}$$

$$\text{and } \quad d_i(t) = \frac{-\left|q_i(t) - q_i(t-\delta t)\right|}{k_i^{(5)}} \tag{7}$$

$$\mathcal{D}_i(t)|_{\dot{q}=0} = \mathcal{D}_i(t-\delta t) \times \left(1 - \frac{\delta t}{k_i^{(6)}}\right) \tag{8}$$

Here, δt denotes the controller sampling time. The parameter $k_i^{(3)}$ is the basic Coulomb parameter and $k_i^{(4)}$ enables us to model the dependence on the applied torque τ_{jnt}. The parameter $k_i^{(5)}$ is the relaxation distance which enables us to model the hysteresis effects observed during small displacements, also known as Dahl friction. Finally, $k_i^{(6)}$ is the relaxation time. It should be noted that, to avoid cumbersome notation, the variable names q_i, $\dot{q}_i$ and τ_{jnt_i} are used in (4), (6) and (7) for simplicity. However, these variables correspond here to the positions, velocities, and torques within the motor reference frame due to the transmission layer and are no longer in the joint reference frame, as shown in Fig. 1.

Ultimately, for each axis i, a motor inertia layer models the motor shaft inertia with a basic proportional model:

$$\mathcal{J}_i(t) = k_i^{(7)} \times \ddot{q}_i(t) \tag{9}$$

where $k_i^{(7)}$ is the shaft inertia of motor i. After this layer, the estimation of the motor torque $\hat{\tau}_p$ is finally obtained by summing all the previously computed torques: the torques due to shaft inertia, viscous and dry frictions, and arm inertia.

The physical parameters P of M_P are either known by construction (for example, the mass, gravity center, and inertia matrix of the robot joints are known thanks to the CAD files) or have been identified using experimental procedures (such as friction parameters) by Stäubli. These identification procedures involve the acquisition of torque and motion data during specific identification trajectories (i.e., at specific speeds, in steady state, free of gravity if necessary, or during small displacements, etc.). Based on these specific acquisition points, the parameters are then identified with Gray-box identification tools. The exact details of these identification procedures are beyond the scope of this paper and will not be described here.

With this architecture, the physics-based model achieves good accuracy. The usual accuracy metrics values reached by M_P can be seen in Table 4. It is important to note that given that we use a hybrid model, achieving high accuracy with M_P is still of great interest. Indeed, as formulated in (1), $\tau = \tau_p + \tau_\epsilon$. Therefore, as the accuracy of M_P increases, the residual torques decrease, meaning M_θ will only have to model small residual phenomena.

2.3. Data-driven component M_θ

The goal of the data-driven model M_θ is to simulate the residual $\tau_e := \tau - \hat{\tau}_p$, and is mainly based on several layers of gated recurrent neural networks.

[1] Note that this condition would never hold with measured signals, due to measurement noise. However, we recall that our models rely on desired signals so the nullification of $\dot{q}$ is possible.

Gated recurrent cells, including GRUs and LSTMs cells, have been introduced to tackle the traditional challenges of vanishing and exploding gradients encountered when training deep networks with long time-series data. These networks are composed of neural layer combinations, commonly referred to as gates, which are designed to concentrate on relevant long-term and short-term dependencies while discarding irrelevant ones. The GRU cells are a more recent and simplified variation of the well-known LSTM cells. Their structure is recalled in Fig. 2.

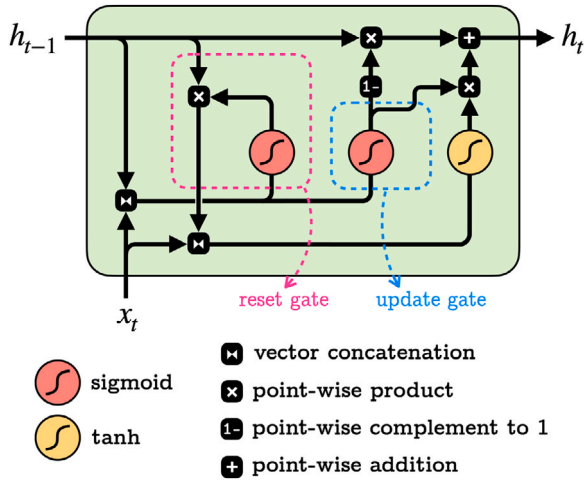

Fig. 2. Structure of a GRU cell annotated with its elementary operators. The cell is based on 2 gates: the reset gate determines what information to discard from the previous hidden state h_{t-1}, and the input gate decides what information from the input x_t to store in the current hidden state h_t.

