The Aeronautical Journal (2025), **129**, pp. 2777–2801
doi:10.1017/aer.2025.10024

RESEARCH ARTICLE

A supervised machine-learning approach for turboshaft engine dynamic modeling under real flight conditions

D. Paniccia[1], F.A. Tucci[1], J. Guerrero[1], L. Capone[1], N. Sanguini[2], T. Benacchio[2,†] and L. Bottasso[2]

[1]Leonardo Labs, Leonardo S.p.A., Genoa 16151, Italy
[2]Leonardo Labs, Leonardo S.p.A., Samarate 21017, Italy
Corresponding author: Damiano Paniccia; Email: damiano.paniccia@leonardo.com

Received: 7 November 2024; **Revised:** 28 April 2025; **Accepted:** 29 April 2025

Keywords: machine learning; rotorcraft engine modeling; sparse regression

Abstract

Rotorcraft engines are highly complex, nonlinear thermodynamic systems operating under varying environmental and flight conditions. Simulating their dynamics is crucial for design, fault diagnostics and deterioration control, requiring robust control systems to estimate performance throughout the flight envelope. Numerical simulations provide accurate assessments in both steady and unsteady scenarios through physics-based and mathematical models, although their development is challenging due to the engine's complex physics and strong dependencies on environmental conditions. In this context, data-driven machine-learning techniques have gained significant interest for their ability to capture nonlinear dynamics and enable online performance estimation with competitive accuracy. This work explores different neural network architectures to model the turboshaft engine of Leonardo's AW189P4 prototype, aiming to predict engine torque. The models are trained on a large database of real flight tests, covering a variety of operational manoeuvers under different conditions, thus offering a comprehensive performance representation. Additionally, sparse identification of nonlinear dynamics (SINDy) is applied to derive a low-dimensional model from the available data, capturing the relationship between fuel flow and engine torque. The resulting model highlights SINDy's ability to recover underlying engine physics and suggests its potential for further investigations into engine complexity. The paper details the development and prediction results of each model, demonstrating that data-driven approaches can exploit a broader range of parameters compared to standard transfer function-based methods, enabling the use of trained schemes to simulate nonlinear effects in different engines and helicopters.

Nomenclature

AI	Artificial Intelligence
ML	Machine Learning
DMD	Dynamic Mode Decomposition
MTOW	Maximum TakeOff Weight
MISO	Multi-Input Single-Output
SISO	Single-Input Single-Output
TRQ	Engine Torque
WF	Fuel Flow
COL	Collective Pitch
T_1	Intake air temperature
T_{oil}	Oil temperature
P_{oil}	Oil pressure
P_0	Intake pressure

[†]Now at: Weather Research, Danish Meteorological Institute, Sankt Kjelds Plads 11, 2100 Copenhagen Denmark

© Leonardo S.p.A., 2025. Published by Cambridge University Press on behalf of Royal Aeronautical Society.

https://doi.org/10.1017/aer.2025.10024 Published online by Cambridge University Press

NR	Main rotor speed
TAT	True air temperature
NP	Power turbine speed
NG	Gas turbine speed
NGR	Corrected gas turbine speed
AIRSPEED	Air speed
FFNN	Feed-Forward Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
MSE	Mean Squared Error
MAE	Mean Absolute Error
rMAE	Relative Mean Absolute Error
SINDy	Sparse Identification of Nonlinear Dynamics
SINDyc	SINDy with control
PySINDy	Python library for Sparse Identification of Nonlinear Dynamics
D_{train}	Training dataset (162 flights)
D_{val}	Validation dataset (18 flights)
D_{test}	Test dataset (16 flights with high-frequency sweep maneuvers)

1.0 Introduction

Real-time control of helicopter turboshaft engines has always been a complex challenge due to the stringent requirements of robust engine models capable of accounting for the strong coupling between the engine and the helicopter drivetrain. In fact, the effective implementation of control systems has always been recognised as essential for enhancing helicopter performance, monitoring and handling, as well as minimising fuel supply oscillations and rotor speed fluctuations [1, 2]. High-fidelity aero-thermodynamic simulations, while invaluable for detailed analysis, are often unfeasible for real-time applications due to their computational complexity and iterative nature, significantly increasing execution time. Simplified aero-thermodynamic models based on reduced-order modeling approach have been proposed as possible alternatives to allow for the real-time applications, still exhibiting many differences in results compared to full transient high-fidelity simulations [3]. Although often built upon static maps and linear approximations with corrections to account for environmental conditions [4], these physics-based models, dating back to the 1980s [5], are grounded in thermodynamic principles and rely on a detailed physical description of engine components that enhances their robustness and reliability. Recent studies (see e.g. [6–8]) have focused on simulating engine behaviour for control applications. As mentioned above, the execution of real-time simulation of physically driven models is challenging, and many compromises are therefore necessary to develop engine models that can easily be exploited for system control, fault detection and diagnostics. By sacrificing some aspects of the engine's physical representation, two of the most straightforward approaches involve simplified real-time engine simulations based on piecewise linear state-space perturbation models or transfer function models. These methods enable coverage of the entire operating range while supporting operational requirements [9–13]. Both rely on tabulated steady-state engine performance at key operating points, with engine dynamics evaluated either through partial derivative approximations around these points (state-space perturbation) or via transfer functions such as lead-lag compensators. Although these models are less accurate than full transient high-fidelity simulations, they strike a practical balance between computational efficiency and fidelity, making them widely applicable in real-world operational scenarios. Still, developing and fine-tuning these models requires a significant effort, as they rely on extensive and costly flight campaigns involving specific manoeuvers and tests. Additionally, a major challenge arises from the inevitable variations in engine performance over time, as well as new flight conditions and transient manoeuvers that were not accounted for during development or deviate significantly from tabulated points. These limitations can gradually degrade even a well-calibrated model, requiring continuous data collection and additional flight tests to maintain and update existing models.

https://doi.org/10.1017/aer.2025.10024 Published online by Cambridge University Press

Over the past decade, turbomachinery and aircraft engine communities have begun to reconsider their design, manufacturing and operational processes in response to the exponential growth in the use of smart technologies [14]. For instance, artificial intelligence (AI) and machine-learning (ML) algorithms have started to be integrated into several engineering applications, leading to a shift from either empirical or purely theoretical approaches to increasingly accurate solutions capable of capturing and extracting complex nonlinear connections directly from data, albeit at the expense of a physical description and understanding. The modeling of turbomachinery is no exception. Considerable efforts and significant progress have already been made to harness ML algorithms for several applications in this field [15–20], from the design and control of the air and fuel flow [21–23], to deterioration modeling and fault prediction [24, 25]. At the same time, despite the great potential, this shift towards purely data-driven models is still viewed with a certain degree of skepticism by industries such as aerospace [26], due to the high-stakes and safety-critical nature of these environments. To address this issue, data-driven control methods like dynamic mode decomposition (DMD) [27, 28] and Koopman analysis [29] enable a structured understanding of system dynamics by extracting dominant patterns from data while preserving some interpretability. Among others, the recently introduced sparse identification of nonlinear dynamics (SINDy) [30] has been attracting increasing attention due to its low-computational cost and its accuracy in modeling non-linear systems, an area where DMD struggles due to its inherent focus on linear patterns [31]. Up to now, the SINDy algorithm has been widely adopted across various domains, demonstrating strong potential in both accuracy and interpretability. Applications span optics [32], plasma dynamics [33], chemical processes [34, 35], structural systems [36] and fluid dynamics [37], leading to very promising results in terms of accuracy and interpretability. Recently, SINDy has been also applied to aircraft dynamics [31, 38], aircraft pressure control [39], collision avoidance for multirotor UAVs [40] and predictive control for diesel engine systems [41], further expanding its relevance in engineering and control applications.

Building on these developments, this paper explores the application of machine learning techniques to helicopter engine modeling, flight mechanics simulation, and performance prediction. Particular attention is given to both fully data-driven methods, which emphasise predictive accuracy and can incorporate external information such as environmental and operational data, and more interpretable approaches that aim to retain physical insight. This study is characterised by at least two key specificities. First, it focuses on a unique type of helicopter turboshaft and its associated drivetrain. Second, it leverages large-scale datasets of actual flight data acquired from extensive flight test campaigns of a modern rotorcraft, including a rich set of manoeuvers and actual engine flight conditions. The dataset was generated by a prototype of Leonardo's AW189[1], an 8-ton maximum takeoff weight (MTOW) twin-engine helicopter (Fig. 1).

To illustrate the complexity of the engine's operation, Fig. 2 presents a typical Leonardo helicopter turboshaft drivetrain and a simple sketch of the engine components. The engine first scoops air through the inlet and increases its pressure within the compressor before it enters the combustion chamber, where fuel is sprayed. The resulting air-fuel mixture then expands in the first turbine, which drives the compressor and is mechanically linked to the same shaft. Before exiting through the exhaust nozzle, the hot gases impinge on a second turbine stage, known as the power turbine. This turbine is mechanically decoupled from the first but is connected to the helicopter rotor through a reduction gearbox. As a result, the two turbines can rotate at different speeds, a key feature of helicopter drivetrain design that allows the rotor to maintain a constant RPM independently of the engine regime. The engine power is thus transferred to the helicopter rotor through a transmission gearbox whose reduction ratio is typically in the order of 100 (achieved in successive stages), from around 30,000 rpm in the power turbine to around 300 rpm in the main rotor.

Due to the intricacies of the drivetrain and the engine's dynamics, accurate helicopter engine simulations play a crucial role in understanding flight mechanics and control. In this context, a realistic engine model is influenced by both pilot control inputs and other critical variables that are of paramount importance for the correct estimation of the helicopter's handling qualities. Specifically, the engine can

[1] https://helicopters.leonardo.com/it/products/aw189-1

https://doi.org/10.1017/aer.2025.10024 Published online by Cambridge University Press



Figure 1. *Leonardo's AW189 twin-engine helicopter. Copyright on the images is held by the contributors. Apart from Fair Use, permission must be sought for any other purpose.*

be simulated as a multi-input single-output (MISO) model where the output variable is torque (*TRQ*) (directly linked to delivered power) which is closely correlated with fuel flow (or weight of fuel, abbreviated as *WF*). In fact, *WF* is an internal variable of the system, not easily monitorable but under the pilot's control by means of the cyclic, pedals and collective setting (*COL*), regulating the power delivered to the rotor. In addition, the main rotor speed (NR) governor ensures that rotor rpm remain stable with minimal fluctuations.

