IEEE Robotics & Automation Magazine (RAM) paper, presented at ICRA 2026, Vienna, Austria. Cite as RAM paper.

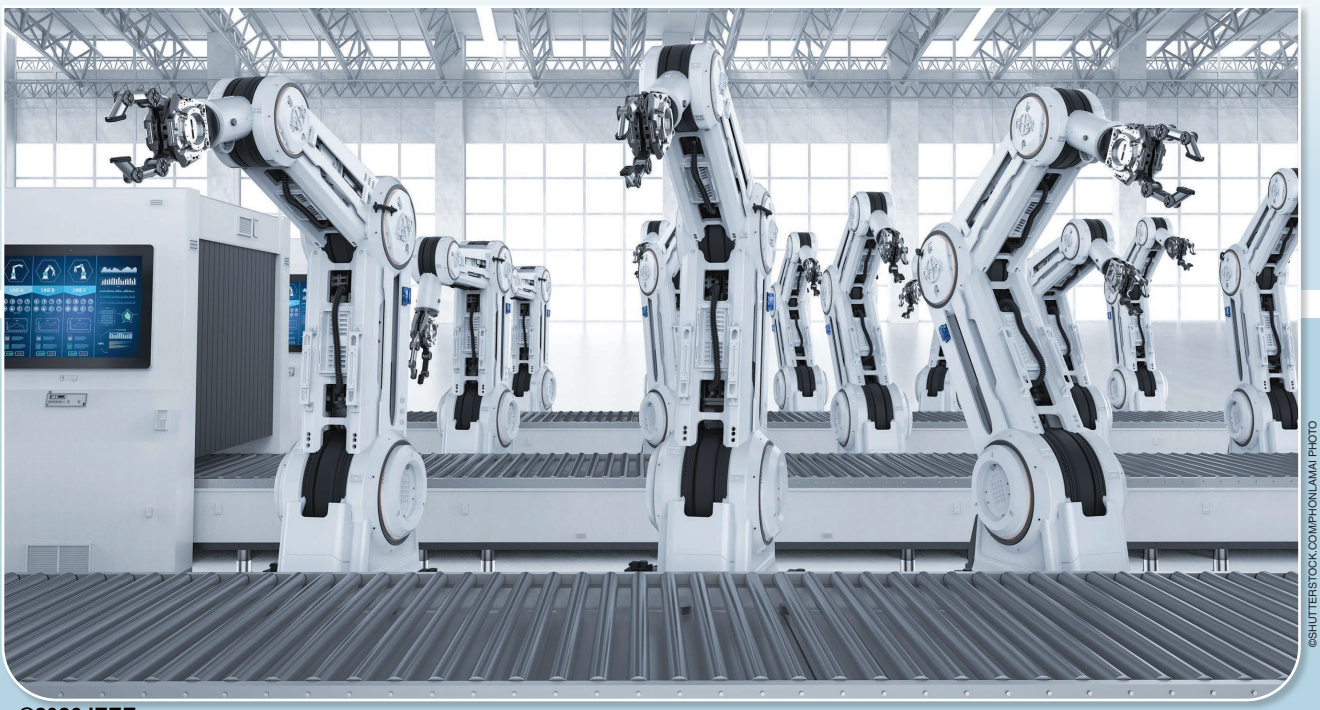
Real-Time Generation of Near Minimum-Energy Trajectories via Constraint-Informed Residual Learning

A Paradigm for Learning From Optimal Solutions

By Domenico Dona', Giovanni Franzese, Cosimo Della Santina, Paolo Boscariol, and Basilio Lenzo

Industrial robotics demands significant energy to operate, making energy-reduction methodologies increasingly important. Strategies for planning minimum-energy trajectories typically involve solving nonlinear optimal control problems (OCPs), which rarely cope with real-time (RT) requirements. In this article, we propose a paradigm for generating near

Digital Object Identifier 10.1109/MRA.2025.3642672



©SHUTTERSTOCK.COM/PHONLAMAI PHOTO

minimum-energy trajectories for manipulators by learning from optimal solutions. Our paradigm leverages a residual learning approach, which embeds boundary conditions (BCs) while focusing on learning only the adjustments needed to steer a standard solution to an optimal one. Compared to a computationally expensive OCP-based planner, our paradigm achieves 87.3% of the performance near the training dataset and 50.8% far from the dataset, while being two to three orders-of-magnitude faster.

INTRODUCTION

The emergence of "green" policies in industry is crucial for mitigating climate change. In the context of robotics, enhancing the efficiency of robotic systems can significantly reduce energy consumption, battery life, and operational costs [1]. This is of particular interest in industrial robotics, as responsible consumption and production is the 12th sustainable development goal.

Trajectory optimization (TO) has emerged as a cost-effective approach for reducing energy consumption of robotic systems, such as single-axis systems [2], quadrotors [3], and manipulators [4]. TO is used to plan trajectories that minimize total energy expenditure by solving the associated OCP. However, closed-form solutions for OCPs are rarely available, requiring the use of numerical routines. Consequently, these numerical routines limit the RT deployment of TO, confining its applicability to precomputed (off-line) trajectories, thus posing a significant limitation in adaptive scenarios.

In this article, we address this limitation by introducing a surrogate model paradigm. Using precomputed optimal trajectories as a training demonstrations, the surrogate model inherently enables RT deployment. Instead of learning the whole solution, we learn only the quantity (or residual) that steers a standard (or prior) solution toward the optimal one, leveraging the idea of *residual learning*, as illustrated in Figure 1. The key idea is that the solution of the planning problem for a fixed-time point-to-point (FT-PTP) motion is rapidly available, e.g., cubic law; what remains is the (residual) component required for achieving optimality.

By proper regressor design, the proposed paradigm: 1) allows the embedding of BCs as hard constraints, 2) requires less training data, and 3) allows deciding the resolution of the solution. Moreover, by using probabilistic regressors, the performance is enhanced by selecting the solution that best satisfies the optimality requirements from multiple samples; the uncertainty measure is used in an active learning (AL) data aggregation routine. The idea is depicted in Figure 2.

In summary, this work proposes an RT capable planner for (near) minimum-energy FT-PTP problems. This is achieved using a residual learning paradigm that allows the embedding of BCs as hard constraints and reduces the number of required training data. The proposed paradigm is tested using two probabilistic regression frameworks, namely neural network (NN) ensembles and Gaussian processes. (GPs) Thanks to its RT capability, the proposed planner can be seamlessly integrated into the robot's control pipeline, enabling energy-efficient motion execution without additional computational overhead. This approach represents an example of embodied artificial intelligence, as the learned model can be embedded into the physical agent to directly guide action [5].

The proposed paradigm is validated through simulated experiments on three robotic systems, namely a pendulum, a SCARA robot, and a six-axis UR5e manipulator.

