UAV Model-based Flight Control with Artificial Neural Networks: A Survey

Journal of Intelligent & Robotic Systems

UAV Model-based Flight Control with Artificial Neural Networks: A Survey

--Manuscript Draft--

Manuscript Number:	JINT-D-20-00078
Full Title:	UAV Model-based Flight Control with Artificial Neural Networks: A Survey
Article Type:	Survey Paper
Keywords:	Model-based control (MBC); Artificial neural network (ANN); Flight control; Hybridization; Unmanned aerial vehicle (UAV)
Corresponding Author:	Weibin Gu University of Denver Denver, UNITED STATES
Corresponding Author Secondary Information:	
Corresponding Author's Institution:	University of Denver
Corresponding Author's Secondary Institution:	
First Author:	Weibin Gu
First Author Secondary Information:	
Order of Authors:	Weibin Gu
	Kimon P. Valavanis
	Matthew J. Rutherford
	Alessandro Rizzo
Order of Authors Secondary Information:	
Funding Information:	National Science Foundation (CMMI-DCSD-1728454) — Dr. Kimon P. Valavanis
	TIM JOL "Crab" — Dr. Alessandro Rizzo
	the Siebel Energy Institute — Dr. Alessandro Rizzo
	Compagnia di San Paolo — Dr. Alessandro Rizzo
Abstract:	Model-Based Control (MBC) techniques have dominated UAV flight controller designs. Despite their success, MBC-based designs rely heavily on the accuracy of the mathematical model of the real plant and they suffer from the explosion of complexity problem. These two challenges may be mitigated by Artificial Neural Networks (ANNs) that have been widely studied due to their unique features and advantages in system identification and controller design. Viewed from this perspective, this survey provides a comprehensive literature review on combined MBC-ANN techniques that are suitable for UAV flight control, i.e., low-level control. The objective is to pave the way and establish a foundation for efficient controller designs with performance guarantees. A reference template is used throughout the survey as a common basis for comparative studies to fairly determine capabilities and limitations of existing research. The end-result offers supported information for advantages, disadvantages and applicability of a family of relevant controllers to UAV prototypes.

Journal of Intelligent & Robotic Systems manuscript No.
(will be inserted by the editor)

UAV Model-based Flight Control with Artificial Neural Networks: A Survey

Weibin Gu · Kimon P. Valavanis · Matthew J. Rutherford · Alessandro Rizzo

Received: date / Accepted: date

Abstract Model-Based Control (MBC) techniques have dominated UAV flight controller designs. Despite their success, MBC-based designs rely heavily on the accuracy of the mathematical model of the real plant and they suffer from the explosion of complexity problem. These two challenges may be mitigated by Artificial Neural Networks (ANNs) that have been widely studied due to their unique features and advantages in system identification and controller design. Viewed from this perspective, this survey provides a comprehensive literature review on combined MBC-ANN techniques that are suitable for UAV flight control, i.e., low-level control. The objective is to pave the way and establish a foundation for efficient controller designs with performance guarantees. A reference template is used throughout the survey as a common basis for comparative studies to fairly determine capabilities and limitations of existing research. The end-result offers supported information for advantages, disadvantages and applicability of a family of relevant controllers to UAV prototypes.

Keywords Model-based control (MBC) · Artificial neural network (ANN) · Flight control · Hybridization · Unmanned aerial vehicle (UAV)

This work is partially supported by an NSF Grant, CMMI-DCSD-1728454, TIM JOL "Crab", the Siebel Energy Institute, and Compagnia di San Paolo.

Weibin Gu · Kimon P. Valavanis · Matthew J. Rutherford
DU Unmanned Systems Research Institute (DU2SRI)
Daniel Felix Ritchie School of Engineering & Computer Science
University of Denver
2390 S. York St., Denver, CO 80208
E-mail: {Weibin.Gu, Kimon.Valavanis, Matthew.Rutherford}@du.edu
www.du2sri.com

Alessandro Rizzo
Department of Electronics and Telecommunications
Politecnico di Torino
Corso Duca degli Abruzzi, 24, 10129 Torino TO, Italia
E-mail: alessandro.rizzo@polito.it

1 Introduction

1.1 Motivation and Rationale

Model-Based Control (MBC) techniques have found great applicability in UAV controller design. This is because control-oriented modeling, being the core of MBC techniques, allows for a systematic way to controller synthesis that basically facilitates the design and development process by reducing the effort to the tuning and calibration processes. Moreover, the focus of MBC techniques is on guaranteeing system stability and performance, improving robustness with respect to uncertainties and disturbances, and finding the optimally designed controller. Recent findings are presented in Feedback Linearization (FL), a.k.a. Nonlinear Dynamic Inversion (NDI) [1], Adaptive Control [2], Model Predictive Control (MPC) [3], Sliding Mode Control (SMC) [4], Backstepping Control [5] and $\mathcal{H}_\infty$ Robust Control [6].

Despite promising results, a challenge MBC-based designs suffer from, is *their dependence on the accuracy of the mathematical model of the real plant* [7]. A poorly derived or defined model, due to imprecise system knowledge and ubiquitous exogenous disturbances, may adversely impact subsequent controller synthesis that leads to unacceptable performance or even instability. Such uncertainties and disturbances may be classified as:

- Parametric uncertainties: These are normally induced by miss-modeling and/or system degradation (e.g., mass and inertia changes, etc).
- Unmodeled dynamics: These refer to difficult-to-model, ill-defined and intentionally ignored aspects of a nonlinear model, which include advanced aerodynamic effects such as blade flapping [8], effect of airflow [9], ground and ceiling effect [10,11], etc.
- Disturbances and noise: Disturbances may include wind gusts and turbulence, while noise mainly refers to sensor noise. Considered assumptions may not be realistic as the statistical properties of sensor noise is mostly non-Gaussian in practice.

To tackle such challenges, a wide spectrum of modern control techniques has been proposed, each with advantages, limitations and drawbacks. For example, although a widely-used technique is Gain Scheduling (GS) [12] that demonstrates good capabilities of coping with parameter variations and nonlinearities, frequent and fast changes of the controller gains may drive the system unstable [13]. In addition, the cost of design and implementation increases with the number of operating points as stated in [14]. On the other hand, robust control is effective when considering bounded parametric uncertainties but it has limitations when considering either unbounded ones or unmodeled dynamics [6,15]. Adaptive control, as a viable solution (due to its strength of real-time adaptation) is suitable to manage parametric uncertainties; however, there barely exist widely accepted solutions to the robust adaptive control problem, so far [16]. Sliding mode control has been shown to be robust with respect to modeling errors and parameter uncertainties, but chattering

is an issue occurred by frequent controller switches. Besides, characteristics of insensitive to parameter changes may pose problems to self-stabilization when exogenous disturbances appear. Last, but not least, Model-Free Control (MFC) techniques that have emerged to overcome unmodeled dynamics and uncertainties of nonlinear systems, have demonstrated great adaptation and estimation capabilities due to the use of a continuously updated model, namely ultra-local model [17,18]. However, this methodology is currently limited to system dynamics that can be converted to Single-Input Single-Output (SISO) subsystems. Besides, there are questions related to analytic stability and convergence proofs.

Another important challenge of MBC techniques relates to *the explosion of complexity* issue [19]. This arises from, but it is not limited to: the mathematical inverse model required in feedback linearization; the repeated differentiation of virtual controllers in backstepping control; the prediction over a future horizon of the plant behavior as well as the online optimization process in MPC. All three limitations may result in a hard-to-implement in real-time controller due to the computational burden, particularly when it comes to highly nonlinear complex systems, such as UAVs, in which the complexity of the controller increases drastically as the system order increases.

To overcome the previously stated challenges of MBC-based designs, Artificial Neural Networks (ANNs) are being exploited to study complex control systems. This is mostly attributed to the perceived advantages of ANNs in system identification and controller design, to say the least [20–22], including: their capability to identify nonlinear and multi-variable systems [23,24]; ability to learn and adapt in real time; relatively easy processing procedure and hardware implementation.

Such advantages allow for ANNs to offer a perfect tool to build the system underlying model with high accuracy and low complexity, even when corrupted with uncertainties and disturbances, and to facilitate the implementation process and to enhance real-time performance. Regardless, there still exist challenges due to their data-driven nature that precludes applicability in industry, to some extent, because of: the requirement for large sets of training data; being prone to learn spurious relationships that may lead to poor generalization capabilities [25]; lack of interpretability due to their black-box characteristics; no systematic approach for related ANN architecture designs (i.e., given a selected ANN architecture, the number of layers and neurons, the type of activation functions, weight updating mechanisms and so forth, are generally arbitrarily determined, rather than in a systematic way).

When focusing on UAVs, a flight control system, being a safety-critical system, requires focus and priority on safety and reliability. As such, the ANN black-box characteristics have contributed to their acceptance and use, at least from the practical point of view. The same problems can be tackled and handled by MBC techniques if such techniques are incorporated properly into the controller design. For example, the explicit system knowledge that MBC techniques heavily rely on, can be leveraged to facilitate the training process (by generating training data and/or speeding up the training process, etc.)

and to improve performance, generalization capabilities, and interpretability of ANNs. This results in a complementary relation between MBC techniques and ANNs, the combination of which, even by intuition, is expected to overcome stated limitations.

1.2 Problem Statement and Contributions

The objective of this paper is to design and implement an MBC flight controller combined with ANNs, MBC-ANN flight controller, to achieve UAV stable and robust autonomous flight under uncertainties and disturbances, and with reduced design and implementation complexity. To achieve this goal, the following issues need to be addressed and solved:

i. (*Stability*): How can system stability and weight convergence of ANNs is guaranteed when combining MBC techniques with ANNs?
ii. (*Robustness*): How ANNs may be exploited during modeling or controller synthesis along with MBC techniques to cope with uncertainties and disturbances?
iii. (*Paucity of data*): How sufficient and suitable training data becomes available, particularly when online training (or adaption) is engaged?
iv. (*Interpretability*): How interpretability of ANNs is improved to make them trustworthy to the users?
v. (*Real-time performance*): What is the computational complexity of the designed controller? Is it implementable in real-time? Is the ANN architecture suitable for hardware implementation?

Technically justified and supported answers to the above questions require a comprehensive literature review of MBC techniques combined with ANNs, with emphasis on UAV low-level flight control. This will pave the way to efficient controller designs with performance guarantees, resulting in the following contributions:

- A thorough review of existing literature related to different techniques, which are compared in a unified framework (i.e., reference template) to analyze advantages and limitations.
- Study and determination of the computational complexity of several frequently used ANNs in UAV flight control applications to provide a-priori knowledge of real-time performance during the design phase.

The remainder of this paper is organized as follows. Section 2 presents a reference template to provide a common basis for comparative studies and to fairly determine the capabilities and limitations of existing work. An analysis on computational complexity is also carried out for several frequently used ANNs, therein. Section 3 and 4 conduct the comparison and classification of approaches based on a thorough literature review on dynamic modeling and control techniques, respectively. Section 5 concludes this survey and sheds light on the future work.

2 Reference Template

It is essential to provide the rationale for the adopted reference template that will be followed. As such, related published surveys on MBC techniques and ANNs for UAV applications have been studied to determine their limitations.