In its final architecture, the model M_θ receives as input sequences of features q, $\dot{q}$, $\ddot{q}$, and $\hat{\tau}_p$, of length 128. These input sequences will be denoted respectively $\tilde{q}$, $\tilde{\dot{q}}$, $\tilde{\ddot{q}}$, and $\tilde{\hat{\tau}}_p$. These time-series first pass through 4 layers of GRU cells, each with a 64-dimensional hidden state, stacked on each other. The hidden state of the final GRU layer is then fed to a Fully Connected (FC) layer which predicts $\hat{\tau}_e$, the n-dimensional vector that estimates the residual τ_e. This final configuration required several experiments, which will be discussed in Section 5.

To train the model M_θ, large datasets containing time-series of q, $\dot{q}$, $\ddot{q}$, $\hat{\tau}_p$ for model input and τ_e for model output must be gathered from the robots. Two datasets were collected, on a 6-axis TX2−90 and a 4-axis TS2−80 Stäubli robots, which enabled us to train a M_θ for each type of robot. To ensure the models' robustness across different robots' trajectories, these datasets must encompass a wide variety of motion scenarios. To do so, the robots performed random point-to-point movements with randomly chosen points in almost their entire workspace, random velocities, and accelerations. During these random movements, the required time-series of q, $\dot{q}$, $\ddot{q}$, and τ were sampled at 250 Hz, then the remaining required signals were post-calculated. First, $\hat{\tau}_p$ was computed with $M_P(q, \dot{q}, \ddot{q})$, then the residual torque τ_e was finally computed as the difference $\tau - \hat{\tau}_p$. No filter was applied to any of the signals. A total of 200 random trajectories were executed for each robot using this data collection process, which led to a first dataset containing 1,623,000 samples for the TS2−80 and a second of 2,575,000 samples for the TX2−90. These datasets are finally normalized with the usual Min-Max normalizer and then split into 3 subsets: 80% of the data are kept for the training set, 17% of the data are allocated for the test set, and the remaining 3% form a small validation set.

Table 3
Datasets description.

	TX2−90	TS2−80
Movements per trajectory	30	24
Motion type	Joint space	Joint space
Joints limits	$q^{(1)}$: [−180°, 180°] $q^{(2)}$: [−90°, 90°] $q^{(3)}$: [−60°, 60°] $q^{(4)}$: [−180°, 180°] $q^{(5)}$: [−90°, 90°] $q^{(6)}$: [−180°, 180°]	$q^{(1)}$: [−180°, 180°] $q^{(2)}$: [−120°, 120°] $q^{(3)}$: [0 m, 0.2 m] $q^{(4)}$: [−500°, 500°]
Number of trajectories	200	200
Sampling frequency	250 Hz	250 Hz
Sampled signals	$q^{(i)}, \dot{q}^{(i)}, \ddot{q}^{(i)}, \tau^{(i)}$ for $i \in \{1, \dots, 6\}$	$q^{(i)}, \dot{q}^{(i)}, \ddot{q}^{(i)}, \tau^{(i)}$ for $i \in \{1, \dots, 4\}$
Post-calculated signals	$\hat{\tau}_p^{(i)}$ and $\tau_e^{(i)}$ for $i \in \{1, \dots, 6\}$	$\hat{\tau}_p^{(i)}$ and $\tau_e^{(i)}$ for $i \in \{1, \dots, 4\}$
Total samples	2.575×10^6	1.623×10^6
Duration	171 min	108 min

The models are then trained, offline, to perform a regression task with the MAE loss function. The optimization of the trainable parameters θ of M_θ is carried out with the well-known Adam optimizer with an exponentially decreasing learning rate from 10^{-4} to 10^{-5} and the default parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. The mini-batch gradient descent is performed with batches of size 64 and a maximum of 200 epochs. An early stopping mechanism based on the validation loss is settled to prevent overfitting: the model that reaches the minimal validation loss is considered optimal. Once training is done, test loss and test metrics are computed in order to assess the performance of the obtained model and to compare various hyperparameter configurations, as will be done in Sections 4 and 5.

2.4. Resulting hybrid model

Once M_P and M_θ are built, the resulting hybrid model can be formed as described in Fig. 3. Since M_θ takes time-series as input, a buffer mechanism is added after M_P to create the appropriate sequences of the features q, $\dot{q}$, $\ddot{q}$, and $\hat{\tau}_p$. Please note that M_θ takes as input the data of all the axes of the robot (q, $\dot{q}$, $\ddot{q}$, $\hat{\tau}_p \in \mathbb{R}^n$) to produce the residual torque estimate of all the axes ($\hat{\tau}_\epsilon \in \mathbb{R}^n$). The sum of $\hat{\tau}_p$ and $\hat{\tau}_\epsilon$ gives $\hat{\tau}$ which should be a robust and precise estimation of τ defined in (1). We recall here that q, $\dot{q}$, $\ddot{q}$ are the desired joint's positions, velocities, and accelerations, while $\hat{\tau}$ is an estimation of the desired motor torques.