The supervised MISO models developed in this study are trained to take as input a specific set of engine and environment variables, known to influence engine behaviour but not easily included in physics-based models (either for a specific time-step or for a specific time window). The models return the desired *TRQ* variable as a final output, demonstrating that data-driven engine models can exploit a wider range of parameters than standard transfer function-based approaches. Specifically, we explore two different neural networks (NN) architectures: feed-forward neural network (FFNN) [42] and long-short-term memory (LSTM) recurrent network [43, 44]. The rationale of the work lies in the expectation that data-driven models, being capable of describing the dynamic behaviour of nonlinear systems [16, 17], can expand the scope of the currently employed transfer function-based approaches by leveraging a broader set of inputs, identifying all those variables that correlate well with the desired output, the engine torque *TRQ*. Therefore, since data-driven engine models can exploit a wide range of parameters describing the turbomachinery, trained schemes can be used to effectively simulate nonlinear effects among different engines, operating regimes and helicopter models. This may allow for an accurate dynamic simulation over the entire flight envelope of helicopter turboshaft performance and may also significantly improve their design, optimisation and maintenance processes [18].

To complement the study on purely data-driven methods, this paper also presents the application of the SINDy algorithm to derive a low-dimensional dynamical model from the available data, trying to capture the relationship between *WF* and *TRQ*, i.e. a single-input single-output (SISO) model. The SINDy methodology identifies the smallest number of terms needed to explain the data without detailed prior physical knowledge, still leading to interpretable governing equations, which may enrich the understanding of this complex system. Unlike other data-driven machine-learning approaches, sparse identification leverages a unique balance in model complexity and descriptive ability, an essential quality in the aviation engine field to meet the highest level of model robustness and validation. The goal

https://doi.org/10.1017/aer.2025.10024 Published online by Cambridge University Press

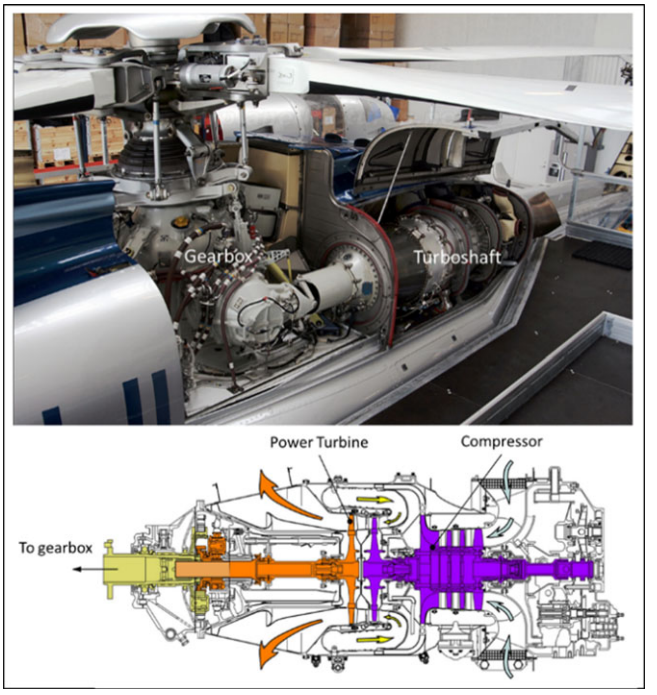


Figure 2. *Helicopter engine installation. Copyright on the images is held by the contributors. Apart from Fair Use, permission must be sought for any other purpose.*

of this investigation is to assess the advantages and limitations of this methodology when applied to real-world flight data, collected during routine helicopter operations throughout its life cycle.

The paper is structured as follows. Section 2 describes the characteristics of the machine learning methods used in the study, while Section 3 contains results of the experiments and related discussion. Section 4 concludes the paper, and the appendix documents further details on hyperparameter tuning and training setup.

2.0 Methodology

This section provides a brief description of the adopted NN architectures for the MISO model, followed by some theoretical background of SINDy. The last subsection provides some details on the Leonardo's AW189 dataset and on the training setup.

2.1 Neural network architectures

For the MISO models, we implemented two distinct NN architectures using the PyTorch library [45] to assess their respective advantages and limitations for modelling the engine dynamics.

The first architecture is a FFNN [42, 46], which establishes direct correlations between the input features of a specific time step and the output feature at the same time step. FFNNs are structured with one or more hidden layers between the input and output layers, where each hidden layer consists of neurons that are fully connected to the neurons in the previous layer. Information flows through these connections, processed by weights, biases and nonlinear activation functions. During training, the network parameters are optimised through backpropagation and gradient descent, with the aim of minimising the loss function. FFNNs are generally efficient in capturing non-linear relationships between input and output variables, but they usually lack the ability to model temporal dependencies, making them less suitable for time series data. In this work, the FFNN is applied to each time step independently. Each

https://doi.org/10.1017/aer.2025.10024 Published online by Cambridge University Press

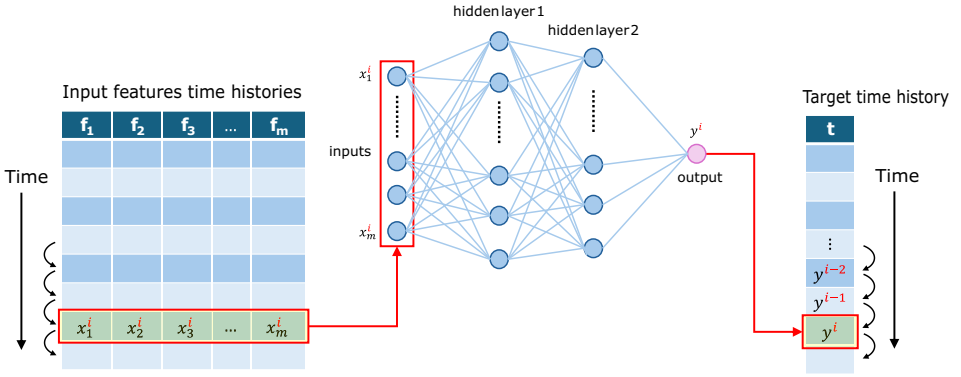


Figure 3. *A simple sketch of the FFNN architecture for the torque time-history prediction.*

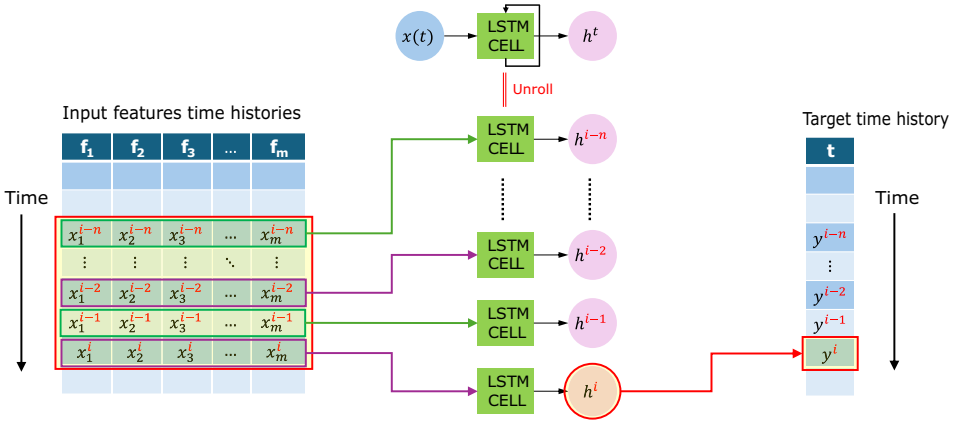


Figure 4. *A simple sketch of the LSTM architecture for the torque time-history prediction.*

time step of the input time series is treated as a separate input sample, and the network predicts the corresponding output for that time step. The full-time series output is reconstructed by sequentially placing the predicted outputs in order (see Fig. 3 for an illustration of the architecture).

To address the FFNN's limitation in modeling temporal relationships, we also implemented a more sophisticated architecture based on LSTM networks. LSTMs are a specialised form of recurrent neural networks (RNNs) and are designed to capture long-term dependencies in sequential data, making them particularly effective for time series. Unlike FFNNs, which process each input independently, LSTMs incorporate memory cells and gating mechanisms that allow the network to retain information across multiple time steps. In our implementation, the network receives as input the input features sequences of n time steps, referred to as a lookback window, and outputs a single prediction corresponding to the final time step in that window (see Fig. 4). As the model moves through the dataset, it generates one prediction per window by shifting the window forward one time step at a time, starting from the n th row. By passing information from one time step to the next, the LSTM can learn temporal patterns and dependencies within the time series. However, this comes at the cost of increased complexity and longer training times compared to more standard FFNN architectures.

In our implementations, we used the mean squared error (MSE) between the predicted and actual output as the loss metric for both architectures. Details on the input and output features and on the dataset are provided in the following sections.

https://doi.org/10.1017/aer.2025.10024 Published online by Cambridge University Press

2.2 Sparse identification of non-linear dynamics

The SINDy approach uses sparse regression to find the smallest number of terms required to model the dynamics of a given system from a library of prescribed candidate functions. In the present work, we employ PySINDy [47, 48], a comprehensive Python package for sparse identification of nonlinear dynamics, to assess the SINDy approach with our AW189 engine database. The goal is to understand the advantages and limitations of this methodology when applied to real data from extensive flight campaigns. However, as a first step, we restrict the analysis to a SISO model, focusing specifically on the relationship between fuel flow (*WF*) and engine torque (*TRQ*), where *WF* is a control variable that directly influences *TRQ*. This makes the *WF*-*TRQ* relationship a natural candidate for benchmarking the 'SINDy with control' (SINDyc) algorithm [49], as this empirical relationship governing *WF* and *TRQ* is well documented in the field of engine modeling. The SINDy with control algorithm, hereafter referred to simply as SINDy, is briefly described below.

Consider a nonlinear dynamical system like

$$\frac{d\mathbf{x}}{dt} = f\left(\mathbf{x}, \mathbf{u}\right) \tag{1}$$

where $\mathbf{x}$ is the state vector collecting all the state variables and $\mathbf{u}$ is the input vector collecting all the control variables. Suppose we have m snapshots of both the state and input vectors arranged in two data matrices $\mathbf{X}$ and $\mathbf{U}$, as

$$\mathbf{X} = \begin{pmatrix} \vdots & \vdots & \dots & \vdots \\ x_1 & x_2 & \dots & x_m \\ \vdots & \vdots & \dots & \vdots \end{pmatrix} \tag{2}$$

$$\mathbf{U} = \begin{pmatrix} \vdots & \vdots & \dots & \vdots \\ u_1 & u_2 & \dots & u_m \\ \vdots & \vdots & \dots & \vdots \end{pmatrix} \tag{3}$$

We may define a library of nonlinear arbitrary candidate functions Θ of the state and the input including nonlinear cross terms

$$\Theta^T = \begin{pmatrix} \dots & X & \dots \\ \dots & X^2 & \dots \\ \vdots & \vdots & \vdots \\ \dots & U & \dots \\ \dots & U^2 & \dots \\ \vdots & \vdots & \vdots \\ \dots & X\,U & \dots \\ \dots & X^2\,U & \dots \\ \dots & X\,U^2 & \dots \\ \vdots & \vdots & \vdots \\ \dots & \sin X & \dots \\ \dots & \sin U & \dots \\ \dots & \sin X \sin U & \dots \\ \vdots & \vdots & \vdots \end{pmatrix} \tag{4}$$

https://doi.org/10.1017/aer.2025.10024 Published online by Cambridge University Press

and finally solve for the sparse regression coefficients matrix Ξ the equations

$$\frac{d\mathbf{X}}{dt} = \mathbf{\Xi}\,\Theta^{\mathrm{T}}\,(\mathbf{X}, \mathbf{U}) \tag{5}$$

by assuming that the input $\mathbf{U}$ corresponds to an external forcing.