- In [26], hardware implementation of ANNs for aeronautical applications is reviewed. Some key issues on FPGA (Field-Programmable Gate Array)-based implementation are posed. However, the design of the flight control system is not treated.
- In [21], deep learning techniques and their applications for UAV-based solutions are surveyed, mainly focusing on high-level control including path planning, situation awareness, etc. Nonetheless, there is a lack of elaboration on low-level control.
- In [27], a general review on advances of modeling and control of UAV maneuvering flights is given, whereas MBC techniques and ANNs are separately discussed.

There is no literature review on how ANNs have been used in the MBC systems either. This survey, although built on previous studies [28], is the first comprehensive review on model-based flight control combined with ANNs.

Next, a unified framework is presented, namely the proposed *reference template* to facilitate comparative studies of existing research on UAV model-based flight control with ANNs. Specifically, the comprehensive literature review is conducted based on set objectives in the reference template and discussed along with strengths and limitations in Section 3 and 4. The set of objectives elaborated in the sequel include: (i) UAV type, (ii) ANN functionalities, (iii) ANN architecture, (iv) ANN training methods, (v) computational complexity, (vi) control techniques, (vii) flight maneuvers, (viii) type of obtained results, and (ix) other problems to be addressed.

2.1 UAV Type

This refers to the general UAV type under control, e.g., rotary wing, fixed-wing, tilt wing, tail sitter, etc. Since any low-level flight controller design is tightly associated with the UAV dynamic model, the type of UAV makes a significant difference. Hence, it is identified for comparison purposes, separately from other components like hardware, software, sensors, etc.

2.2 ANN Functionalities

ANNs are composed of layers, of which applications fall into two mathematical problems: *regression* and *classification*. From the control perspective, however, ANN functionalities may be defined based on the role the ANN plays in the control system, e.g., UAV dynamic modeling, adaptive control augmentation, observer and estimator, fault detection component, etc.

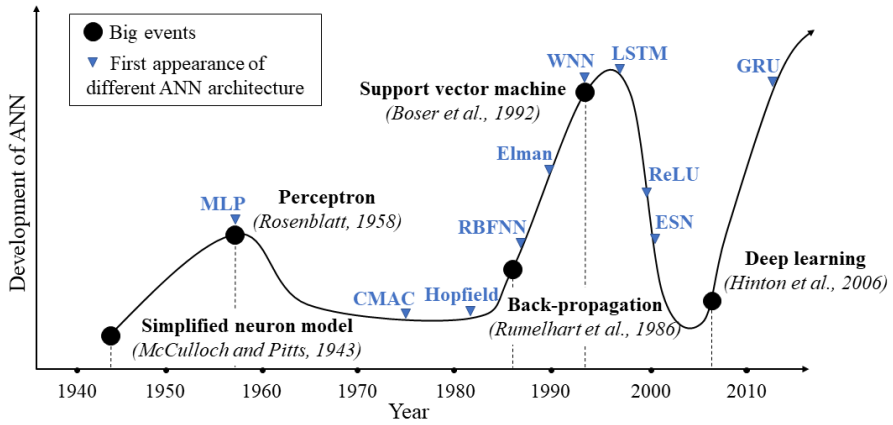

Fig. 1 Historical development of ANN

2.3 ANN Architecture

It is considered essential to provide a brief summary of ANN mathematical models. With this knowledge in mind, it will be easier for the reader to comprehend classifications of different ANN architecture as well as their advantages and limitations for flight controller design.

2.3.1 Historical perspective

The smallest element in ANNs is named as *neuron* by analogy with neurophysiology, whose simplified model was firstly studied by McCulloch and Pitts in the 1940s [29]. Few years later, the very first simple model of ANN, namely perceptron, was proposed by Rosenblatt [30], which was a supervised learning model receiving wide attention. However, subsequent research progress was somehow impeded at that time until Back-Propagation (BP) algorithm appeared in 1986 [31], a powerful tool for training feed-forward networks proposed by Rumelhart et al. Coupled with Multi-Layer Perceptron (MLP) network, a number of problems on classification and regression could be practically solved with satisfactory results. Nonetheless, the thriving development of Support Vector Machine (SVM) starting from 1992 witnessed AI winter for the next decade. In 2006, Hinton et al. proposed the concept of Deep Learning (DL) as opposed to shallow learning [32]. Given many more layers, Deep Neural Networks (DNNs) are endowed with much powerful potential, especially in pattern recognition and object detection [33]. Motivated by DL, an increasing number of ANNs with more complicated architecture have been popping up since then. Figure 1 depicts the brief history of ANN indicated with big events happened.

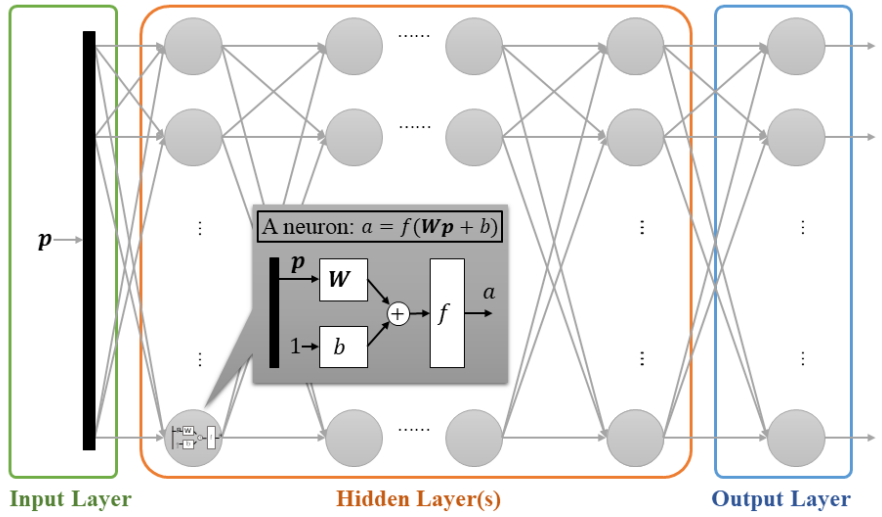

Fig. 2 A typical architecture of feed-forward neural networks

2.3.2 Mathematical model of neuron and ANN

An artificial neuron (or simply, a neuron) behaves like a function in principle. The simple model of a neuron generally consists of a scalar-valued activation function $f : \mathbb{R} \to \mathbb{R}$ and two training parameters, namely input weight matrix $\mathbf{W} \in \mathbb{R}^{1\times N}$ and bias weight $b \in \mathbb{R}^{1\times 1}$, where N is the number of elements in the input vector $\mathbf{p} \in \mathbb{R}^{N\times 1}$. The output $a \in \mathbb{R}^{1\times 1}$ is generated through $a = f(\mathbf{W}\mathbf{p} + b)$ where the engaged activation function $f(\cdot)$ could be hard-limit, linear, log-sigmoid or tan-sigmoid function [34], chosen by the designers considering the requirements of network performance.

An ANN is a typical network (e.g., see Fig. 2), which is made up of a single input layer, single or multiple hidden layers, and a single output layer. Each layer (except the input) comprises a number of neurons as introduced earlier, possibly governed by different activation functions. To draw consistent analyses in the sequel, we follow the convention throughout this survey that the total number of layers of an ANN is only associated with hidden and output layers, regardless of input layer. For example, an ANN as depicted in Fig. 2 with single hidden layer is considered to have two layers in total (i.e., one hidden, one output). In the rest of this survey, we denote the size of an ANN by $n_L \times n_N$, where $n_L, n_N \in \mathbb{Z}^+$ represent the number of layers and neurons, respectively. Therefore, given a particular two-layer ANN with 50 hidden neurons, we can simply specify its size by 2×50. Furthermore, we use the punctuation mark — colon (:), along with a number ahead to denote the total number of a specified ANN being used. With $2 : 2\times 50$, it means that this particular two-layer ANN with 50 hidden neurons is used twice in the design (in other words, two 2×50 networks).

2.3.3 Feed-forward and recurrent ANNs

In terms of neuron connections, ANNs can be categorized into one of the following two types: *feed-forward* and *recurrent*. Neurons in a feed-forward network only receives input from previous layer, while feedback connections exist in a recurrent network. Due to the additional connections, recurrent networks are endowed with large dynamical memory which feed-forward networks generally do not possess, however, at the cost of increasing complexity.

In terms of dynamic characteristics, recurrent networks may also be named as *dynamic* networks as opposed to *static* networks referring to feed-forward networks. Note that with the use of Tapped Delay Lines (TDLs), feed-forward networks can also be regarded as dynamic networks. For instance, Focused Time-Delay Neural Network (FTDNN)[1] (as part of a general class of dynamic networks, namely focused networks) refers to feed-forward networks with tapped delay lines only at the input.

Some typical and widespread ANNs are presented in Table 1, classified based on neuron connections. Only a brief introduction to these ANNs is given here, mainly recalling their origin and key features. Multi-Layer Perceptron (MLP) networks [30] and Radial Basis Function Neural Networks (RBFNNs) [35] are the two well-studied architecture of feed-forward networks originated in the late 1900s, mainly differing in the underlying activation functions. Subsequently, many variations have appeared with different functions taking on the role of activation function. For instance, Wavelet Neural Network (WNN) was proposed by Zhang et al. in 1992 to decrease the number of nodes required in the network by using Mexican hat wavelet as activation function [36]; Rectified-Linear Unit (ReLU) was firstly introduced to a dynamical network by Hahnloser et al. in 2000 to solve gradient vanishing issues [37]. On the other hand, motivated by cerebellum neurophysiological model, Cerebellar Model Articulation Controller (CMAC, a.k.a. cerebellar model arithmetic computer) was initially proposed by Albus for robotic manipulator control in 1975 [38]. As for recurrent networks, Hopfield networks serve as content-addressable memory systems with saturated linear transfer functions, popularized by Hopfield in 1982 [39]. Later in 1990, Elman networks were proposed which are normally two-layer recurrent networks having a feedback connection from the output of the hidden layer to its input[40]. Due to this additional connection, Elman networks have the capabilities of detecting and generating time-varying patterns. Echo State Networks (ESNs) are a special type of recurrent networks, initially introduced by Jaeger in 2001 [41]. Thanks to the concept of Reservoir Computing (RC), the training of such recurrent networks becomes conceptually simple and computationally inexpensive. Long Short-Term Memory (LSTM) networks are a typical type of recurrent networks, firstly proposed by Hochreiter and Schmidhuber in 1997 [42], capable of learning long-term dependencies by solving the problem of vanishing gradients [43]. Similar to LSTM units,

[1] More details please refer to MathWorks Documentation on "Design Time Series Time-Delay Neural Networks": https://www.mathworks.com/help/deeplearning/ug/design-time-series-time-delay-neural-networks.html.

Gated Recurrent Units (GRUs) are a gating mechanism particularly used in recurrent networks while involving fewer parameters, firstly introduced by Cho et al. in 2014 [44]. All these representatives are also marked in Fig. 1 for visualization purposes.

