Learning uncertainties online for quadrotor flight control: A comparative study

Journal of Intelligent & Robotic Systems (2025) 111:98
https://doi.org/10.1007/s10846-025-02305-5

REGULAR PAPER

Learning uncertainties online for quadrotor flight control: A comparative study

Weibin Gu[1] · **Jiance Zhao**[2] · **Alessandro Rizzo**[3]

Received: 25 October 2024 / Accepted: 19 August 2025

Abstract
Traditional adaptive and robust control methods sometimes struggle to address the complex and dynamic uncertainties inherent in quadrotor flight, frequently resulting in overly conservative or unstable behavior due to reliance on presumed uncertainty bounds. To overcome these limitations, we propose a unified online-learning-based control framework that integrates data-driven uncertainty estimation with a nominal flight controller. Furthermore, we systematically evaluate four representative learning methods, namely Extreme Learning Machines (ELMs), Radial Basis Function Neural Networks (RBFNNs), Echo State Networks (ESNs), and Gaussian Processes (GPs), spanning both parametric and non-parametric techniques. Through extensive simulation and hardware experiments, we benchmark these methods in terms of (i) uncertainty estimation accuracy, (ii) trajectory tracking performance, and (iii) real-time computational efficiency. Our results reveal key trade-offs: GPs yield highest accuracy but require approximations for real-time use. ELMs and RBFNNs enable fast inference but suffer initialization sensitivity. ESNs provide the best balance, offering both stability and real-time performance. This work underscores the critical role of model selection in learning-based flight control, providing insights for deploying these algorithms in resource-constrained aerial systems operating under uncertainty.

Keywords Online learning · Uncertainty estimation · Trajectory tracking · Quadrotor

1 Introduction

Addressing uncertainties in quadrotor flight presents a significant challenge within the control community. This is mainly due to the inherent complexities involved in designing controllers for tasks such as trajectory tracking and attitude stabilization [1]. These challenges arise from the nonlinearities and strong coupling in quadrotor dynamics, further complicated by uncertainties and external disturbances [2–4]. The difficulty is exacerbated when quadrotors must perform rapid or aggressive maneuvers [5] or operate in dynamic environments [6].

Over the past decade, traditional adaptive and robust control methods have been well-developed to address various uncertainties in quadrotor flight. These uncertainties include inaccuracies in mass and inertia matrix estimation, unknown wind gusts, and complex aerodynamic effects [7–13]. These methods typically rely on predefined bounds for uncertainties to ensure system stability and performance. However, as the complexity of tasks and the nature of uncertainties continue to escalate, they often become overly conservative, sacrificing performance to maintain stability, or they may become unstable when the actual uncertainties exceed the assumed bounds [14]. This inherent limitation underscores the growing potential of data-driven approaches in flight controller design.

This article belongs to the Topical Collection: *Topical Collection I, ICUAS 2024*.

✉ Alessandro Rizzo
alessandro.rizzo@polito.it

Weibin Gu
guweibin@air.tsinghua.edu.cn

Jiance Zhao
Zhaojiance@buaa.edu.cn

[1] Institute for AI Industry Research (AIR), Tsinghua University, Beijing 100084, PR China

[2] School of Aeronautic Science and Engineering, Beihang University, Beijing 100191, PR China

[3] Department of Electronics and Telecommunications, Politecnico di Torino, Turin 10129, Italy

Published online: 03 September 2025

Data-driven methods, with their strong approximation capabilities, are highly effective for modeling and compensating for complex uncertainties [15]. In this realm, *offline learning* techniques like supervised learning and Reinforcement Learning (RL), which utilize historical or simulation data, have demonstrated significant potential in scenarios where a known distribution is assumed for full dynamics [16–18] or for uncertainties [19–23]. These approaches excel at learning and predicting system behaviors within the training data's distribution, resulting in robust performance without distribution shifts. However, real-world flight scenarios often present challenges with Out-Of-Distribution (OOD) uncertainties not captured in the offline training. In such situations, the performance of offline methods can degrade, leading to suboptimal or unsafe flight behavior [24]. Thus, *adaptive or online learning* methods are crucial; they can continuously update and refine learned models in real-time, effectively compensating for emerging uncertainties during flight. This adaptability is essential for maintaining the robustness and reliability of quadrotor operations in dynamic and unstructured environments.

Online modeling and control techniques are increasingly indispensable for robotic systems operating under complex nonlinearities and unknown dynamics. They are essential for enhancing the generalizability and adaptability of robots across diverse and dynamic environments. For example, Koopman operator theory has been integrated into Model Predictive Control (MPC) for online model estimation, demonstrating its effectiveness in enabling precise and adaptive control of soft multi-fingered grippers [25]; similarly, a knowledge-based neural Ordinary Differential Equation (ODE) framework has been introduced for continual updating of dynamic models in MPC, facilitating better adaptation to disturbances and enhancing quadrotor trajectory tracking [26]. In addition to learning the full dynamics of the system under control, online learning has also been applied to partial dynamics such as wind gusts and ground effect acting on quadrotors [24, 27] as well as model mismatches within adaptive cruise control systems [28]. These examples collectively highlight the critical role of online learning methods in improving robustness, adaptability, and overall performance of robotic systems across a broad spectrum of applications, ranging from aerial robotics to autonomous driving and beyond.

As the emphasis on online learning in control systems continues to grow, a fundamental question arises: *which online learning method is most appropriate for specific applications?* There are a variety of approaches suitable for online learning tasks [29], ranging from parametric to non-parametric models, each offering distinct advantages and limitations. *Parametric models* are characterized by a fixed number of parameters, with Artificial Neural Networks (ANNs) serving as prominent examples. These models can be further divided by architecture, such as feedforward networks like Extreme Learning Machines (ELMs) [30] and Radial Basis Function Neural Networks (RBFNNs) [31], which rely on unidirectional data flow, and recurrent networks like Echo State Networks (ESNs) [32], which incorporate temporal dynamics via feedback loops. In contrast, *non-parametric models* offer greater flexibility as their complexity grows with data, with Gaussian Processs (GPs) [33] being a typical example, providing a probabilistic framework and the ability to quantify uncertainty in predictions. Despite the growing body of literature on these methods, many studies adopt a single approach without clear justification for its selection, leading to a practical gap in this field. A comprehensive benchmarking of different online learning methods under consistent conditions is lacking, which is essential for systematically comparing the strengths and limitations of these techniques in conjunction with controller design. Such efforts would ultimately guide the selection of the most effective method for real-time control applications.

In this work, we propose an online-learning-based control framework, building on our earlier study [24]. The framework is designed to be modular, allowing for easy integration of other online learning methods beyond those discussed in this paper. We conduct a comprehensive evaluation of multiple online learning techniques for uncertainty estimation, integrated with a nominal flight controller. The methods examined include ELM, RBFNN, ESN, and GP, representative of a spectrum of both parametric and non-parametric models. To ensure fair comparison among parametric models, Bayesian Optimization (BO) is adopted for hyperparameter selection, thereby eliminating manual bias in configuration and guaranteeing that each method operates near its best achievable performance. We carry out extensive simulation campaigns for performance assessment, focusing on the benefits and limitations of each method. The evaluation is structured around three key performance metrics: (i) uncertainty estimation error, (ii) computational efficiency in terms of online execution time, and (iii) trajectory tracking error. Notably, we deploy the framework on an NVIDIA Jetson Orin NX embedded platform, experimentally evaluating its compliance with the stringent demands of 200 Hz real-time control. These experiments highlight the inherent trade-offs between prediction accuracy, computational overhead, and control stability under uncertain operating conditions, providing actionable insights for learning-based control in resource-constrained aerial systems. In summary, the main contributions of this paper are:

- We present a modular framework integrating multiple learning methods for real-time uncertainty estimation in flight control.

- We benchmark these approaches under a unified setting, using both simulation and hardware experiments.
- We analyze the tradeoffs between accuracy, control performance, and stability to guide practical deployment.

The remainder of this paper is organized as follows: Section 2 reviews prior work in learning-based quadrotor control and highlights the gaps addressed by our study. Section 3 presents the quadrotor dynamics and formally defines the problem of trajectory tracking under uncertainty. Section 4 details our unified framework, beginning with an architectural overview (Section 4.1), followed by the online learning methods under evaluation (Section 4.2) and the design of the learning-augmented controller (Section 4.3). Section 5 evaluates the framework through both simulations and hardware experiments on the NVIDIA Jetson Orin NX platform, offering a comparative assessment of the performance of different learning methods. Finally, the paper draws to a close in Section 6, where future research directions are discussed.

2 Related work

2.1 Robust and adaptive control

To address uncertainties in quadrotor flight, a substantial body of research is classified under model-based control, as reviewed in [34]. For example, [7] developed a robust adaptive attitude tracking controller on $\mathsf{SO}(3)$ (special orthogonal group) capable of asymptotically tracking attitude commands without prior knowledge of the inertia matrix. Similarly, [10] and [11] proposed a geometric nonlinear adaptive controller on $\mathsf{SO}(3)$ and $\mathsf{SE}(3)$ (special Euclidean group), respectively, to ensure stability while compensating for uncertainties in quadrotor dynamics. In another approach, [8, 9] introduced adaptive backstepping controllers designed to estimate unknown dynamic parameters such as mass and moment of inertia; [12] formulated an adaptive sliding backstepping control law to manage attitude control in the presence of matched uncertainties; and [13] employed robust integral of the signum of the error and invariance-based adaptive control methodologies to address both disturbances and parametric uncertainties. The main advantage of these approaches is their ability to rigorously guarantee stability and robustness through Lyapunov-based analysis, with relatively low computational demands compared to learning-based methods. However, as previously noted, these methods can become overly conservative, leading to performance degradation or instability when the actual uncertainties exceed the assumed bounds.

2.2 Offline learning for control

Advances in computational power have rendered learning-based methods increasingly viable for quadrotor applications, particularly for modeling quadrotor dynamics. By leveraging pre-collected data, ANNs can approximate unknown dynamics or develop control policies through supervised or RL. For example, a multi-step prediction model combining deep Recurrent Neural Networks (RNNs) with classical system identification has been introduced for long-horizon predictions [16]. Physics-informed Neural Networks (PINNs) have been developed to learn quadrotor dynamics purely from experience, which can be subsequently embedded in MPC for precise trajectory tracking [17, 18]. However, learning full quadrotor dynamics from data requires substantial resources and sample sizes, making it impractical in some circumstances [35]. As a result, incorporating prior knowledge into the learning process is encouraged, leading to a shift in focus toward modeling partial dynamics, such as aerodynamic effects [4]. In fact, integrating learning algorithms with model-based control strategies has demonstrated significant benefits in addressing the uncertainties associated with quadrotor dynamics. Notable examples of quadrotor applications include agile flight in turbulence [21], deep-learning-based robust controller for landing [19], learning interactions among heterogeneous multirotors in a swarm [20], training robust RL policies in simulation [22], and deep RL for autonomous drone racing [23]. Despite these advancements, offline learning methods often exhibit performance degradation when encountering OOD data. This paper aims to address this limitation by employing online learning approaches for quadrotor control, which allows for real-time adaptation to uncertainties and extends applicability to previously unencountered scenarios.

2.3 Online learning for control

Previous research on online learning methods for handling uncertainties in robotic platforms has investigated various approaches. For example, ESNs have been employed for real-time estimation of the forces induced by wind gusts and ground effect acting on quadrotors, providing robust performance under varying uncertain conditions [24]. ELMs have demonstrated efficacy in compensating for model mismatches within adaptive cruise control systems, thereby improving the reliability and safety of autonomous vehicles [28]. GPs have been applied to the estimation of wind gusts in quadrotors, offering a probabilistic framework that accounts for uncertainty and enhances control performance in dynamic environments [27]. However, there is still a lack of comprehensive comparative analysis on the effectiveness of

the aforementioned approaches for quadrotors and the factors that influence their performance. This study addresses this gap by building upon previous work [24] through the integration of several established online learning methods into a baseline flight controller and systematically evaluating their respective advantages and limitations. Notably, while episodic learning, which acquires knowledge from discrete episodes, has been explored in some studies [36], we here focus specifically on learning with a circular buffer. Such paradigm continuously stores and overwrites data in a fixed memory, enabling random sampling of recent experiences and maintaining up-to-date information. Lastly, although there exists some literature that reviews offline and online learning algorithms for quadrotors [29], our study uniquely provides a focused comparative analysis of online learning approaches specifically aimed at addressing uncertainties in quadrotor flight control.