Note that the sparsity of the coefficients matrix $\mathbf{\Xi}$ is key to obtaining an easily readable and interpretable form of the original system and is assured by the assumption of a threshold parameter ε, so that each coefficient lower than ε is automatically set to zero. The threshold should be treated as a hyperparameter of the SINDy algorithm and its correct tuning is essential for each specific case.

The SINDy algorithm has been previously presented in its general form, assuming a continuous-time framework. However, the practical implementation of SINDy necessitates a discrete representation of the time signals and their corresponding derivatives. Therefore, a dataset of measurements sampled at discrete time instances $t_1, t_2, t_3, \ldots, t_m$ is required. If the time derivatives of the state vector and any external forcing terms (input vector) are not directly available in the dataset, they can be estimated numerically. This can be done either through custom numerical differentiation or automatically via the PySINDy package, which uses the total variation regularised derivative under the hood. It is worth noting that several differentiation techniques were tested, ranging from finite difference methods to the Savitzky-Golay filter. However, no significant influence was observed on the results, likely due to the pre-processing applied to our dataset, which will be briefly described in the following paragraph.

As anticipated, our goal is to describe the relationship between the fuel flow *WF* and the engine torque *TRQ*. In this case, we may express the state vector as

$$\mathbf{X_i} = \begin{pmatrix} TRQ_{i1} & TRQ_{i2} & TRQ_{i3} & \ldots & TRQ_{im_i} \end{pmatrix} \tag{6}$$

and the input vector as

$$\mathbf{U_i} = \begin{pmatrix} WF_{i1} & WF_{i2} & WF_{i3} & \ldots & WF_{im_i} \end{pmatrix} \tag{7}$$

where the subscript i refers to the specific manoeuver of each flight in the dataset.

2.3 Dataset and training setup

For the development and training of the data-driven MISO and SISO models, we utilised an extensive AW189 engine dataset consisting of time series data for key features of interest, as outlined in Table 1. Due to the specific configuration of the AW189 data acquisition system, all signal sampling frequencies are exact multiples of 8 Hz. This property allows for efficient decimation while preserving the original dynamic characteristics of the signals. To prevent aliasing effects during the resampling process, a 20th-order low-pass finite impulse response (FIR) anti-aliasing filter is applied to continuous signals, preserving a bandwidth of up to 3.2 Hz before uniformly downsampling all signals to 8 Hz. The filtering process effectively smooths the signals, reducing high-frequency noise and enabling more stable and accurate numerical differentiation, a crucial requirement for the subsequent application of the SINDy algorithm.

The dataset covers 196 individual flights and spans more than 35 hours of recorded flight time. Out of the 196 available flights in the dataset, 180 are used for the models' development. These are further divided into a training dataset (D_{train}) and a validation dataset (D_{val}), consisting of 162 and 18 flights, respectively. These flights are never involved in the training phase and include a variety of standard manoeuvers listed and briefly described in Table 2. However, for the purpose of this analysis, some of the manoeuvers present in the original dataset, such as autorotations, normal shutdown, and taxiing, have little or no significance and were therefore excluded.

The remaining 16 flights form the test dataset (D_{test}) used for the final evaluation of all models. These flights were not randomly selected. They specifically include high-frequency sweep manoeuvers, such as very fast collective and pedal inputs, which exhibit significantly more complex dynamic behaviour

https://doi.org/10.1017/aer.2025.10024 Published online by Cambridge University Press

Table 1. *List of variables*

Parameter	Description
TRQ	Engine torque [Nm]
COL	Collective [%]
T_1	Intake air temperature [°C]
T_{45}	Gas turbine temperature [°C]
T_{Oil}	Oil temperature [°C]
P_{Oil}	Oil pressure [psi]
P_0	Intake pressure [psi]
NR	Main rotor speed [%]
TAT	True air temperature [°C]
NP	Power turbine speed [%]
NG	Gas turbine speed [%]
NGR	Corrected gas turbine speed [%]
WF	Fuel flow [lb/h]
AIRSPEED	Air speed [kts]

Table 2. *List of manoeuvers in the training and validation datasets*

Manoeuver	Description
Cruise	Forward flight at constant speed and altitude
Hover	Steady manoeuver, where the rotorcraft has constant altitude and is fixed with respect to the ground, can be in-ground effect (HIGE) or out of ground effect (HOGE)
Take-off/Landing	Initial/final flight phase
Bank turn	A change of heading manoeuver where the rotorcraft tilts around the roll axis to balance centrifugal force with lift
Climb/descent	Altitude variation due to a combination of collective and cyclic control input
Reversal	An abrupt flight control input (pedal, cyclic) applied in opposite directions over a 2 sec timespan
Pull-up	Application of cyclic control input to achieve a positive pitch change from steady level flight
Transition	The transition between two different manoeuvers e.g. hover and cruise
Acceleration/deceleration	Change in forward flight speed
Azimuth forward	From in-ground effect (HIGE) condition, establish a steady ground speed then recover hover
Sideway	Flight along the transversal axis
Rearward movements	Flight along the backwards direction
Spot turn	360° rotation around the yaw axis during hover executed in 12 sec
Sideslip	Level flight condition where the rotorcraft velocity vector briefly maintains both a longitudinal and a transversal component with no change in heading
Approach	Descent and alignment phase of the aircraft
Quarter manoeuver	Manoeuvers involving partial changes in attitude or direction (of approximately 90 degrees)

https://doi.org/10.1017/aer.2025.10024 Published online by Cambridge University Press

Table 3. *List of manoeuvers in the test dataset*

Man0euver	Description
Collective sweep	A rapid cycling of the collective pitch performed at increasing frequency, designed to excite the dynamic response of the rotorcraft and typically used for system identification
Yaw sweep	A rapid cycling of the pedals performed at increasing frequency, designed to excite the dynamic response of the rotorcraft to changes in directional control
Doublet	A quick change in control input (e.g., yaw, pitch, or roll) followed by an opposite input, testing the rotorcraft's dynamic response

Table 4. *MISO models input-output features set*

Input Features	Target Feature
COL, T_1,P_0, NR, $AIRSPEED$	TRQ

than those present in the training and validation datasets. As for the training and validation datasets, we present in Table 3 a list of the sweep manoeuvers, along with a brief description of each.

Identical training and cross-validation phases are carried out for the MISO models and for SISO SINDy models by exploiting *davinci-1*[2], Leonardo S.P.A.'s High Performance Computing Facility and one of the most powerful supercomputers in the aerospace, defence and security sector.

As ML models rely on statistical analysis and are strongly influenced by the distribution and quality of the data, we performed a correlation analysis to identify highly and poorly correlated variables to be included and/or excluded from the training database (Fig. 5). The correlation between two features ranges between -1 and $+1$. Features with a correlation coefficient of 1 are directly correlated, while those with a correlation coefficient of -1 are inversely correlated. Features that are uncorrelated have values close to zero. Since datasets typically contain a large number of features describing each sample, analysing their correlations may help in selecting the most relevant features and discarding those that are not needed to build an effective predictive model. For instance, highly or poorly correlated features can cause multicollinearity in prediction models, a situation in which predictors are linearly dependent, potentially leading to bias in the results. In such cases, highly or poorly correlated features should be removed from the model's input to reduce information redundancy in the data and potentially improving the model performance. In our case, for example, T_{45} has a strong correlation (0.86) with TRQ, and is also closely related to the collective COL (0.87), which in turn shows a significant correlation with the torque (0.79) as well. One approach would be to remove COL from the list of inputs and consider only T_{45} so to reduce the problem dimensionality. However, many of the variables that, as shown in Fig. 5, would be the best candidates, are not typically available for engine monitoring because they are internal to the engine itself. From a practical perspective, the aim of this work is to train a model for engine control using only information that is always accessible. In most cases, the engine manufacturer does not permit such detailed monitoring of the system, for obvious reasons related to intellectual property protection. This is why, despite the high correlation, variables like T_{45}, NG, T_{Oil}, and P_{Oil} cannot and should not be considered for model training, even though it is highly probable that some of these could lead to more accurate models.

Table 4 summarises the final set of variables included into the MISO input-output feature set after the analysis of Fig. 5.

The accuracy of all developed models is evaluated using the relative mean absolute error ($rMAE$) index, where a lower value indicates better performance. This score is calculated for each manoeuver of

[2]https://www.leonardo.com/en/innovation-technology/davinci-1

https://doi.org/10.1017/aer.2025.10024 Published online by Cambridge University Press

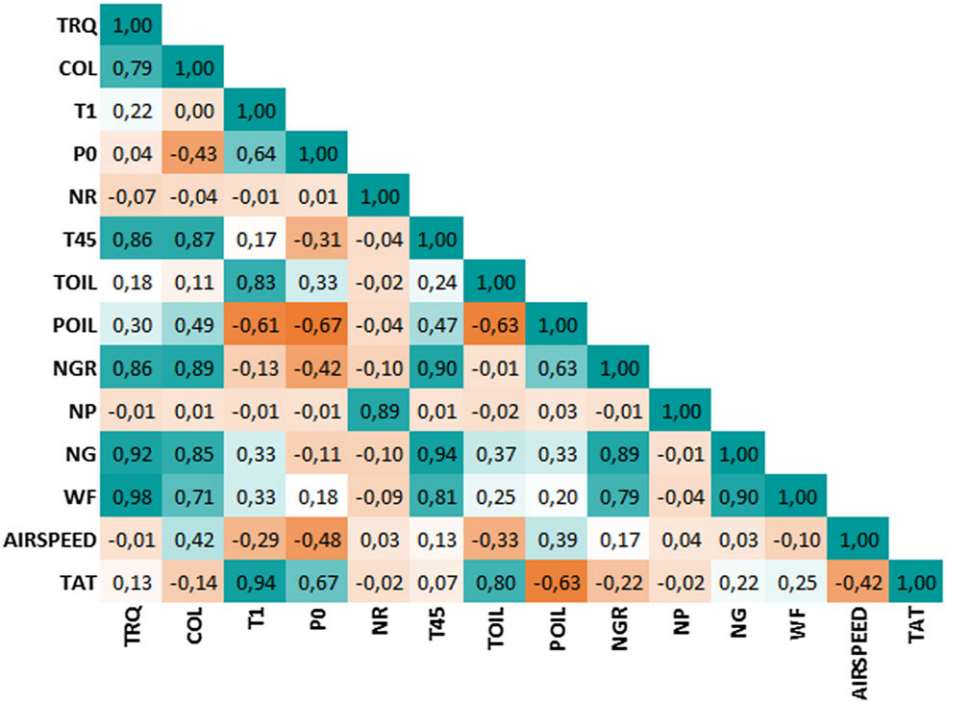


Figure 5. *Features correlation matrix for the engine variables used in the models in this paper.*

interest in each flight using the following formula:

$$rMAE_i = \frac{MAE_i}{\overline{TRQ_j}} \tag{8}$$

where MAE_i is the mean absolute error of the $i-th$ manoeuver of the $j-th$ flight and $\overline{TRQ_j}$ is the average TRQ of that specific flight. Then, to provide an overall index to compare the accuracy of the training phases of each developed model, we first calculate the $rMAE_j$, i.e., the average of the $rMAE_i$ scores for the $j-th$ flight and next global $rMAE$ score is obtained by averaging all the $rMAE_j$.