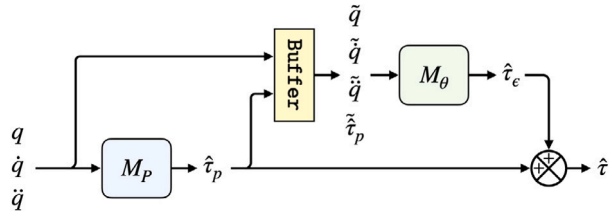

Fig. 3. Final architecture of M_H.

With this hybrid model, significantly higher accuracy can be reached compared to the initial white-box model M_P, as we can see in Table 4.

Table 4
Accuracy metrics of M_P and M_H for the two robots on the 40 test trajectories. The rows correspond to the different joints of the robots. The columns correspond to the metrics computed between τ and $\hat{\tau}_p$ for M_P and between τ and $\hat{\tau}$ for M_H.

		nMAE		nMSE	
		M_P	M_H	M_P	M_H
TX2-90	1	0.179 ± 0.009	0.139 ± 0.008	0.033 ± 0.003	0.020 ± 0.003
	2	0.114 ± 0.007	0.035 ± 0.002	0.015 ± 0.001	0.002 ± 0.000
	3	0.110 ± 0.013	0.050 ± 0.004	0.015 ± 0.003	0.003 ± 0.000
	4	0.287 ± 0.082	0.225 ± 0.067	0.158 ± 0.100	0.117 ± 0.082
	5	0.399 ± 0.030	0.105 ± 0.009	0.155 ± 0.019	0.013 ± 0.002
	6	0.378 ± 0.038	0.116 ± 0.007	0.123 ± 0.021	0.016 ± 0.002
TS2-80	1	0.211 ± 0.007	0.055 ± 0.004	0.044 ± 0.002	0.003 ± 0.000
	2	0.183 ± 0.007	0.064 ± 0.005	0.040 ± 0.003	0.004 ± 0.001
	3	0.193 ± 0.012	0.078 ± 0.006	0.042 ± 0.004	0.007 ± 0.001
	4	0.231 ± 0.012	0.188 ± 0.012	0.067 ± 0.007	0.050 ± 0.006

3. Detailed analysis of M_H behavior

This section provides insights and results regarding the behavior of M_H. A brief study on the contribution of each prior physical sub-model is first proposed, followed by a visualization of the phenomena not captured by the model. Finally, results from a short feedforward control experiment utilizing M_H are presented.

3.1. Contributions of prior physical models

To enhance the explainability of our hybrid architecture, the impact of each prior physical sub-model has been studied. For this purpose, M_H was tested while progressively activating the sub-models of M_P: starting with none activated (Ø), then activating only the Newton-Euler algorithm (NE), adding friction models (NE+F), and finally adding motor shaft inertia models (NE+F+MS). For each configuration of M_P, the residual model M_θ was trained accordingly. This experiment was conducted on the TS2−80 dataset. The metrics values reached by each configuration on the test trajectories are gathered in Table 5.

Table 5
Impact of each prior model. Results presented for the nMAE metric only.

		M_P			
		Ø	NE	NE+F	NE+F+MS
TS2-80	1	1.0 ± 0.0	0.562 ± 0.033	0.326 ± 0.005	0.211 ± 0.007
	2	1.0 ± 0.0	0.541 ± 0.044	0.204 ± 0.009	0.183 ± 0.007
	3	1.0 ± 0.0	0.577 ± 0.046	0.203 ± 0.014	0.193 ± 0.012
	4	1.0 ± 0.0	0.981 ± 0.024	0.353 ± 0.025	0.231 ± 0.012
		M_H			
		Ø	NE	NE+F	NE+F+MS
TS2-80	1	0.083 ± 0.005	0.059 ± 0.005	0.057 ± 0.004	0.055 ± 0.004
	2	0.081 ± 0.005	0.061 ± 0.005	0.061 ± 0.006	0.064 ± 0.005
	3	0.082 ± 0.007	0.083 ± 0.007	0.079 ± 0.007	0.078 ± 0.006
	4	0.191 ± 0.012	0.189 ± 0.012	0.189 ± 0.012	0.188 ± 0.012

The results show, for M_P, that the Newton-Euler algorithm first captures a significant part of the torques, except on the last joint. Then, the friction and motor shaft inertia models have reasonable contributions, especially on the 4th joint, where they help to reduce the residual to a level comparable to the other joints, revealing that the dynamics of this joint is particularly governed by friction and motor's inertia. In contrast, within the overall hybrid architecture, gradually activating friction and motor shaft inertia models yields only minor accuracy gains. This suggests that M_θ is greatly assisted by the Newton-Euler algorithm, which notably models the complex coupled dynamics of robot joints, but has no difficulty in replacing the terms of friction and motor inertia if they are lacking from the physical model.