Type of connection	**ANNs**
Feed-forward	Multi-Layer Perceptron (MLP) network Radial Basis Function Neural Network (RBFNN) Rectified-Linear Unit (ReLU) network Wavelet Neural Network (WNN) Cerebellar Model Articulation Controller (CMAC)
Recurrent	Elman network Hopfield network Echo State Network (ESN) Long Short-Term Memory (LSTM) network Gated Recurrent Unit (GRU) network

Table 1 Classification of ANNs

2.4 ANN Training Methods

Back-Propagation (BP) algorithm is a gradient-based approach belonging to supervised learning. The procedures for updating network weights by using BP algorithm are as follows:

- Take a batch of training data.
- Perform forward-propagation to obtain the loss (i.e., the difference between actual/predicted and desired output).
- Perform back-propagation on the loss to derive gradients.
- Update network weights using the gradients.

However, BP algorithm works only for feed-forward networks and requires modifications to be applied for recurrent networks. For instance, Back-Propagation Through Time (BPTT) [45] and Real Time Recurrent Learning (RTRL) [46] are the two alternatives for the use of recurrent network training.

Apart from gradient-based approach, linear regression is also an efficient and effective way for some typical ANN architecture (e.g., ESN) due to its one-shot training process. Additionally, network weights can also be adjusted by using Lyapunov stability theory which provides mathematical proof of stability when network is employed in the control design.

Based upon the mode of network training, ANNs can also be categorized into *offline* and *online* networks. Offline networks refer to those with pre-trained and fixed network parameters. For applications where offline training time is not critical, such networks are prevalent as they offer higher accuracy models in general. On the contrary, online networks refer to those with online

adaptation of network parameters. Due to this unique learning ability, online networks are of great interest to applications where online adaptation is essential. For example, a dynamic model of UAV constructed by an online network would be more ideal. As such, all the dynamic changes can be automatically captured by adjusting the network weights in real time. However, since online networks involves online adaptation, the resulting training process can be computationally expensive and may suffer from divergence owing to a paucity of data.

2.5 Computational Complexity

Despite the parallel processing capability that ANNs uniquely possess, not all the implementations can rely on it if not supported by the suitable hardware such as FPGAs. This is not uncommon in practice since many ANN-based applications are still developed on CPU due to simplicity reasons (e.g., relatively simple development and deployment in computer systems), where the data cannot be processed in parallel unfortunately. More importantly, the real-time performance should be stressed from the perspective of controller design when ANNs are incorporating into MBC system design. Hence, no matter the hardware foundation, there is always a point to carry out analyses on computational complexity of ANNs being used beforehand.

Computational complexity refers to the computational resources such as computational time and amount of memory required to solve a computational problem. In the context of this paper, the main focus has been placed on the computational time for ANNs, also known as Time Complexity (TC). Motivated by [47], one of the very few works that have treated computational issues meticulously, an analysis on TC of ANNs is detailedly addressed in the sequel, offering a clear mind on the required computational resources during system design.

The TC of an ANN can be attributed to three phases: (i) offline training, (ii) online training, and (iii) feed-forward propagation. For offline ANNs, TC can be simply associated with feed-forward propagation, given that the time for offline training is not critical for the applications. For online ANNs, the time for online training has to be considered along with that for feed-forward propagation. Similarly, the time for offline training can be negligible depending on applications. Since there is a wide variety of ANN architecture and online training techniques, the analysis of TC for online training phase can be very complicated. Hence, we will leave this topic as one of our future research topics. In the following, we show that TC with regard to feed-forward propagation can be easily expressed as a function of input nodes, hidden neurons and output nodes. We denote the number of input nodes, output nodes and hidden neurons by $N, K,$ and $L \in \mathbb{Z}^+$, respectively, in the formulation.

The time required for running algorithms, generally expressed as the number of elementary operations included, can be varying with respect to the computer being used. However, since the elementary operations are normally

assumed to take constant execution time on a specific computer, executing on different computers can be hence regarded as changing the execution time by a constant factor determined by the computer configurations. Motivated by [47], Table 2 presents TC for different operations that might be used in the subsequent ANNs. It is noted that the choice of multiplication algorithm directly affects the complexity of division, exponentiation, exponential function and square root. In our case, we select 3-way Toom-Cook algorithm for multiplication in spite of the fact that there are other variations of algorithm available such as Schoolbook long multiplication and Karatsuba algorithm [48]. The scalar constants $K_i \in \mathbb{R}^+$ with $i = A, E, M, \phi$ represent the ratios between the computational time cost for different operations calculated with algorithms detailed in Table 2 and the unit time cost (which is exactly defined as the time cost for addition/subtraction operation), hence it always holds that $K_i \geqslant 1$ [47]. Since the constants K_i for multiplication, division and square root are identical which are all determined by multiplication algorithm, these constants are equivalently denoted by K_M with a slight abuse of notation. Additionally, subscript A denotes addition/subtraction operation, E denotes exponentiation, and ϕ denotes exponential function. Similarly, $N_i \in \mathbb{R}$ with $i = A, E, M, \phi$ denote the number of each operation involved in a network. To make the results more numerical, all the computations are assumed to be performed on a 32-bit microcontroller, although industrial embedded controllers vary in number of bits. Therefore, $n = k = 32$ in Table 2, where n denotes the input size in units of bits needed to represent the input and k is the number of digits of the exponent. Table 3 concludes the TC of several frequently-used (fully-connected) ANNs in terms of feed-forward propagation, of which derivations can be found in our previous study [28].

Operation	**Algorithm**	**Complexity**	**Constant** K_i $(n = k = 32)$
Addition and Subtraction	Basic	$O(n)$	$K_A = 1$
Multiplication	3-way Toom-Cook multiplication	$O(n^{1.465})$	$K_M = n^{0.465} = 5.01$
Division	Newton-Raphson division	$O(n^{1.465})$	$K_M = n^{0.465} = 5.01$
Exponentiation	Exponentiation by squaring	$O(k \cdot n^{1.465})$	$K_E = k \cdot n^{0.465} = 160.34$
Exponential function	Taylor series	$O(n^{0.5} \cdot n^{1.465})$	$K_\phi = n^{0.5} \cdot n^{0.465} = 28.34$
Square root	Newton's method	$O(n^{1.465})$	$K_M = n^{0.465} = 5.01$

Table 2 Time complexity of different operations

ANN architecture	**Time complexity** (TC $= N_A \cdot K_A + N_M \cdot K_M + N_\phi \cdot K_\phi$)
Feed-forward network	$\text{TC} = L(N+K+1) + L(N+K+1)K_M + LK_\phi$
RBFNN	$\text{TC} = L(2N+K-1) + L(N+K+3)K_M + LK_\phi$
ReLU network	$\text{TC} = L(N+K+1) + L(N+K)K_M$
ESN	$\text{TC} = ((N+K+L)L + (N+L-1)K) + ((N+K+L+3)L + (N+L)K)K_M + 2LK_\phi$

Table 3 Time complexity of feed-forward propagation of different ANNs

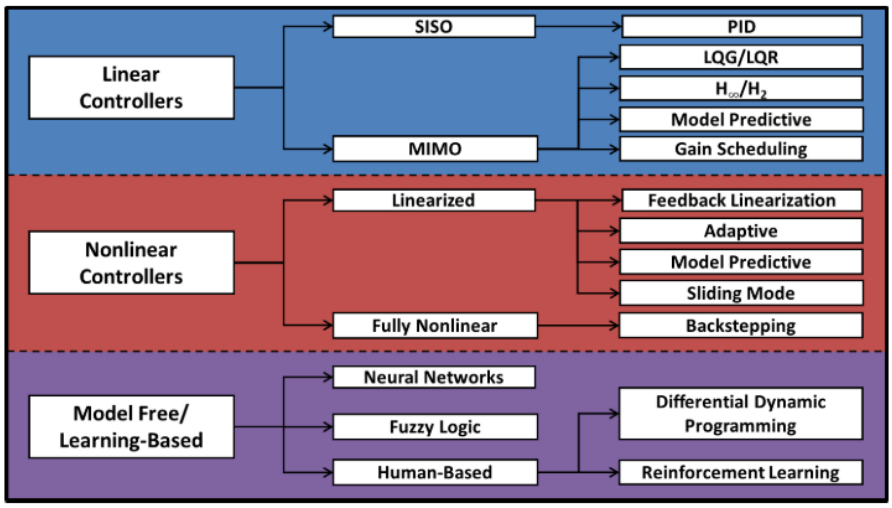

Fig. 3 Classification of control techniques (reprinted from [49])

2.6 Control Techniques

Control techniques can be generally classified into one of the following three categories: *linear*, *nonlinear*, and *model-free*, as illustrated in Fig. 3. Linear controllers can be further classified into Single-Input Single-Output (SISO) and Multi-Input Multi-Ouput (MIMO) methods. For instance, Proportional-Integral-Derivative (PID) control belongs to SISO methods while MIMO methods include Linear Quadratic Regulator (LQR), Linear Quadratic Gaussian (LQG), $\mathcal{H}_\infty$ control, etc. Nonlinear controllers can be further divided into linearized and fully nonlinear methods. Linearized methods start with a nonlinear model and proceed with linearization process to derive a linear model for subsequent control design, which include adaptive control, Model Predictive Control (MPC), Sliding Mode Control (SMC), etc. Fully nonlinear methods, for insatnc, backstepping control, can be directly applied on nonlinear dynamics without the need of linearization process. As this survey is focused on the combination of MBC techniques and ANNs, model-free approaches will not be discussed in detail. Table 4 summarizes the advantages and disadvantages of linear and nonlinear controllers presented in Fig. 3.

Control techniques	**Advantages**	**Disadvantages**
PID	Easy implementation.	(i) Lack of robustness; (ii) Coupling of dynamics is ignored.
LQR/LQG	Guaranteed stability margins for LQR.	(i) Limited to certain operating conditions; (ii) Required full state feedback for LQR; (iii) Performance degradation of Kalman filter for LQG; (iv) Output limitations are not considered.
$\mathcal{H}_\infty$	Parametric uncertainties and unmodeled dynamics can be handled.	(i) Conservative design for structural uncertainties; (ii) Good knowledge of system model is required.
GS	Large range of flight envelope and operating conditions can be covered.	(i) Unsteady transition issues between switches; (ii) Cumbersome design procedures.
Adaptive control	Bounded unmodeled dynamics, parametric uncertainties, and disturbances can be handled.	(i) Complex analysis and good knowledge of system dynamics is required; (ii) Parameter drift caused by process noise; (iii) Large adaptation rates are non-implementable.
Feedback linearization	Nonlinearities can be handled.	(i) High computational complexity; (ii) High requirement for model accuracy.
MPC	(i) Prediction capabilities; (ii) Abilities of explicitly handling constraints on control input.	(i) High requirement for accuracy of prediction model; (ii) Time-consuming for online optimization.
SMC	Control performance is insensitive to modeling errors and parameter uncertainties.	(i) Chattering is induced by frequent controller switches; (ii) Highly nonlinear sliding mode surface poses difficulties to stability analysis.
Backstepping control	Good for underactuated systems.	(i) Nonlinear model of a lower triangle form (known as pure-feedback form) is required; (ii) High computational complexity for computing the derivative of pseudo control inputs.

Table 4 Advantages and disadvantages of control techniques. The top half of the table belongs to linear controllers, while the bottom part of the table belongs to nonlinear controllers.

2.7 Flight Maneuvers

Flight maneuvers of UAVs can be categorized into *basic* and *aggressive* maneuvers. Basic maneuvers, including hover, level flight, turns, and climbs or descents, are associated with a single operating condition, thereby can be normally realized using a linear controller. On the other hand, more complex controllers such as gain scheduling or nonlinear methods are required for aggressive maneuvers that go through a wider flight envelope, for example, Figure-∞, split-S and other aerobatics.