3.0 Results and Discussion

To evaluate the capabilities of the developed data-driven models in describing the engine's dynamic behaviour, the following sections compare the results obtained during the testing phase on the D_{test} dataset, consisting of 16 flights. The best and worst results are presented by plotting the model predictions against the actual TRQ values for these flights, each of which is uniquely identified by an ID (e.g., ID1, ID2, up to ID16). All the data in the figures are scaled in the range [0, 1] with a Min-Max scaler.

3.1 MISO neural network results

In this section, we present and analyse the performance of the MISO data-driven models by highlighting both the best and worst results obtained from the prediction of flight manoeuvers. Table 5 details the training setup of the two MISO data-driven models tuned through an extensive campaign of training and tests to determine the most suitable network structures, while Table 6 summarises the overall $rMAE$ indices for the training phase of each model. The best FFNN model consists of four

https://doi.org/10.1017/aer.2025.10024 Published online by Cambridge University Press

Table 5. *MISO models training setup*

Hyperparameter	FFNN Model	LSTM Model
Activation function	ReLU	ReLU
Optimiser	RMSprop	ADAM
Loss Function	*MSE*	*MSE*
Weights initialisation	Xavier Init	–
Batch size	64	64
N° epochs	500	100
Learning rate	$1x10^{-4}$	$5x10^{-4}$
Lookback	–	20

Table 6. *Overall Training mean relative MAE per model on the Validation set*

Model	Overall Mean *rMAE*
MISO FFNN	0.0361
MISO LSTM	0.0335

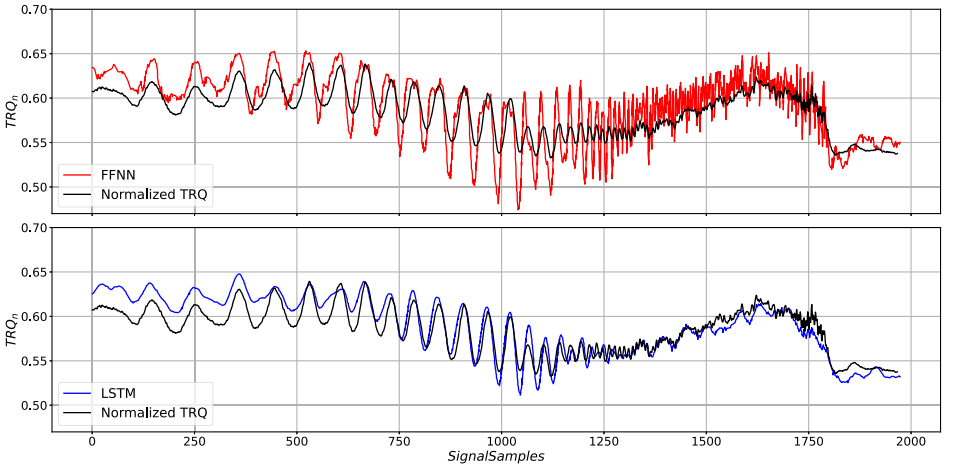


Figure 6. *Comparison of FFNN and LSTM predictions of normalised TRQ for a manoeuver of Flight ID1 in the D_{test} dataset.*

hidden layers of 24 neurons each and the best LSTM model encodes three recurrent layers and a hidden size equal to 6. More details on the hyperparameter and network structure choices are given in Appendix 4.

Figures 6 and 7 present results obtained for two different manoeuvers of two different flights from the test set. In particular, Fig. 6 refers to flight ID1 and compares the *TRQ* predictions of FFNN ($rMAE_{FFNN} = 3.75\%$) and LSTM ($rMAE_{LSTM} = 2.34\%$) with the normalised actual *TRQ*. Although in terms of *rMAE* per manoeuver the results obtained by the two MISO models may seem comparable, it is evident from the graph that FFNN is unable to capture the engine dynamics. In fact, its weakness emerges especially for manoeuvers (such as the collective sweep) in which the dynamic response of the engine is intentionally excited. A second problem is the lack of smoothness of the reconstructed *TRQ* time series, which has a general 'noisy' and 'spiky' behaviour. The lack of smoothness is a direct consequence of network structure and architecture, where each time instant is treated in isolation without

https://doi.org/10.1017/aer.2025.10024 Published online by Cambridge University Press

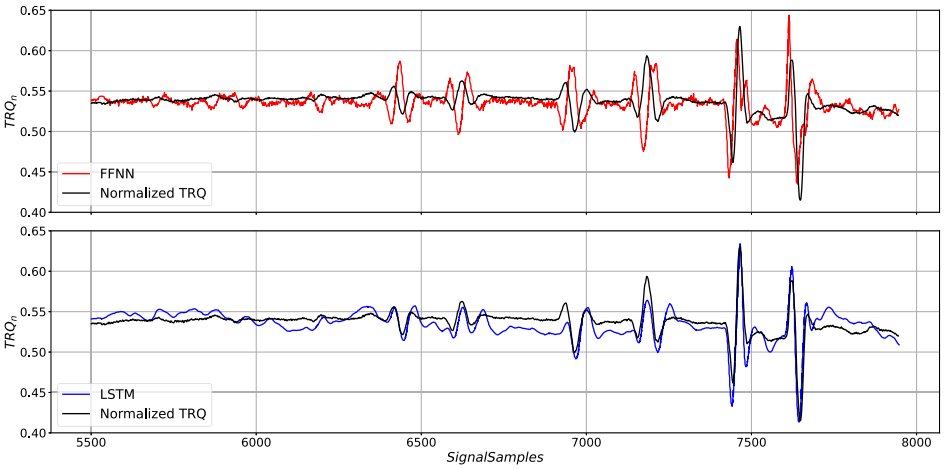


Figure 7. *Comparison of FFNN and LSTM predictions of normalised TRQ for a manoeuver of Flight ID6 in the D_{test} dataset.*

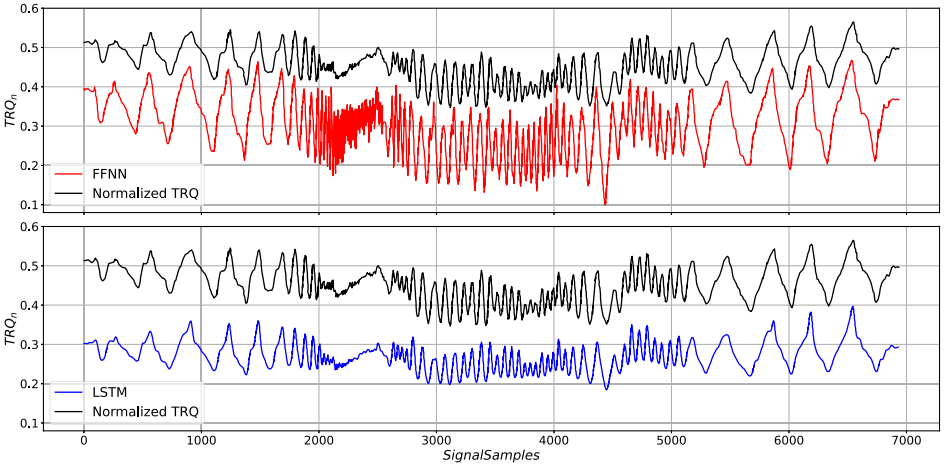


Figure 8. *Comparison of FFNN and LSTM predictions of normalised TRQ for a manoeuver of Flight ID11 in the D_{test} dataset.*

taking into account the time history of the input variables. On the contrary, the predictions of LSTMs fit the original signal better; their feedback connections allow them to handle the sequential nature of time series data, capturing long-term patterns and dependencies, which makes them ideal for predicting time series data.

Figure 7 is a second example of a comparison between the prediction results of MISO models and the normaliased real *TRQ*. Again, it is evident that the LSTM network ($rMAE_{LSTM} = 1.82\%$) provides a better prediction of the dynamic behaviour of the *TRQ* with respect to the FFNN ($rMAE_{FFNN} = 2.21\%$), which continues to produce a very noisy result and, more importantly, fails to capture sudden spikes in the measured *TRQ*.