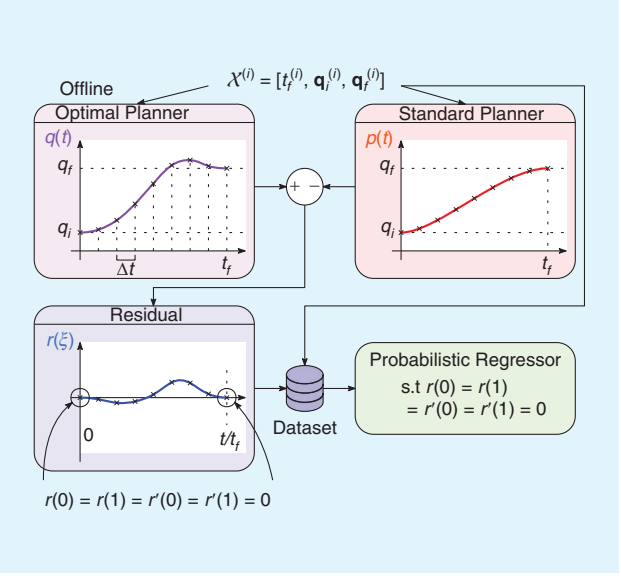


FIGURE 1. Schematic of the proposed residual learning paradigm. Offline, a dataset of residuals is generated and used to train a specifically designed probabilistic regressor that embeds the BCs.

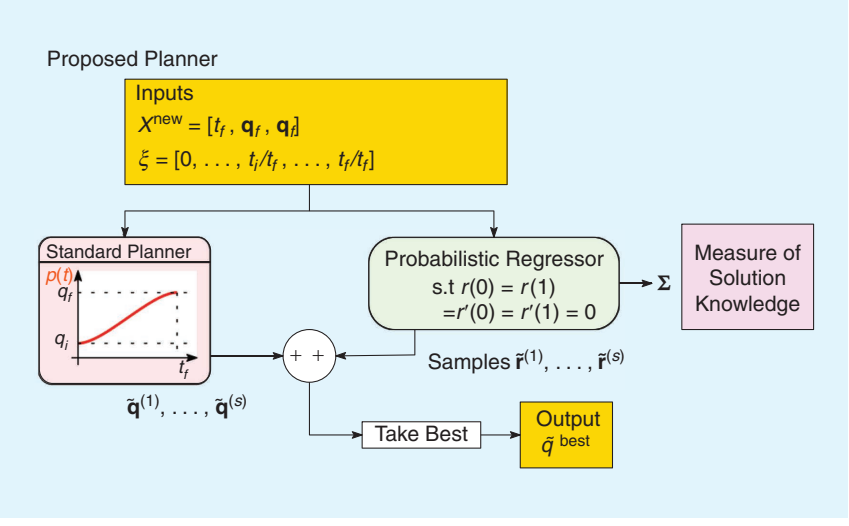


FIGURE 2. The proposed planner: The trained probabilistic regressor is deployed online by summing its output with the standard planner output. Moreover, the regressor outputs the epistemic uncertainty of the solution, that can be used as a measure of the lack of knowledge (data).

RELATED WORKS

Two major families of numerical methods for solving OCP can be distinguished: direct and indirect methods. The former transform an infinity-dimensional optimization problem into a nonlinear programming (NLP) problem. Alternatively, the latter formulate the problem as a two-point boundary value problem (TPBVP) resulting from the Pontryagin maximum principle (PMP).

A main idea to speed up the generation process is to avoid the *numerical routines* and to use surrogate models. An early example can be seen in [6], where precomputed solutions were deployed online for a model predictive control application. However, a clear structure for the solution is not always available. Recently, data-driven methods have emerged as a viable option for speeding up the TO problem [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25]. The proposed regression models are trained to generate optimal trajectories, but different input/output choices are proposed in the literature. As such, we can identify three main categories: 1) indirect models, 2) state feedback models, 3) parameterized solution models. Indirect models leverage the PMP ordinary differential equations to retrieve the solution. Given the initial and final state, the regressor predicts the initial costate and the total time (if not provided as input). Relevant examples are found in [9], [10], and [11] for spacecraft applications. However, to retrieve the trajectory, the TPBVP has to be solved, resulting in possible convergence issues given the quality of the predicted initial costate [26].

State feedback models are based on the fact that the OCP solution is a function of the state. In [12] and [13], the state-feedback map was learned for the optimal landing problem using an NN, while quadcopter applications can be found in [14] and [15]. Different recurrent models can be found in the literature, based on long short-term memory architectures [20] or Transformer models [23], [24], [25]. Note that, these examples do not guarantee BC satisfaction in the hard sense, resulting in possible unfeasible solutions.

Finally, parameterized modules output a complete sequence of the states and/or control inputs given the initial/final state and the total time. The resulting trajectory can also be used as the initial guess to warm up an optimizer [16], [17]. For reducing the dimensionality of the output, an autoencoder can be used to learn a latent representation of the trajectory [19]. However, these approaches suffer from the fact that either the length and/or the resolution of the trajectory has to be fixed to perform the regression.

In light of the above, learning solutions from data is promising for RT applications but existing methods: a) do not impose BCs as hard constraints, b) do not leverage prior knowledge, c) still need numerical routines, or d) have a predetermined length/resolution for the output. The next section introduces our proposed residual learning paradigm.

METHOD

The paradigm can be divided into the following steps:

a) Formulate the minimum-energy problem as an OCP and solve it numerically to create a dataset of residuals.
b) Define a structure for the regressors that 1) embeds the BCs for the residuals and 2) gives uncertainty information of the output.
c) When a new set of problem variables is given, generate a near-optimal solution using the selected regressor and (eventually) evaluate the uncertainty.
d) If needed, upgrade the training set by generating new data based on uncertainty information, i.e., AL.

OCP FORMULATION

We are interested in FT-PTP trajectories for manipulators, as usually task time is decided externally by production constraints. The study case can be formalized as a fixed-time fixed-endpoint OCP. The functional that we want to minimize is the energy expenditure, i.e., the integral over time of the electrical power intake. We take as state the vector of configurations and velocities $\mathbf{x} = (\mathbf{q}, \mathbf{v}) \in \mathbb{R}^{2n}$ and as control the torque exerted by the motors $\mathbf{u} = \boldsymbol{\tau} \in \mathbb{R}^n$, where n is the number of degrees-of-freedom of the system. Using the DC equivalent circuit model, the absorbed electrical power is simply given by $\mathcal{P} = \sum_i^n \mathcal{P}_i = \sum_i^n (r_i u_i^2 + v_i u_i)$, where the first term is due to Joule losses and the second is due to the mechanical power. The ith control weight coefficient is given by $r_i = R_a^i / k_t^{i2}$, where R_a^i is the armature resistance and k_t^i the torque constant of the ith motor, respectively. With the convenient definitions above, it is possible to state the optimal planning problem as follows

$$\underset{\mathbf{x}(\cdot),\mathbf{u}(\cdot)}{\text{minimize}} \quad \varepsilon = \int_0^{t_f} \mathcal{P}(\mathbf{x}(t), \mathbf{u}(t))\, \mathrm{d}t \tag{1a}$$

$$\text{subject to:} \quad \mathbf{x}(0) = \mathbf{x}_0, \quad \mathbf{x}(t_f) = \mathbf{x}_f \tag{1b}$$