3 Problem setup

Henceforth, we consider the mathematical model for the kinematics and dynamics of a quadrotor as follows:

$$\dot{\mathbf{p}} = \mathbf{v}, \tag{1a}$$
$$m\dot{\mathbf{v}} = mg\mathbf{e}_3 + \mathbf{R}\mathbf{f}_u + \mathbf{f}_a, \tag{1b}$$
$$\dot{\mathbf{R}} = \mathbf{R}\boldsymbol{\omega}^{\wedge}, \tag{1c}$$
$$\mathbf{J}\dot{\boldsymbol{\omega}} = -\boldsymbol{\omega}^{\wedge}\mathbf{J}\boldsymbol{\omega} + \boldsymbol{\tau}_u + \boldsymbol{\tau}_a, \tag{1d}$$

where $\mathbf{p}, \mathbf{v} \in \mathbb{R}^3$ denote the position and linear velocities in the inertial frame, $\boldsymbol{\omega} = [p, q, r]^\top \in \mathbb{R}^3$ denotes the body-fixed angular rate, and $\mathbf{R} \in \mathsf{SO}(3)$ denotes the rotation matrix that transforms coordinates from the body frame to the inertial frame, which is constructed by the Euler angles $\boldsymbol{\eta} = [\phi, \theta, \psi]^\top$ (roll, pitch, yaw angle). The control inputs consist of the body wrench produced by the four rotors, denoted by $\mathbf{f}_u = [0, 0, T]^\top \in \mathbb{R}^3$ and $\boldsymbol{\tau}_u = [\tau_{u,x}, \tau_{u,y}, \tau_{u,z}]^\top \in \mathbb{R}^3$, where $T \in \mathbb{R}$ denotes the collective thrust. The lumped uncertainties are denoted by $\mathbf{f}_a \in \mathbb{R}^3$ and $\boldsymbol{\tau}_a \in \mathbb{R}^3$, which encompass both unmodeled dynamics and disturbances. Note that both system states and control inputs are inherently time-dependent, represented in continuous form as $(\cdot)(t)$ or in discrete form as $(\cdot)[k]$. We omit explicit time-dependent notation throughout the paper for brevity, except where necessary to avoid ambiguity.

The quadrotor's mass and inertia matrix are denoted by $m \in \mathbb{R}$ and $\mathbf{J} \in \mathbb{R}^{3\times3}$, respectively. The gravitational constant is denoted by $g \in \mathbb{R}$, and $\mathbf{e}_3 = [0, 0, 1]^\top$ denotes a vector pointing downward in the unit 2-sphere $\mathbb{S}^2 = \{\mathbf{a} \in \mathbb{R}^3 : \|\mathbf{a}\| = 1\}$. The hat operator, denoted by $(\cdot)^{\wedge} : \mathbb{R}^3 \to \mathfrak{so}(3)$, converts real vectors into the Lie algebra associated with $\mathsf{SO}(3)$. Finally, the North-East-Down (NED) and Forward-Right-Down (FRD) conventions are adopted for the inertial and body-fixed reference frame, respectively.

We aim to solve the tracking control problem for a quadrotor, governed by the kinematics and dynamics described in Eq. 1, of desired reference trajectories $\boldsymbol{\xi}_d(t) = [\mathbf{p}_d^\top(t), \psi_d(t)]^\top \in \mathbb{R}^4$, while accounting for unknown uncertainties, denoted as $\mathbf{f}_a$ and $\boldsymbol{\tau}_a$. The reference trajectories $\boldsymbol{\xi}_d(t)$ are assumed to belong to $\mathcal{C}^3$, meaning they have continuous derivatives up to the third order, and these derivatives are bounded. Similarly, the uncertainties are assumed to be continuous and bounded. The control objective is to ensure asymptotic tracking performance in the presence of uncertainties.

4 Method

4.1 System overview

The online-learning-based control framework, depicted in Fig. 1, integrates a nominal controller[1] with an online residual learning module for non-parametric uncertainty estimation, such as ground effects or wind disturbances. The framework is *modular* due to the standalone nature of the online residual learning module, offering two key advantages. First, it supports the seamless incorporation of various online learning methods beyond those discussed in this paper. Second, it provides enhanced flexibility compared to fully data-driven control approaches, such as those presented in [37], by enabling integration with existing controllers. Within the learning module, one of four learning methods, namely ELM, RBFNN, ESN, or GP, is employed to capture and adapt to system uncertainties in real time. A circular data buffer is implemented to facilitate this process, storing feature and label data from flight history in a sliding window fashion. The following sections will elaborate on the selected learning methods, provide a detailed account of the online residual learning process, and derive the corresponding learning-based control laws.

4.2 Online learning methods

4.2.1 Extreme learning machine (ELM)

As illustrated in Fig. 2(a), ELMs are fundamentally single-hidden-layer Feedforward Neural Networks (FNNs) comprising N_u input nodes, N_h hidden nodes, and N_y output

[1] This controller may be either model-based or learning-based; in this paper, a model-based approach is employed.

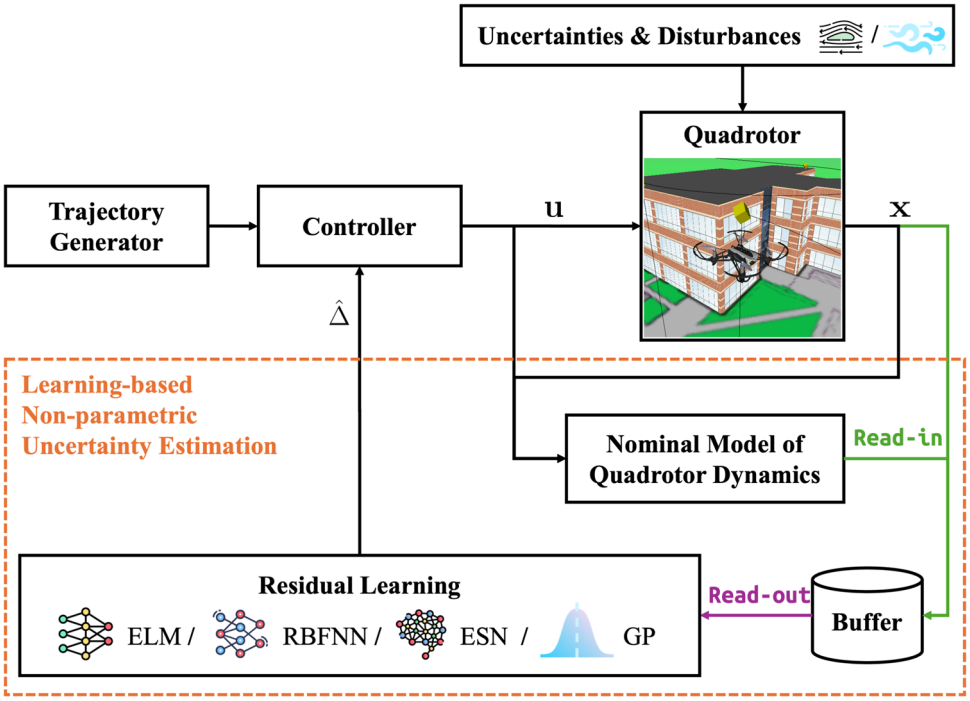

Fig. 1 Overview of control architecture. The online learning module for non-parametric uncertainty estimation, enclosed within the orange dashed box, takes as inputs the online measurements of quadrotor states $\mathbf{x} = [\mathbf{p}^\top, \mathbf{v}^\top, \boldsymbol{\eta}^\top, \boldsymbol{\omega}^\top]^\top$ and control inputs $\mathbf{u} = [\mathbf{f}_u^\top, \boldsymbol{\tau}_u^\top]^\top$, and generates uncertainty estimates $\hat{\Delta} = [\hat{\mathbf{f}}_a^\top, \hat{\boldsymbol{\tau}}_a^\top]^\top$

nodes [30]. Given an input $\mathbf{u} \in \mathbb{R}^{N_u}$ and its corresponding output $\mathbf{y} \in \mathbb{R}^{N_y}$, the input-output mapping can be expressed as

$$\mathbf{y} = \mathbf{W}_{\text{out}}\mathbf{h}(\mathbf{u}) = \mathbf{W}_{\text{out}}g(\mathbf{W}_{\text{in}}\mathbf{u}), \tag{2}$$

where $\mathbf{h} = \{h_1, h_2, ..., h_{N_h}\} \in \mathbb{R}^{N_h}$ denotes the outputs of the hidden layer, $g(\cdot)$ denotes the activation functions (e.g., ReLU or sigmoid functions), and $\mathbf{W}_{\text{in}} \in \mathbb{R}^{N_h \times N_u}$ and $\mathbf{W}_{\text{out}} \in \mathbb{R}^{N_y \times N_h}$ denote the input and output weight matrices, respectively.

In contrast to traditional FNN, ELM utilize a *fixed* input weight matrix $\mathbf{W}_{\text{in}}$, which is initialized with random values, thereby eliminating the need for parameter training. Consequently, the objective of network learning is to determine the optimal $\mathbf{W}_{\text{out}}$ by solving the following optimization problem

$$\mathbf{W}_{\text{out}} = \arg\min \|\mathbf{W}_{\text{out}}\mathbf{H} - \mathbf{Y}_{\text{label}}\|_2, \tag{3}$$

where $\|\cdot\|$ denotes the l_2-norm, which quantifies the distance between the predicted output collection $\mathbf{W}_{\text{out}}\mathbf{H} =$

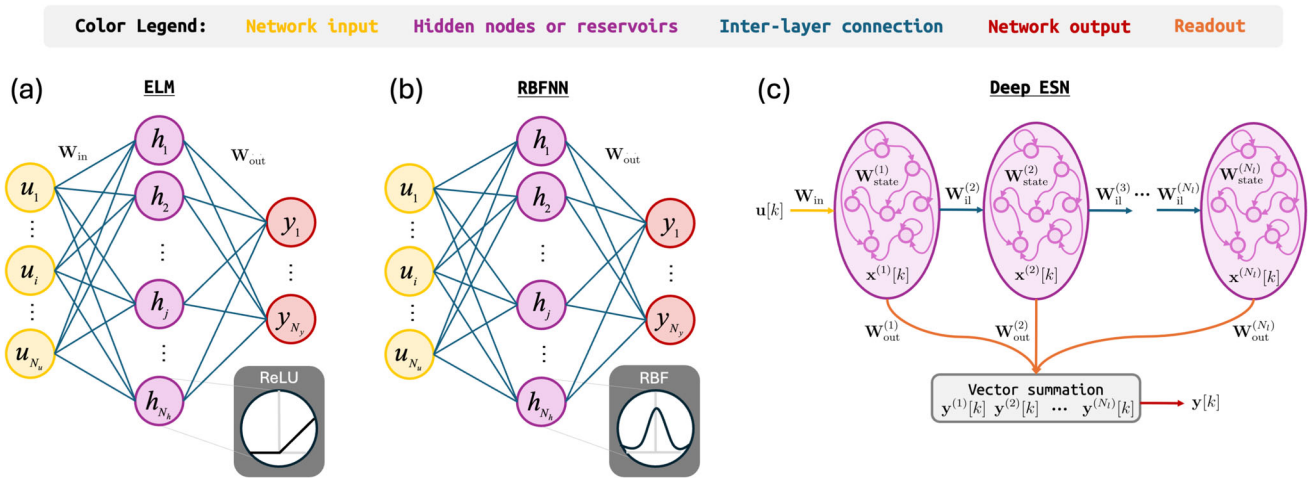

Fig. 2 Neural network architecture. (a) The ELM is a three-layer FNN comprising an input layer, a single hidden layer, and an output layer. (b) The RBFNN similarly adopts a three-layer feedforward structure, where the hidden layer computes local activations based on the distance between input patterns and predefined center vectors. (c) The deep ESN incorporates hierarchical reservoir layers, with inter-layer connections and recurrent feedback within each reservoir, enabling the capture of complex temporal dependencies

$\mathbf{W}_{\text{out}}[\mathbf{h}(1), \cdots, \mathbf{h}(N)]$ and the true label collection $\mathbf{Y}_{\text{label}} = [\mathbf{y}_{\text{label}}(1), \cdots, \mathbf{y}_{\text{label}}(N)]$ over the time span $k = 1, \cdots, N$. Therefore, the update rule for ELMs can be expressed as a least-squares solution using pseudoinverse as $\mathbf{W}_{\text{out}} = \mathbf{H}^{+}\mathbf{Y}_{\text{label}}$. As such, the lengthy training phases typically associated with iterative adjustments of network parameters, such as learning rate and number of iterations, can be circumvented. This makes ELMs a well-suited candidate for online learning of uncertainties during quadrotor flight.

4.2.2 Radial basis function neural network (RBFNN)

Similar to ELMs, RBFNNs are a class of single-hidden-layer FNNs distinguished by their use of Radial Basis Functions (RBFs) as activation functions, which can be used for continuous nonlinear function approximation [31]. As shown in Fig. 2(b), in RBFNNs, the network output $\mathbf{y} \in \mathbb{R}^{N_y}$ is computed as a linear combination of the RBF outputs, which can be formulated as

$$\mathbf{y} = \mathbf{W}_{\text{out}}\mathbf{h}(\mathbf{u}), \tag{4a}$$

$$\mathbf{h}_j = \exp\left(-\frac{\|\mathbf{u} - \mathbf{c}_j\|^2}{b_j^2}\right), \tag{4b}$$

where $\mathbf{u} \in \mathbb{R}^{N_u}$ is the input vector, and j indexes the nodes in the hidden layer. The Gaussian function Eq. 4b generates the outputs of the hidden layer $\mathbf{h} = \{h_1, h_2, ..., h_{N_h}\} \in \mathbb{R}^{N_h}$, with $\mathbf{c}_j$ representing the center of the j-th Gaussian basis function and b_j denoting its width. The weight matrix $\mathbf{W}_{\text{out}} \in \mathbb{R}^{N_y \times N_h}$ maps the outputs of the hidden layer to the final output. The network performs online adaptive weight updates, akin to ELMs, bypassing traditional training.

RBFNNs are highly effective in modeling complex nonlinear relationships due to their localized learning capabilities, enabling responsiveness to distinct regions of the input space. This localized learning also offers advantages in training efficiency, particularly with smaller data sets. RBFNNs excel in interpolation across multi-dimensional spaces, which is critical for control systems requiring accurate predictions in unobserved regions. However, the accuracy of these representations is strongly dependent on the a priori selection of RBF centers, limiting the model's effectiveness to areas of the state space where the centers are properly placed as we will see in Section 5.