3.2. Vibratory effects

By looking closely at the predictions of M_H, we found that our model was particularly good at simulating low-frequency phenomena, but had difficulty to capture some vibratory effects. For example, this can be seen in the carefully chosen zooms on the predictions of M_H for the TX2−90 dataset, in Fig. 4. First, it is clear that M_θ succeeds in compensating the inaccuracies of M_P. However, we can see that the model struggles to reproduce the vibratory phenomena starting at 191.7 s and is finally completely out of touch at 192 s, especially for the joints 1 and 6.

These observations are confirmed by looking at the Power Spectral Densities of our model on joint 1 and 6, shown in Fig. 5. We see that M_H manages to model precisely the effects below 10 Hz, but misses important parts of the frequency content between 10 and 50 Hz. These vibration phenomena could be caused by joint elasticity which the model struggles to capture for long durations, probably because of vanishing gradients.

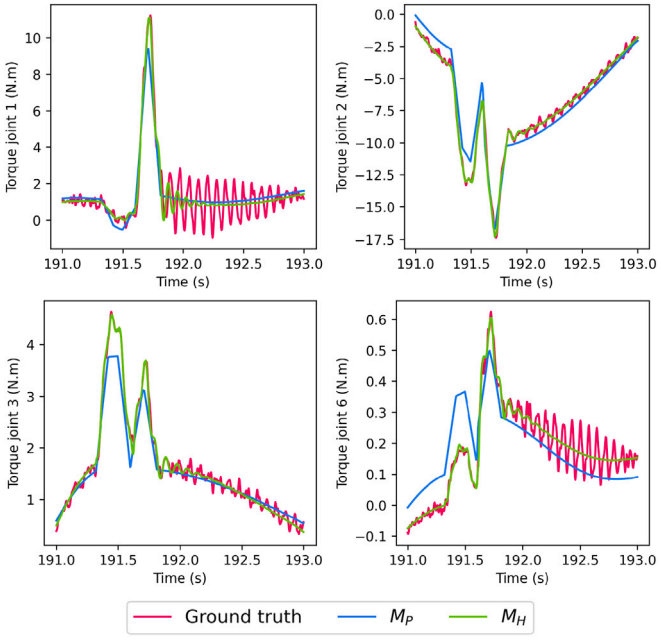

Fig. 4. Zooms on M_H predictions for the TX2−90 robot.

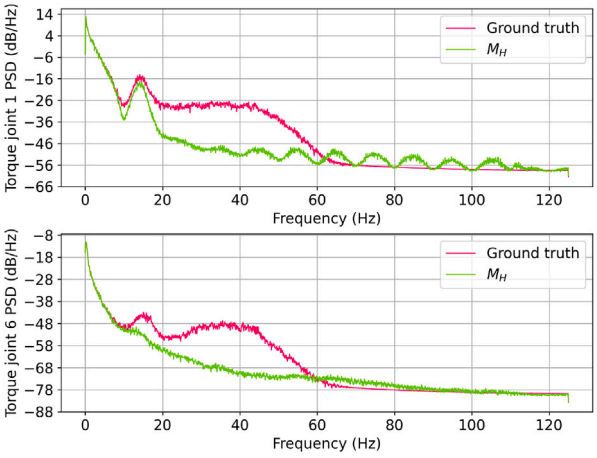

Fig. 5. Power spectral densities for joints 1 and 6.

3.3. Feedforward control experiment

To validate the relevance of our model, we used it in a short feedforward control experiment on the TS2-80. A simplified diagram of the initial control loop of Stäubli robots is shown in Fig. 6. It is made up of a classic PID equipped with a feedforward torque unit named Advanced Torque Control (ATC).

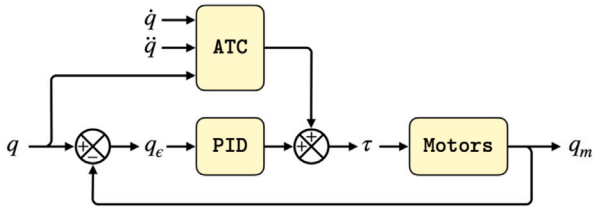

Fig. 6. Control loop of Stäubli robots.

In our experiment, we ordered the robot to perform exactly the same predefined movements under the same operating conditions, first with the original control loop and then with the ATC unit being replaced by M_H. The motions were new motions performed in the joint space, unseen by our model during its training, and lasted 31 s. During these motions, we recorded the position tracking error signal q_ϵ at 250 Hz. Excerpts of the measured absolute tracking error can be compared in Fig. 7.