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) \tag{1c}$$

where t_f is the prescribed total time, $\mathbf{x}_0 = (\mathbf{q}_0, \mathbf{v}_0)$ and $\mathbf{x}_f = (\mathbf{q}_f, \mathbf{v}_f)$ are the BCs. Since it is a PTP motion, the velocities at the boundaries are zero ($\mathbf{v}_0 = 0$ and $\mathbf{v}_f = 0$). The problem is a nonlinear OCP, as the dynamics $\mathbf{f}$ is (highly) nonlinear. It can be made explicit by splitting the position and velocity components in the usual form

$$\dot{\mathbf{x}} = \begin{cases} \dot{\mathbf{q}} = \mathbf{v} \\ \dot{\mathbf{v}} = \mathbf{M}^{-1}(\mathbf{q})(\mathbf{u} - \mathbf{c}(\mathbf{q}, \mathbf{v}) - \mathbf{g}(\mathbf{q}) - \boldsymbol{\tau}_{\text{fr}}) \end{cases} \tag{2}$$

where $\mathbf{M}$ is the mass matrix, $\mathbf{c}$ is the vector of centrifugal effects, $\mathbf{g}$ is the gravity vector, and $\boldsymbol{\tau}_{\text{fr}}$ is the vector of friction torques.

REGRESSOR DESIGN

The explicit solution of the OCP can be seen as a map from the inputs of the problem, i.e., the time t and the trajectory parameters $\mathcal{X}$ to the trajectory $\mathbf{x}(t)$. In particular, $t \in [0, t_f]$ and $\mathcal{X} = (t_f, \mathbf{q}_0, \mathbf{q}_f)$.

In this work, we parameterize the solution of the OCP in the explicit manner where we focus solely on predicting the

position over time, $\mathbf{q}(t)$, i.e.,

$$\mathbf{q}(t) = \mathbf{g}(t, \mathcal{X}). \tag{3}$$

This choice alleviates the curse of output dimensionality of the regressor; nevertheless, the velocity can still be retrieved numerically or analytically from the position.

Using a dataset of optimal trajectories, one can learn an approximation $\tilde{\mathbf{g}}$ for the mapping $\mathbf{g}$. However, this "naive" approach has the following problems[1]:

1) The trajectories must be learned from scratch.
2) There are no guarantees of BCs satisfaction.
3) It requires a high amount of data.

The above problems can be addressed using the *residual learning* paradigm presented in this article. A schematic of the paradigm is depicted in Figure 1.

RESIDUAL LEARNING OF EXPLICIT SOLUTIONS

The key idea is that we know how to plan an FT-PTP trajectory, but we do not know how to plan it optimally. Therefore, we want to learn only the residual component that steers our standard (prior) solution toward the optimal one. This reduces the burden of the regressor by learning only a part of the solution. Mathematically, let us define the *prior* as $\mathbf{p}(t)$ and the *residual* as $\mathbf{r}(t)$

$$\mathbf{q}(t) = \mathbf{p}(t) + \mathbf{r}(t) \tag{4}$$

where the prior $\mathbf{p}(t)$ can be any continuous function that satisfies the BCs in (1b), for instance a proper cubic polynomial.

BOUNDARY CONDITIONS SATISFACTION

Since the prior satisfies the BCs, the residual must have zero values at the boundaries, i.e.,

$$\mathbf{r}(0) = \mathbf{r}(t_f) = \dot{\mathbf{r}}(0) = \dot{\mathbf{r}}(t_f) = \mathbf{0}. \tag{5}$$

At this point, even if the values at the boundaries are the same, their position depends on the value of t_f. For overcoming this issue, we can define the normalized time ξ as the ratio between the physical time t and the total time t_f, such that the boundaries are at $\xi = 0$ and $\xi = 1$. The mapping $\mathbf{r}(\cdot)$ can be approximated using a regressor $\tilde{\mathbf{r}}(\xi, \mathcal{X})$ that, according to (5), must satisfy the following:

$$\tilde{\mathbf{r}} \,|\, \tilde{\mathbf{r}}(0) = \tilde{\mathbf{r}}(1) = \tilde{\mathbf{r}}'(0) = \tilde{\mathbf{r}}'(1) = \mathbf{0} \tag{6}$$

where $\square'$ denotes the partial derivative with respect to ξ. The second argument has been omitted for brevity. Depending on the type of nonlinear regressor chosen, different techniques can be applied to satisfy (6). In the following, NNs and GPs are considered, as they are two of the most established nonlinear regressors.

To enforce property (6) using an NN, it is sufficient to multiply the output of the NN element-wise by a "scaling term" s, as shown in Figure 3. One possible choice for s is

$$s(\xi) = \xi^2 (1 - \xi)^2 \tag{7}$$

and calling $\mathbf{o}$ the output of the NN, then the regressor

$$\tilde{\mathbf{r}}(\xi, \mathcal{X}) = s(\xi) \odot \mathbf{o}(\xi, \mathcal{X}) \tag{8}$$

satisfies (6). The symbol $\odot$ is used for element-wise multiplication.

Recalling the objectives outlined at the beginning of the "Methods" section, at this stage we lack uncertainty information for the output. This issue can be addressed by using an ensemble of NNs, providing two main benefits: 1) estimation of the epistemic uncertainty using the standard deviation of outputs, and 2) selection of the best solution (in energy sense) among the set of predictions. The ensemble approach introduces only a minor computational cost, as it can be easily parallelized.