4.2.3 Echo state network (ESN)

A typical ESN, characterized by shallow architecture with feedback connections and leaky integration, can be modeled as a *dynamical system*. The corresponding state transition and output equations are expressed as

$$\mathbf{x}[k] = (1-\alpha)\mathbf{x}[k-1] + \alpha \tanh(\mathbf{W}_{\text{in}}\mathbf{u}[k] + \mathbf{W}_{\text{state}}\mathbf{x}[k-1] + \mathbf{W}_{\text{fb}}\mathbf{y}[k-1]), \tag{5a}$$

$$\mathbf{y}[k] = \mathbf{W}_{\text{out}}(\mathbf{x}[k]; \mathbf{u}[k]), \tag{5b}$$

where $\mathbf{u} \in \mathbb{R}^{N_u}$ denotes the input sequence, $\mathbf{x} \in \mathbb{R}^{N_r}$ denotes the reservoir state, $\mathbf{y} \in \mathbb{R}^{N_y}$ denotes the output sequence, $\alpha \in [0, 1]$ denotes the leaking rate, $\mathbf{W}_{\text{in}} \in \mathbb{R}^{N_r \times N_u}$, $\mathbf{W}_{\text{state}} \in \mathbb{R}^{N_r \times N_r}$, $\mathbf{W}_{\text{fb}} \in \mathbb{R}^{N_r \times N_y}$, $\mathbf{W}_{\text{out}} \in \mathbb{R}^{N_y \times (N_r + N_u)}$ denote input-to-reservoir, reservoir, readout-to-reservoir, and readout weight matrix, respectively. For better learning capability, the reservoir layer can be enhanced by stacking multiple layers, similar to the architecture employed in deep learning models. As depicted in Fig. 2(c), a deep ESN [2] with N_l reservoir layers can be expressed as

$$\mathbf{x}^{(l)}[k] = \begin{cases} \tanh(\mathbf{W}_{\text{in}}\mathbf{u}[k] + \mathbf{W}_{\text{state}}^{(l)}\mathbf{x}^{(l)}[k-1]) & \text{if } l = 1, \\ \tanh(\mathbf{W}_{\text{il}}^{(l)}\mathbf{x}^{(l-1)}[k] + \mathbf{W}_{\text{state}}^{(l)}\mathbf{x}^{(l)}[k-1]) & \\ \qquad \text{if } l = 2, \cdots, N_l, \end{cases} \tag{6a}$$

$$\mathbf{y}^{(l)}[k] = \mathbf{W}_{\text{out}}^{(l)}\mathbf{x}^{(l)}[k], \tag{6b}$$

where the superscript (l) is used to represent the network parameters at the l-th layer, and $\mathbf{W}_{\text{il}} \in \mathbb{R}^{N_r \times N_r}$ denotes the inter-layer reservoir weight matrix. Note that reservoir states (depicted as purple circles in Fig. 2(c)) are synonymous with the system's state in dynamical systems, while a reservoir (shown by purple-shaded ellipses) comprises a collection of these states, potentially connected through feedforward or recurrent links.

Unlike other RNNs, which typically adopt back-propagation as the standard training technique, the training process of an ESN follows the *Reservoir Computing (RC)* paradigm. In such paradigm, only the readout weight matrix is updated, while the remaining weight matrices are randomly initialized [38]. Training the readouts of an ESN without feedback connections can generally be achieved through linear regression, which involves solving the following optimization problem

$$\mathbf{W}_{\text{out}}^{(l)} = \arg\min \|\mathbf{W}_{\text{out}}^{(l)}\mathbf{X}^{(l)} - \mathbf{Y}_{\text{label}}\|_2^2, \tag{7}$$

where $\mathbf{X}^{(l)} = [\mathbf{x}^{(l)}(1), \cdots, \mathbf{x}^{(l)}(N)]$, $\mathbf{Y}_{\text{label}} = [\mathbf{y}_{\text{label}}(1), \cdots, \mathbf{y}_{\text{label}}(N)]$ are the collection of reservoir states and target

[2] The mathematical expression of each layer in the deep ESN can be obtained from Eqs. 5a and 5b by setting $\alpha = 1$, omitting readout-to-reservoir connections, and removing the connections from input to readout.

values over the time span $k = 1, \cdots, N$. Hence, Moore-Penrose pseudo-inversion $\mathbf{W}_{\text{out}} = \mathbf{Y}_{\text{label}}\mathbf{X}^{+}$ or ridge regression $\mathbf{W}_{\text{out}} = \mathbf{Y}_{\text{label}}\mathbf{X}^{\top}(\mathbf{X}\mathbf{X}^{\top} + \lambda_r \mathbf{I})^{-1}$ ($\lambda_r \in \mathbb{R}_{>0}$ is the regularization coefficient) can be used for offline or semi-online training.

Since all reservoir weight matrices are initialized at random, Echo State Property (ESP) [32] was shown to be a critical property to ensure valid dynamics in ESNs. From a dynamical system perspective, it guarantees that the reservoir states should asymptotically depend soley on the driving input signal, independent of initial conditions [39]. By assuming the Euclidean distance metric in the reservoir space and using the hyperbolic tangent (`tanh`) as the activation function, satisfying the following condition practically ensures ESP:

$$\rho(\mathbf{W}_{\text{state}}) < 1, \tag{8}$$

where $\rho(\cdot)$ denotes the spectral radius (i.e., the largest absolute eigenvalue) of the given square matrix. In the context of dynamical systems, Eq. 8 also implies that $\mathbf{W}_{\text{state}}$ is *contractive*.

4.2.4 Gaussian process (GP) regression

GPs are commonly employed for *non-parametric* regression in machine learning [33], with the goal of approximating a nonlinear mapping $g(\mathbf{u}) : \mathbb{R}^{N_u} \to \mathbb{R}$, where $\mathbf{u} \in \mathbb{R}^{N_u}$ denotes the input vector. This approximation is based on the assumption that the function values $g(\mathbf{u})$, evaluated at different points of $\mathbf{u}$, are treated as random variables. Moreover, any finite number of these random variables is assumed to follow a joint Gaussian distribution depending on the values of $\mathbf{u}$.

In the context of non-parametric regression, it is essential to define a prior for the mean of $g(\mathbf{u})$ and for the covariance between any two function values, $g(\mathbf{u}_i)$ and $g(\mathbf{u}_j)$, where i, $j \in \mathbb{Z}$. Then, the posterior mean and variance for a new test point $\mathbf{u}_*$ can be computed as

$$\mu(\mathbf{u}_*) = \mathbf{k}_n^{\top}(\mathbf{K} + \sigma_n^2\mathbf{I})^{-1}\hat{\mathbf{g}}_n, \tag{9a}$$

$$\sigma^2(\mathbf{u}_*) = k(\mathbf{u}_*, \mathbf{u}_*) - \mathbf{k}_n^{\top}(\mathbf{K} + \sigma_n^2\mathbf{I})^{-1}\mathbf{k}_n, \tag{9b}$$

where $\hat{\mathbf{g}}_n = [\hat{g}(\mathbf{u}_1), ..., \hat{g}(\mathbf{u}_{N_u})]$ denotes the vector of observed outputs, affected by Gaussian noise $v \sim \mathcal{N}(0, \sigma_n^2)$. Moreover, the covariance matrix $\mathbf{K} \in \mathbb{R}^{n\times n}$ has elements $\mathbf{K}_{i,j} = k(\mathbf{u}_i, \mathbf{u}_j)$, where $k(\cdot,\cdot)$ denotes the kernel function that determines the similarity between inputs with a popular choice of being RBFs as defined in Eq. 4b. The vector $\mathbf{k}_n = [k(\mathbf{u}_1, \mathbf{u}_*), \cdots, k(\mathbf{u}_n, \mathbf{u}_*)]^{\top}$ contains the covariances between the test input and each training input, and $\mathbf{I} \in \mathbb{R}^{n\times n}$ denotes the identity matrix. Learning the function from an observed dataset also involves determining the hyperparameters of its kernel.

GPs are non-parametric models, meaning their complexity grows with the amount of data, unlike parametric models such as RBFNNs, which require predefined centers, or ELMs, which fix the number of hidden-layer nodes. GPs are particularly effective for regression tasks involving smooth underlying functions, offering a convex formulation for learning, which is beneficial for smaller data sets. In comparison to ELMs, RBFNNs, and ESNs, which require tuning of parameters, GPs are often more suitable for online learning due to their capability to provide uncertainty estimates.

However, GPs come with notable limitations in online applications. Their computational complexity scales cubically with the number of data points, rendering them inefficient for large data sets. Additionally, they require substantial memory to store all past data, which may pose practical challenges. Moreover, the performance of GPs heavily depends on the choice of kernel function [33]. Finally, while GPs provide uncertainty estimates, their interpretability in terms of feature contributions can be less intuitive compared to other neural network models such as ESNs [24].

4.3 Learning-based control

4.3.1 Online residual learning

To estimate the unknown and potentially dynamic uncertainties, an online residual learning module is developed, centered on a general residual learning approach. This module leverages the efficient training capabilities of ELM, RBFNN, ESN, and GP, to provide online learning. Consequently, this online residual learning module is anticipated to "generalize" (or more precisely, actively adapt) more effectively than networks trained offline when facing unseen scenarios.

Given the integration of four online learning methods (i.e., ELM, RBFNN, ESN, and GP) into a unified online-learning-based control framework, as illustrated in Fig. 1, which maintains a consistent interaction with the data buffer, we will concentrate on the detailed implementation of ESN in the rest of this section. This focus is warranted due to its relative underrepresentation in the existing literature and the complexity of its dynamics. The online residual learning module utilizing ESN is depicted in Fig. 3. As shown, the module features a *deep ESN* (described by Eq. 6) for the learning and inference of uncertainties, along with a *circular data buffer* that stores the feature and label data from flight history in a sliding window fashion, thereby facilitating

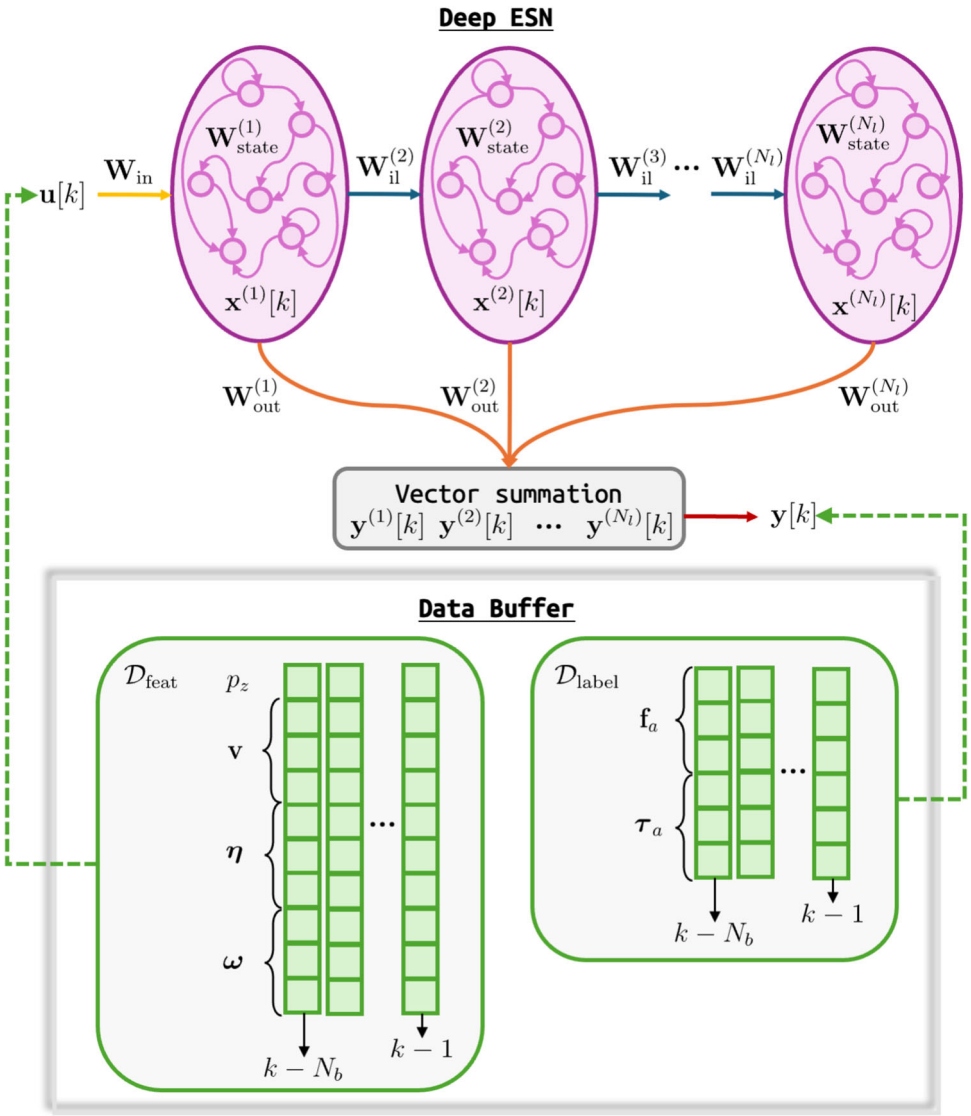

Fig. 3 The online residual learning module utilizing ESN consists of a deep ESN and a data buffer. The data buffer of length N_b stores the feature and label data from flight history in a column-wise fashion (shown by green blocks), which are subsequently used for online training of readout weights of ESN. The figure is redrawn after [24]

efficient online training. The algorithm for online residual learning using ESN is outlined as follows:

(i) **Initializing ESN**: We consider an ESN that takes system state[3] as network input $\mathbf{u} = [p_z, \mathbf{v}^\top, \boldsymbol{\eta}^\top, \boldsymbol{\omega}^\top]^\top \in \mathbb{R}^{10}$ and predicted forces and torques as network output $\mathbf{y} = [\hat{\mathbf{f}}_a^\top, \hat{\boldsymbol{\tau}}_a^\top]^\top \in \mathbb{R}^6$. Besides configuring the ESN with designated numbers of input units (N_u), readout units (N_y), reservoir units (N_r), and reservoir layers (N_l), this step includes random initialization of all weight matrices (i.e., $\mathbf{W}_{\text{in}}$, $\mathbf{W}_{\text{il}}$, $\mathbf{W}_{\text{state}}$), adhering to the ESP. Note that suitable modifications to network parameters (as later shown in Table 1) are necessary for each learning method for this step.