Figures 8 and 9 show additional results for two different manoeuvers in the test set. The dynamic component appears to be well predicted, at least by the LSTM model, but in this case the predicted

https://doi.org/10.1017/aer.2025.10024 Published online by Cambridge University Press

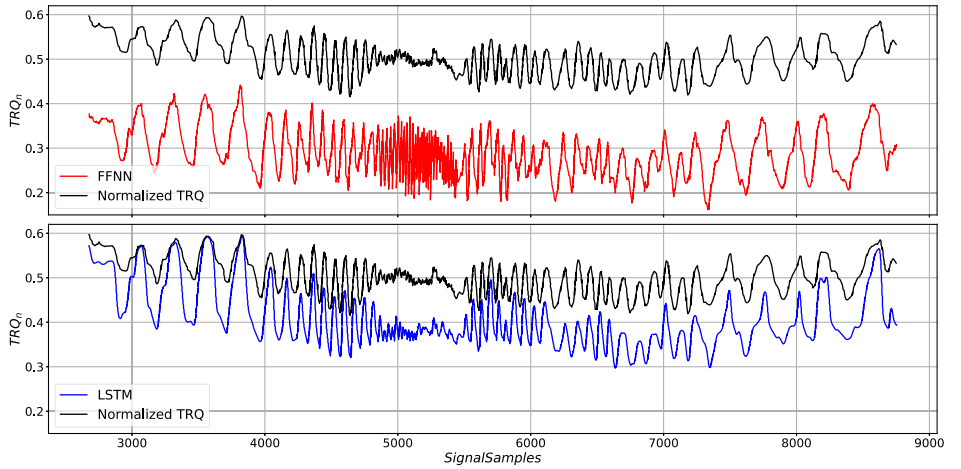


Figure 9. *Comparison of FFNN and LSTM predictions of normalised TRQ for a manoeuver of Flight ID10 in the D_{test} dataset.*

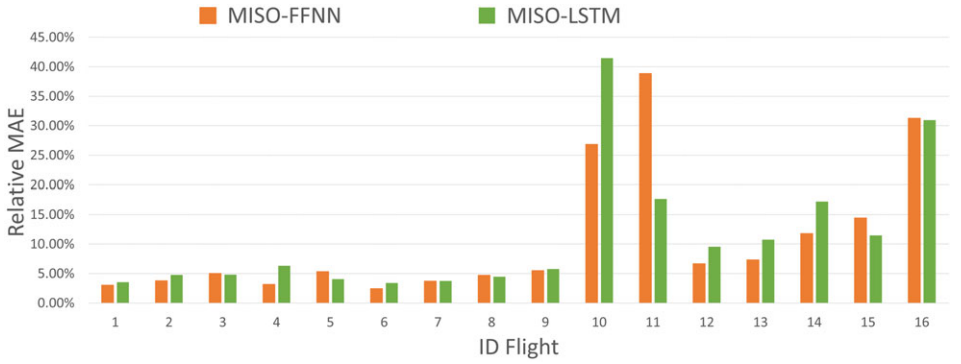


Figure 10. *MISO models rMAE score per single test flight.*

values seem to be consistently shifted by a constant. This suggests that the static component of the dynamics is not accurately captured.

An analysis of the *rMAE* scores per single test flight (Fig. 10) shows that, while for flights between ID1 and ID9 the $rMAE_j$ is acceptable and comparable to the training and testing scores, for the remaining flights (ID10 to ID15) the $rMAE_j$ is significantly larger, with $rMAE_j$ scores as high as 42.42%, because of a static offset of the predicted *TRQ*.

The causes of this behaviour can be related to errors and uncertainties in the collected flight data.

The data from flights ID1 to ID9 pertains to flight tests conducted in 2016, while flights ID10 to ID15 correspond to tests carried out in 2018. This two-year gap introduces potential changes in both the measurement system and the engine's performance, which could affect the accuracy and generalisability of the machine learning model trained exclusively on the 2016 data. Over this period, it is possible that onboard sensors, including those used to measure torque or some input variables, may have undergone maintenance, recalibration, or even replacement, which could result in discrepancies in the data collected in 2018 compared to 2016. Furthermore, the engine may have

https://doi.org/10.1017/aer.2025.10024 Published online by Cambridge University Press

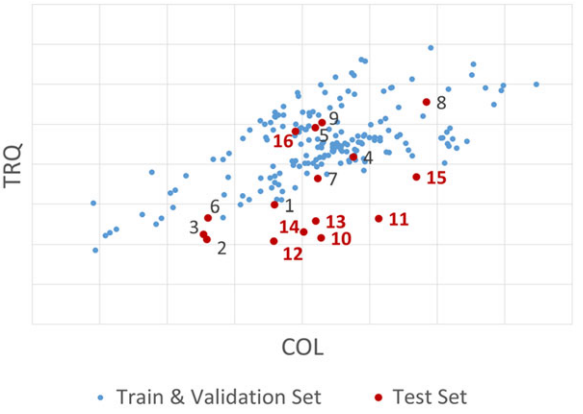


Figure 11. *Scatter plot of average COL and TRQ values for training/validation and test datasets. Red labels highlight the ID-FLIGHTS in the test set with hight rMAE score.*

experienced wear and tear, leading to alterations in its performance characteristics, including its torque response.

As components age and accumulate flight hours, their ability to perform at peak efficiency can degrade, leading to subtle but significant changes in the engine's behaviour during flight tests. Moreover, not only the engine, but also the helicopter's overall setup and other components, such as the rotor blades or transmission system, could have been replaced or modified over time. These changes in the airframe or mechanical systems could further affect the torque output and the overall performance of the helicopter. Unfortunately, we are unaware of the specific details regarding any replacements or modifications to the helicopter's components between 2016 and 2018. This uncertainty complicates the model's ability to account for all potential variations in the system, further emphasising the need for caution when applying the model to newer data. As a result, the machine learning model, which has been trained on data from a less-worn engine and potentially different components, might struggle to accurately predict torque values from a more aged system or modified helicopter. This issue highlights the importance of considering the evolving nature of the system when developing predictive models, as the model's performance may degrade if it fails to account for changes in the instrumentation and the engine's operational characteristics over time. Therefore, regular updates to the model, incorporating new data that reflects the current state of the engine and its components, are crucial to maintaining its accuracy and robustness.

Another potential cause for reduced prediction accuracy could be the absence of essential variables that are not included in the input feature set. For instance, the helicopter's climb rate is a critical feature that directly impacts the torque response. Even at the same Collective Pitch (*COL*), a higher climb rate generally results in a higher torque requirement due to the increased engine power demand on the engine as the helicopter ascends. The torque response can vary significantly depending on the rate of climb, as different flight profiles (such as steady, climbing or descending flights) impose distinct aerodynamic and mechanical stresses on the engine and rotor system.

All these hypotheses are supported by the relationship between the average *COL* and average *TRQ* per flight (Fig. 11): a discrepancy may be observed between the distribution of the training dataset and one of the test dataset. In fact, the majority of flights with the highest relative *MAEs* do not fall within the distribution of training flights. Hence, neural networks struggle to accurately predict *TRQ* because the training data do not cover these behaviours. To achieve greater accuracy in *TRQ* prediction on D_{est} flights, a new training phase of MISO models is performed by including 11 D_{test}

https://doi.org/10.1017/aer.2025.10024 Published online by Cambridge University Press

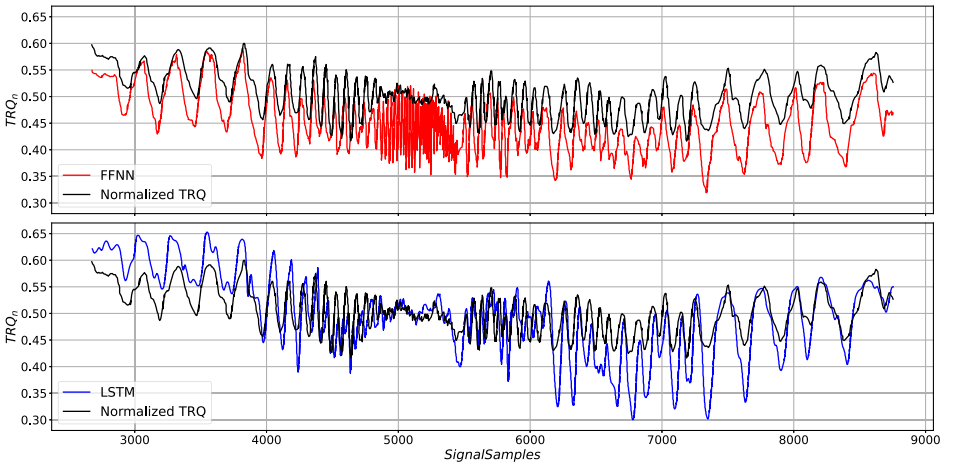


Figure 12. *Comparison of FFNN and LSTM predictions of normalised TRQ for a manoeuver of Flight ID10 in the D_{test} dataset after MISO models re-training.*

flights in D_{train}. The 11 test flights added to D_{train} include both flights with ID1 to ID9 (for which the MISO models already work quite well) and flights with ID10 to ID15, which are not part of the initial training set.

By incorporating the test flights into the training set, MISO model performance improves; in fact, the scores obtained in the test phase after models re-training phase go from a $rMAE_{FFNN} = 13.50\%$ to a $rMAE_{FFNN} = 5.78\%$ and from a $rMAE_{LSTM} = 8.32\%$ to a $rMAE_{LSTM} = 4.06\%$.

The performance improvements of the two MISO models are even more evident when looking at Fig. 12, which shows the same ID10 flight manoeuver as in Fig. 9. Even though the predictions are now mostly superimposed to the normalised real *TRQ*, the FFNN model still fails in properly predicting high-frequency and small amplitude *TRQ* oscillations for sweep flights.

3.2 SINDy Results

In this section, we present the results of the SINDy model training, which aims to identify the underlying dynamic relationships from the dataset. The following relation was obtained after selecting a polynomial candidate functions library:

$$\dot{TRQ}\,(t) = -a - b\,TRQ\,(t) + c\,WF\,(t) \tag{9}$$

where a, b and c are all positive coefficients and t is time. In fact, this result is not unexpected. In standard transfer-function modeling approaches, the *WF*-*TRQ* relation is modeled by a transfer function as follows:

$$\frac{TRQ\,(s)}{WF\,(s)} = \frac{\mu}{1+\tau s} \tag{10}$$

which may be easily reconducted to a first-order ODE in time domain. Even though this result shows how SINDy is able to correctly recover the system dynamics from data without previous knowledge of the underlying physics, the first order Equation (9) is not so useful in terms of accuracy, as intuitively shown by the overall mean *rMAE*, reported in Table 5. The simulations are obtained for each manoeuver in D_{test} by using as initial conditions the value of *TRQ* at the beginning of the manoeuver itself, and then integrating in time via the time integration algorithm provided

https://doi.org/10.1017/aer.2025.10024 Published online by Cambridge University Press

Table 7. *Overall Training mean relative MAE per SINDy model*

Model	Overall Mean *rMAE*
SISO SINDy first-order	0.137
SISO SINDy second-order	0.016

by PySINDy package. A step further is obtained by forcing a second-order model for SINDy prediction. The strategy is to modify the state and input vectors by including the time derivatives of *TRQ* and *WF*:

$$\mathbf{X_i} = \begin{pmatrix} TRQ_{i1} & TRQ_{i2} & TRQ_{i3} & \dots & TRQ_{im_i} \\ \dot{T}RQ_{i1} & \dot{T}RQ_{i2} & \dot{T}RQ_{i3} & \dots & \dot{T}RQ_{im_i} \end{pmatrix} \tag{11}$$

$$\mathbf{U_i} = \begin{pmatrix} WF_{i1} & WF_{i2} & WF_{i3} & \dots & WF_{im_i} \\ \dot{W}F_{i1} & \dot{W}F_{i2} & \dot{W}F_{i3} & \dots & \dot{W}F_{im_i} \end{pmatrix} \tag{12}$$

so as to look for a system of equations instead.