Another common nonlinear regression model is given by GPs. Similar to NNs, GPs do not satisfy property (6) out-of-the-box. However, although the process is less straightforward, one can modify the GP kernel to enforce it. To do so, we propose taking a general-purpose kernel $k_g(\cdot,\cdot)$, such as a radial basis function, and multiplying it by two "scaling" functions $s(\cdot)$ that depends only on the normalized time ξ. This custom kernel is defined as

$$k(\xi_1, \xi_2) = s(\xi_1) k_g(\xi_1, \xi_2) s(\xi_2) \tag{9}$$

with $s(\xi)$ being the scaling function introduced in (7). Here, ξ_1 and ξ_2 are the normalized time points at which the function is evaluated. It is easy to prove that the kernel (9) satisfies (6). Although the kernel degenerates at the boundaries, this does not cause numerical issues in practice because: 1) during training the noise measurement stabilizes the Cholesky decomposition, 2) the mean prediction does not require any matrix inversion, and 3) the jitter term included by default in GPyTorch stabilizes the Cholesky decomposition

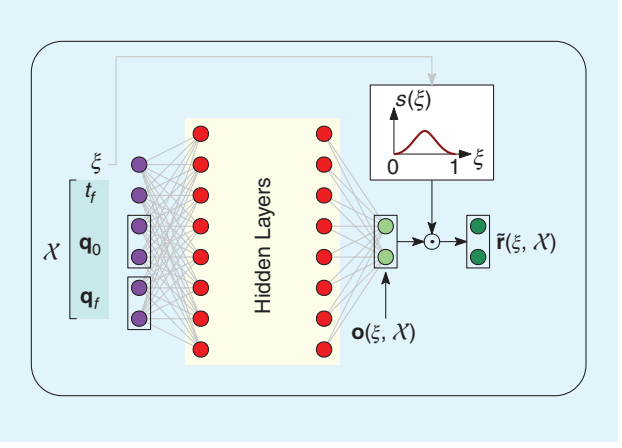


FIGURE 3. Proposed NN scheme. The scaling function is applied element-wise.

[1]An ablation study to demonstrate the statement is reported in the "Ablation Study" section.

during sampling. This jitter introduces a negligible violation at the boundaries, which is well below the resolution of standard robotic systems. Unlike NNs, where an ensemble was needed to estimate uncertainty, this regressor inherently provides epistemic uncertainty information. Additionally, multiple solutions are sampled from the distribution, and again the one with the lowest energy consumption is selected.

ACTIVE LEARNING

Our paradigm reduces data quantity by enforcing structure through the prior. However, solution quality may vary across the input space due to limited data, as data generation is costly. However, we can query the optimal planner as needed to update the dataset. We propose employing an AL strategy based on the idea that the further the input is from the existing dataset, the higher the uncertainty in the output. As a result, the epistemic uncertainty of the output serves as a measure of the quality of the solution, allowing us to generate new data where the uncertainty is higher. The idea is depicted in Figure 4.

The uncertainty is retrieved out-of-the-box for the GP, while for the ensemble of NN, the variance of the outputs is used as uncertainty. The way new data are incorporated differs significantly between the types of regressors used. GPs can easily incorporate new data by adding the new data to the inducing points. On the other hand, NNs do not possess the same flexibility when it comes to incorporating new information. In this work, we employed fine-tuning, by training the model for additional epochs while enriching the original training batches with the new data. Clearly, the AL process cannot be performed online, as the generation of data are costly.

RESULTS

DATASETS

The models designed in the "Methods" section were compared across three different mechanical systems, namely an actuated pendulum, a SCARA robot, and a six-axis manipulator. The datasets of residuals were generated using the Rockit [27] OC framework and the IPOPT [28] NLP solver.

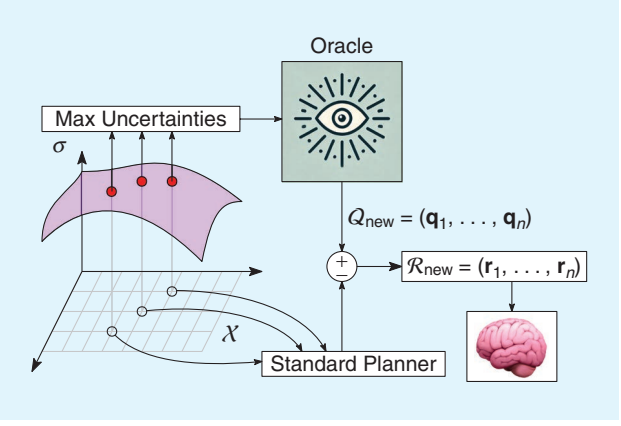


FIGURE 4. AL approach: The uncertainty of the solution is used to generate new data.

The dynamics of the pendulum system was modeled using Casadi [29], while the Pinocchio library [30] was employed for the robotic manipulators. The standard (or prior) planner was chosen as a cubic polynomial law for its simplicity.

The SCARA dataset was constructed by noting the system's symmetry with respect to the first initial joint, which made it possible to reduce the dataset dimension.

Finally, a dataset of a six-axis manipulator was constructed. The universal robot UR5e was chosen, as its dynamical parameters are well studied in the literature and its model can be trusted in simulation. The latter model is of particular interest as its model is representative of an industrial robot and the large number of states make the minimum-energy problem numerically intensive. In this case, we stressed the proposed paradigm while leveraging prior knowledge of the workspace. In particular, no simplification was made for the first joint symmetry. At the same time, the initial and final configurations were chosen by discretizing the workspace in a grid and then retrieving the joint values by means of inverse kinematics. The orientation of the initial and final pose was chosen equal in all of the trajectories as well as the configuration along the trajectory.[2] The details on the number of trajectories, data, and sampling frequency are summarized in Table 1. We will refer to these data as the *inside dataset*, while data outside these boundaries will be referred to as the *outside dataset*.

The dynamic parameters of the first two systems are in the supplementary information (supplementary downloadable material available at https://doi.org/10.1109/MRA.2025.

[2]The inverse kinematics generally has eight solutions for the UR5e away from singularities. We use the same righty-below-noflip.

TABLE 1. Details of the training datasets' construction.

	PENDULUM	SCARA	UR5e
No. data	808	43,344	33,658
No. trajectories	8	144	231
Sampling frequency (Hz)	100	100	50
Ranges t_f (s)	[1.0, 1.5]	[2.5, 3.5]	[2.5, 3.0]
No. samples t_f	2	3	2
Ranges $\mathbf{q}_i$ (rad or m)	$[-\pi/4, \pi/4]$	[0, 0]	[0.3, 0.5]
		$[0, (3/4)\,\pi]$	[0.0, −0.5]
			[0.0, 0.5]
No. samples $\mathbf{q}_i$	4	1 × 4	3 × 3 × 3
Ranges $\mathbf{q}_f$ (rad or m)	$[-\pi/4, \pi/4]$	$[0, \pi]$	[0.3, 0.5]
		$[0, (3/4)\,\pi]$	[0.0, 0.5]
			[0.0, 0.5]
No. samples $\mathbf{q}_f$	4	4 × 4	3 × 3 × 3
Additional requirement	$\lvert q_f \rvert > \lvert q_i \rvert$	$q_{f,2} > q_{i,2}$	$\Sigma_i \lvert \Delta x_i \rvert > 0.1$
	$\lvert \Delta q \rvert > \pi/16$		$z_f - z_i > 0$

For the UR5e, the $\mathbf{q}_i$ quantities are defined as (x, y, z) (with abuse of notation) positions of the end effector.

3642672, provided by the authors). For the six-axis robot, the dynamical parameters were taken from the company's official Unified Robot Description Format, while the friction parameters were taken from [31].