[3] The altitude p_z is chosen as a network feature owing to its contribution to certain aerodynamic phenomena such as ground effect.

(ii) **Collecting flight data and computing label data**: The continuous-time model Eq. 1 needs to be converted into its discrete-time counterpart with sampling time T_s as given in Eq. 10 for implementation. Through the latter, we can calculate the label data at the previous time step from the current and previous state.

$$\mathbf{v}[k+1] = \mathbf{v}[k] + \frac{T_s}{m}(mg\mathbf{e}_3 + \mathbf{f}_u[k] + \mathbf{f}_a[k]), \tag{10a}$$

$$\boldsymbol{\omega}[k+1] = \boldsymbol{\omega}[k] + T_s\mathbf{J}^{-1}(-\boldsymbol{\omega}[k]^\wedge\mathbf{J}\boldsymbol{\omega}[k] + \boldsymbol{\tau}_u[k] + \boldsymbol{\tau}_a[k]). \tag{10b}$$

(iii) **Updating data buffer**: The buffer retains the training data pair within a specified time span (denoted by N_b). Upon the arrival of a new data pair that exceeds the buffer size, the earliest stored data will be replaced.

(iv) **Training ESN and inferring from the latest flight data**: We use ridge regression for training the ESN.

Table 1 Model parameters for online learning assessment

Model	Parameter	Value
ELM	Number of input/output/hidden units	$(N_u, N_y, N_h) = (10, 6, 140)$
	Input-to-hidden weight	$\mathbf{W}_{\text{in}} \sim \mathcal{U}_{[0,1]}$
RBFNN	Number of input/output/hidden units	$(N_u, N_y, N_h) = (10, 6, 100)$
	Input-to-hidden weight	$\mathbf{W}_{\text{in}} \sim \mathcal{U}_{[\min(\mathcal{D}_{\text{feat}}),\max(\mathcal{D}_{\text{feat}})]}$
ESN	Number of input/readout/reservoir units	$(N_u, N_y, N_r) = (10, 6, 23)$
	Number of reservoir layers	$N_l = 5$
	Input-to-reservoir weight	$\mathbf{W}_{\text{in}} \sim \mathcal{U}_{[-1,1]}$
	Inter-layer reservoir weight	$\mathbf{W}_{\text{il}} \sim \mathcal{U}_{[-1,1]}$
	Reservoir weight	$\mathbf{W}_{\text{state}} \sim \mathcal{U}$ with $\rho = 0.9$
	Leaking rate	$\alpha = 0$
	Washout	$N_w = 5$
	Regularization coefficient	$\lambda_r = 0.1$

The training data pairs are drawn from the data buffer, and training initiates once the buffer is fully populated (hence, requiring an initial data collection period). The RC paradigm enables online training, specifically within our sampling time, as demonstrated empirically in Section 5. Upon completion of training, the ESN takes the latest flight data for inference.

The pseudo-code for our implementation of the above outlined online residual learning is presented in Algorithm 1. Note that except for Step (i), which will be executed only once, Steps (ii) to (iv) will be iteratively performed during the flight. Moreover, the outlined steps are applicable for ELM, RBFNN, and GP, but require appropriate modifications. For example, the initialization of these methods (see `InitESN` in Algorithm 1) should adhere to the structure or working principles described in Section 4.2, and the training method (see `TrainESN`) must be adjusted accordingly. Nonetheless, the data buffer mechanism (see `UpdateBuffer`) remains unchanged for all online learning methods, as previously noted. This consistency enables the straightforward integration of new online learning methods into the module. We leave the discussion of technical details on network architecture and computational efficiency to Section 5.

4.3.2 Trajectory tracking control law

The decoupled nature of our adopted architecture allows for the selection of any nominal controller independently of the overall control architecture. Here, we use the nominal controller proposed in [1] as a starting point, whereby the tracking control laws for thrust and torque determination of Eq. 10 under nominal conditions are given by

$$\bar{\mathbf{f}}_u[k] = -\mathbf{K}_p\tilde{\mathbf{p}}[k] - \mathbf{K}_v\tilde{\mathbf{v}}[k] - mg\mathbf{e}_3 + m\dot{\mathbf{v}}_d[k], \tag{11}$$

$$\bar{\boldsymbol{\tau}}_u[k] = -\mathbf{K}_r\tilde{\mathbf{r}}[k] - \mathbf{K}_\omega\tilde{\boldsymbol{\omega}}[k], \tag{12}$$

where the tracking errors of position, linear velocity, attitude, and angular rate are defined as

$$\tilde{\mathbf{p}}[k] = \mathbf{p}[k] - \mathbf{p}_d[k], \tag{13a}$$

$$\tilde{\mathbf{v}}[k] = \mathbf{v}[k] - \mathbf{v}_d[k], \tag{13b}$$

$$\tilde{\mathbf{r}}[k] = \frac{1}{2}(\mathbf{R}_d^\top[k]\mathbf{R}[k] - \mathbf{R}^\top[k]\mathbf{R}_d[k])^\vee, \tag{13c}$$

$$\tilde{\boldsymbol{\omega}}[k] = \boldsymbol{\omega}[k] - \boldsymbol{\omega}_d[k], \tag{13d}$$

with $(\cdot)^\vee : \mathfrak{so}(3) \to \mathbb{R}^3$ denoting the vee operator, which is an inverse operation of the hat operator. Moreover, the desired body z-axis can be computed from $\mathbf{k}_{b,d} = -\bar{\mathbf{f}}_u/\|\bar{\mathbf{f}}_u\| \in \mathbb{S}^2$, and hence the desired rotation matrix $\mathbf{R}_d$ can be derived using the differential flatness property of quadrotors as

$$\mathbf{j}_c = [-\sin(\psi_d),\ \cos(\psi_d),\ 0]^\top, \tag{14a}$$

$$\mathbf{i}_{b,d} = \mathbf{j}_c^\wedge\, \mathbf{k}_{b,d} \,/\, \|\mathbf{j}_c^\wedge\, \mathbf{k}_{b,d}\|, \tag{14b}$$

$$\mathbf{j}_{b,d} = \mathbf{k}_{b,d}^\wedge\, \mathbf{i}_{b,d}, \tag{14c}$$

$$\mathbf{R}_d = [\mathbf{i}_{b,d},\ \mathbf{j}_{b,d},\ \mathbf{k}_{b,d}]. \tag{14d}$$

In [1], it is rigorously demonstrated that this nominal controller guarantees faithful tracking performance of given reference trajectories with proper selections of controller gains $\mathbf{K}_\Xi$ ($\Xi \in \{p, v, r, \omega\}$).

Our proposed controller augments the nominal controller with the online residual learning module as detailed in Section 4.3.1. Hence, the learning-based control laws for thrust and torque determination are given as

$$\mathbf{f}_u[k] = \bar{\mathbf{f}}_u[k] - \hat{\mathbf{f}}_a[k], \tag{15}$$

$$\boldsymbol{\tau}_u[k] = \bar{\boldsymbol{\tau}}_u[k] - \hat{\boldsymbol{\tau}}_a[k], \tag{16}$$

where $\hat{\mathbf{f}}_a$ and $\hat{\boldsymbol{\tau}}_a$ are the estimates of the uncertainties predicted by one of the chosen online learning methods among

Algorithm 1 Online residual learning using ESN.

Parameter: N_b, $\mathcal{E} = (N_r, N_l, N_u, N_y)$
Input: $\mathcal{X}[k] = (\mathbf{v}[k]^\top, \boldsymbol{\omega}[k]^\top, \mathbf{f}_u[k]^\top, \boldsymbol{\tau}_u[k]^\top)^\top$, $\mathcal{P} = (m, \mathbf{J}, T_s)$
Output: $\mathcal{Y}[k] = (\hat{\mathbf{f}}_a[k]^\top, \hat{\boldsymbol{\tau}}_a[k]^\top)^\top$
Data: $\mathcal{D}_{\text{feat}}$, $\mathcal{D}_{\text{label}}$ (from flight data)

Function `CalculateLabel(`$\mathcal{X}[k]$, $\mathcal{X}[k-1]$, $\mathcal{P}$`)`**:**
  Calculate $\mathbf{f}_a[k-1]$ from Eq. 10a
  Calculate $\boldsymbol{\tau}_a[k-1]$ from Eq. 10b
  return $\mathbf{f}_a[k-1]$, $\boldsymbol{\tau}_a[k-1]$

Function `UpdateBuffer(`$\mathcal{X}[k]$, N_b`)`**:**
  if $\mathcal{D}_{feat}$, $\mathcal{D}_{label}$ *are undefined* **then**
    $\mathcal{D}_{\text{feat}} = \mathcal{D}_{\text{label}} = [\,]$ `// initialization`
  if $\mathcal{D}_{feat}$ *has more than one column* **then**
    $\mathbf{f}_a[k-1], \boldsymbol{\tau}_a[k-1]$ = `CalculateLabel(`$\mathcal{X}[k]$, $\mathcal{X}[k-1]$, $\mathcal{P}$`)`
    $\mathcal{D}_{\text{label}}$.append($\mathbf{f}_a[k-1]$, $\boldsymbol{\tau}_a[k-1]$)
  if $\mathcal{D}_{feat}$, $\mathcal{D}_{label}$ *have more than* N_b *columns* **then**
    Delete the first column (i.e., the earliest data)
  $\mathcal{D}_{\text{feat}}$.append($\mathcal{X}[k]$) `// append column`
  return $\mathcal{D}_{\text{feat}}$, $\mathcal{D}_{\text{label}}$

Function `InitESN(`$\mathcal{E}$`)`**:**
  Instantiate and initialize an ESN using $\mathcal{E}$
  return net

Function `TrainESN(`*net*, $\mathcal{D}_{feat}$, $\mathcal{D}_{label}$`)`**:**
  Train *net* on $\mathcal{D}_{\text{feat}}$, $\mathcal{D}_{\text{label}}$ by solving Eq. 7
  return 0

Function `Main`**:**
  Specify N_b and $\mathcal{E}$
  net = `InitESN(`$\mathcal{E}$`)`
  $k \leftarrow 1$
  while *non-stop* **do**
    Get data $\mathcal{X}[k]$
    $\mathcal{D}_{\text{feat}}, \mathcal{D}_{\text{label}}$ = `UpdateBuffer(`$\mathcal{X}[k]$, N_b`)`
    if $\mathcal{D}_{label}$ *has* N_b *columns* **then**
      `TrainESN(`*net*, $\mathcal{D}_{feat}(:, 1:N_b)$, $\mathcal{D}_{label}$`)`
    $k \leftarrow k + 1$
  return 0

ELM, RBFNN, ESN, and GP, upon completion of network training within each sampling interval. Taking ESN as an example,

$$\mathbf{y}[k] = \text{ESN}(\mathbf{u}[k] \,|\, \mathbf{W}_{\text{out}}[k]), \quad \text{where} \tag{17a}$$

$$\mathbf{y}[k] = [\hat{\mathbf{f}}_a^\top[k], \hat{\boldsymbol{\tau}}_a^\top[k]]^\top, \; \mathbf{u}[k] = [p_z[k], \mathbf{v}^\top[k], \boldsymbol{\eta}^\top[k], \boldsymbol{\omega}^\top[k]]^\top. \tag{17b}$$

The desired body z-axis for constructing the desired rotation matrix is modified to $\mathbf{k}_{b,d} = -\mathbf{f}_u / \|\mathbf{f}_u\|$, taking into account the compensation of unknown forces predicted by the learning methods.

5 Results and discussion

5.1 Model selection and training

The online learning capabilities of the four methods were systematically evaluated using the model configurations summarized in Table 1. For the ELM, input-to-hidden weights were initialized randomly from a uniform distribution. Similarly, the RBFNN employed randomly initialized input-to-hidden weights, which were then scaled according to the maximum and minimum values of the dataset. The widths of the RBFs were calculated as the average distance between all training input data points and their corresponding centers. In the case of the ESN, the reservoir weights were also randomly initialized, with the reservoir state matrix constrained to satisfy the ESP, using a spectral radius of $\rho = 0.93$. For the GP, N_y regression processes were designed, each taking N_u inputs and yielding a single output, with the collective output formed by these processes.