2.8 Type of Obtained Results

Verification and validation results of the design of a flight control system can be categorized into *theoretical*, *simulation*, and *experimental*. Theoretical results provide rigorous mathematical proofs demonstrating the control design can fulfill the set of control objectives. Simulation results can be achieved coupled with the nonlinear aircraft model, for example, either in MATLAB/Simulink or in X-Plane Simulator. Experimental results are the most desired ones as they explicitly validate the effectiveness of the system. Nonetheless, they are also the most difficult to obtain due to technical and practical reasons.

2.9 Problems that Remain Unsolved

During the review, we will examine to what extent the problems listed in Section 1.2 are well addressed. The problems that remain unsolved will be indicated by the corresponding Roman Numerals in the final analysis table to reveal the limitations of each previous work as well as clarifying the future research direction.

3 Dynamic Modeling

As one of the most ubiquitous methods for years, mathematical modeling such as First Principle Model (FPM) is efficient and effective for plant dynamics that are well studied and easy to model. However, modeling of complex systems such as UAVs, normally engaged with identifying nonlinearities and uncertainties, remains a challenging topic in system identification. To handle these issues, there has been explosive growth in developing ANN-based models in recent years thanks to their abilities of approximating nonlinear functions and real-time learning. Such modeling technique is also known as *data-driven* or *black-box* approach. Instead of building the model from explicit physical laws, ANN-based models are trained based on data science techniques given sufficient informative data, which greatly eases the modeling process and reduces the computational complexity of mathematical formulation.

Apart from data-driven approach, there is another branch of dynamic modeling called *hybrid* approach. The core idea of hybrid modeling can be described as one of the two following: (i) Besides ANN-based model, a physical model is augmented (either in series or in parallel) to improve the overall performance; (ii) Explicit or implicit physical knowledge is incorporated into ANN-based model to enhance its interpretability. Consequently, hybrid modeling tends to yield more accurate model comprehending physical knowledge, thereby having improved generalization capability and interpretability.

In the sequel, we first separately review the existing literature on dynamic modeling based on data-driven and hybrid approach. Then, comparison and classification of approaches are conducted, followed with possible open problems.

3.1 Literature Review

3.1.1 Data-driven approach

Data-driven approach, which builds an ANN-based model entirely from data without the need of a-priori physical knowledge, can be classified into one of the three categories based on network architecture: feed-forward, recurrent, and *mixed*. In Section 2.3, feed-forward and recurrent networks have already been discussed in detail. Mixed networks refer to the combination of feed-forward and recurrent networks in one single model. Noticed that these networks may have other names such as "hybrid networks" or "complementary networks" used in different literature, however, we find it more appropriate to use the term "mixed modeling" to distinguish from the concept of hybrid modeling as defined earlier.

In [50] (1997), feed-forward network models are used for inverse modeling of a fixed-wing aircraft for feedback linearization. Representing the inverse longitudinal and lateral dynamics, both of the two networks comprise two layers using a combination of RBF and sigma-pi units [51] (polynomial representations). The training data, consisting of angular accelerations (second derivatives of Raw-Pitch-Yaw angles) and control inputs, are generated by simulation software at different flight conditions. Furthermore, angular accelerations are converted into moment coeffcients as input data so as to facilitate a inverse mapping that is less sensitive to altitude and Mach number variation. Since network weights appear linearly, standard linear Least Square (LS) approximation method is adopted for offline learning algorithm. Simulation results show that the offline trained networks provide satisfactory average performance over a given flight envelope. However, they might also exhibit large errors within a portion of envelope having a high degree of mapping complexity.

In [52] (2006), a feed-forward network is used for online estimation of plant dynamics of a damaged aircraft to avoid aggressive learning induced by significant discrepancy between the model and real flight dynamics. Motivated

by [50], single-hidden-layer sigma-pi network is adopted to estimate incremental plant matrices in dynamic inversion control. The network parameters are updated by an adaptive law derived from Lyapunov stability theory and a recursive LS formula using gradient method for comparative studies.

In [53,54] (2006), a mixed model of a mini-helicopter is proposed through offline training. Due to the flight dynamics of helicopter, the model can be divided into two subsystems (namely, radio-signal-to-attitude and attitude-to-position) in cascade form. For each subsystems, a mixed architecture is adopted, consisting of a series of contextual neurons [40] to memorize previous states and a two-layer feed-forward network (MLP or RBFNN). The training patterns include attitude, position and 4 control inputs, which all coming from real flight data. For radio-signal-to-attitude subsystem, 54 input neurons are selected for two different types of feed-forward networks (MLP and RBFNN) for comparative studies, each including 60 hidden neurons and 500 radial neurons, respectively. For attitude-to-position subsystem, 48 input neurons are used associated with 60 hidden neurons for MLP and 5 radial neurons for RBFNN. For both subsystems, number of outputs are 3 (attitude and position, respectively). Simulation results show that mixed model with RBFNN has better performance in terms of local approximation (especially for take-off and landing phase affected by the ground effect) while mixed model with MLP demonstrates better capability of global approximation as lower error is achieved during the flight. Moreover, it is founded that the offline training time required for RBFNN ($\sim$ a few minutes) is much less than that for MLP ($\sim$ 18hrs) to obtain the final results of same level.

In [55] (2007), a comparison is carried out between an online ANN-based model and an offline ANN-based model for modeling of a fixed-wing UAV. The offline model is trained as a single block with control inputs as 4 inputs and linear velocities along with angular rates as 6 outputs. The online model, on the other hand, consists of two individually trained ANN subsystems connected in parallel which represent longitudinal and lateral dynamics. To take into account the coupling effects between two dynamics, roll rate and yaw rate (for longitudinal ANN-based model) and pitch rate (for lateral ANN-based model) are augmented as additional inputs apart from control inputs in online model. Both offline and online ANN are feed-forward networks with size of 2×8 and 2×4 for each subsystem, respectively, and with a maximum of three past inputs used for the prediction of the present output. All the training is performed on real flight data using Levenberg-Marquardt (LM) algorithm. HIL simulation shows that offline model demonstrates smoother approximation of flight dynamics while online model exhibits better adaptability.

In [56] (2012), an offline feed-forward network is trained for learning heave (or vertical) dynamics of an unmanned helicopter. The network has a single hidden layer consisting of 4 hidden neurons. Vertical flight data are collected for network training using LM algorithm. Simulation shows consistent results on prediction of vertical acceleration. Therefore, the resulting network model is subsequently used for heave control design.

In [57] (2014), the inverse model of a fixed-wing UAV is trained by a two-layer feed-forward network, specifically MLP, for the use in feedback linearization. The model is first trained offline with training data generated from the validated nonlinear aircraft model using sinusoids as inputs. Then, the network weights are adjusted online using the real-time data from flight simulator in SIL simulation. Different from offline model, momentum term is added for online model. Despite the satisfactory results achieved, the network architecture is not presented in details and more investigations would be necessary on online model to be effectively used along with real flight data.

In [58] (2015), an offline ESN model is trained for modeling of a quadrotor, a system with 4 control inputs and 6 outputs (position and attitude), using real flight data. Thanks to RC techniques, the training of ESN is conceptually simpler and less time-consuming than that of other recurrent networks. Initially, 100 dynamic reservoirs and spectral radius of 0.1 are chosen empirically. Then, an evolutionary algorithm for parameter optimization, namely Covariance Matrix Adaptation Evolution Strategy (CMA-ES), is applied, yielding the finalized network with 127 dynamic reservoirs and spectral radius of 0.9191. The optimized ESN shows imperceptible results in simulation compared with real flight data.

In [59] (2015), an offline recurrent architecture is employed for MIMO modeling of quadrotor in the presence of noise and ground effect, namely Modular Deep Recurrent Neural Network (MODERNN) [60], which is proposed to address the problem of vanishing/exploding gradient in deep networks [61]. Thanks to the presence of feed-forward connections throughout MODERNN, the proposed recurrent network can be trained using ordinary gradient descent approach such as LM algorithm instead of BPTT or RTRL as frequently used for recurrent networks. The training data (including 4 control inputs as inputs, positions and heading angle as outputs) are sampled from simulated trajectory considering the sum of 10 sinusoids with uniformly random frequencies ranging from 0 to 10Hz. To reduce the computational load, three separate MODERNNs are trained with size of 3×10, 4×20, and 3×8. Consistent modeling results are achieved after approximately 24-hour training for each MODERNN on an i7 core machine with 16GB of RAM.

In [62] (2015), a novel offline ReLU network model is proposed along with parameter initialization and optimization methods to solve challenging issues of helicopter modeling by extracting information about the hidden (difficult-to-measure) state such as airflow. Motivated by Takens Theorem [63,64], the proposed model consists of a quadratic lag model (with control lag) and a two-layer ReLU network (with control lag and time-lagged output including linear velocities and angular rates). The use of quadratic lag model is to account for nonlinear dependence on current state and control, while ReLU network is employed to learn the correction to quadratic lag model to better approximate the helicopter dynamics locally. Noted that the ReLU network with delayed units used in the paper has a similar architecture as nonlinear autoregressive

network with exogenous inputs (NARX)[2] in which ReLU rather than sigmoid activation function is used in the hidden layer. Since the proposed model is non-convex, LS regression and SGD (with momentum term and variable learning rate) are applied separately for training, which can be also regarded as one iteration of Block-Coordinate Descent. Real-world data across 19 maneuvers are used for both training and testing. However, no label in terms of maneuvers is given during training, resulting in a global model which is applicable across all maneuvers. With 2500 hidden neurons for ReLU network, it takes nearly 1 hour for 200,000 iterations of training on a 6-core Intel i7 server with 32GB RAM. Simulation results show the proposed ReLU model improves 58% overall in Root Mean Square (RMS) acceleration prediction over state-of-the-art methods.

In [65] (2016), two ReLU networks are employed for learning quadrotor dynamics (translational and rotational, respectively) through offline training. The paper mainly investigates whether a simple feed-forward network model can be used for controller synthesis for unspecified trajectories. To this end, two-layer ReLU networks only fed by current state and control input with 100 hidden neurons are trained using resilient back-propagation with a momentum term and fixed learning rate. Training data consist of linear velocities, angular rates, Euler angles and control inputs, all from real flight data of only translational and only rotational motion (e.g., sinusoids in XY plane without yaw motion). Experimental results show good control performance of tracking sinusoid-yaw trajectory (i.e., the quadrotor undergoes a sinusoidal motion in XY plane while yawing) when coupled with LQR and PD controller. It can be hence concluded that ANN-based model is capable of generalizing the dynamics to learn nonlinear couplings between translational and rotational motions, even though these are not explicitly captured by the training data.

In [66] (2016), an offline ESN with output feedback is used for modeling the inverse dynamics of a fixed-wing UAV for feedback linearization. As an extension work of [57], the paper investigates the feasibility of using a different ANN architecture following a similar design procedure. The ESN is trained using Weiner-Hopf method for linear regression with sinusoidal simulated data from flight simulator as training data. However, detailed specifications and training procedure for ESN with output feedback are not given. As output feedback has been used, stability issues are not addressed as well [67].