In this case, the time-derivatives of the state and input signals are evaluated by numerical differentiation and are used to train SINDy model, leading to the following result

$$\begin{cases} \dot{T}RQ\,(t) = a'\dot{T}RQ \\ \ddot{T}RQ\,(t) = b'\dot{T}RQ\,(t) + c'\dot{W}F\,(t) \end{cases} \tag{13}$$

where the first is clearly a dummy equation with $a' = 1$, which emerges naturally from the SINDy algorithm, while b' and c' are positive coefficients. It is interesting to point out that the dummy equation involving the coefficient $a' = 1$ results directly from the algorithm, as the second-order model expands the state space by including the derivative of *TRQ*. Nevertheless, since this is a dummy equation, the value of a' produced by the algorithm serves as a meaningful indicator of the quality of the regression. If it differs from the expected value of 1, it would suggest that the model fails to accurately represent the system–even when reconstructing a simple identity.

By using the second-order SINDy model given by system 13 to simulate the *TRQ* time-histories of the test dataset, a significant improvement is obtained. As reported in Table 7, we obtain a reduction of one order of magnitude with respect to the first-order model in terms of overall mean relative *MAE*. In this case, an additional initial condition is required on the $\dot{T}RQ$, while the new input variable is the time-derivative of *WF*.

At first glance, the very large difference between the two SINDy models may appear counterintuitive since the second equation of system 13 is easily traced back to a first-order model by integrating in time. Actually, we obtain

$$\int \ddot{T}RQ\,(t) = \int -b'\dot{T}RQ\,(t)\,dt + \int c'\dot{W}F\,(t)\,dt \tag{14}$$

leading to

$$\dot{T}RQ\,(t) = -b'TRQ\,(t) + c'WF() + const \tag{15}$$

which is the same expression as Equation (9). However, the coefficients resulting from the two different training processes are quite different, and the results provided by the second-order SINDy model are sufficiently accurate even for D_{val}.

https://doi.org/10.1017/aer.2025.10024 Published online by Cambridge University Press

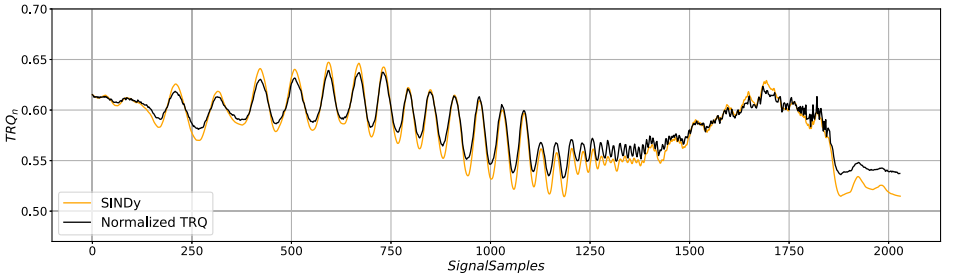


Figure 13. *SINDy second-order prediction of normalised TRQ for a manoeuver of Flight ID1 in the D_{test} dataset.*

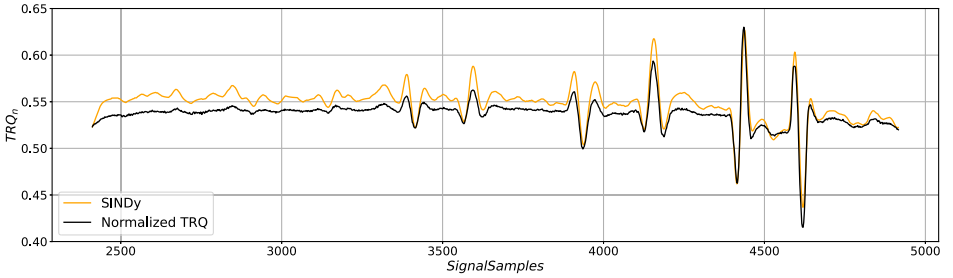


Figure 14. *SINDy second-order prediction of normalised TRQ for a manoeuver of Flight ID6 in the D_{test} dataset.*

Before proceeding with the discussion of the result for the validation dataset, it should be noted that the SINDy model differs in a very important aspect from the FFNN and LSTM models presented in the previous paragraphs. The SINDy model takes only an internal variable of the system as input, while ignoring the real pilot control variables, namely collective (*COL*) and main rotor speed (*NR*). It follows that a direct comparison of SINDy 2^{nd} order scores with those of FFNN and LSTM may be misleading, as Equation (15) pertains only to a single engine component, providing a direct relation between *WF* and *TRQ*. Still, it is interesting to discuss the following results to demonstrate SINDy's potential to extrapolate accurate physical models from normal flight data.

Figures 13 through 16 present the same flight manoeuvers discussed in subsection 3.1 for the MISO neural networks. It is clear that the simulation results obtained from Equation (13) are highly accurate, reproducing *TRQ* dynamics very well with only slight under/overestimations of some oscillation peaks. Again, an estimate of the total error is given by the rMAE score per test flight, which in this case is always lower than 3%. As anticipated above, these results demonstrate the potential of the SINDy approach in extrapolating a simple and interpretable dynamic model even without a deep understanding of the underlying physics of the system. However, in this case, the transition to the second order is crucial to achieve more than satisfactory accuracy. This behaviour, in our experience, is not easily predictable, as *WF*-*TRQ* relationship is typically modeled using transfer functions associated to firstorder ODE (see Equation (10)). It is likely that the information carried by the temporal derivative of fuel flow, the new control variable for the system of Equation (13), together with the additional initial condition on $\dot{TRQ}$, may constitute the key for the improvement achieved in terms of model performance.

https://doi.org/10.1017/aer.2025.10024 Published online by Cambridge University Press

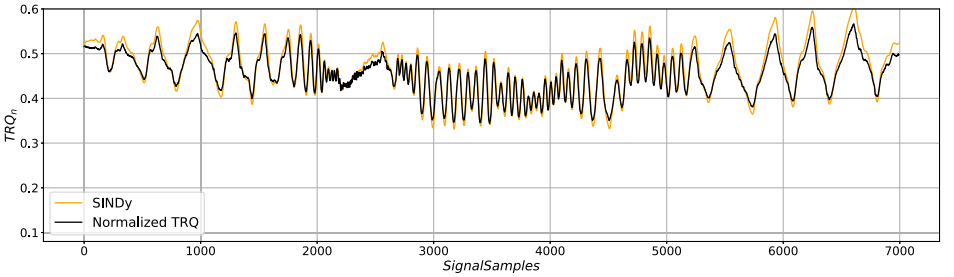


Figure 15. *SINDy second=order prediction of normalised TRQ for a manoeuver of Flight ID11 in the D_{test} dataset.*

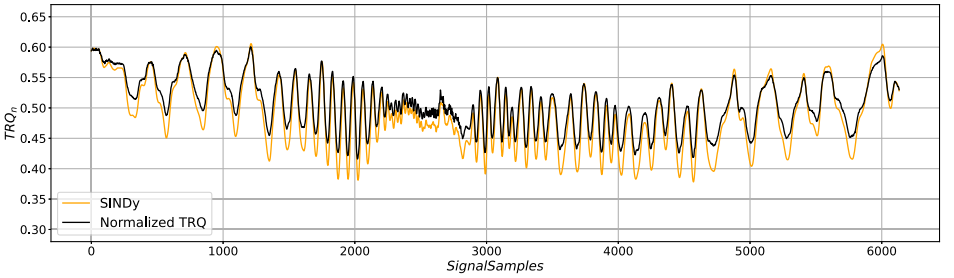


Figure 16. *SINDy second-order prediction of normalsed TRQ for a manoeuver of Flight ID10 in the D_{test} dataset.*

4.0 Conclusions

In this paper, three different supervised data-driven approaches have been presented with the goal of developing a dynamic turboshaft engine model to extend the scope of commonly used transfer function-based and state-space perturbation models. The developed data-driven engine models have been trained, validated and tested on a database of real-flight tests of Leonardo's AW189 prototypes and accurately predict the engine torque over the entire flight envelope of the helicopter, demonstrating that neural networks may effectively be used to simulate nonlinear behaviours of a helicopter engine. Specifically, two different MISO neural network architectures have been developed, namely a FFNN and a LSTM network, which take as input the time-histories of engine and environmental variables to predict the desired engine torque. The results show that the FFNN's predictions lack accuracy and feature strong oscillations that are unacceptable for engine control. On the other hand, the LSTM, being a recurrent network, yields better predictions of *TRQ* dynamic behaviour since its output is based on a time window that improves performance in terms of signal smoothness. The LSTM results are promising, but as shown by the analysis on the test dataset, which includes flights conducted long after the original training flights, the trained model lacks sufficient flexibility and fails to maintain satisfactory performance. As a consequence, a new training phase is necessary to update the model in response to the aircraft's aging process. As future work, it would be of interest to further expand the dataset by including different aircraft types and flight campaigns, in order to evaluate whether more complex network architectures could generalise more effectively, thus leveraging the predictive capability demonstrated on the training and validation sets.

https://doi.org/10.1017/aer.2025.10024 Published online by Cambridge University Press

The predictive capability of purely data-driven neural network models, however, comes at a cost. Unlike physics-based approaches, they often act as 'black boxes', making it difficult to understand the underlying decision-making processes. This lack of transparency complicates diagnostics and compromises reliability, especially in safety-critical contexts. As a result, the trade-off between accuracy and interpretability remains a key issue in the adoption of such models, particularly in aerospace where explainability is essential for certification and trustworthiness. For this reason, with the intent of developing an interpretable ML dynamical model, we have used the SINDy approach to derive an accurate interpretable representation of the system dynamics directly from flight data even in the absence of a prior deep understanding of the underlying physics of the system. The obtained results, although preliminary as they focus on a specific engine component, show that the SISO SINDy model can effectively reproduce the *TRQ* dynamics, but only when a second-order model is explicitly enforced within the SINDy algorithm. In this configuration, the SINDy model proves to be significantly more flexible than the purely data-driven neural network models analysed in this study, and the results demonstrate that its predictions remain highly accurate even on the test dataset, including sweep manoeuvers. Considering the encouraging results obtained with the SINDy model, future developments of this work will aim at extending the methodology to derive a complete engine model capable of predicting torque directly from real pilot control inputs.

Competing interests. The authors declare that they have no competing interests in relation to the work described in this manuscript.

References

[1] Corliss, D. *A helicopter handling-qualities study of the effects of engine response characteristics height-control dynamics and excess poweron nap-of-the-earth operations,* Tech. rep. U.S. Army Aeromechanics Laboratory, NASA Ames Research Center, 1982.