To ensure fair comparisons, the number of hidden neurons was fixed at 100 for all parametric models during the baseline evaluation. Hyperparameters of these models were then tuned using BO, a sample-efficient global optimization framework particularly suitable for objective functions that are costly, non-differentiable, or treated as black boxes. Essentially, BO constructs a probabilistic surrogate model to approximate the objective function while leveraging an acquisition function to determine the most informative next evaluation point. This approach enables efficient convergence to a near-optimal configuration within a limited number of evaluations. In our context, the surrogate model was implemented via GP regression, and the Expected Improvement Plus (EI^+) function was employed to guide the trade-off between exploration and exploitation. Specifically, the Kernel function was adaptively scaled by multiplying its parameters with the current iteration number [40]. Compared to the standard EI, the EI^+ function better accounts for predictive uncertainty.

The overall BO procedure adopted herein is illustrated in Fig. 4, which consists of the following stages. It begins with an initialization phase, where a set of sample points is randomly drawn from the search space to construct an initial GP surrogate model. In each subsequent iteration, the GP is updated using all observed input-output pairs, providing both predictive mean and uncertainty estimates. Based on the updated surrogate, the EI^+ acquisition function is evaluated over the input space to identify the most promising candidate point that maximizes the trade-off between exploration and exploitation. The true objective function is then evaluated at this selected point, and the resulting observation is

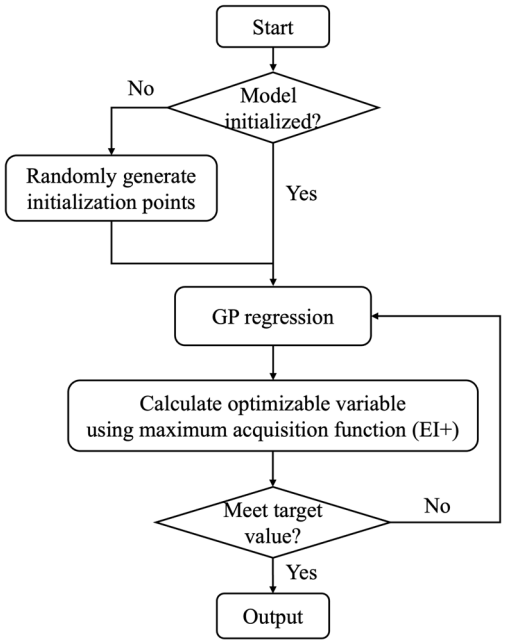

Fig. 4 Flowchart of Bayesian optimization procedure

incorporated into the data set. This procedure is repeated iteratively until a predefined stopping criterion such as maximum number of iterations or convergence threshold is satisfied.

As mentioned in Section 4.3.1, we adopted a deep ESN architecture consisting of five reservoir layers. This depth was selected based on empirical insights from our prior study [24], balancing approximation capability with computational cost. To reduce the complexity associated with network depth tuning, the number of neurons per layer was chosen as the primary optimization parameter in the BO process. For consistency, the number of hidden neurons in ELM and RBFNN was constrained to multiples of five, and for deployment relevance, the total number of neurons in each model was restricted to the range from 0 to 200. All BO experiments were conducted using MATLAB's `bayesopt` function, with the offline MSE on the training set as the objective. The maximum number of BO iterations was set to 200. The convergence behavior of BO for each model is shown in Fig. 5 with the final optimized hyperparameters reported in Table 1.

To assess real-time learning performance, we trained the ELM, RBFNN, ESN, and GP models following Algorithm 1 on two simulated data sets generated in MATLAB, with a sampling time of $T_s = 0.02$ seconds. The buffer size was chosen to be $N_b = 50$. The data sets represent two distinct scenarios: periodic wind disturbances and the quadrotor ground effect [2]. Each model was trained over 50 trials, with different random seeds, on a laptop equipped with a 12$^{\text{th}}$ Gen Intel Core i7-12700KF processor, without utilizing GPU acceleration.

5.2 Learning performance and sensitivity analysis

The online learning performance of the models under discuss concerning the quadrotor's ground effect is showcased in Fig. 6, while Fig. 7 reveals that of a periodic wind disturbance. Notably, both scenarios exhibited an initial phase during which the ELM, RBFNN, ESN, and GP models were unable to produce predictions, as they awaited the data buffer to accumulate a total of N_b samples. In summary, our comparative analysis of online learning capabilities yielded the following rankings: for ground effect, the learning performance follows the order of GP, ESN, RBFNN, and ELM; for periodic wind disturbances, the hierarchy is GP, ESN, ELM, and RBFNN. The results underscore the varying efficacy of the models in adapting to different uncertainties.

In addition, we present the computational efficiency in terms of online execution time for the ELM, RBFNN, ESN, and GP models in Fig. 8. Notably, the GP model requires the longest average execution time, despite achieving the highest prediction accuracy. This prolonged execution time is mostly

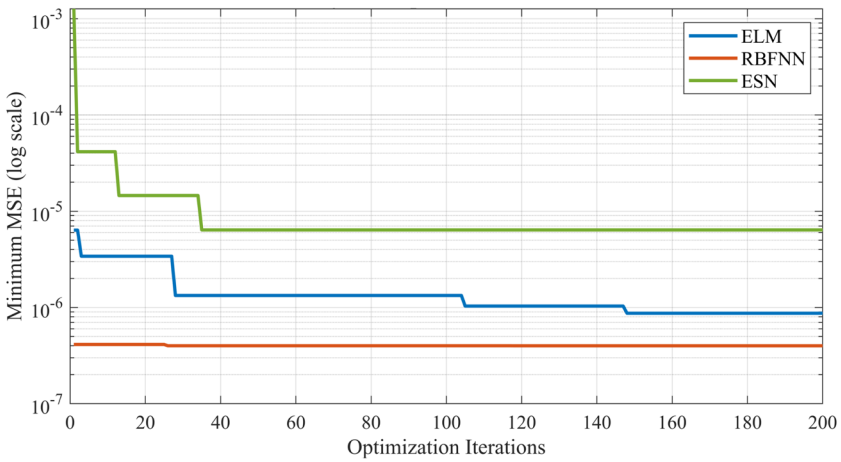

Fig. 5 Convergence of Bayesian optimization for hyperparameter tuning of parametric models

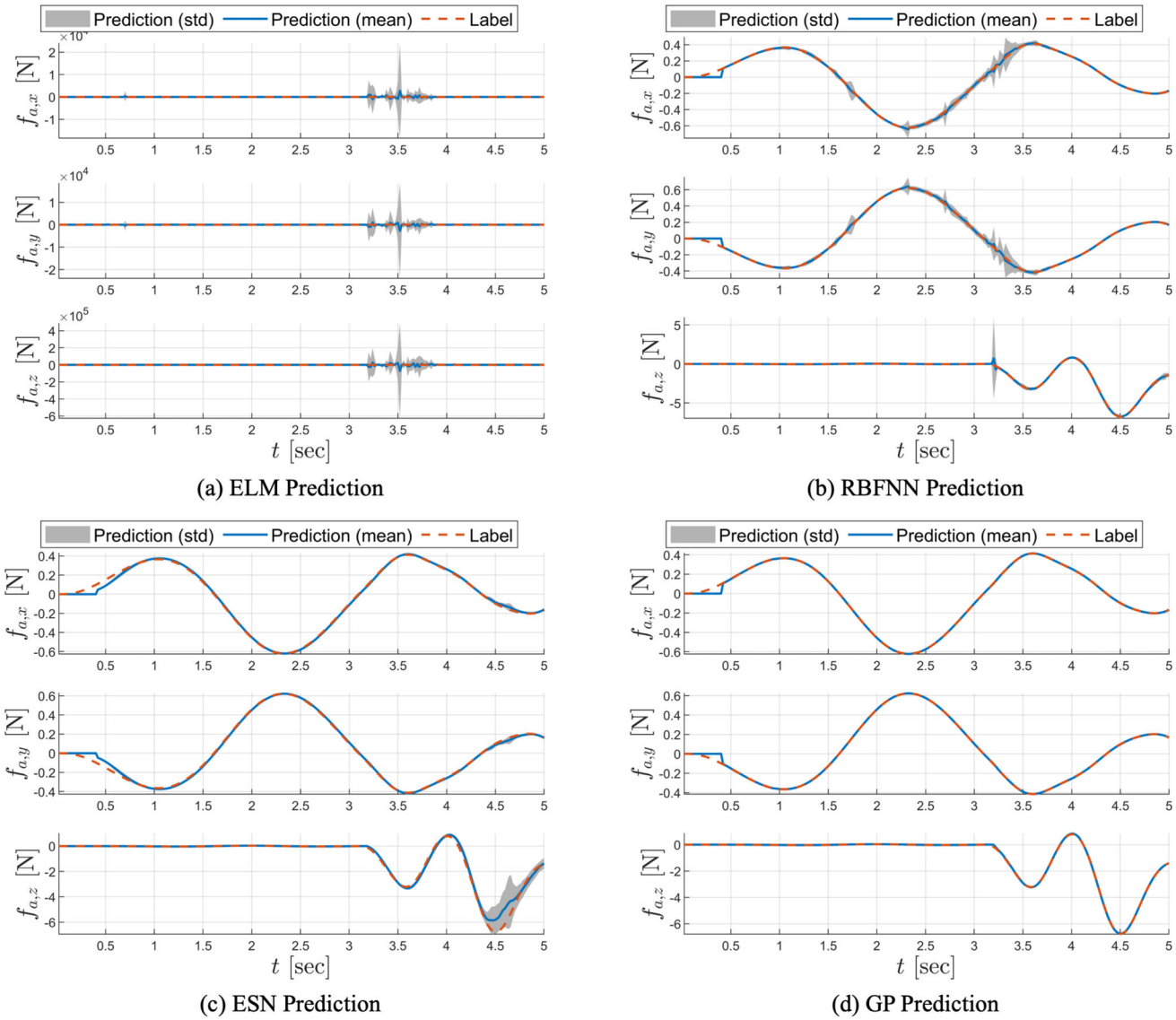

Fig. 6 Comparison of online learning performance over 50 simulation campaigns under ground effect for (a) ELM, (b) RBFNN, (c) ESN, and (d) GP

caused by the necessity of inverting an N-by-N matrix,[4] where N denotes the number of data points, in the online kernel update. This matrix inversion results in a computational complexity of $\mathcal{O}(N^3)$, which might become untenable for larger data sets (in consideration of our sampling time $T_s = 0.02$ sec). The ESN model, due to its recurrent structure within five reservoir layers, incurs additional computational costs arising from inter-layer operations, resulting in slightly longer training and inference time compared to single-layer FNN models such as ELM and RBFNN. Still, ESN demonstrates potential, offering a well-considered balance between accuracy and real-time performance. Lastly, a higher execution time is observed across all models at the initial stage of training, coinciding with the point when the data buffer reaches capacity. This effect is evident in the impulse-like fluctuations in standard deviation, as shown in Fig. 8 (particularly for GP), and can be attributed to the cold start phase of the training process.

Lastly, we investigated the influence of varying buffer sizes on the learning performance of the ELM, RBFNN, ESN, and GP models. Buffer sizes (N_b) of 50, 100, 150, and 200 were employed for online training. The resulting outcomes in terms of execution time and MAE in network prediction are summarized in Table 2. The analysis reveals that the online execution time for the four methods exhibit an increasing trend as buffer size increases. However, no

[4] Mathematically, the inverse operation is applied to $\mathbf{K} + \sigma_N^2\mathbf{I}$ as expressed in Eq. 9.