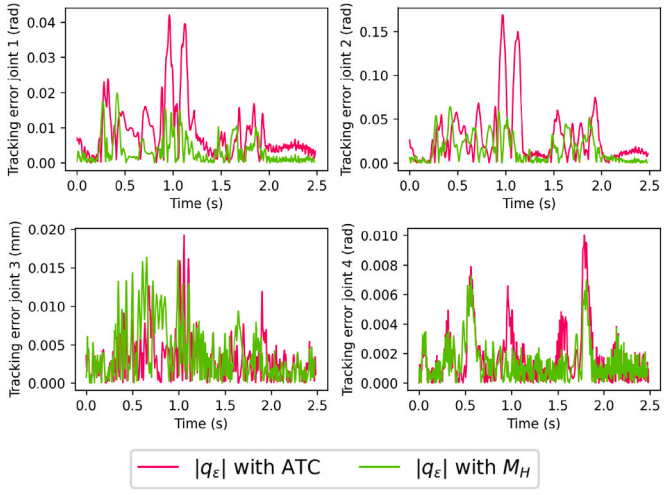

Fig. 7. Tracking error comparison.

The average values of $|q_\epsilon|$ obtained for each joint are presented in Table 6. We can observe a general improvement in the tracking errors in almost all joints, which is an encouraging result that validates the relevance of the model in a feedforward control context. Nevertheless, a slight deterioration can be seen for the third joint. This side effect is still under investigation and could be explained by the fact that this axis is the only translation axis of the robot. The mechanical transmission of this axis is provided by a ball screw system, making its dynamics unique and significantly influenced by friction phenomena, which could hinder the modeling task of M_H.

Table 6
Mean absolute tracking errors.

		With ATC	With M_H	Unit
TS2-80	1	0.0098	0.0058	rad
	2	0.035	0.023	rad
	3	0.0036	0.0041	mm
	4	0.0019	0.0016	rad

4. Black-box vs. hybrid

We also wanted to validate the relevance of using our hybrid model rather than a black-box model. To do so, we trained a model with the same architecture as M_θ to predict the whole signal τ instead of the residual $\tau - \hat{\tau}_p$. This model, called M_B, is a black box version of M_H since it uses the same input features q, $\dot{q}$, and $\ddot{q}$ to model the same target variable τ. This model was trained with the exact same optimization configuration used for M_θ (i.e. same optimizer, same number of epochs, same early stopping mechanism, etc.). We performed this experiment while increasing the number of trainable parameters of the two models to see if a black-box model with a sufficient complexity would be able to reach a better or at least similar precision than a hybrid model.

The results are shown in Fig. 8, with the displayed metric being the nMAE obtained on the TS2-80 test dataset, averaged across all axes of the robot. The right y-axis shows the relative accuracy deviation for each model compared to the optimal model. This second y-axis will be present in all remaining graphs to facilitate interpretation. The results reveal that the black-box version has significantly lower prediction performances than the hybrid model, even when increasing the model's complexity up to more than a million parameters. The use of black-box models leads to a loss of at least 8% in accuracy regardless of the number of trainable parameters. This confirms the idea that all the prior knowledge about Stäubli robots contained in M_P is of great importance for M_H.

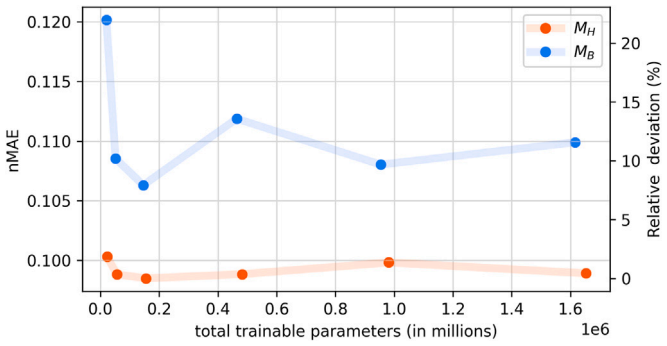

Fig. 8. Comparison of M_H and M_B on the test dataset.

4.1. Generalization to a new motion type

Subsequently, we intended to verify the hypothesis that M_H had better generalization capabilities than M_B. To achieve this, a first experiment on the TS2-80 was conducted and consisted in validating the models on an entirely new type of motion. Therefore, the robot executed predefined movements in the linear space, contrasting with the training movements performed in the joint space. These new movements differed significantly in terms of positions, velocities, and accelerations from those used for training. The trajectory lasted approximately 12 s and the appropriate data were sampled at 250 Hz.

The performance metrics of both models during this trajectory are presented in Table 7. We see a clear superiority of M_H, especially on the first three axes. This suggests that M_H is still more accurate than M_B when it comes to generalize. In future work, this validation experiment will be performed on longer trajectories.