REGRESSORS SETTINGS

The hyperparameters for the three test cases are summarized in Table 2 for the NNs; in all cases, AdamW was used as optimizer [32] and the activation functions are tanh. The ensemble has six models for the UR5e case and 10 for the other two. For the GPs, we used a stochastic variational GP due to the large amount of data; the Adam optimizer was used with a learning rate of 10^{-2} for 100 epochs, and 100 inducing points were selected. The number of samples was 100 for the pendulum and 10 for the other two cases. During training, we split the data into 80% for training and 20% for testing. Timing results are obtained using a laptop equipped with an AMD R9 8945HS CPU, 32 GB of RAM, and an NVIDIA RTX 4060 GPU. The NNs are developed using PyTorch [33], while the GPs were implemented using GPyTorch [34].

TABLE 2. Common hyperparameters of the NNs.

TEST CASE	LR	WD	# HLs	HL WIDTH
Pendulum	10^{-3}	10^{-2}	2	50
SCARA	10^{-3}	10^{-3}	2	50
UR5e	10^{-3}	10^{-1}	2	100

LR: learning rate; WD: weight decay.; HL: hidden layer.

DISCUSSION

ABLATION STUDY

First of all, to evaluate the superiority of the residual learning paradigm, an ablation study was performed. This study aims to demonstrate that a standard GP or NN generate unfeasible results. Specifically, Figure 5(a) illustrates a trajectory example trained with a vanilla NN and a vanilla GP, showing boundary violations on training data. Statistical evidence is provided in Figure 5(b), where the two models were evaluated across the entire dataset. On average, the violations inside the dataset, considering only the position, amount to about 0.2 rad for the NN and 0.3 rad for the GP, suggesting a lack of feasibility.

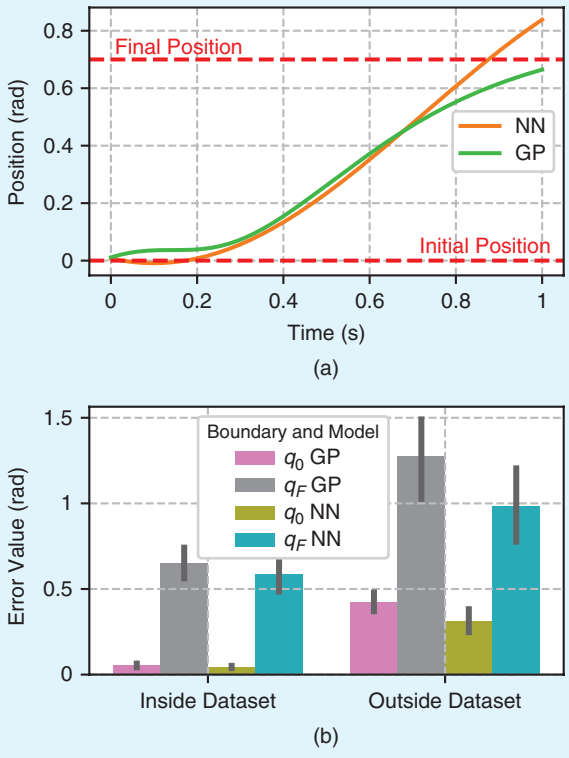


FIGURE 5. (a) Example trajectories of the NN and GP vanilla models and (b) bar plot of violations of the boundaries inside and outside the dataset for the NN and GP vanilla models. The violations of the BCs are evident.

SAMPLING EFFECT

The energy savings for all considered systems (the pendulum, SCARA, and UR5e) compared to the standard solutions are shown in Figures 6 and 7. The figures present both the best samples from the NN ensembles and GP sampling, as well as the corresponding mean results, μ_{NN} and μ_{GP}. Notably, the violin plots illustrate that picking the best sample tends to shift the distribution of energy savings toward higher values, highlighting the superiority of the sampling feature of probabilistic regressors. A quantitative summary is provided in Table 3.

ENERGY SAVING CAPABILITIES

Remarkably, both the NN- and GP-based models enable energy savings inside and outside of the dataset. A comparison across the robot test cases is depicted in Figure 7, alongside the results for the pendulum in Figure 6. A summary table is provided in Table 3. Overall, the NN-based model performs slightly better than the GP-based model. Further discussion on the comparison of the two models is provided in the "Active Learning" section that follows.

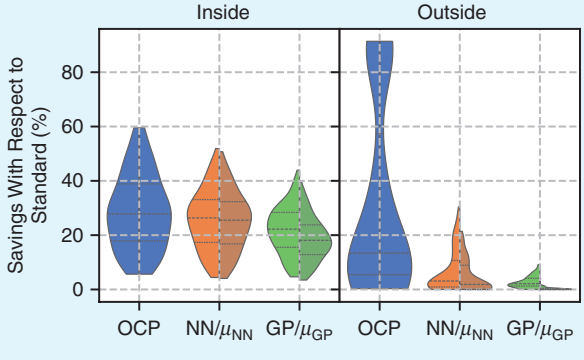


FIGURE 6. Inside and outside savings with respect to the standard solution for the pendulum. The right side of the violin plot represents the output mean of the regressors, showing the advantage of sampling.

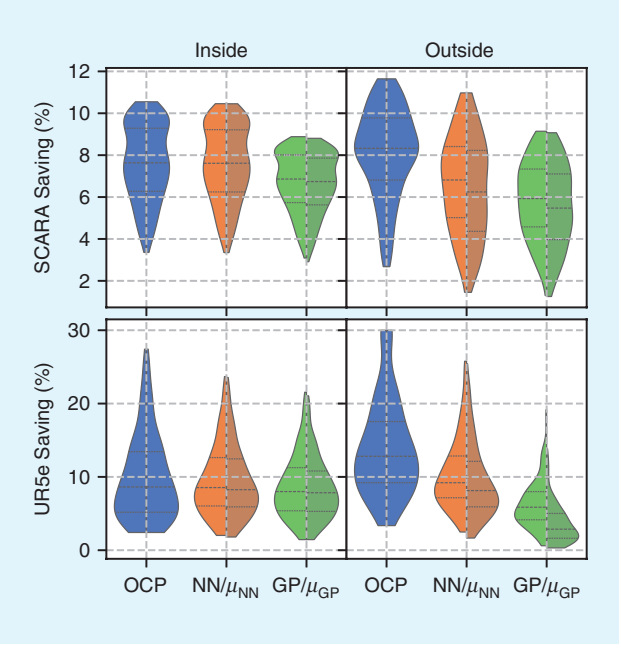


FIGURE 7. Energy savings with respect to the standard solution for the manipulators. Brighter-colored violins indicate the savings of the models' mean predictions, while darker violins represent the best individual samples. Overall, sampling achieves superior energy savings.