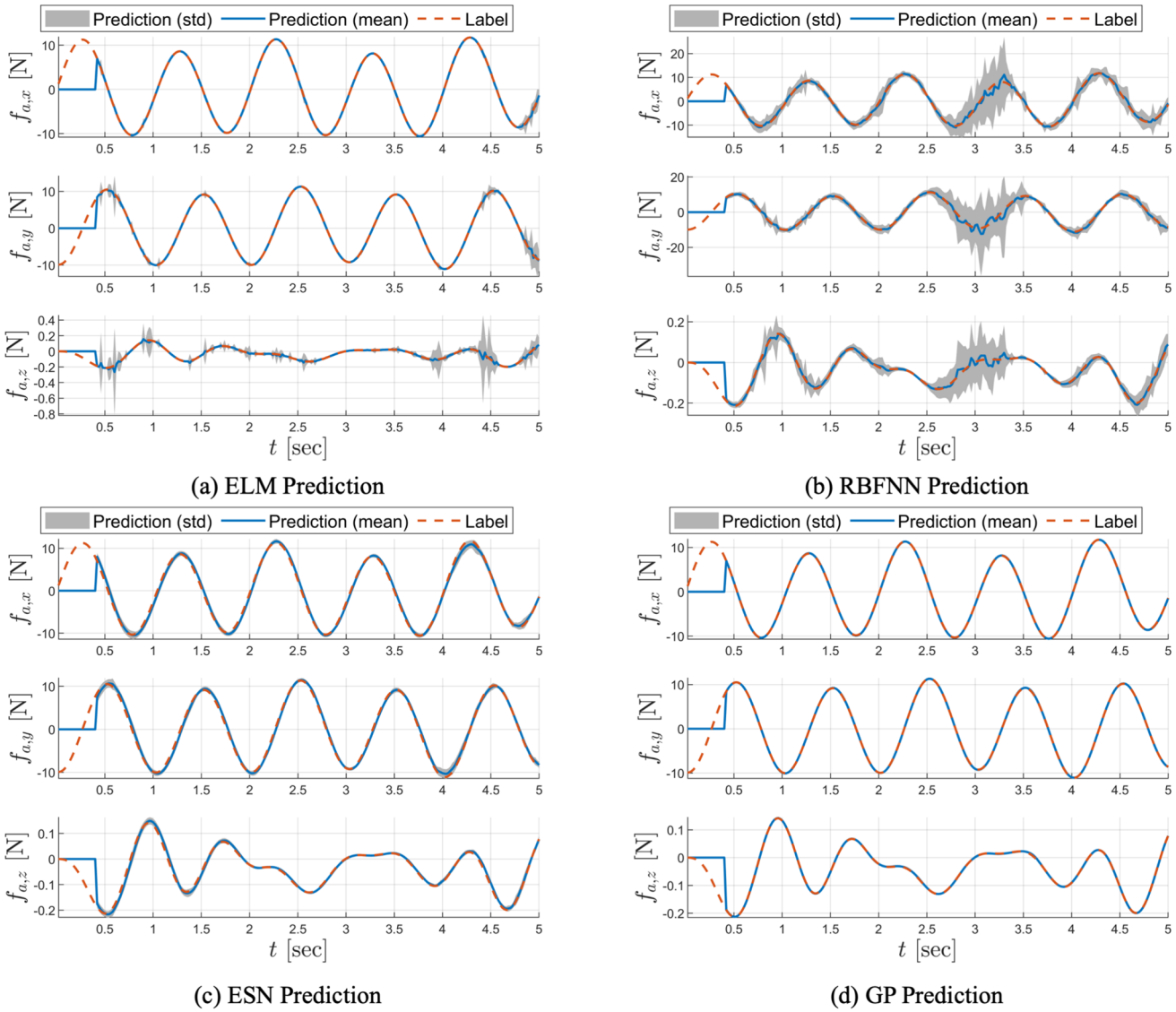

Fig. 7 Comparison of online learning performance over 50 simulation campaigns under periodic wind for (a) ELM, (b) RBFNN, (c) ESN, and (d) GP

statistically significant correlation was consistently[5] found between buffer size and prediction accuracy.

5.3 Flight control performance with online learning

The tracking performance of our online-learning-based controller was evaluated by commanding the quadrotor to track a Fig. 8 trajectory defined by $\mathbf{p}_d(t) = [\cos(0.5t), \sin(t), -5]^\top$. The quadrotor was subject to various wind disturbance models: (i) constant wind disturbances represented by $\mathbf{f}_a(t) = [5, 3, 0]^\top$, (ii) periodic wind disturbances characterized by $\mathbf{f}_a(t) = [2\sin(2\pi t), -2\sin(2\pi t), 0]^\top$, and (iii) additive Gaussian wind disturbances described as

$$f_{\text{wind}}(a, \mu, \sigma) = a \exp\left(-\frac{(t-\mu)^2}{2\sigma^2} \right), \tag{18a}$$

$$\mathbf{f}_a(t) = [\underbrace{f_{\text{wind}}(35, 3, 10) + f_{\text{wind}}(20, 5, 5)}_{(*)}, *, 0], \tag{18b}$$

representing a more intricate wind profile that comprises the summation of two Gaussian functions. We considered the quadrotor mass $m = 2.95\text{kg}$, with the inertia matrix given by $\mathbf{J} = \text{diag}([0.5, 0.5, 0.5])\text{kg m}^2$. The initial conditions were set as $\mathbf{p}(0) = [0, 0, -5]^\top$, $\mathbf{v}(0) = \boldsymbol{\eta}(0) = \boldsymbol{\omega}(0) = \mathbf{0}_{3\times 1}$.

[5] While more training data generally improves accuracy, challenges arise regarding data representativity and overfitting. For example, the ELM with a buffer size of 200 does not demonstrate the best performance across all trials, while the ESN with a buffer size of 200 performs worse than that with a buffer size of 100. We will leave the design of a data selection mechanism for future work.

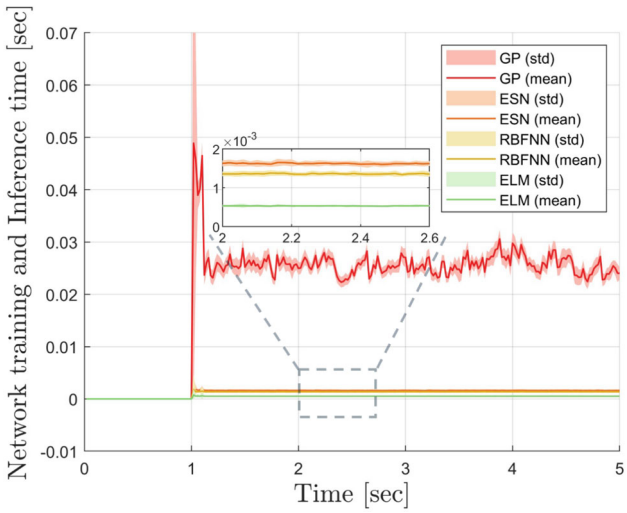

Fig. 8 Comparison of online execution time over 50 simulation campaigns under ground effect for ELM, RBFNN, ESN, and GP

Same model parameters were used for learning methods as presented in Table 1. The simulation duration was selected to be 12.5 sec, with a sampling time of $T_s = 0.02$ sec. The buffer size was chosen to be $N_b = 50$.

Figure 9 presents the tracking performance of our online-learning-based controller, leveraging four learning methods, for the quadrotor subject to constant wind, periodic wind, and additive Gaussian wind. To facilitate comparison, the tracking error was normalized for each uncertainty scenario. In Fig. 9(a) and (c), all methods demonstrate comparable performance, with the larger deviation from the reference trajectory in Fig. 9(a) attributed to the significant initial force caused by constant wind at the start of the flight. In Fig. 9(b), the controllers using ESN and GP exhibit superior performance compared to that using ELM, with RBFNN's counterpart performing the worst. This underperformance of RBFNN is consistent with its previously observed lower prediction accuracy relative to other learning methods.

Moreover, the online-learning-based controller with RBFNN encountered instability and crashes during the simulation due to significant prediction errors, as shown in Fig. 7(b). Similar instability was observed in simulation when the online-learning-based controller employed an ELM (caused by erratic prediction, see Fig. 6(a)) for landing tasks under ground effect, though these results are not detailed here. In contrast, the other methods did not experience such issues. This instability may be attributed to the random initialization of weights between the input and hidden layers of RBFNN and ELM, which can produce excessively large prediction values. By comparison, while ESN also utilizes random initialization, it adheres to the ESP, which ensures the stability of the reservoir dynamics and thus guarantees consistent prediction accuracy. Besides that, all learning methods demonstrated satisfactory tracking performance across all scenarios, including those involving distribution shifts, as will be discussed in Section 5.4. These results emphasize the critical role of online learning methods in our proposed control framework, effectively compensating for uncertainties and ensuring asymptotic tracking performance.

5.4 Comparison with an offline-learning-based controller

We demonstrate the superiority of online learning by comparing our proposed control framework with an offline-trained FNN controller. For this comparison, we considered the additive Gaussian wind disturbance given by Eq. 18, and that the quadrotor was tasked with executing the same figure-8 trajectory as described in Section 5.3. Unlike online-learning-based controllers, which require a data buffer for training on the fly, the offline-trained FNN controller is deployed without a buffer, as all training is completed prior to flight.

For fair comparisons, we implemented a 5-layer cascaded FNN, mirroring the network complexity of the ESN as shown

Table 2 Comparison of online execution time and prediction accuracy in relation to buffer size for ELM, RBFNN, ESN, and GP

N_b		ELM	RBFNN	ESN	GP
Online	50	6.166e-4 ± 6.381e-05	0.0014 ± 9.5818e-05	0.0018 ± 1.498e-04	0.0255 ± 0.0023
Execution	100	6.592e-4 ± 2.133e-05	0.0013 ± 3.233e-05	0.0029 ± 0.303e-04	0.0421 ± 0.0014
Time [sec]	150	7.955e-4 ± 3.441e-05	0.0020 ± 0.4544e-05	0.0041 ± 0.667e-04	0.0966 ± 0.0030
	200	8.235e-4 ± 0.606e-05	0.0021 ± 0.3813e-05	0.0052 ± 0.193e-04	0.0891 ± 0.0014
MAE in	50	0.2146 ± 0.1137	21.6977 ± 0.2395	1.5207 ± 0.0124	0.1439 ± 0.1285
Network	100	0.0856 ± 0.0167	28.1841 ± 9.1657	1.4691 ± 0.1002	0.1389 ± 0.1278
Prediction [N]	150	0.0916 ± 0.0270	23.9275 ± 0.6210	1.6958 ± 0.1247	0.0861 ± 0.1317
	200	0.1128 ± 0.0349	16.6837 ± 2.3527	2.2815 ± 0.1820	0.0593 ± 0.1335

The best performance across all trials is indicated in teal, while the worst performance is indicated in violet

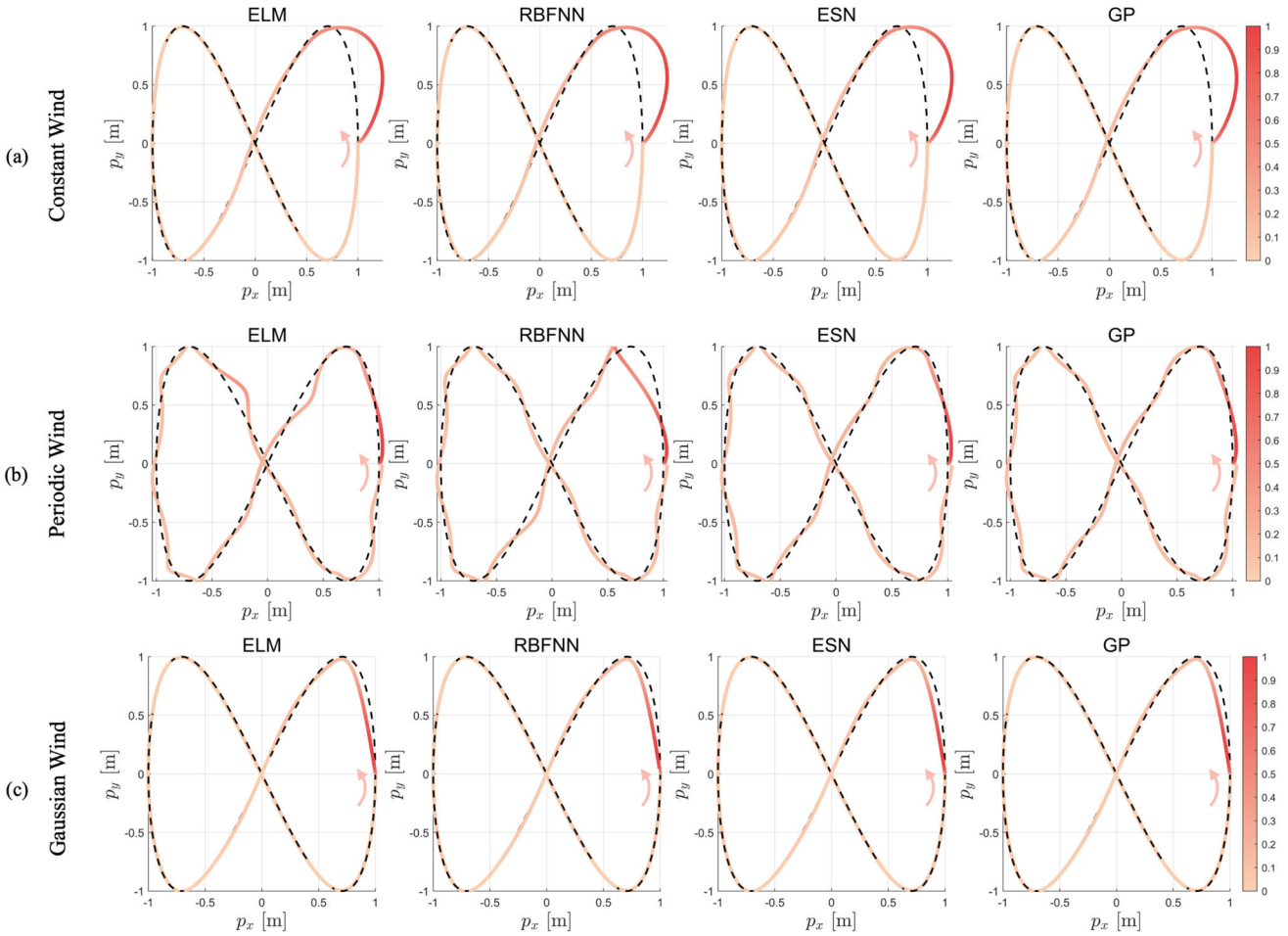

Fig. 9 Comparison of tracking performance of online-learning-based controller utilizing ELM, RBFNN, ESN, and GP for executing a figure-8 maneuver (viewed in 2D) under constant wind, periodic wind, and additive Gaussian wind conditions. The arrow indicates the starting point of the quadrotor and its flight direction, while the colorbar represents the normalized tracking error for each uncertain scenario

in Table 1. We collected 50 sec of simulated flight data, with the time interval from 0 to 28 sec used for training, while the remaining interval from 28 to 50 sec was reserved for testing. The offline training was conducted using the LM algorithm, with early stopping applied to mitigate overfitting. Notably, as illustrated in Fig. 10(a)-(b), the test data includes both ID data, covering the intervals from 28 to 31 sec and 39 to 50 sec (shaded in blue), and OOD data, spanning from 31 to 39 sec (shaded in red). A distribution shift is evident in both the network inputs (i.e., Euler angles) and the predictions (i.e., aerodynamic forces exceeding the horizontal dotted line in Fig. 10(b)), introducing considerable challenges for learning tasks.