In [68] (2017), quadrotor dynamics are decoupled into 6 subsystems (i.e., roll/pitch/yaw and X/Y/Z position), each of which is modeled by a single feed-forward network. Different from conventional two-layer ANN, there are only one input layer which receives 5 inputs (state and control input with TDLs) and one output layer which generates 1 output (state) for each feed-forward network. Such architecture can be simply regarded as a two-layer feed-forward network having linear output layer with identity weight matrix and no bias. All

[2] More details refer to MathWorks Documentation on "Design Time Series NARX Feedback Neural Networks": https://www.mathworks.com/help/deeplearning/ug/design-time-series-narx-feedback-neural-networks.html.

the feed-forward networks are trained online using BP algorithm. Simulation results demonstrate satisfactory learning ability.

3.1.2 Hybrid approach

Hybrid approach refers to modeling methods that either use ANNs in conjunction with a mathematical model (e.g., FPM) or incorporate physical knowledge directly into the training process of ANNs. As we will see in the following review, all the existing work in flight control is focused on the former idea.

In [47] (2018), a hybridization of analytical and empirical techniques is proposed for linear velocity estimation of quadrotor to improve the performance in terms of model precision and computational time. The hybrid model is made up of a parametric model (e.g., ARX, ARMAX or OE) and a data-driven model (e.g., MLP, RBFNN, or Adaptive Neural Fuzzy Inference System (ANFIS)) in series. Using real flight data, the best parametric model is first determined. Then, soft computing technique is applied for training the offline data-driven model using simulated data as input and real flight data as output. Specifically, LM algorithm is adopted for RBFNN training. Unlike the conventional training process where the loss function is only an MSE of output error, a novel loss function is defined as the average of the sum of MSE of three error signals, that are system output, step response, and the gain of the output sinusoidal signal of the system given a sinusoidal signal of unitary amplitude and frequency as input. Moreover, computational complexity is analyzed for both parametric and data-driven model to estimate the complexity of different identification technique. Simulation results show improved performance of hybridization in terms of model accuracy despite a low rising of computational cost. In the end, online MLP is also discussed for online modeling taking consecutive samples as input. However, it is not investigated how the online ANN will influence the model accuracy and computational complexity in hybridization.

In [69] (2018), Multi-Step (MS) prediction problem of quadrotor dynamics is studied, as opposed to a majority of literature focusing on Single-Step (SS) prediction. An offline hybrid model is proposed, consisting of three modules, namely input modeler, motion model and output modeler, in which motion model is an FPM while the rest two adopt LSTM networks. The hybrid model receives 4 control inputs and predicts linear velocities as well as angular rates. Real flight data of various flight regimes are collected and used for training by TensorFlow. Comparison results are drawn through simulation, indicating the hybrid model with hybrid-parallel configuration outperforms either FPM or black-box approach. However, the paper does not provide detailed information on LSTM networks (e.g., their size and training details).

3.2 Comparison of Approaches

To fairly determine the capabilities and limitations of existing work on UAV dynamic modeling, reference template, as elaborated in Section 2, is applied which yields a detailed review as summarized in Table 6. For better visualization and understanding purposes, acronyms and abbreviations are used and explained with their full name in Table 5. Specifically, the ANN architecture is briefly specified in the form of "AA-BB-CC-DD", where "AA=OFF/ON", "BB=INV/(feed-forward modeling by default)", "CC=MS/(Single-step prediction by default)", and "DD=SINGLE/DE" (which will be explained in the sequel). Moreover, the input-output specifications of trained model are also specified in "ANN_TRAIN" column along with the type of training data using right arrow and tuple. For example, "RDATA: $\delta \rightarrow$ (vel, pqr)" denotes that the model is trained by real flight data with control inputs as input and linear velocities and angular rates as output.

OFF	Offline modeling	ON	Online modeling
INV	Inverse modeling	MS	Multi-step prediction
SINGLE	Single model architecture	DE	Decoupled model architecture
FF	Feed-forward network	TDL	Tapped delay line
LS	Least square	SGD	Stochastic gradient descent
BP	Back-propagation algorithm	SDATA	Simulated data for training
RDATA	Real flight data for training		
SIM	Simulation results	THE	Theoretical results
HIL	Hardware-in-the-loop		
δ	Control inputs	pos	Position (X, Y, Z)
att	Attitude (ϕ, θ, ψ)	vel	Linear velocities (u, v, w)
rpy	Angular rates (p, q, r)		
N/A or ?	Not applicable	w/	with

Table 5 Acronyms and abbreviations used in Table 6

REF	UAV	ANN_FUNC	ANN_ARCH	ANN_TRAIN	TC	RESULT	ISSUE
[50] (1997)	Fixed-wing	OFF-INV-DE	$2:2\times$? RBF+sigma-pi	LS, SDATA: $(\ddot{\phi},\ddot{\theta},\ddot{\psi})\rightarrow\delta$	N/A	SIM	ii
[52] (2006)	Fixed-wing	Online plant estimation	$2\times$? sigma-pi	Lyapunov/recursive LS	N/A	THE, SIM	ii
[53, 54] (2006)	Helicopter	OFF-DE	Contextual neurons w/ $2:2\times 60$ MLPs or $2\times 500, 2\times 5$ RBFNNs	RDATA: $\delta\rightarrow$ (att,pos)	A few minutes for RBFNNs and $\sim$ 18hrs for MLPs	SIM	i, ii, iv
[55] (2007)	Fixed-wing	OFF-SINGLE; ON-DE	2×8 FF w/ TDLs for offline; $2:2\times 4$ FF w/ TDLs for online	LM algorithm, RDATA: $\delta\rightarrow$ (vel,pqr) for offline, RDATA: $\delta+(p,r/q)\rightarrow$ (vel,pqr) for online	N/A	HIL	i, ii
[56] (2012)	Helicopter	OFF-SINGLE	2×4 FF	LM algorithm, RDATA: $\delta\rightarrow w$	N/A	SIM	i, ii, iv
[57] (2014)	Fixed-wing	OFF-INV-SINGLE; ON-INV-SINGLE	$2\times$? MLP	Gradient descent, SDATA	N/A	SIM	i, ii, iv
[58] (2015)	Quadrotor	OFF-SINGLE	2×127 ESN	CMA-ES, RDATA: $\delta\rightarrow$ (pos,att)	N/A	SIM	i, ii, iv
[60] (2014), [59] (2015)	Quadrotor	OFF-DE	$3\times 10, 4\times 20, 3\times 8$ MODERNNs	LM algorithm, SDATA: $\delta\rightarrow$ (pos, ψ)	24hrs (i7, 16GB RAM)	SIM	iv
[62] (2015)	Helicopter	OFF-SINGLE	2×2500 ReLU w/ TDLs	LS and SGD, RDATA: $\delta\rightarrow$ (vel,rpy)	1hr (i7, 32GB RAM)	SIM	i, ii, iv
[65] (2016)	Quadrotor	OFF-DE	$2:2\times 100$ ReLUs	Resilient BP, RDATA: $(\delta$, pos,vel,att,rpy) $\rightarrow$ (vel,rpy)	N/A	SIM	i, ii, iv
[66] (2016)	Fixed-wing	OFF-INV-SINGLE	ESN w/ output feedback	Weiner-Hopf method for linear regression, SDATA	N/A	SIM	i, ii, iv
[68] (2017)	Quadrotor	ON-DE	$6:1\times 1$ FFs w/ TDLs	BP algorithm, $\delta\rightarrow$ (pos,att)	N/A	SIM	i, ii, iv
[47] (2018)	Quadrotor	OFF-SINGLE	RBFNN	LM algorithm, RDATA: $\delta\rightarrow$ vel	Analytical results for feed-forward propagation	SIM	i, ii
[69] (2018)	Quadrotor	OFF-MS-DE	LSTMs w/ TDLs	TensorFlow, RDATA: $\delta\rightarrow$ vel,pqr	N/A	SIM	i, ii

Table 6 This table summarizes the review of ANN-based dynamic modeling for UAV applications as addressed in Section 3.1. The columns represent the referenced work and the year published (REF), the type of UAV under control (UAV), the functionalities of ANN (ANN_FUNC), the architecture of ANN (ANN_ARCH), the training methods for ANN (ANN_TRAIN), time complexity (TC), achieved results (RESULT), and problems that remain unsolved (ISSUE), as early defined in the reference template in Section 2.

3.3 Classification of Approaches

As the first step of dynamic modeling, one should clarify the requirements for the model based on the specific application. For example, is the model going to be used in feedback linearization? If the answer is yes, then an inverse model is required and it will definitely not work if one ends up with a feed-forward model. Apart from that, there are two more questions that need to be identified, i.e., what kind of prediction is expected from the model, and are subsystems preferable? The answer to the former question will determine whether a single-step or a multi-step prediction model is desired, whereas the answer to the latter question will settle the model architecture. For cases where longitudinal and lateral dynamics can be approximately separated, one might prefer to have decoupled model architecture (either in parallel or in series); otherwise, a single block of model might be preferred. The design flow of selecting an appropriate model architecture based on specific requirements is depicted in Fig. 4.

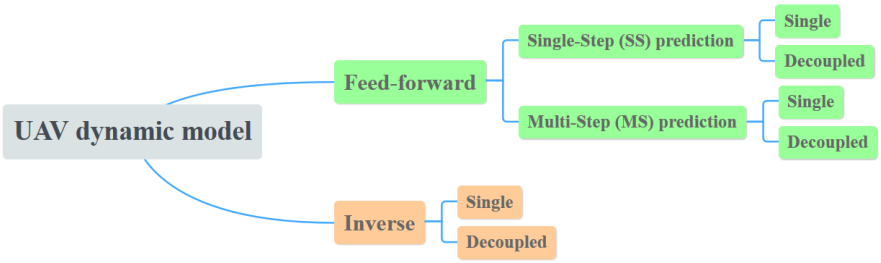

Fig. 4 Design flow of selecting an appropriate model architecture for UAV dynamic model

As soon as a suitable model architecture is determined to meet the requirements for specific task, it is then the turn to choose ANN architecture and corresponding training methods for building the ANN-based model. Based on the definitions of ANN categorization in Section 2.3 and 2.4, the existing work on UAV dynamic modeling can be classified as shown in Table 7 with distribution given by pie chart for visualization purposes as illustrated in Fig. 5. It is straightforward to see that more than a half of literature is focused on feed-forward networks due to their simple architecture and working principle. However, almost all the feed-forward networks are used along with Tapped Delay Lines (TDLs), resulting in dynamic networks. This is simply because static networks are incapable of capturing complex flight dynamics of UAV from time-series flight data. Moreover, only about one-fifth of literature works on online networks and all of them fall under feed-forward category. This, again, shows the preference for feed-forward networks over the rest thanks to their easy training process and implementation. However, the small amount of work on online networks also indicates that most of the existing modeling approaches do not possess learning ability to deal with dynamic changes during

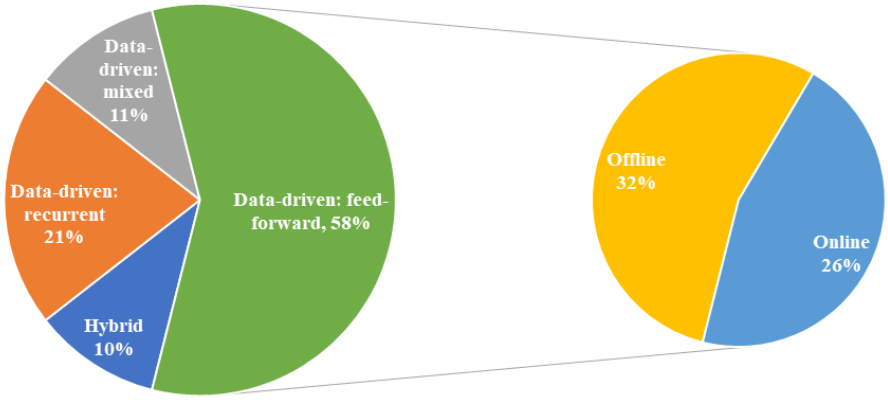

Fig. 5 Pie chart distribution of literature on UAV dynamic modeling. **Left:** Distribution of literature categorized by ANN architecture. **Right:** Distribution of literature using feed-forward networks categorized by training methods.

the flight. Lastly, it can be concluded that hybrid approach has been actually very little studied so far, thereby remaining an open research topic for UAV dynamic modeling.