Table 7
Generalization metrics on a movement in the linear space.

		nMAE		nMSE	
		M_B	M_H	M_B	M_H
TS2-80	1	0.118	0.098	0.016	0.010
	2	0.190	0.177	0.033	0.028
	3	0.170	0.156	0.034	0.029
	4	0.320	0.309	0.114	0.109

4.2. Extrapolation on unknown position space

A second and more robust experiment was conducted to validate the superiority of M_H for extrapolation tasks. For this experiment, a third training dataset was collected with the exact same acquisition settings as the initial TS2−80 dataset described in Table 3, except for the joints limits. The limits of the joint positions were deliberately reduced such that $q^{(1)}, q^{(2)}, q^{(4)} \in [-20°, 20°]$, and $q^{(3)} \in$[0 m, 0.04 m].

The two models M_H and M_B were first trained with this dataset collected on a limited workspace. The models were then validated on the 40 test trajectories of the initial dataset collected in a wide workspace, which therefore includes a large majority of positions not seen by the models during training. Specifically, only 17 samples out of the 3.12×10^5 test samples were collected with the robot's joints positions located within the training set limits. This dataset was therefore legitimate for assessing the extrapolation abilities of the models on completely unknown positions.

The performance metrics obtained during this experiment are shown in Table 8. The accuracy gap between M_H and M_B is even clearer in this experiment: a ratio of at least 3 is observed across all metrics. These results confirm the great superiority of M_H for extrapolating along with the hypothesis that the physics-based part of M_H is a key advantage for extrapolation.

Table 8
Extrapolation metrics on positions not seen during training.

		nMAE		nMSE	
		M_B	M_H	M_B	M_H
TS2-80	1	1.090 ± 0.201	0.316 ± 0.044	1.214 ± 0.452	0.084 ± 0.022
	2	1.060 ± 0.180	0.341 ± 0.041	1.189 ± 0.418	0.103 ± 0.020
	3	0.973 ± 0.170	0.262 ± 0.036	1.047 ± 0.268	0.076 ± 0.014
	4	2.353 ± 0.199	0.772 ± 0.074	5.439 ± 0.738	0.585 ± 0.100

5. Tuning of hyperparameters

Here we present some of the experiments that we have conducted in order to find the best hyperparameter configuration that minimizes the precision metrics computed on the test data. These experiments were essentially conducted with the data collected on the TS2−80 robot, however we believe the results are transposable to some extent.

5.1. Model's size

First, the most crucial hyperparameters of M_θ that must be tuned are those determining the complexity of the model. This can be done with the parameter `n_layer` which corresponds to the number of GRU layers used in the model, and the parameter `hidden_dim` which corresponds to the dimension of the GRUs' hidden states. These two parameters drastically influence the number of trainable parameters of M_θ. For example, a configuration with `n_layer` = 4 and `hidden_dim` = 256 has nearly 1.6×10^6 trainable parameters, while taking `n_layer` = 1 with `hidden_dim` = 16 leads to only 1.2×10^4 trainable parameters. Therefore, these must be cautiously tuned to achieve good predicting performances while avoiding overfitting.

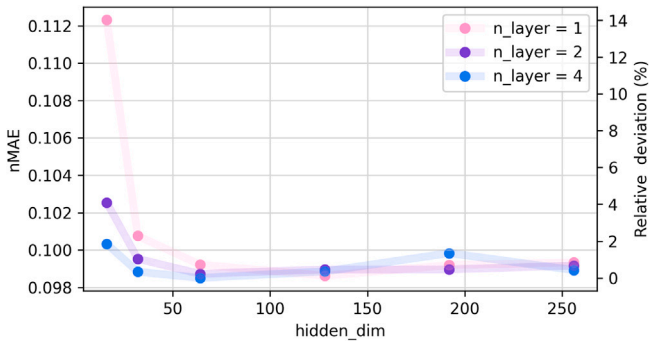

Fig. 9. Impact of `n_layer` and `hidden_dim`.

To do so, several experiments have been conducted by varying these two parameters, Fig. 9 shows some of the results obtained for `n_layer` varying from 1 to 4 and `hidden_dim` varying from 16 to 128. The results show that, beyond 64 hidden dimensions, adding more layers does not have an impact on the model's accuracy and can even lead to overfitting. Based on Fig. 9, the optimal architecture is composed of 4 layers with 64 hidden dimensions and is therefore the configuration that will be used in the following sections.