As shown in Fig. 10(b), the offline-trained FNN demonstrated poor generalization to OOD data, resulting in substantial prediction errors. These inaccurate predictions further degraded the tracking performance of the offline-trained FNN controller, leading to unacceptable behaviors in the OOD data regime. Figure 10(c) provides a 3D illustration of such failure. In contrast, our online-learning-based controller (taking ESN as an example) achieved superior tracking performance thanks to the online learning capability of ESN, which effectively compensated for dynamic uncertainties even within the OOD regime. This underscores the critical limitations of offline-trained methods and emphasizes the significant potential of online learning approaches for adaptation in dynamic environments.

5.5 Real-time performance on an embedded platform

To demonstrate that our approach meets the stringent latency requirements of real-time control on a resource-constrained onboard platform, we deployed all online learning modules directly on an NVIDIA Jetson Orin NX (16 GB) and integrated them with the open-source `gym-pybullet-drones` simulator [41] (see Fig. 11). Executing both the simulator and learning algorithms on the same embedded

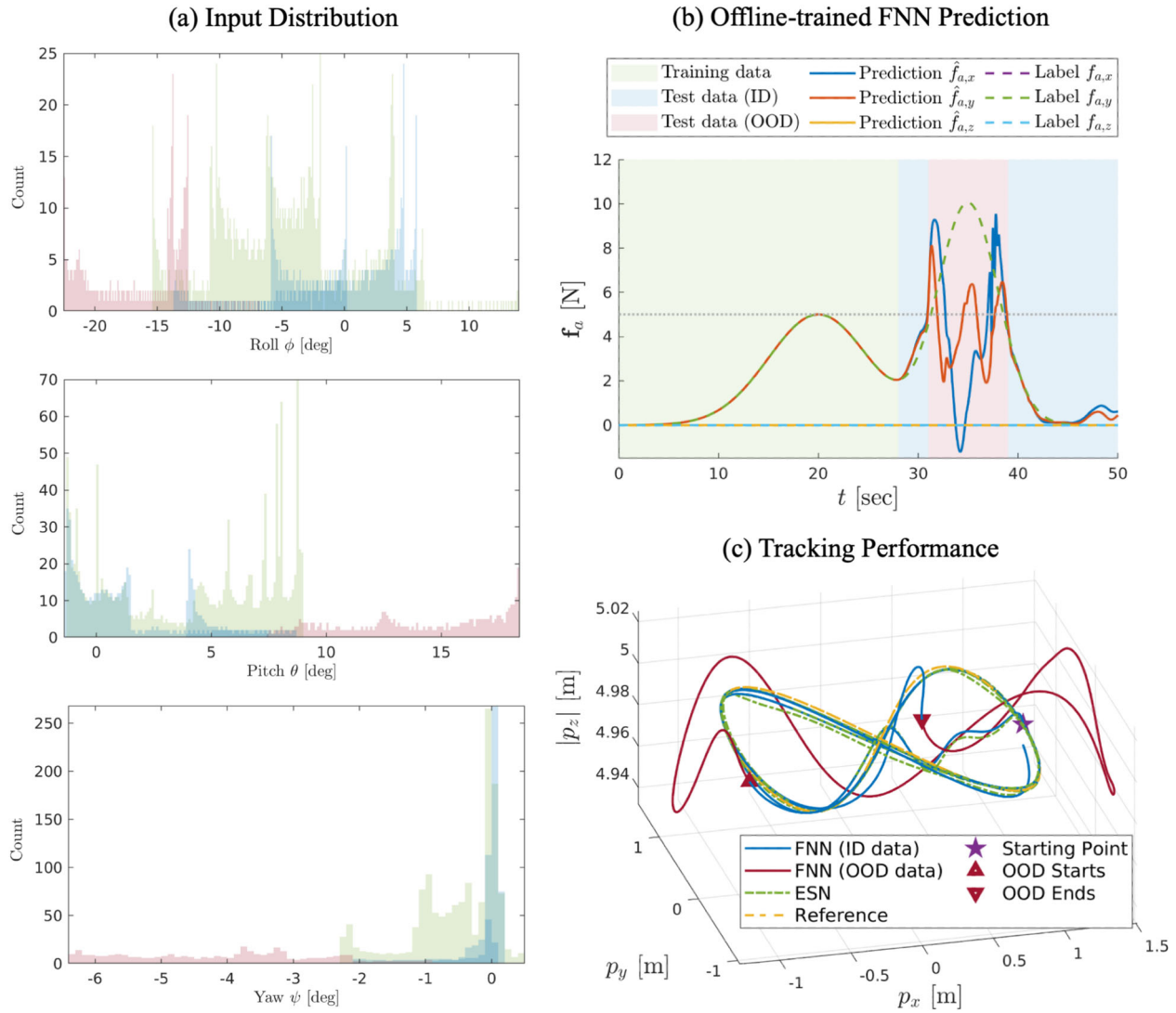

Fig. 10 Comparison of control behavior between online and offline learning-based controllers. (a) Distribution shift in Euler angles (network inputs). (b) Distribution shift in network predictions and FNN learning performance. (c) Tracking performance (viewed in 3D) of a figure-8 trajectory. This figure is redrawn after [24]

device enables a realistic evaluation of computational cost and control responsiveness under deployment conditions.

From a computational standpoint, GP incurs $\mathcal{O}(N^3)$ time complexity (for matrix inversion) and $\mathcal{O}(N^2)$ memory (for kernel matrix storage) during training (as discussed in Section 5.2), which becomes prohibitive for moderate sample sizes (e.g., $N \gtrsim 10^4$) on embedded hardware. While sparse or local GP approximations (e.g., Stochastic Variational Gaussian Process (SVGP) [42]) can reduce this to $\mathcal{O}(NM^2)$ (where $M \ll N$ is the number of inducing points), they introduce approximation errors and hyperparameter tuning overhead. In contrast, ESN, ELM, and RBFNN train only a linear readout layer. That being said, it takes only $\mathcal{O}(N_h^3)$ for closed-form solutions via ridge regression during training, and $\mathcal{O}(N_h d)$ for inference per step, where N_h is the number of hidden neurons and d is the input/output dimension. This efficiency makes them inherently suitable for online learning and real-time embedded execution. To further improve numerical robustness and speed for the readout estimation, we addressed the output-layer least-squares problem with an Singular Value Decomposition (SVD)-based solver (`np.linalg.lstsq`) rather than a direct linear solver. This choice is more stable under high-dimensional states and potential rank deficiency and can significantly reduce training time on the embedded platform.

We benchmarked training and inference times for GP, ESN, ELM, and RBFNN in both simulation and hardware settings, reporting means and standard deviations on a logarithmic time axis (in seconds) and marking the 5 ms sampling time corresponding to 200 Hz control (see Fig. 12). Across all

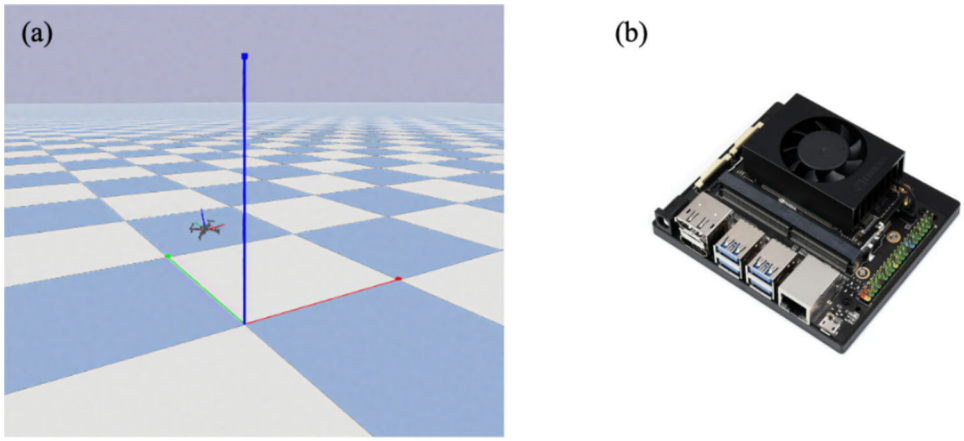

Fig. 11 Experimental setup for real-time performance evaluation. (a) The open-source `gym-pybullet-drones` simulation environment. (b) The NVIDIA Jetson Orin NX (16 GB) embedded computing platform used for onboard deployment

models, wall-clock times on hardware are roughly an order of magnitude lower than in simulation thanks to the SVD-based solver, with execution time in the millisecond regime and meeting the 5 ms real-time threshold. Model-specific behaviors align with their computational characteristics. In simulation, GP requires substantially longer training time relative to inference; on hardware, the training time is reduced to the millisecond scale, with inference accounting for a larger portion of the total cost. ESN exhibits lower overall computational load than GP, and hardware execution further reduces both training and inference times. ELM and RBFNN achieve consistently fast inference in both simulation and hardware, although with larger variance compared to ESN and GP. In summary, hardware experiments show that all models except GP satisfy the 200 Hz real-time requirement, whereas GP demands hardware acceleration and/or sparse approximations to achieve feasible training times. Among the lightweight models, ELM and RBFNN provide the fastest execution, while ESN offers a more balanced trade-off between real-time performance, accuracy, and stability.

6 Conclusion and future work

In this work, we presented a unified online-learning-based control framework that integrates a residual learning module for uncertainty estimation with a baseline controller for quadrotor trajectory tracking. Within this framework, we systematically compared representative parametric and non-parametric approaches, including ELM, RBFNN, ESN, and GP. To ensure a fair comparison, we employed BO to tune the hyperparameters of all parametric models under consistent

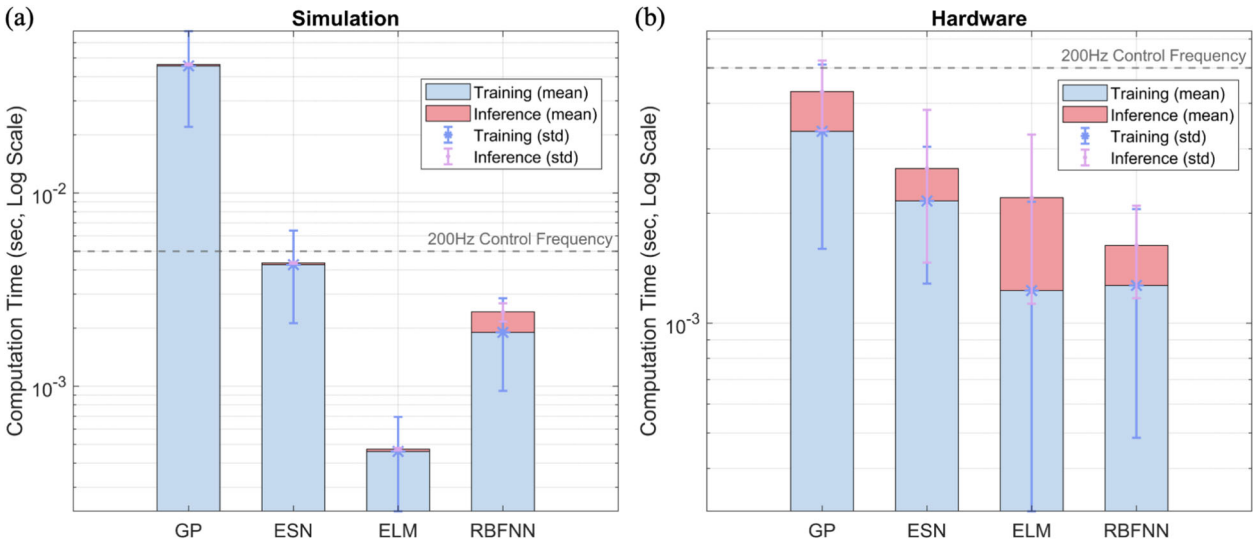

Fig. 12 Comparison of computation times for different online learning models in (a) simulation and (b) embedded hardware execution (NVIDIA Jetson Orin NX, 16 GB). Bars show mean training (blue) and inference (red) times with standard deviations indicated by error bars. The dashed line denotes the 200 Hz control frequency threshold (5 ms)

conditions, allowing each to operate near its best achievable performance. Furthermore, we deployed the entire framework on an embedded NVIDIA Jetson Orin NX platform and evaluated its ability to meet the stringent requirements of 200 Hz real-time control.

The results of extensive simulation campaigns, corroborated by hardware experiments, reveal clear trade-offs among the methods. The GP model achieves the highest accuracy in uncertainty estimation and trajectory tracking, but its cubic training complexity limits real-time feasibility without hardware acceleration or sparse approximations. ELM and RBFNN demonstrate extremely fast runtimes and low computational cost, yet their sensitivity to initialization leads to unstable prediction performance that may compromise control stability. By contrast, the ESN, benefiting from its recurrent dynamics and the ESP, provides the most balanced trade-off, achieving robust real-time performance with improved stability compared to the lightweight alternatives and comparative accuracy compared to the GP model.