	Data-driven			**Hybrid**
	Feed-forward	**Recurrent**	**Mixed**	
Offline	[50, 55–57, 62, 65]	[58–60, 66]	[53, 54]	[47, 69]
Online	[47, 52, 55, 57, 68]	N/A	N/A	N/A

Table 7 Literature classification of UAV dynamic modeling. **Column:** ANN architecture. **Row:** Training methods.

3.4 Challenges and Opportunities

We recall the five questions thrown in Section 1.2 about the integration between MBC techniques and ANNs, which are stability, robustness, paucity of data, interpretability, and real-time performance. Based on the classification and comparison as carried out in the previous sections, we investigate to what extent these problems have been resolved as well as identifying what are the current challenges and opportunities in the sequel.

i. (*Stability*): In the context of dynamic modeling, stability issue refers to the convergence of network weights within specified time interval. Despite being widely recognized as one of most powerful tools for training feed-forward networks, BP algorithm (especially Batch Gradient Descent

(BGD)) suffers from slow convergence and sometimes yields suboptimal solutions due to the utilization of steepest descent method [70]. To achieve faster convergence and better performance, there are currently a dozen of superior options available for network training, for example, Stochastic Gradient Descent (SGD), Mini-Batch Gradient Descent (MBGD), conjugate gradient algorithms such as Fletcher-Reeves and Polak-Ribiére update, BFGS algorithm, Levenberg-Marquardt (LM) algorithm, etc. Besides, one can also use some simple tricks including adaptive learning rate and momentum term to obtain the desired performance. Fortunately, all the aforementioned approaches have already been supported and well documented by quite a few open source platforms such as TensorFlow, PyTorch, and MATLAB, which greatly ease the design and implementation process for developers. However, it should be noted that there is still no formula to guarantee the convergence of the network and all the approaches we have mentioned only increase the likelihood of finding a better model in a faster speed. The only way to guarantee the convergence, perhaps, is to use adaptive control theory as shown in [52] rather than using purely data-driven training methods (which are the mainstream in the literature). Under such circumstance, the network weights are updated through Lyapunov stability theory and all the signals can be proved bounded analytically.

ii. (*Robustness*): The majority of existing work aims to model flight dynamics under nominal conditions. That being said, no uncertainties and disturbances are taken into account explicitly. Despite the inclusion of noise data in network training as demonstrated in some literature, the objective is focused on achieving better generalization capability as well as avoiding overfitting. From control perspectives, there is no doubt that a robust dynamic model is desired in most occasions which should remain valid even though there are some small variations in parameters or modeling mismatch. In addition, substantial discrepancy between model and real flight dynamics should be avoided because this might result in large control input that would be practically unrealizable. Hence, online networks are of great value on developing a real-time adaptive model thanks to their learning abilities. To this end, however, stability issue and paucity of data problem should be all taken into considerations.

iii. (*Paucity of data*): Preparing the dataset for training, testing and validation is a delicate task for training an accurate ANN-based model. Foremost, one has to identify the appropriate input and output variables. For example, what kind of aircraft states should be engaged in training, position (x, y, z) or linear velocity (u, v, w), Roll-Pitch-Yaw angle (ϕ, θ, ψ) or angular rate (p, q, r)? According to our review, it seems that the use of linear and angular velocity will result in model of better performance. Next, since we aim to train a model that fully describes the nonlinear flight dynamics over the entire flight regimes, collecting sufficient and suitable data is a laborious work. Table 8 shows different type of training data that are used in the literature. Among them, real flight data for all possible maneuvers required for mission are mostly desired, but an experienced UAV pilot is a must.

Using simulated data can remove this requirement and reduce the cost as a simulator software will be employed rather than a real UAV. However, the quality of data is tightly linked to the fidelity of the simulator. Some literature also uses step signals as inputs to generate training data. This can be even simpler to extract the maximum possible information from the dynamic system at the expense of less representative data if not carefully dealt with. For offline networks, it is usually not a big deal for dataset preparation given enough time and experts following what we have just mentioned. However, things are completely different for online networks where real flight data are the main source for network training. Due to the limited data available at each sampling period, the trained model would most probably not accurate enough for subsequent control synthesis. This issue is referred to as "paucity of data", which occurs mainly in online training. To counter this issue, one might find TDLs useful (e.g., see applications [55,68]) because TDLs form a larger input vector consisting of the current time-step data instance and the past time-step data instances.

Priority	**Type of data**	**Data information**
1	Real flight data	All possible maneuvers required for mission
2	Simulated data	All possible maneuvers required for mission
3	Simulated data	Steps with different amplitude as input signals

Table 8 Different type of training data and their priorities

iv. (*Interpretability*): It is apparent that modeling with data-driven approach suffers from interpretability issue, i.e., the internal working principle of the black box is hard to come by. As the only two papers ([47,69]) researching on modeling with hybrid approach, the objectives lie in improving the performance, particularly in terms of model precision and computational time, rather than the interpretability. Hence, how to open the black box of ANN remains an ongoing challenge and it is worthwhile devoting much effort to this.

v. (*Real-time performance*): Generally speaking, offline networks do not have issues with real-time application. This is because the time complexity of feed-forward propagation is generally negligible compared to other process. Nonetheless, considerable attention should be paid to online networks. Specifically, the real-time performance of online networks is associated with the speed of convergence of online adaptation.

From the above analysis, we may conclude that *an online hybrid model* is of the most preferable to dynamic modeling of real-time complex systems such as UAVs. According to Table 7, there is no research on this direction so far. It is remarked that the hybrid approach should put emphasis not only on improving the model accuracy but also on enhancing the intertepretability by leveraging the complementary strengths of physics-based model and ANN. Recent

work has proposed two frameworks, namely Physics-Guided Neural Network (PGNN) and Physics-Guided Recurrent Neural Network (PGRNN), for offline modeling of lake temperature, which seek models to have physical consistency and good generalization capability apart from superior accuracy by incorporating explicit physical knowledge into network training [71, 72]. The theory behind these two frameworks is called Theory-Guided Data Science(TGDS) [73], which deserves further investigations for UAV applications. On the other hand, specific considerations on both dataset preparation (pre-processing) and training algorithms are also required from implementation point of view.

4 Control Techniques

In this section, we will review different flight control techniques combined with ANNs for flight control design. Due to the intrinsic learning ability of ANN, such combination can be substantially regarded as adaptive control. Hence, some literature coins the terms such as "feedback-linearization-based adaptive control" to represent the control architecture that combines feedback linearization technique with ANN-based adaptive augmentation. The review in the sequel is classified based on the main control techniques adopted for flight control so as to avoid those lengthy terminologies for better categorization purposes.

4.1 Literature Review

4.1.1 Proportional-integral-derivative control

In [74] (2009), an online CMAC is proposed for baseline PID controller under turbulence in aircraft automatic landing system. Lyapunov stability theory is applied to determine the learning rate and weight updating rule of CMAC. Simulation shows the adaptive CMAC controller is capable of tracking a desired path under severe turbulence environment. Nonetheless, network structure and parameters are not given in detail.

4.1.2 Feedback linearization

In [50] (1997), three feed-forward networks are used as adaptive control augmentation for each channel of control input of a fixed-wing aircraft. Each feed-forward network consists of 21 RBF units (one-dimensional Gaussian functions) and 40 sigma-pi units (polynomial representations), resulting in a total number of 840 weights. To ensure the uniform boundedness of all the signals in the loop as well as the convergence of online network weights, a stable weights adjustment rule is also derived via Lyapunov analysis. Through simulation results, it is shown that online adaptive networks are capable of canceling out model inversion error without full knowledge of uncertainties

under a high-g, fixed throttle turn manueuver. A related work on control of rotational dynamics of helicopter refers to [75].

In [76] (2002), a two-layer feed-forward network with 5 hidden neurons is used as adaptive control augmentation for an autonomous helicopter to account for inversion error. A stable adaptive law is derived from Lyapunov stability analysis to update network parameters online, which guarantees signal boundedness. Although simulation and flight tests show satisfactory tracking performance for various maneuvers, the potential of ANN-based adaptive element has not been investigated under uncertainties and disturbances.

In [77] (2011), a two-layer feed-forward network with 8 hidden neurons is used as adaptive control augmentation for a helicopter to reduce model inversion error, similarly to [76]. Concurrent-learning adaptive law is developed to improve the weight convergence properties and tracking performance by alleviating the rank-1 condition of traditional BP-based training law that use merely the current data. With such modification, the controller is able to adapt using current and stored data without affecting the responsiveness of the adaptive law to current data. Simulation and flight test results show improved tracking performance for both repeated forward-step and aggressive trajectory tracking manuevers.

In [78] (2015), a two-layer feed-forward network with 50 hidden neurons is used as adaptive control augmentation for a fixed-wing aircraft to cope with inversion error. Lyapunov stability analysis is applied for updating network weights. Simulation results show remarkable performance under mass changes and loss of an aileron.

4.1.3 Optimal feedback control

In [56] (2012), a two-layer feed-forward network with 4 hidden neurons is trained offline using LM algorithm to mimic the control inputs generated from an optimal feedback control law for the heave control of a helicopter. Both simulation and flight test results show satisfactory tracking performance. However, the offline ANN-based controller does not possess learning ability. Hence, repeated ANN training cannot be avoided if operating condition has been changed significantly (e.g., see the initial flight test results in gusty conditions therein).

4.1.4 Adaptive control

In [79] (2016), a two-layer static structure RBFNN[3] with 125 centers is used to compensate the unknown aerodynamic forces for an autonomous helicopter in real time. It is then exploited by a new structural identifier to identify the unknown inertial matrix facing unknown aerodynamic forces and external disturbances. To reduce online computational burden, an optimized algorithm

[3] Static structure RBFNN refers to a specific type of RBFNN of which only weights are updated while centers and widths of Gaussian function (if selected as the receptive field function) are fixed.

is incorporated in RBFNN mechanism. A barrier Lyapunov function is used to avoid kinematic singularity problem in updating network weights. Simulation shows the designed controller guarantees a good tracking performance for both fixed-point and dynamic trajectory tracking.