5.2. Input features and temporal span

We also wanted to determine the optimal input sequences for M_θ. First, the temporal span of the input sequences had to be tuned. This was done by adjusting the `seq_len` hyperparameter which represents the length of the input sequences. For example, a `seq_len` = 125 corresponds to input time-series that have a temporal span of 0.5 s, since the data are sampled at 250 Hz. A second way to tune this parameter was to adjust the hyperparameter `x_sampling` that corresponds to the subsampling ratio of the input data. For example, time-series of `seq_len` = 125 with `x_sampling` = 1 (i.e. no subsampling) correspond to 0.5 s, while the same time-series with `x_sampling` = 2 correspond to 1.0 s, etc. Several experiments were conducted varying these two parameters, Fig. 10 shows some results obtained for `seq_len` varying from 4 to 256 and `x_sampling` varying from 1 to 4. The results show that, once the corresponding temporal span is sufficiently large, these two hyperparameters do not greatly affect the model's accuracy. However, using single point, non-sequential inputs (i.e. `seq_len` = 1) would lead to poor precision performances. Finally, the optimal configuration is achieved rapidly with `seq_len` set to 128 and `x_sampling` set to 2, which corresponds to a temporal span of 1.024 s.

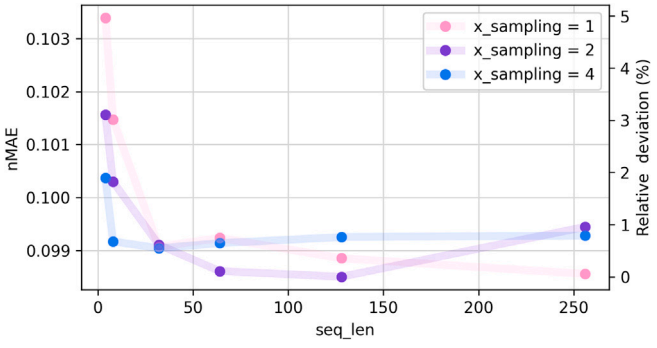

Fig. 10. Impact of `seq_len` and `x_sampling`.

We also had to choose which input features were the best to use, among all the collected signals. To do so, we performed experiments using various subsets of input features. Table 9 shows the performance metrics obtained for several subsets. The results show that the use of

Table 9
Impact of the choice of input features. The metrics are averaged over all robot's axes.

	nMAE	nMSE
$\ddot{q}$	0.116	2.11×10^{-3}
$\dot{q}$	0.111	1.90×10^{-3}
q	0.108	1.74×10^{-3}
$\hat{\tau}_p$	0.102	1.14×10^{-3}
q, $\dot{q}$, $\ddot{q}$	0.100	1.13×10^{-3}
q, $\hat{\tau}_p$	0.099	1.07×10^{-3}
q, $\dot{q}$, $\ddot{q}$, $\hat{\tau}_p$	0.098	1.04×10^{-3}

τ_p has a strong impact on the accuracy of the model. Indeed, once this feature is used, adding more input features has a small impact on the accuracy of the model. This could be explained by the fact that this feature is a combination of the others, since it is computed by M_P with q, $\dot{q}$ and $\ddot{q}$. However, it can be observed that the use of all features q, $\dot{q}$, $\ddot{q}$ and τ_p seems to be the optimal configuration.

5.3. Size of the training set

Finally, given the complexity of the chosen model, we wanted to know how much data was necessary to properly train it. This question is of interest, as collecting large training datasets on a robot can take several hours, and Stäubli might need to minimize this data collection time. The amount of data required depends, of course, on the richness of the scenarios realized by the robot. However, we believe that it could be useful for the community to know, for this type of robot and model, an order of magnitude of the time required to collect a sufficient amount of data with the robot performing random trajectories.

Experiments were carried out with various portions of the original TS2−80 dataset of 1,623,000 samples. Fig. 11 shows the performance metrics reached by different models, for different portions starting from 5% (equivalent to 4 min of recording) to 100% (83 min of recording). The x-axis has been converted in equivalent recording time for easier interpretation.

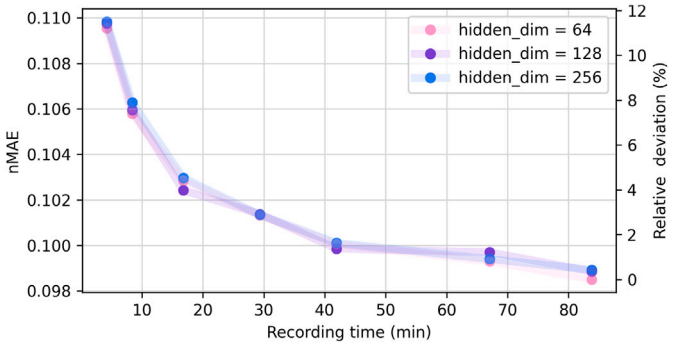

Fig. 11. Impact of the training set's size.

The results show that good prediction performances are rapidly achieved, with only 50% of our initial dataset, which corresponds to 43 min of data recording on the robot. Beyond this point, the precision gain becomes very small. We can also notice that models of varying complexities learn equivalently, regardless of the amount of data used. Therefore, the idea of collecting more data to use more complex models with the aim of achieving better performances would be pointless: it seems that we have reached a limit in terms of accuracy with this hybrid structure.