For future work, we plan to extend the framework to address the impact of noisy sensor measurements, which can pose additional challenges for online adaptation. We also aim to validate the approach experimentally on a real quadrotor platform to assess robustness and adaptability in realistic operating conditions. Beyond this, we envision investigating hybrid strategies, such as combining lightweight parametric models with non-parametric uncertainty quantification, to further enhance reliability while maintaining real-time feasibility.

Author Contributions W. Gu (Conceptualization, Methodology, Software, Data Curation, Original Draft Preparation); J. Zhao (Investigation, Visualization, Original Draft Preparation); A. Rizzo (Review and Editing, Funding Acquisition, Supervision)

Funding Open access funding provided by Politecnico di Torino within the CRUI-CARE Agreement. This study was carried out within the MOST – Sustainable Mobility National Research Center and received funding from the European Union Next-GenerationEU (PIANO NAZIONALE DI RIPRESA E RESILIENZA (PNRR) – MISSIONE 4 COMPONENTE 2, INVESTIMENTO 1.4 – D.D. 1033 17/06/2022, CN00000023).
Also, it was carried out within the FAIR - Future Artificial Intelligence Research and received funding from the European Union Next-GenerationEU (PIANO NAZIONALE DI RIPRESA E RESILIENZA (PNRR) – MISSIONE 4 COMPONENTE 2, INVESTIMENTO 1.3 – D.D. 1555 11/10/2022, PE00000013).
This manuscript reflects only the authors' views and opinions, neither the European Union nor the European Commission can be considered responsible for them.

Data Availability Data sets generated during the current study are available from the corresponding author on reasonable request.

Declarations

Ethics approval and consent to participate Not applicable.

Consent for publication Not applicable.

Conflicts of Interest Alessandro Rizzo is a Senior Editor-at-Large of the Journal of Intelligent and Robotic Systems. The authors have no other conflicts of interest that are relevant to the contnt of this article.

References

1. Mellinger, D., Kumar, V.R.: Minimum snap trajectory generation and control for quadrotors. IEEE International Conference on Robotics and Automation. **2011**, 2520–2525 (2011)
2. Conyers, S.A., Rutherford, M.J., Valavanis, K.P.: An empirical evaluation of ground effect for small-scale rotorcraft. In: 2018 IEEE international conference on robotics and automation (ICRA). pp. 1244–1250. IEEE (2018)
3. David Du Mutel de Pierrepont Franzetti, I., Parin, R., Capello, E., Rutherford, M., Valavanis, K.: Ground, ceiling and wall effect evaluation of small quadcopters in pressure-controlled environments. J. Intell. Robotic Syst. **08**, 110. https://doi.org/10.1007/s10846-024-02155-7
4. Bauersfeld, L., Kaufmann, E., Foehn, P., Sun, S., Scaramuzza, D.: NeuroBEM: Hybrid Aerodynamic Quadrotor Model. (2021). ArXiv:abs/2106.08015
5. Penicka, R., Song, Y., Kaufmann, E., Scaramuzza, D.: Learning Minimum-Time Flight in Cluttered Environments. IEEE Robotics and Automation Letters. **7**(3), 7209–7216 (2022). https://doi.org/10.1109/LRA.2022.3181755
6. Gu, W., Hu, D., Cheng, L., Cao, Y., Rizzo, A., Valavanis, K.P.: Autonomous wind turbine inspection using a quadrotor. In: 2020 international conference on unmanned aircraft systems (ICUAS). pp. 709–715 (2020)
7. Lee, T.: Robust Adaptive Attitude Tracking on SO(3) With an Application to a Quadrotor UAV. IEEE Trans. Control Syst. Technol. **21**, 1924–1930 (2013)
8. Ou, T.W., Liu, Y.: Adaptive backstepping tracking control for quadrotor aerial robots subject to uncertain dynamics. Am. Control Conf. (ACC) **2019**, 1–6 (2019)
9. Zhu, B., Huo, W.: Adaptive backstepping control for a miniature autonomous helicopter. In: 2011 50th IEEE conference on decision and control and european control conference. pp. 5413–5418 (2011)
10. Gu, W., Primatesta, S., Rizzo, A.: Robust adaptive control for aggressive quadrotor maneuvers via SO(3) and backstepping techniques. Robot. Auton. Syst. **188**, 104942 (2025)
11. Goodarzi, F.A., Lee, D., Lee, T.: Geometric adaptive tracking control of a quadrotor unmanned aerial vehicle on SE(3) for agile maneuvers. J. Dyn. Syst. Meas. Control-Trans. Asme. **137**, 091007 (2014)

12. Chingozha, T., Nyandoro, O.T.: Adaptive sliding backstepping control of quadrotor UAV attitude. IFAC Proc **47**, 11043–11048 (2014)
13. Zhao, B., Xian, B., Zhang, Y., Zhang, X.: Nonlinear robust adaptive tracking control of a quadrotor UAV via immersion and invariance methodology. IEEE Trans. Industr. Electron. **62**(5), 2891–2902 (2015). https://doi.org/10.1109/TIE.2014.2364982
14. Ioannou, P.A., Sun, J.: Robust Adaptive Control. (2012)
15. Gu, W., Valavanis, K.P., Rutherford, M.J., Rizzo, A.: UAV model-based flight control with artificial neural networks: A survey. J. Intell. Robotic Syst. **100**, 1469–1491 (2020)
16. Mohajerin, N., Mozifian, M., Waslander, S.: Deep learning a quadrotor dynamic model for multi-step prediction. In: 2018 IEEE international conference on robotics and automation (ICRA), p.p 2454–2459 (2018)
17. Saviolo, A., Li, G., Loianno, G.: Physics-inspired temporal learning of quadrotor dynamics for accurate model predictive trajectory tracking. IEEE Robotics Autom. Lett. **7**(4), 10256–10263 (2022)
18. Gu, W., Primatesta, S., Rizzo, A.: Physics-informed neural network for quadrotor dynamical modeling. Robot. Auton. Syst. **171**, 104569 (2024). https://doi.org/10.1016/j.robot.2023.104569
19. Shi, G., Shi, X., O'Connell, M., Yu, R., Azizzadenesheli, K., Anandkumar, A., et al.: Neural Lander: Stable Drone Landing Control Using Learned Dynamics. Int. Conf. Robotics Autom (ICRA) **2018**, 9784–9790 (2019)
20. Shi, G., Honig, W., Shi, X., Yue, Y., Chung, S.J.: Neural-swarm2: Planning and control of heterogeneous multirotor swarms using learned interactions. IEEE Trans. Rob. **38**, 1063–1079 (2020)
21. O'Connell, M., Shi, G., Shi, X., Azizzadenesheli, K., Anandkumar, A., Yue, Y., et al.: Neural-Fly enables rapid learning for agile flight in strong winds. Sci. Robotics. **7** (2022)
22. Deshpande, A.M,. Minai, A.A., Kumar, M.: Robust deep reinforcement learning for quadcopter control. IFAC-PapersOnLine. *54*(20):90–95. Modeling, Estimation and Control Conference MECC 2021. (2021). https://doi.org/10.1016/j.ifacol.2021.11.158
23. Song, Y., Steinweg, M., Kaufmann, E., Scaramuzza, D. :Autonomous drone racing with deep reinforcement learning. In: 2021 IEEE/RSJ international conference on intelligent robots and systems (IROS), pp. 1205–1212 (2021)
24. Gu, W., Rizzo, A.: Online residual learning using interpretable reservoir computing for quadrotor control. In: 2024 international conference on unmanned aircraft systems (ICUAS), pp. 23–30 (2024)
25. Shi, L., Mucchiani, C., Karydis, K.: Online modeling and control of soft multi-fingered grippers via koopman operator theory. In: 2022 IEEE 18th international conference on automation science and engineering (CASE), pp. 1946–1952 (2022)
26. Jiahao, T.Z., Chee, K.Y., Hsieh, M.A.: Online dynamics learning for predictive control with an application to aerial robots. In: Liu, K., Kulic, D., Ichnowski, J., (eds.) Proceedings of the 6th conference on robot learning of proceedings of machine learning research, vol. 205, pp. 2251–2261. PMLR (2023). Available from: https://proceedings.mlr.press/v205/jiahao23a.html
27. Zheng, L., Yang, R., Pan, J., Cheng, H.: Safe learning-based tracking control for quadrotors under wind disturbances. In: American control conference (ACC). **2021**, 3638–3643. IEEE (2021)
28. Munoz, E., Kalaria, D., Lin, Q., Dolan, J.M.: Online adaptive compensation for model uncertainty using extreme learning machine-based control barrier functions. In: 2022 IEEE/RSJ international conference on intelligent robots and systems (IROS). pp. 10959–10966 IEEE (2022)
29. Sönmez, S., Rutherford, M.J., Valavanis, K.P.: A survey of offline- and online-learning-based algorithms for multirotor UAVs. Drones. **8**(4) (2024). https://doi.org/10.3390/drones8040116
30. Huang, G.B., Zhu, Q.Y., Siew, C.K.: Extreme Learning Machine: Theory and Applications. Neurocomputing **70**(1–3), 489–501 (2006)
31. Park, J., Sandberg, I.W.: Universal Approximation using Radial-Basis-Function Networks. Neural Comput. **3**(2), 246–257 (1991)
32. Jaeger, H.: The echo state approach to analysing and training recurrent neural networks; (2001). Available from: https://api.semanticscholar.org/CorpusID:15467150
33. Rasmussen, C.E.: Gaussian processes in machine learning. In: Summer school on machine learning. pp. 63–71. Springer (2003)
34. Martini, S., Stefanovic, M., Rizzo, A., Rutherford, M.J., Livreri, P., Valavanis, K.P.: A benchmark framework for testing, evaluation, and comparison of quadrotor linear and nonlinear controllers. In: 2023 international conference on unmanned aircraft systems (ICUAS). pp. 471–478 (2023)
35. Saviolo, A., Loianno, G.: Learning quadrotor dynamics for precise, safe, and agile flight control. Annu. Rev. Control. **55**, 45–60 (2023). https://doi.org/10.1016/j.arcontrol.2023.03.009
36. Folkestad, C., Pastor, D., Burdick, J.W.: Episodic Koopman learning of nonlinear robot dynamics with application to fast multirotor landing. In: 2020 IEEE international conference on robotics and automation (ICRA). pp. 9216–9222. IEEE (2020)
37. Kaufmann, E., Bauersfeld, L., Loquercio, A., Mueller, M., Koltun, V., Scaramuzza, D.: Champion-level drone racing using deep reinforcement learning. Nature **08**(620), 982–987 (2023). https://doi.org/10.1038/s41586-023-06419-4
38. Lukoševičius, M., Jaeger, H.: Reservoir computing approaches to recurrent neural network training. Comput. Sci. Rev. **3**(3), 127–149 (2009). https://doi.org/10.1016/j.cosrev.2009.03.005
39. Gallicchio, C., Micheli, A.: Echo state property of deep reservoir computing networks. Cogn. Comput. **9**, 337–350 (2017)
40. Bull, A.D.: Convergence rates of efficient global optimization algorithms. J. Mach. Learn. Res. 2011;12(10)
41. Panerati, J., Zheng, H., Zhou, S., Xu, J., Prorok, A., Schoellig, A.P.: Learning to fly—a gym environment with pybullet physics for reinforcement learning of multi-agent quadcopter control. pp. 7512–7519. (2021). https://doi.org/10.1109/IROS51168.2021.9635857
42. Hensman, J., Fusi, N., Lawrence, N.D.: Gaussian processes for big data. (2013) arXiv preprint arXiv:1309.6835

Weibin Gu is currently a Postdoctoral Researcher at the Institute for AI Industry Research (AIR), Tsinghua University, China. He earned his Ph.D. in Electronics and Telecommunications from Politecnico di Torino, Italy, in 2024, his M.Sc. in Mechatronic Engineering from the same institution in 2017, and his B.Sc. in Mechanical Engineering through a dual-degree program at Tongji University, China, and Politecnico di Torino in 2015. From September 2019 to October 2020, he served as a Flight Control Engineer and Project Lead at Shanghai FOlA Co., Ltd., China, where he led a UAV-based autonomous project for industrial applications. Between 2017 and 2020, he also collaborated on UAV research with the University of Denver, U.S., and Politecnico di Torino. His research interests are centered on the intersection of machine learning and control theory, with a focus on realworld robotic and autonomous systems.

Jiance Zhao is currently a Master's student at the School of Aeronautic Science and Engineering, Beihang University. He received his B.Eng. degree in Aircraft Design and Engineering from Shenyang Aerospace University in 2023. His research interests mainly focus on optimization-based control and learning-based control for robotics and autonomous systems.

Alessandro Rizzo is an Associate Professor at the Department of Electronics and Telecommunications, Politecnico di Torino, Italy, where he coordinates the Complex Systems Laboratory. He received his Laurea degree in Computer Engineering, summa cum laude, and his PhD in Automation and Electronics Engineering from the University of Catania, in 1996 and 2000, respectively. Dr. Rizzo's previous affiliations include JET Joint Undertaking, ST Microelectronics ,the University of Messina, Politecnico di Bari, and New York University. His research interests span complex networks and systems, robotics, and the modeling and control of nonlinear systems. He has authored one book, holds two international patents, and has published over 200 papers in peer-reviewed international journals and conference proceedings. A Distinguished Lecturer for the IEEE Nuclear and Plasma Sciences Society, he received the Best Application Paper Award at the 2002 IFAC World Congress, as well as two Amazon Research Awards in Robotics (2019, 2021).