In [80] (2018), a deep RNN combined with CNN network is developed for regression of actuator failure model from state trajectories for fault tolerant design of a fixed-wing aircraft. The network has 2 CNNs, 2 LSTMs, and a dense regression layer (For more details on network parameters, please refer to the original paper). Offline training is conducted on a database of different failure scenarios using TensorFlow library and a NVIDIA GTX 1060 GPU. The controller adopts a control methodology that combines Structure Model Reference Adaptive Control (SMRAC) [81] and Nonlinear Dynamic Inversion (NDI), namely Structured Adaptive Model Inversion (SAMI). Thanks to the use of deep neural network, faster convergence of controller coefficent is achieved. Simulation provides satisfactory recovery results under severe engine and elevator failure scenario.

4.1.5 Sliding mode control

In [82] (2015), a two-layer static structure RBFNN with 6 hidden nodes is adopted as adaptive uncertainty observer for a coaxial 8-rotor UAV. Along with Backstepping Sliding Mode Controller (BSMC), the lumped uncertainties, which include external disturbance and inertia matrix uncertainty, can be effectively estimated and handled without the need of knowing their bounds. Lyapunov stability theory is applied to update network weights of the observer and prove the signal boundedness. Simulation results show robust performance under model uncertainties and external disturbances.

In [83] (2016), RBFNN is used along with a double-loop Integral Sliding Mode Control (IntSMC) for position and attitude tracking of a quadrotor subject to disturbances and parametric uncertainties. Lyapunov theory is applied to update the network weights of RBFNN as well as providing stability proof. Faster convergence of state variables to their desired values is noted through simulation. However, network structure and parameters of RBFNN are not given in detail.

4.1.6 Backstepping control

In [84] (2008), two-layer MLPs with 15 hidden neurons are used to estimate unknown linearities as well as physical parameters for flight control of a helicopter. Network weights are updated online using Lyapunov theory. From simulation, it is demonstrated that the proposed backstepping controller achieves good position tracking performance even with a sudden mass change perturbation.

4.2 Comparison of Approaches

Similar to Section 3.2, reference template is applied to summarize the comprehensive review of previous studies, yielding Table 10, to fairly determine the advantages and disadvantages of existing work on UAV flight control combined with ANNs. For better visualization and understanding purposes, acronyms and abbreviations are used and explained with their full name in Table 9.

OFF	Offline network	ON	Online network
ADAUG	Adaptive control augmentation	OBSV	Uncertainty observer
CTRLID	Controller coefficient identification	CTRLMM	Controller mimicking
FL	Feedback linearization	OP	Optimal feedback
AC	Adaptive control	SMC	Sliding mode control
BS	Backstepping control	PID	Proportional-integral -derivative

Table 9 Acronyms and abbreviations used in Table 10 and Table 11

REF	UAV	ANN_FUNC	ANN_ARCH	ANN_TRAIN	TC	CTRL	MANUEUVER	RESULT	ISSUE
[50] (1997)	Fixed-wing	ON-ADAUG	3 : 2 × (21 RBF + 40 sigma-pi)	Lyapunov	N/A	FL	High-g	THE, SIM	iii, v
[76] (2002)	Helicopter	ON-ADAUG	2 × 5 FF	Lyapunov	N/A	FL	Various	THE, SIM, EXP	
[74] (2009)	Fixed-wing	ON-ADAUG	CMAC	Lyapunov	N/A	PID	Landing	THE, SIM	iii, v
[77] (2011)	Helicopter	ON-ADAUG	2 × 8 FF	Lyapunov	N/A	FL	Tracking	THE, SIM, EXP	ii
[56] (2012)	Helicopter	OFF-CTRLMM	2 × 4 FF	LM algorithm	N/A	OP	Tracking	SIM, EXP	i, ii, iii, iv
[78] (2015)	Fixed-wing	ON-ADAUG	2 × 50 FF	Lyapunov	N/A	FL	Roll	THE, SIM	iii, v
[82] (2015)	Coaxial 8-rotor	ON-OBSV	2 × 6 RBFNN with Gaussian function (static structure)	Lyapunov	N/A	SMC	Attitude tracking	THE, SIM	iii, v
[79] (2016)	Helicopter	ON-OBSV	2×125 RBFNN with Gaussian function (static structure)	Lyapunov	N/A	AC	Tracking	THE, SIM	iii, v
[83] (2016)	Quadrotor	ON-Control	RBFNN	Lyapunov	N/A	SMC	Tracking	THE, SIM	iii, v
[84] (2008)	Helicopter	ON-OBSV	2 × 15 MLPs	Lyapunov	N/A	BS	Tracking	THE, SIM	iii, v
[80] (2018)	Fixed-wing	OFF-CONTID	2 CNNs + 2 LSTMs + 1 dense regression layer	N/A	N/A	FL	Command tracking	SIM	i, ii, iii, iv

Table 10 This table summarizes the review of ANN-based flight control for UAV applications as addressed in Section 4.1. The columns represent the referenced work and the year published (REF), the type of UAV under control (UAV), the functionalities of ANN (ANN_FUNC), the architecture of ANN (ANN_ARCH), the training methods for ANN (ANN_TRAIN), time complexity (TC), main control techniques adopted for flight control (CTRL), flight manueuvers (MANUEUVER), achieved results (RESULT), and problems that remain unsolved (ISSUE), as early defined in the reference template in Section 2.

4.3 Classification of Approaches

Table 11 presents the classification of existing literature on UAV flight control combined with ANNs. It is straightforward to see that online networks are used more frequently than offline ones due to their intrinsic learning ability. This is quite different from the situation where we saw for dynamic modeling (as shown in Table 7) mainly because either control augmentation or uncertainty observer requires real-time adaptability which offline networks do not possess.

As for the role that ANNs play in the controller design, control augmentation is mostly associated with feedback linearization technique whereas uncertainty observer usually works together with backstepping and sliding mode control techniques. Additionally, ANNs have also been used to mimic static controller or identify the online controller coefficient.

	Offline	**Online**	**ADAUG**	**OBSV**	**Other**
PID	N/A	[74]	✓		
FL	[80]	[50,76–78]	✓		✓
OP	[56]	N/A			✓
AC	[80]	[79]		✓	✓
SMC	N/A	[82,83]		✓	✓
BS	N/A	[82,84]		✓	

Table 11 Literature classification of UAV flight control. **Left-Column:** Training methods. **Right-Column:** ANN functionalities. **Row:** Control techniques.

4.4 Challenges and Opportunities

Noted from Table 10 and Table 11, the main purpose of using ANNs in flight control proposed by most of the literature is to increase the robustness against uncertainties and disturbances, either by means of adaptive control augmentation or uncertainty observers. It should also be emphasized that Lyapunov stability theory is adopted as the mainstream approach to update network weights, rather than using data-driven approaches, to provide rigorous analytical proof of stability and convergence. Hence, it can be concluded that through these efforts, i-(*Stability*) and ii-(*Robustness*) have been intentionally addressed. However, iii-(*Paucity of data*) and v-(*Real-time performance*) remain to be under investigation since only very few work has presented experimental results and little concentration has been placed on implementation aspects.

5 Conclusions

In this survey, we began with the investigation of the rationale for model-based flight control with ANNs for UAV applications. It has been shown that there is a complementary relation between MBC techniques and ANNs. To design and implement an MBC-ANN flight controller, however, several issues need to be addressed and solved, which include stability, robustness, paucity of data, interpretability, and real-time performance.

To provide technical justifications as well as supported answers to these questions, we carried out a thorough review on the existing literature focusing on the combination of model-based control techniques and ANNs. Specifically, a reference template was proposed and used as a unified framework to fairly determine the capabilities and limitations of each approach of dynamic modeling and control techniques. Bearing in mind the real-time implementation and performance, analysis on time complexity was also performed for several frequently-used ANNs in the literature. Through classification and comparison results, we also shed light on the direction of future development.

Based on this survey, our future work will be the design of a UAV flight control system with combined MBC-ANN techniques which aims to be equipped with stable and robust autonomous flight capability under uncertainties and disturbances. On top of that, we will explore methods to open the black box of ANNs by incorporating domain knowledge. From implementation perspectives, we will continue our study on the time complexity of online training for different ANN architecture and research suitable hardware platforms for deployment and experimental tests for MBC-ANN flight controller.

References

1. H. Voos, "Nonlinear control of a quadrotor micro-uav using feedback-linearization," *2009 IEEE International Conference on Mechatronics*, pp. 1–6, 2009.
2. C. Nicol, C. Macnab, and A. Ramirez-Serrano, "Robust adaptive control of a quadrotor helicopter," *Mechatronics*, vol. 21, no. 6, pp. 927 – 938, 2011.
3. K. Alexis, G. Nikolakopoulos, and A. Tzes, "Experimental model predictive attitude tracking control of a quadrotor helicopter subject to wind-gusts," *18th Mediterranean Conference on Control and Automation, MED'10*, pp. 1461–1466, 2010.
4. R. Xu and U. Ozguner, "Sliding mode control of a quadrotor helicopter," *Proceedings of the 45th IEEE Conference on Decision and Control*, pp. 4957–4962, 2006.
5. A. Das, F. L. Lewis, and K. Subbarao, "Backstepping approach for controlling a quadrotor using lagrange form dynamics," *Journal of Intelligent and Robotic Systems*, vol. 56, pp. 127–151, 2009.
6. G. V. Raffo, M. G. Ortega, and F. R. Rubio, "An underactuated h infinity control strategy for a quadrotor helicopter," *2009 European Control Conference (ECC)*, pp. 3845–3850, 2009.
7. F. Santoso, M. A. Garratt, and S. G. Anavatti, "State-of-the-art intelligent flight control systems in unmanned aerial vehicles," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 2, pp. 613–627, 2018.
8. R. W. Prouty, "Helicopter performance, stability, and control," 1986.
9. G. Hoffmann, H. Huang, S. Waslander, and C. Tomlin, "Quadrotor helicopter flight dynamics and control: Theory and experiment," in *AIAA Guidance, Navigation and Control Conference and Exhibit*, 2007, p. 6461.