[2] Mihaloew, D.J., Ballin, M.G. and Ruttledge, G. *Rotorcraft flight-propulsion control integration: An eclectic design concept*, Technical Report 2815, NASA, 1988.

[3] Fuksman, I. and Sirica, S. Real-time execution of a high fidelity aero- thermodynamic turbofan engine simulation, *J Eng Gas Turbines Power*, 2012, **134**, (5), p 054501. issn: 0742-4795. https://asmedigitalcollection.asme.org/gasturbinespower/article-pdf/134/5/054501/5894871/054501_1.pdf. URL: https://doi.org/10.1115/1.4004856

[4] Volponi, A.J. Gas Turbine parameter corrections, *J Eng Gas Turbines Power*, 1999, **121**, (4), pp 613–621. https://doi.org/10.1115/1.2818516

[5] Ballin, M.G. *A high fidelity real-time simulation of a small turboshaft engine*, Tech. rep. NTRS Research Center, 1988.

[6] Yazar, I. Simulation of a high fidelity turboshaft engine-alternator model for turboelectric propulsion system design and applications, *Int J Turbo Jet-Engines*, 2018, **36**, (3), pp 271–281. issn: 0334-0082, 2191-0332. https://doi.org/10.1515/tjj-2018-0036

[7] Zheng, Q., et al. A turboshaft engine NMPC scheme for helicopter autorotation recovery Maneuver, Aerosp Sci Technol, 2018, 76, pp 421–432. issn: 12709638. https://doi.org/10.1016/j.ast.2018.01.034

[8] Mao, H., et al. Versatile simulation platform for turboshaft engine control system, *2019 Chinese Control Conference (CCC),* 2019, pp 7211–7216. https://doi.org/10.23919/ChiCC.2019.8865902

[9] Duyar, A., Gu, Z. and Litt, J.S. A simplified dynamic model of the T700 turboshaft engine, *J Am Helicopter Soc.*, 1995, **40**, (4), pp 62–70.

[10] Litt, J.S. *A real-time simulator of a turbofan engine*. Vol. **100869**. National Aeronautics and Space Administration Technical Memorandum (TM), Lewis Research Center, Cleveland, OH, 1989.

[11] Merrill, W.C., et al. *HYTESS: A hypothetical turbofan engine simplified simulation*. Tech. rep. 1984.

[12] Duyar, A., et al. A simplified dynamic model of the space shuttle main engine, *J Dyn Sys, Meas, Control*, 1994, **116**, (4), pp 815–819. https://doi.org/10.1115/1.2899286

[13] Castiglione, T., et al. Linear model of a turboshaft aero-engine including components degradation for control-oriented applications, *Energies*, 2023, **16**, (6), p 2634. issn: 1996-1073. https://doi.org/10.3390/en16062634

[14] Xie, R., et al. Digital twin technologies for turbomachinery in a life cycle perspective: A review, *Sustainability*, 2021, **13**, (5), p 2495.

[15] Asgari, H., Ory, E., and Lappalainen, J. Recurrent neural network based simulation of a single shaft gas turbine, *Proceedings of the 61st SIMS Conference on Simulation and Modelling SIMS*. Finland Linko¨ping Electronic Conference Proceedings, 2020, pp 22–24.

https://doi.org/10.1017/aer.2025.10024 Published online by Cambridge University Press

[16] De Giorgi, M.G. and Quarta, M. Hybrid multigene genetic programming-artificial neural networks approach for dynamic performance prediction of an aeroengine, *Aerosp Sci Technol.*, 2020, **103**, p 105902.

[17] Zheng, Q., et al. Aero-engine on-board dynamic adaptive MGD neural network model within a large flight envelope, *IEEE Access*, 2018, **6**, pp 45755–45761.

[18] Kiakojoori, S. and Khorasani, K. Dynamic neural networks for gas turbine engine degradation prediction, health monitoring and prognosis, *Neural Comput Appl.*, 2016, **27**, (8), pp 2157–2192.

[19] Wang, W., et al. Helicopter dynamic modeling and system development for flight simulation, *2021 IEEE 3rd International Conference on Civil Aviation Safety and Information Technology (ICCASIT),* 2021, pp 1220–1224. https://doi.org/10.1109/ICCASIT53235.2021. 9633434.

[20] Gu, Z., et al. An online data-driven LPV modeling method for turbo-shaft engines. *Energies*, 2022, **15**, (4), 1255. issn: 1996-1073. https://doi.org/10.3390/en15041255

[21] Badihi, H., Shahriari, A. and Naghsh, A. Artificial neural network application to fuel flow function for demanded jet engine performance, *2009 IEEE Aerospace Conference,* IEEE, 2009, pp 1–7.

[22] Corsini, A., et al. Cascade with sinusoidal leading edges: Identification and quantification of deflection with unsupervised machine learning, *Proceedings of the ASME Turbo Expo 2021, Vol. 1,* ASME, 2021, pp 1–10.

[23] Tieghi, L., et al. A machine-learnt wall function for rotating diffusers, *J Turbomach.*, 2021, **143**, (8), p 081012.

[24] Tayarani-Bathaie, S.S, Vanini, Z.S. and Khorasani, K. Dynamic neural network-based fault diagnosis of gas turbine engines, *Neurocomputing*, 2014, **125**, pp 153–165.

[25] Vatani, A., Khorasani, K. and Meskin, N. Health monitoring and degradation prognostics in gas turbine engines using dynamic neural networks, *Turbo Expo: Power for Land, Sea, and Air,* Vol. 56758, American Society of Mechanical Engineers, 2015, V006T05A030.

[26] Brunton, S.L., et al. Data-driven aerospace engineering: reframing the industry with machine learning, *Aiaa J.*, 2021, 59, (8), pp 2820–2847.

[27] Schmid, P.J. Dynamic mode decomposition of numerical and experimental data, *J Fluid Mech.*, 2010, **656**, pp 5–28.

[28] Proctor, J.L., Brunton, S.L. and Kutz, J.N. Dynamic mode decomposition with control, *SIAM J Appl Dyn Syst.*, 2016, **15**, (1), pp 142–161.

[29] Williams, M.O, Kevrekidis, I.G. and Rowley, C.W. A data–driven approximation of the koopman operator: Extending dynamic mode decomposition, *J Nonlinear Sci.*, 2015, **25**, pp 1307–1346.

[30] Brunton, S.L., Proctor, J.L. and Kutz, J.N. Discovering governing equations from data by sparse identification of nonlinear dynamical systems, *Proc Natl Acad Sci.*, 2016, 113, (15), pp 3932–3937.

[31] Liu, Y., et al. Data-based modeling and control of the nonlinear aircraft system using extended implicit sparse identification, *IEEE Trans Aerosp Electron Syst.*, 2025.

[32] Sorokina, M., Sygletos, S. and Turitsyn, S. Sparse identification for nonlinear optical communication systems: SINO method, *Opt Express*, 2016, 24, (26), pp 30433–30443.

[33] Dam, M., et al. Sparse identification of a predator-prey system from simulation data of a convection model, *Phys Plasmas*, 2017, **24**, (2).

[34] Narasingam, A. and Kwon, J.S.I. Data-driven identification of interpretable reduced-order models using sparse regression, *Comput Chem Eng.*, 2018, **119**, pp 101–111.

[35] Hoffmann, M., Fro¨hner, C. and Noe,́F. Reactive SINDy: Discovering governing reactions from concentration data, *J Chem Phys.*, 2019, **150**, (2), p 025101. https://doi.org/10.1063/1.5066099. PMID: 30646700.

[36] Lai, Z. and Nagarajaiah, S. Sparse structural system identification method for nonlinear dynamic systems with hysteresis/inelastic behavior, *Mech Syst Signal Process.*, 2019, **117**, pp 813–842.

[37] Loiseau, J.C., Noack, B.R. and Brunton, S.L. Sparse reduced-order modelling: Sensor-based dynamics to full-state estimation, *J Fluid Mech.*, 2018, **844**, pp 459–490.

[38] Elsayed, E., El-Badawy, A. and Zometa, P. Data-driven discovery of the equations of motion of twin-tailed fighter aircraft, *2024 28th International Conference on System Theory, Control and Computing (ICSTCC),* IEEE, 2024, pp 362–367.

[39] Mukhachev, P., et al. Evaluation of ml algorithms for system dynamics identification of aircraft pressure control system, *PHM Society European Conference*, 2021, 6, (1), pp 7–7.

[40] Lee, J.D., et al. Sparse identification of nonlinear dynamics-based model predictive control for multirotor collision avoidance. *arXiv preprint arXiv:2412.06388*, 2024.

[41] Yahagi, S., et al. Sparse identification and nonlinear model predictive control for diesel engine air path system, *Int J Control Autom Syst.*, 2025, 23, (2), pp 620–629.

[42] Goodfellow, I., et al. *Deep learning*, vol. 1, No. 2, MIT Press, 2016, Cambridge.

[43] Hochreiter, S. Long short-term memory, *Neural Computation MIT-Press,* 1997.

[44] Staudemeyer, R.C. and Morris, E.R. Understanding LSTM–a tutorial into long short-term memory recurrent neural networks, *arXiv preprint arXiv:1909.09586*, 2019.

[45] Paszke, A., et al. Pytorch: An imperative style, high-performance deep learning library, *Proc 33rd Int Conf Neural Inf Process Syst.*, 2019, **721**, p 12.

[46] LeCun, Y., Bengio, Y. and Hinton, G. Deep learning, *Nature*, 2015, **521**, (7553), pp 436–444.

[47] de Silva, B.M., et al. Pysindy: a python package for the sparse identification of nonlinear dynamics from data, *arXiv preprint arXiv:2004.08424*, 2020.

https://doi.org/10.1017/aer.2025.10024 Published online by Cambridge University Press

[48] Kaptanoglu, A.A., et al. PySINDy: A comprehensive Python package for robust sparse system identification, *arXiv preprint arXiv:2111.08481*, 2021.
[49] Brunton, S.L., Proctor, J.L. and Kutz, J.N. Sparse identification of nonlinear dynamics with control (SINDYc), *IFAC-PapersOnLine*, 2016, **49**, (18), pp 710–715.

Appendix

Hyperparameter Tuning and Training Setup

To determine the most suitable hyperparameters for the MISO data-driven models (e.g. number of layers, number of neurons, learning rate and batch size), we compared different network structures and training setups. Tables 8 and 9 present the hyperparameter grids for the FFNN and LSTM models, respectively. The best training setup parameters identified during training are highlighted in **bold**.