Naturally, industrial contexts could prohibit hours of data acquisition due to costs caused by downtime or because of technical issues related to the production environment. Strategies to reduce the required amount of operational data could therefore be considered. As an example, one strategy would involve pre-training a base model using an extensive dataset historically acquired by Stäubli. This model would then be fine-tuned with transfer learning techniques, using small datasets quickly collected at the end customer's production site. Incremental learning strategies could also be used to ensure the consistency of the model throughout the robot's lifetime. The base model would be periodically re-trained with the original dataset judiciously updated with small amounts of data frequently collected on the robot during its use on site.

6. Conclusion and future work

This paper has proposed an extensive study of hybrid models based on GRU neural networks for accurately capturing the dynamics of Stäubli robots in a complete feedforward manner, which has never been done in the literature, as far as we know. The key hyperparameters to optimize its precision have been presented and analyzed. In addition, valuable insights have been proposed on the explainability and behavior of such hybrid architectures, and their great superiority over black-box architectures has been demonstrated and validated in generalization and extrapolation experiments.

The use of the model for feedforward control seems promising and will be further explored in the future. Future work could also involve the study of transfer learning or incremental learning to address the challenges associated with real world industrial contexts. Finally, utilizing the model for anomaly detection tasks is highly considered.

CRediT authorship contribution statement

S. Clavel: Writing – review & editing, Writing – original draft, Investigation, Data curation. **M. Alamir:** Writing – review & editing, Methodology. **J. Faure-Favre:** Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The data that has been used is confidential.

References

[1] Chung J, Gülçehre Ç, Cho K, Bengio Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. 2014, CoRR, abs/1412.3555.
[2] Mukhopadhyay R, Chaki R, Sutradhar A, Chattopadhyay P. Model learning for robotic manipulators using recurrent neural networks. In: TENCON 2019 IEEE region 10 conference. 2019, p. 2251–6.
[3] Liu N, Li L, Hao B, Yang L, Hu T, Xue T, Wang S. Modeling and simulation of robot inverse dynamics using LSTM-based deep learning algorithm for smart cities and factories. IEEE Access 2019;7:173989–98.
[4] Rueckert E, Nakatenus M, Tosatto S, Peters J. Learning inverse dynamics models in O(n) time with LSTM networks. 2017.
[5] Liu N, Li L, Hao B, Yang L, Hu T, Xue T, Wang S, Shao X. Semiparametric deep learning manipulator inverse dynamics modeling method for smart city and industrial applications. Complexity 2020:1–11.
[6] Wang S, Shao X, Yang L, Liu N. Deep learning aided dynamic parameter identification of 6-DOF robot manipulators. IEEE Access 2020.
[7] Valencia-Vidal B, Ros E, Abadia I, Luque N. Bidirectional recurrent learning of inverse dynamic models for robots with elastic joints: a real-time real-world implementation. Front Neurorobotics 2023;17.
[8] Çallar T-C, Böttger S. Hybrid learning of time-series inverse dynamics models for locally isotropic robot motion. IEEE Robot Autom Lett 2022;8:1061–8.
[9] Tao Y, Chen S, Liu H, Wan J, Wei H, Wang T. Robot hybrid inverse dynamics model compensation method based on the BLL residual prediction algorithm. Robotica 2024;1–18.
[10] Lutter M, Ritter C, Peters J. Deep Lagrangian networks: Using physics as model prior for deep learning. 2019, CoRR, abs/1907.04490.

[11] Lutter M, Silberbauer J, Watson J, Peters J. A differentiable Newton-Euler algorithm for real-world robotics. 2021, CoRR, abs/2110.12422.
[12] Reuss M, van Duijkeren N, Krug R, Becker P, Shaj V, Neumann G. End-to-end learning of hybrid inverse dynamics models for precise and compliant impedance control. 2022.
[13] Yao Z, Xu F, Jiang G-P, Yao J. Data-driven control of hydraulic manipulators by reinforcement learning. IEEE/ASME Trans Mechatronics 2024;29(4):2673–84.
[14] Yao Z, Liang X, Wang S, Yao J. Model-data hybrid driven control of hydraulic Euler–Lagrange systems. IEEE/ASME Trans Mechatronics 2025;30(1):131–43.
[15] Zhang Z, Jin Z, Gu GX. Efficient pneumatic actuation modeling using hybrid physics-based and data-driven framework. Cell Rep Phys Sci 2022;3(4):100842.
[16] Yang W-T, Liao J-M, Lin P-C. Physics-data hybrid dynamic model of a multi-axis manipulator for sensorless dexterous manipulation and high-performance motion planning. 2024.
[17] Khalil W, Dombre E. Modeling, identification and control of robots. 2004.