10. S. A. Conyers, M. J. Rutherford, and K. P. Valavanis, "An empirical evaluation of ceiling effect for small-scale rotorcraft*," *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 243–249, 2018.
11. ——, "An empirical evaluation of ground effect for small-scale rotorcraft," *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1244–1250, 2018.
12. J. S. Shamma and J. R. Cloutier, "Gain-scheduled missile autopilot design using linear parameter varying transformations," *Journal of guidance, Control, and dynamics*, vol. 16, no. 2, pp. 256–263, 1993.
13. K. S. Tsakalis and P. A. Ioannou, *Linear time-varying systems: control and adaptation*. Prentice-Hall, Inc., 1993.
14. P. A. Ioannou and J. Sun, *Robust adaptive control*. PTR Prentice-Hall Upper Saddle River, NJ, 1996, vol. 1.
15. M. Sadraey and R. Colgren, "Robust nonlinear controller design for a complete uav mission," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2006, p. 6687.
16. S. Fekri, M. Athans, and A. Pascoal, "Issues , progress and new results in robust adaptive control," 2006.
17. Y. A. Younes, A. Drak, H. N. Noura, A. Rabhi, and A. E. hajjaji, "Model-free control of a quadrotor vehicle," *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 1126–1131, 2014.
18. A. N. Chand, M. Kawanishi, and T. Narikiyo, "Non-linear model-free control of flapping wing flying robot using ipid," *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2930–2937, 2016.
19. P. P. Yip and J. K. Hedrick, "Adaptive dynamic surface control: a simplified algorithm for adaptive backstepping control of nonlinear systems," *International Journal of Control*, vol. 71, no. 5, pp. 959–979, 1998.
20. K. J. Hunt, D. Sbarbaro, R. Żbikowski, and P. J. Gawthrop, "Neural networks for control systems—a survey," *Automatica*, vol. 28, no. 6, pp. 1083–1112, 1992.
21. A. Carrio, C. Sampedro, A. Rodriguez-Ramos, and P. Campoy, "A review of deep learning methods and applications for unmanned aerial vehicles," *Journal of Sensors*, vol. 2017, 2017.
22. Y. Jiang, C. Yang, J. Na, G. Li, Y. Li, and J. Zhong, "A Brief Review of Neural Networks based Learning and Control and their Applications for Robots," vol. 2017, pp. 1–30, 2017.
23. K. Hornik, M. Stinchcomb, and H. White, "Multilayer feedforward networks are universal approximator," *IEEE Transactions on Neural Networks*, vol. 2, 01 1989.
24. G. Cybenko, "Approximations by superpositions of a sigmoidal function," *Mathematics Control Signals Systems*, vol. 2, 1989.
25. D. Lazer, R. Kennedy, G. King, and A. Vespignani, "The parable of google flu: Traps in big data analysis," *Science (New York, N.Y.)*, vol. 343, pp. 1203–5, 03 2014.
26. T. Cheng, P. Wen, and Y. D. Li, "Research status of artificial neural network and its application assumption in aviation," *2016 12th International Conference on Computational Intelligence and Security (CIS)*, pp. 407–410, 2016.
27. R. Zhang, J. Zhang, and H. Yu, "Review of modeling and control in uav autonomous maneuvering flight," *2018 IEEE International Conference on Mechatronics and Automation (ICMA)*, pp. 1920–1925, 2018.
28. W. Gu, K. P. Valavanis, M. J. Rutherford, and A. Rizzo, "A survey of artificial neural networks with model-based control techniques for flight control of unmanned aerial vehicles," *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 362–371, 2019.
29. W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity. 1943." *Bulletin of mathematical biology*, vol. 52 1-2, pp. 99–115; discussion 73–97, 1990.
30. F. F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain." *Psychological review*, vol. 65 6, pp. 386–408, 1958.
31. D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 1986.
32. G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks." *Science*, vol. 313 5786, pp. 504–7, 2006.

33. J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks : the official journal of the International Neural Network Society*, vol. 61, pp. 85–117, 2015.
34. M. H. Beale, M. T. Hagan, and H. B. Demuth, "Neural network toolbox^{TM} user's guide," *The Mathworks Inc*, 1992.
35. D. S. Broomhead and D. Lowe, "Multivariable functional interpolation and adaptive networks," *Complex Systems*, vol. 2, 1988.
36. Q. Zhang and A. Benveniste, "Wavelet networks," *IEEE transactions on neural networks*, vol. 3 6, pp. 889–98, 1992.
37. R. H. R. Hahnloser, R. Sarpeshkar, M. A. Mahowald, R. J. Douglas, and H. S. Seung, "Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit," *Nature*, vol. 405, pp. 947–951, 2000.
38. J. S. Albus, "I a new approach to manipulator control: The i cerebellar model articulation controller," 1975.
39. J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities." *Proceedings of the National Academy of Sciences of the United States of America*, vol. 79 8, pp. 2554–8, 1982.
40. J. L. Elman, "Finding structure in time," *Cognitive science*, vol. 14, no. 2, pp. 179–211, 1990.
41. H. Jaeger, "The"echo state"approach to analysing and training recurrent neural networks," 2001.
42. S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, pp. 1735–1780, 1997.
43. S. Hochreiter and Y. Bengio, "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies," 2001.
44. K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *ArXiv*, vol. abs/1409.1259, 2014.
45. P. J. Werbos, "Backpropagation through time: What it does and how to do it," 1990.
46. R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural Computation*, vol. 1, pp. 270–280, 1989.
47. J. E. Sierra and M. S. Peñas, "Modelling engineering systems using analytical and neural techniques: Hybridization," *Neurocomputing*, vol. 271, pp. 70–83, 2018.
48. D. J. Struik, *A source book in mathematics, 1200-1800*. Harvard University Press, 1969, vol. 11.
49. J. Alvarenga, N. I. Vitzilaios, K. P. Valavanis, and M. J. Rutherford, "Survey of unmanned helicopter model-based navigation and control techniques," *Journal of Intelligent & Robotic Systems*, vol. 80, no. 1, pp. 87–138, 2015.
50. B. S. Kim and A. J. Calise, "Nonlinear flight control using neural networks," *Journal of Guidance, Control, and Dynamics*, vol. 20, no. 1, pp. 26–33, 1997.
51. B. W. Mel and C. Koch, "Sigma-pi learning: On radial basis functions and cortical associative learning," in *NIPS*, 1989.
52. N. T. Nguyen, K. S. Krishnakumar, J. Kaneshige, and P. Nespeca, "Dynamics and adaptive control for stability recovery of damaged asymmetric aircraft," 2006.
53. R. S. Martin, A. Barrientos, P. Gutiérrez, and J. del Cerro, "Unmanned aerial vehicle (uav) modelling based on supervised neural networks," *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pp. 2497–2502, 2006.
54. R. S. Martín, A. Barrientos, P. Gutiérrez, and J. D. Cerro, "Neural networks training architecture for uav modelling," *2006 World Automation Congress*, pp. 1–6, 2006.
55. V. R. Puttige and S. G. Anavatti, "Comparison of real-time online and offline neural network models for a uav," *2007 International Joint Conference on Neural Networks*, pp. 412–417, 2007.
56. M. Garratt and S. Anavatti, "Non-linear control of heave for an unmanned helicopter using a neural network," *Journal of Intelligent & Robotic Systems*, vol. 66, no. 4, pp. 495–504, Jun 2012. [Online]. Available: https://doi.org/10.1007/s10846-011-9634-9
57. S. Bhandari, A. Raheja, D. Tang, K. Ortega, O. Dadian, and A. Bettadapura, "Nonlinear control of uavs using multi-layer perceptrons with off-line and on-line learning," *2014 American Control Conference*, pp. 2875–2880, 2014.

58. A. Vargas, M. Ireland, and D. Anderson, "System identification of multi-rotor uavs using echo state networks," 2015.
59. N. Mohajerin and S. L. Waslander, "Modelling a quadrotor vehicle using a modular deep recurrent neural network," *2015 IEEE International Conference on Systems, Man, and Cybernetics*, pp. 376–381, 2015.
60. ——, "Modular deep recurrent neural network: Application to quadrotors," *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 1374–1379, 2014.
61. Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.
62. A. Punjani and P. Abbeel, "Deep learning helicopter dynamics models," *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3223–3230, 2015.
63. F. Takens, "Detecting strange attractors in turbulence," 1981.
64. J. C. Robinson, "A topological delay embedding theorem for infinite-dimensional dynamical systems," 2005.
65. S. Bansal, A. K. Akametalu, F. J. Jiang, F. Laine, and C. J. Tomlin, "Learning quadrotor dynamics using neural network for flight control," *2016 IEEE 55th Conference on Decision and Control (CDC)*, pp. 4653–4660, 2016.
66. O. Dadian, S. Bhandari, and A. Raheja, "A recurrent neural network for nonlinear control of a fixed-wing uav," in *American Control Conference (ACC), 2016.* IEEE, 2016, pp. 1341–1346.
67. M. Lukoševičius, "A practical guide to applying echo state networks," in *Neural networks: Tricks of the trade.* Springer, 2012, pp. 659–686.
68. Y.-F. Teng, B. Hu, Z.-W. Liu, J. Huang, and Z.-H. Guan, "Adaptive neural network control for quadrotor unmanned aerial vehicles," *2017 11th Asian Control Conference (ASCC)*, pp. 988–992, 2017.
69. N. Mohajerin, M. Mozifian, and S. L. Waslander, "Deep learning a quadrotor dynamic model for multi-step prediction," *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2454–2459, 2018.
70. M. Gori and A. Tesi, "On the problem of local minima in backpropagation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, pp. 76–86, 1992.
71. A. Karpatne, W. Watkins, J. Read, and V. Kumar, "Physics-guided neural networks (pgnn): An application in lake temperature modeling," *arXiv preprint arXiv:1710.11431*, 2017.
72. X. Jia, J. Willard, A. Karpatne, J. S. Read, J. Zward, M. Steinbach, and V. Kumar, "Physics guided rnns for modeling dynamical systems: A case study in simulating lake temperature profiles," *ArXiv*, vol. abs/1810.13075, 2018.
73. A. Karpatne, G. Atluri, J. H. Faghmous, M. Steinbach, A. Banerjee, A. R. Ganguly, S. Shekhar, N. F. Samatova, and V. Kumar, "Theory-guided data science: A new paradigm for scientific discovery from data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, pp. 2318–2331, 2016.
74. T.-C. Yang and J.-G. Juang, "Aircraft landing control based on adaptive cmac," *2009 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pp. 791–796, 2009.
75. J. A. Leitner, A. J. Calise, and J. V. R. Prasad, "Analysis of adaptive neural networks for helicopter flight control," 1995.
76. S. K. Kannan and E. N. Johnson, "Adaptive trajectory based control for autonomous helicopters," 2002.
77. G. V. Chowdhary and E. N. Johnson, "Theory and flight-test validation of a concurrent-learning adaptive controller," *Journal of Guidance, Control, and Dynamics*, vol. 34, no. 2, pp. 592–607, 2011.
78. D. T. Ocaña, H.-S. Shin, and A. Tsourdos, "Development of a nonlinear reconfigurable f-16 model and flight control systems using multilayer adaptive neural networks," *IFAC-PapersOnLine*, vol. 48, no. 9, pp. 138–143, 2015.
79. G. Lai, Z. D. Liu, Y. Zhang, and C. L. P. Chen, "Adaptive position/attitude tracking control of aerial robot with unknown inertial matrix based on a new robust neural identifier," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, pp. 18–31, 2016.

80. B. Eroglu, C. Sahin, B. Yuksek, N. K. Ure, and G. Inalhan, "Deep recurrent and convolutional networks for accelerated fault tolerant adaptive flight control under severe failures," in *2018 Annual American Control Conference (ACC)*, June 2018, pp. 6559–6565.
81. M. R. Akella, J. L. Junkinst, and R. D. Robinett, "Structured model reference adaptive control with actuator saturation limits," 1998.
82. C. Peng, Y. Bai, X. Gong, Q. Gao, C. Zhao, and Y. Tian, "Modeling and robust backstepping sliding mode control with adaptive rbfnn for a novel coaxial eight-rotor uav," *IEEE/CAA Journal of Automatica Sinica*, vol. 2, pp. 56–64, 2015.
83. S. Li, Y. Wang, J. Tan, and Y. Zheng, "Adaptive rbfnns/integral sliding mode control for a quadrotor aircraft," *Neurocomputing*, vol. 216, pp. 126–134, 2016.
84. T. Madani and A. Benallegue, "Adaptive control via backstepping technique and neural networks of a quadrotor helicopter," *IFAC Proceedings Volumes*, vol. 41, no. 2, pp. 6513–6518, 2008.