RT CAPABILITY

As explained in the Introduction, the main challenge addressed in this work is generating the solution in RT, as standard optimal control frameworks do not meet this requirement. The times required to compute the solution are reported in Figure 8, where the RT limit is set as a 10th of the average task time t_f. Notably, both models guarantee RT capabilities for the UR5e case.

Notably, the convergence time for the OCP is one to three orders-of-magnitude greater than the RT limit. Relevant literature [15], [17], [25] shows that warm-start techniques can accelerate convergence by up to one order-of-magnitude (at best) excluding the inference time required to compute the warm-start guess. It is also worth noting that warm-start models with dimensionality comparable to the proposed ones may not leave sufficient time for the optimization routine to even begin. Therefore, although warm-starting can improve solver performance, it is not suitable in this case, as the time requirement is too tight even for inference.

TABLE 3. Percentual savings or performance with respect to the standard solution for the pendulum, SCARA, and UR5e robots.

SYSTEM	DATASET	OCP	NN	μNN	GP	μGP
Pendulum	Inside	28.5	25.8	25.2	21.8	18.5
Pendulum	Outside	31.6	6.8	5.0	2.8	0.4
SCARA	Inside	7.61	7.56	7.56	6.71	6.6
SCARA	Outside	8.09	6.69	6.26	5.91	5.51
UR5e	Inside	9.95	9.52	9.34	8.69	8.54
UR5e	Outside	14.25	10.51	9.37	6.36	3.64

ACTIVE LEARNING

As discussed in the "Energy Saving Capabilities" section, the NN-based model slightly outperforms the GP-based one. The main difference between the two models, as introduced in the "Active Learning" section of the "Methods," is in the way they can incorporate new data. For this reason, the four cases with the higher uncertainty of the outside dataset were used to update the models, for the pendulum system. The savings before and after the AL phase are reported in Table 4.

Notably, the GP can incorporate data more efficiently, and is much easier since the ensemble of NNs requires additional training for each model. In particular, 200 more epochs with a learning rate of 10^{-4} were performed for each model. On the other hand, before AL, the NN-based model performs better. This suggests that depending on the deployment phase, one can choose one over the other.

EXAMPLES

Simulated examples are provided to illustrate differences between the two regressors. In addition to the regressor solutions, the prior (Cubic) and the OCP solution are shown.

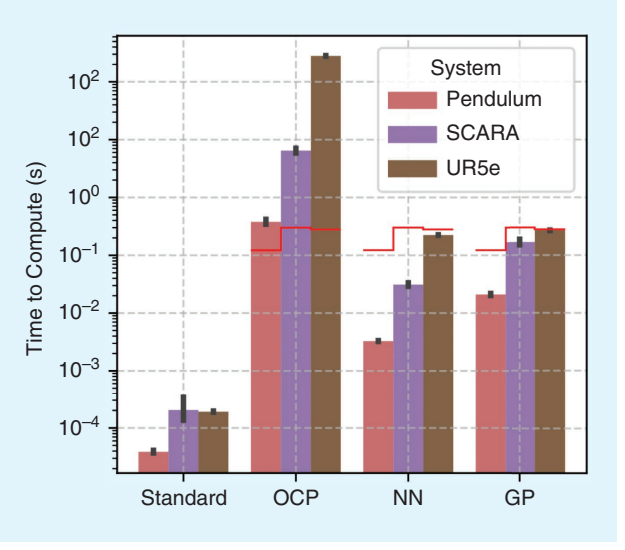


FIGURE 8. Computation times for each model in seconds. The piecewise constant red line represents the RT limit, defined as one-tenth of the total task time, t_f. The y scale is logarithmic.

TABLE 4. Savings with respect to the standard solution before and after the AL phase.

CASE	PRE-AL (%)	POST-AL (%)	OCP REFERENCE (%)
GP	2.98	50.93	53.84
NN	17.63	49.65	55.46

Within the dataset, the regressors closely follow the OCP solution, while outside the dataset, greater variability is observed, with the GP solution tending toward the prior. The results are reported for each testcase, both inside and outside the dataset, and for each model: in Figure 9 for the pendulum and in Figure 10 for the SCARA and UR5e robots.

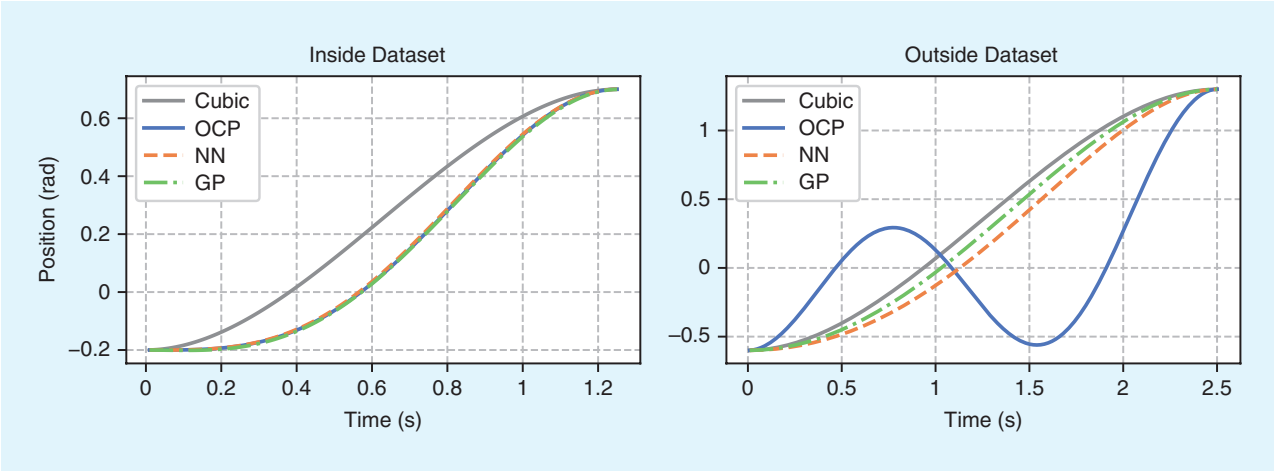


FIGURE 9. Comparison of example trajectories for the pendulum, both within and outside the dataset. Inside the dataset, the regressor accurately captures the solution, while outside the dataset, the complexity of the solution is less represented. Notably, the GP solution tends to align with the prior.