With the aim of illustrating the performance of the investigated FFNN structures and providing insights into the selection of the best architecture, Fig. 17 shows a table reporting the values of the mean absolute error (MAE) and the relative mean absolute error (rMAE) for both the Training and Validation datasets, along with the training time of each model, where the best architectures with the same number of nodes are highlighted with different colors. As expected, smaller networks show worse

Table 8. *FFNN hyperparameter grid*

Training setup	
Optimiser	**RMSprop**, ADAM
Loss function	**MSE** (Mean Square Error)
Weights init.	None, **Xavier Init.**
Batch size	32, **64**, 128
N° epochs	100, **500**
Learning Rate	$1 \cdot 10^{-3}$, $\mathbf{1 \cdot 10^{-4}}$, $5 \cdot 10^{-4}$
Network Structure	
Activation function	ReLU (Rectified Linear Unit)
Hidden layers	2, 4, 6, 8, 10
N. of nodes	8, 16, 24, 32

Table 9. *LSTM hyperparameter grid*

Training setup	
Optimiser	RMSprop, **ADAM**
Loss function	**MSE** (mean square error)
Weights init.	None, **Xavier Init.**
Batch size	32, **64**, 128
N° epochs	**100**, 500
Learning rate	$1 \cdot 10^{-3}$, $1 \cdot 10^{-4}$, $\mathbf{5 \cdot 10^{-4}}$
Network Structure	
Activation function	ReLU (Rectified Linear Unit)
Recurrent layers	1, 2, 3, 4
Hidden size	2, 4, 6, 8, 10
Lookback	20, 40, 60, 80

https://doi.org/10.1017/aer.2025.10024 Published online by Cambridge University Press

FFNN Structure		TRAIN		VALIDATION		Training Time
hidden layers	nodes	MAE	rMAE	MAE	rMAE	
2	8	14.60	4.47%	13.16	3.99%	01:00
4	8	14.58	4.47%	13.76	4.16%	01:30
6	8	13.66	4.19%	13.15	3.98%	01:32
8	8	12.94	3.97%	12.22	3.70%	01:49
10	8	12.75	3.91%	12.57	3.80%	02:07
2	16	13.11	4.02%	13.01	3.94%	01:05
4	16	12.06	3.70%	12.18	3.69%	01:18
6	16	11.78	3.61%	12.17	3.68%	01:39
8	16	11.38	3.49%	12.13	3.67%	01:53
10	16	11.43	3.50%	12.18	3.69%	02:18
2	24	12.61	3.87%	12.15	3.68%	01:04
4	24	11.77	3.61%	11.93	3.61%	01:35
6	24	10.56	3.24%	12.90	3.91%	01:35
8	24	10.20	3.13%	12.17	3.68%	02:03
10	24	9.54	2.92%	12.54	3.80%	02:11
2	32	11.87	3.64%	12.18	3.69%	01:02
4	32	11.09	3.40%	12.90	3.90%	01:39
6	32	9.93	3.04%	12.53	3.79%	01:36
8	32	9.54	2.92%	12.95	3.92%	01:57
10	32	8.79	2.69%	12.99	3.93%	02:15

Figure 17. *Performance and training time of different FFNN structures. Training time is expressed in hours and minutes (hh:mm).*

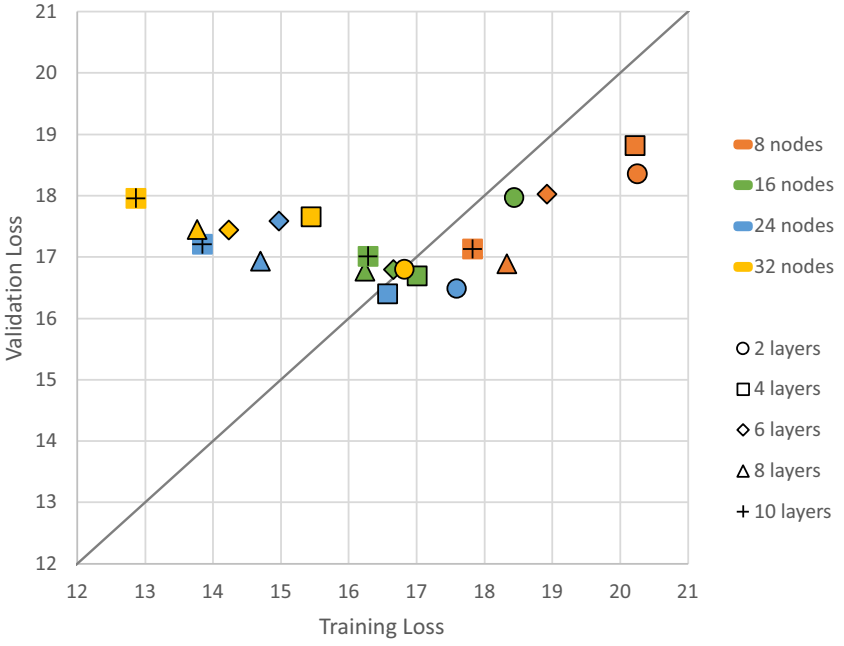


Figure 18. *Validation loss vs training loss for different FFNN structures.*

https://doi.org/10.1017/aer.2025.10024 Published online by Cambridge University Press

LSTM Structure		TRAIN		VALIDATION		
recurrent layers	hidden state	MAE	rMAE	MAE	rMAE	Training Time
1	2	14.88	4.56%	13.56	4.11%	01:34
1	4	12.79	3.91%	12.11	3.66%	01:36
1	6	11.43	3.50%	11.61	3.51%	01:41
1	8	10.94	3.35%	11.32	3.43%	01:47
1	10	10.62	3.25%	11.40	3.45%	01:55
2	2	14.10	4.32%	13.20	4.00%	02:47
2	4	11.52	3.53%	11.14	3.37%	02:51
2	6	11.08	3.39%	11.65	3.53%	02:59
2	8	9.98	3.06%	11.17	3.38%	03:11
2	10	9.30	2.85%	10.85	3.28%	03:20
3	2	13.40	4.10%	12.47	3.77%	04:03
3	4	11.71	3.59%	11.58	3.50%	04:09
3	6	10.37	3.18%	11.07	3.35%	04:18
3	8	9.46	2.90%	11.02	3.33%	04:35
3	10	8.86	2.71%	10.86	3.29%	04:43
4	2	13.35	4.09%	12.47	3.77%	05:11
4	4	11.31	3.46%	10.74	3.25%	05:30
4	6	10.17	3.11%	10.92	3.30%	05:38
4	8	9.20	2.82%	10.58	3.20%	05:56
4	10	8.41	2.58%	11.24	3.40%	06:14

Figure 19. *Performance and training time of different LSTM structures. Training time is expressed in hours and minutes (hh:mm).*

performance in terms of rMAE. For instance, the first two structures, namely 2 and 4 hidden layers with 8 nodes each, achieved an rMAE of 4.47% on the training set, which is significantly higher compared to larger networks. On the other hand, the largest model (10 hidden layers with 32 nodes each) is clearly affected by overfitting. In fact, although the smallest rMAE was obtained on the training set (2.69%), this was not the case for the validation set, where the model performed poorly with an rMAE of 3.93%. Again, this is an expected behaviour, as it is well known that larger machine learning models are often prone to overfitting issues. Figure 18 depicts this behaviour more clearly, reporting on the x and y axes the values of the loss function for the training set and the validation set, respectively. The best models should be the ones lying on the diagonal of the plane. The same analysis was done for the LSTM model. Figure 19 shows the performances of the different networks structure. First of all, looking at the last column, we notice that LSTM networks take much longer to train than feed-forward NN networks (Fig. 17). Second, in general LSTM networks obtain greater accuracy: in fact, while the relative MAE of feed-forward networks is on average around 3.8%, here the relative MAE of LSTM networks is on average equal to 3.5%. The best performing networks are the ones highlighted in colors, which have a relative MAE of about 3.3% for the training set around and about 3.4% for the test set. The scatter plot of the loss functions in Fig. 20 also confirms that LSTM models have greater accuracy than feed-forward networks. In fact, the points of the different networks in the chart are more aligned along the diagonal, showing less tendency of overfitting and lower loss values. The network that has the highest accuracy is the one characterised by four recurrent layers and a hidden size equal to six, with an rMAE equal to 3.30% on the validation set. However, the performance increase of this network structure compared to

https://doi.org/10.1017/aer.2025.10024 Published online by Cambridge University Press

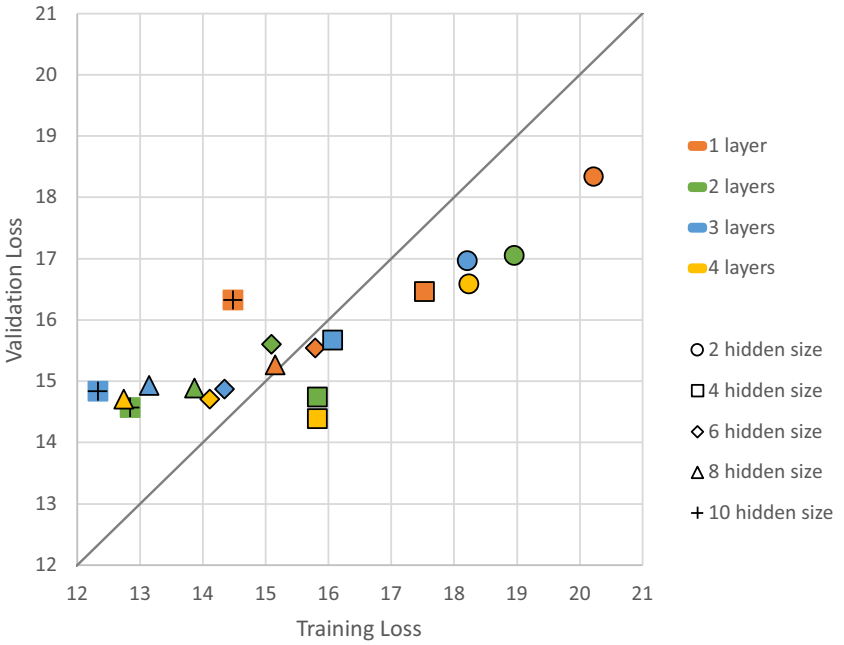


Figure 20. *Validation loss vs training loss for different LSTM structures.*

the one with three recurrent layers and six as hidden size is only 1.5%, while the computational cost increases by more than 30%. For this reason, the latter architecture was considered the optimal one for the study presented in this work.

Cite this article: Paniccia D., Tucci F.A., Guerrero J., Capone L., Sanguini N., Benacchio T. and Bottasso L. (2025). A supervised machine-learning approach for turboshaft engine dynamic modeling under real flight conditions. *The Aeronautical Journal*, **129**, 2777–2801. https://doi.org/10.1017/aer.2025.10024

https://doi.org/10.1017/aer.2025.10024 Published online by Cambridge University Press