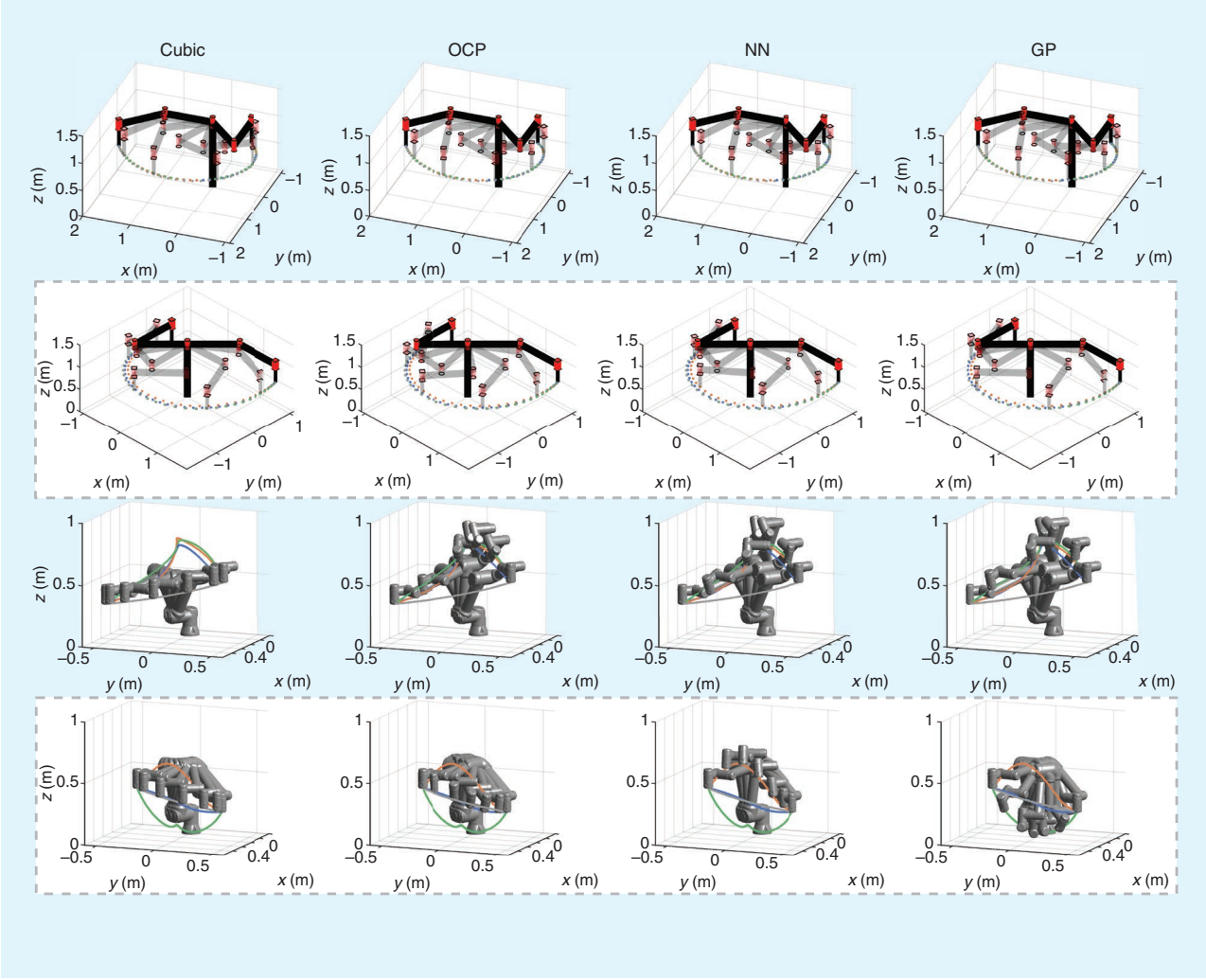


FIGURE 10. Comparison of example trajectories for the robotic manipulators. The grey dashed boxes indicate regions outside the dataset.

CONCLUSIONS

In this article, by focusing only in the residual between an optimal planner solution and a given standard planner, the objective of learning optimal solutions has been achieved. Through numerical simulation, the advantage of the proposed paradigm has been shown to be twofold: it is possible to embed BCs while also reducing the amount of needed training data.

By using variational regressors, it is possible to select the best output from the samples and using uncertainty information to upgrade the training dataset. Between the two regressor frameworks used, GPs turned out to be more flexible in incorporating new data during the AL phase, while the ensemble of NNs demonstrated better results before adding new evidence.

Future work will consider the inclusion of other types of constraints, such as torque or speed limits, to make the paradigm applicable in more demanding situations.

ACKNOWLEDGMENT

This research was partially funded by Fondazione Aldo Gini (Call 2023). The work was carried out at the Delft Technical University. Domenico Dona' acknowledges the financial support of the Erasmus+ program. This article has supplementary downloadable material available at https://doi.org/10.1109/MRA.2025.3642672, provided by the authors.

AUTHORS

Domenico Dona', Department of Management and Engineering, University of Padua, 36100 Vicenza, Italy. E-mail: domenico.dona@unipd.it.

Giovanni Franzese, Department of Cognitive Robotics, Delft University of Technology, 2628 CD Delft, The Netherlands. E-mail: franzesegiovanni@outlook.com.

Cosimo Della Santina, Department of Cognitive Robotics, Delft University of Technology, 2628 CD Delft, The Netherlands; and Institute of Robotics and Mechatronics, German Aerospace Center, 82234 Oberpfaffenhofen, Germany. E-mail: c.dellasantina@tudelft.nl.

Paolo Boscariol, Department of Management and Engineering, University of Padua, 36100 Vicenza, Italy. E-mail: paolo.boscariol@unipd.it.

Basilio Lenzo, Department of Industrial Engineering, University of Padua, 35131 Padova, Italy. E-mail: basilio.lenzo@unipd.it.

REFERENCES

[1] G. Carabin, E. Wehrle, and R. Vidoni, "A review on energy-saving optimization methods for robotic and automatic systems," *Robotics*, vol. 6, no. 4, 2017, Art. no. 39, doi: 10.3390/robotics6040039.

[2] D. Dona', B. Lenzo, P. Boscariol, G. Rosati, "A real-time capable method for planning minimum energy trajectories for one degree-of-freedom mechatronic systems," *Control Eng. Pract.*, vol. 142, Jan. 2024, Art. no. 105766, doi: 10.1016/j.conengprac.2023.105766.

[3] F. Morbidi and D. Pisarski, "Practical and accurate generation of energy-optimal trajectories for a planar quadrotor," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, Piscataway, NJ, USA: IEEE Press, 2021, pp. 355–361, doi: 10.1109/ICRA48506.2021.9561395.

[4] G. Field and Y. Stepanenko, "Iterative dynamic programming: An approach to minimum energy trajectory planning for robotic manipulators," in *Proc. IEEE Int. Conf. Robot. Automat.*, Piscataway, NJ, USA: IEEE Press, 1996, pp. 2755–2760, doi: 10.1109/ROBOT.1996.506579.

[5] Y. Liu et al., "Aligning cyber space with physical world: A comprehensive survey on embodied AI," *IEEE/ASME Trans. Mechatron.*, early access, Jul. 28, 2025, doi: 10.1109/TMECH.2025.3574943.

[6] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, "The explicit solution of model predictive control via multiparametric quadratic programming," in *Proc. Am. Control Conf. ACC (IEEE Cat. No. 00CH36334)*, Piscataway, NJ, USA: IEEE Press, 2000, vol. 2, pp. 872–876, doi: 10.1109/ACC.2000.876624.

[7] K. Hauser, "Learning the problem-optimum map: Analysis and application to global optimization in robotics," *IEEE Trans. Robot.*, vol. 33, no. 1, pp. 141–152, Feb. 2017, doi: 10.1109/TRO.2016.2623345.

[8] R. Sambharya, G. Hall, B. Amos, and B. Stellato, "End-to-end learning to warm-start for real-time quadratic optimization," in *Proc. Learn. Dyn. Control Conf.*, PMLR, 2023, pp. 220–234.

[9] L. Cheng, Z. Wang, F. Jiang, and C. Zhou, "Real-time optimal control for spacecraft orbit transfer via multiscale deep neural networks," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 55, no. 5, pp. 2436–2450, Oct. 2019, doi: 10.1109/TAES.2018.2889571.

[10] Y. Zang, J. Long, X. Zhang, W. Hu, E. Weinan, and J. Han, "A machine learning enhanced algorithm for the optimal landing problem," in *Proc. Math. Sci. Mach. Learn.*, PMLR, 2022, pp. 319–334.

[11] G. Tang and K. Hauser, "A data-driven indirect method for nonlinear optimal control," *Astrodynamics*, vol. 3, no. 4, pp. 345–359, 2019, doi: 10.1007/s42064-019-0051-3.

[12] C. Sánchez-Sánchez and D. Izzo, "Real-time optimal control via deep neural networks: Study on landing problems," *J. Guid., Control, Dyn.*, vol. 41, no. 5, pp. 1122–1135, 2018, doi: 10.2514/1.G002357.

[13] L. Zhu, J. Ma, and S. Wang, "Deep neural networks based real-time optimal control for lunar landing," *IOP Conf. Ser.: Mater. Sci. Eng.*, vol. 608, no. 1, 2019, Art. no. 012045, doi: 10.1088/1757-899X/608/1/012045.

[14] D. Tailor and D. Izzo, "Learning the optimal state-feedback via supervised imitation learning," *Astrodynamics*, vol. 3, no. 4, pp. 361–374, 2019, doi: 10.1007/s42064-019-0054-0.

[15] R. Ferede, G. de Croon, C. De Wagter, and D. Izzo, "End-to-end neural network based optimal quadcopter control," *Rob. Auton. Syst.*, vol. 172, Feb. 2024, Art. no. 104588, doi: 10.1016/j.robot.2023.104588.

[16] G. Tang, W. Sun, and K. Hauser, "Learning trajectories for real-time optimal control of quadrotors," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Piscataway, NJ, USA: IEEE Press, 2018, pp. 3620–3625, doi: 10.1109/IROS.2018.8593536.

[17] S. Banerjee et al., "Learning-based warm-starting for fast sequential convex programming and trajectory optimization," in *Proc. IEEE Aerosp. Conf.*, Piscataway, NJ, USA: IEEE Press, 2020, pp. 1–8, doi: 10.1109/AERO47225.2020.9172293.

[18] Y. Wu, X. Sun, I. Spasojevic, and V. Kumar, "Deep learning for optimization of trajectories for quadrotors," *IEEE Robot. Autom. Lett.*, vol. 9, no. 3, pp. 2479–2486, Mar. 2024, doi: 10.1109/LRA.2024.3357399.

[19] M. Berniker and K. P. Kording, "Deep networks for motor control functions," *Front. Comput. Neurosci.*, vol. 9, Mar. 2015, Art. no. 32, doi: 10.3389/fncom.2015.00032.

[20] M. J. Bency, A. H. Qureshi, and M. C. Yip, "Neural path planning: Fixed time, near-optimal path generation via oracle imitation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Macau, China, 2019, pp. 3965–3972, 2019, doi: 10.1109/IROS40897.2019.8968089.

[21] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.

[22] A. Vaswani, "Attention is all you need," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst. (NIPS)*, 2017, pp. 6000–6010.

[23] T. Guffanti, D. Gammelli, S. D'Amico, and M. Pavone, "Transformers for trajectory optimization with application to spacecraft rendezvous," in *Proc. IEEE Aerosp. Conf.*, Piscataway, NJ, USA: IEEE Press, 2024, pp. 1–13, doi: 10.1109/AERO58975.2024.10521334.

[24] D. Celestini, D. Gammelli, T. Guffanti, S. D'Amico, E. Capello, and M. Pavone, "Transformer-based model predictive control: Trajectory optimization via sequence modeling," *IEEE Robot. Autom. Lett.*, vol. 9, no. 11, pp. 9820–9827, Nov. 2024, doi: 10.1109/LRA.2024.3466069.

[25] J. Briden, C. Choi, K. Yun, R. Linares, and A. Cauligi, "Constraint-informed learning for warm starting trajectory optimization," 2023, *arXiv:2312.14336*.

[26] A. E. Bryson, *Applied Optimal Control: Optimization, Estimation and Control*. New York, NY, USA: Routledge, 2018.

[27] J. Gillis, B. Vandewal, G. Pipeleers, and J. Swevers, "Effortless modeling of optimal control problems with rockit," in *Proc. 39th Benelux Meeting Syst. Control*, Elspeet, The Netherlands, vol. 138, 2020.

[28] L. T. Biegler and V. M. Zavala, "Large-scale nonlinear programming using IPOPT: An integrating framework for enterprise-wide dynamic optimization," *Comput. Chem. Eng.*, vol. 33, no. 3, pp. 575–582, 2009, doi: 10.1016/j.compchemeng.2008.08.006.

[29] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi: A software framework for nonlinear optimization and optimal control," *Math. Program. Comput.*, vol. 11, no. 1, pp. 1–36, 2019, doi: 10.1007/s12532-018-0139-4.

[30] J. Carpentier, R. Budhiraja, and N. Mansard, "Proximal and sparse resolution of constrained dynamic equations," in *Proc. Robot.: Sci. Syst.*, 2021, doi: 10.15607/RSS.2021.XVII.017. [Online]. Available: https://hal.inria.fr/hal-03271811

[31] E. Clochiatti, L. Scalera, P. Boscariol, and A. Gasparetto, "Electro-mechanical modeling and identification of the UR5 e-series robot," *Robotica*, vol. 42, no. 7, pp. 2430–2452, 2024, doi: 10.1017/S0263574724000833.

[32] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," 2017, *arXiv:1711.05101*.

[33] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in *Proc. 33rd Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 8026–8037.

[34] J. Gardner, G. Pleiss, K. Q. Weinberger, D. Bindel, and A. G. Wilson, "GPyTorch: Blackbox matrix-matrix Gaussian process inference with GPU acceleration," in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst. (NIPS)*, 2018, pp. 7587–7597.