Reliability Engineering and System Safety 217 (2022) 107961

Contents lists available at ScienceDirect

Reliability Engineering and System Safety

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

Fusing physics-based and deep learning models for prognostics

Manuel Arias Chao[a], Chetan Kulkarni[b], Kai Goebel[c], Olga Fink[a,*]

[a] *Chair of Intelligent Maintenance Systems, ETH Zurich, Switzerland*
[b] *KBR, Inc., NASA Ames Research Center, USA*
[c] *Luleå University of Technology, Operation and Maintenance Engineering, Luleå, Sweden*

ARTICLE INFO

Keywords:
Prognostics
Deep learning
Hybrid model
CMAPSS

ABSTRACT

Physics-based and data-driven models for remaining useful lifetime (RUL) prediction typically suffer from two major challenges that limit their applicability to complex real-world domains: (1) the incompleteness of physics-based models and (2) the limited representativeness of the training dataset for data-driven models. Combining the advantages of these two approaches while overcoming some of their limitations, we propose a novel hybrid framework for fusing the information from physics-based performance models with deep learning algorithms for prognostics of complex safety-critical systems. In the proposed framework, we use physics-based performance models to infer unobservable model parameters related to a system's components health by solving a calibration problem. These parameters are subsequently combined with sensor readings and used as input to a deep neural network, thereby generating a data-driven prognostics model with physics-augmented features. The performance of the hybrid framework is evaluated on an extensive case study comprising run-to-failure degradation trajectories from a fleet of nine turbofan engines under real flight conditions. The experimental results show that the hybrid framework outperforms purely data-driven approaches by extending the prediction horizon by nearly 127%. Furthermore, it requires less training data and is less sensitive to the limited representativeness of the dataset as compared to purely data-driven approaches. Furthermore, we demonstrated the feasibility of the proposed framework on the original CMAPSS dataset, thereby confirming its superior performance.

1. Introduction

The prediction of the failure time of complex systems has been successfully addressed on the basis of models that capture the physics of failure. Despite significant progress using physics-based models for prognostics [1–3], physical degradation processes are only well understood for critical or relatively simple components. As a result, the widespread deployment of physics-based models in practical applications has been limited.

The increased availability of system condition monitoring data has facilitated the broader use of data-driven approaches for prognostics and health management (PHM) of complex engineered systems. The underlying assumption of data-driven approaches is that the relevant information concerning the evolution of the system health and the failure time can be learned from past data [4]. In particular, deep learning has recently gained attention due to its ability to learn fault patterns directly from raw sensor data [5]. A variety of supervised and semi-supervised deep learning models have shown promise in estimating the remaining useful life (RUL) from sensor data [6–10] based on prognostics benchmark datasets, e.g., [11]. Most research studies on deep learning applications in PHM require a representative dataset of run-to-failure degradation trajectories to obtain accurate prognostics models. These trajectories must comprise a set of time series sensor readings along with the corresponding time-to-failure labels. However, the collection of such a representative dataset for systems subjected to periodic maintenance interventions can take a long time because (a) failures may be rare, (b) the system can operate in different environments and follow different mission profiles, thus resulting in a large range of possible deterioration trajectories, and (c) maintenance is performed before failure. In real application scenarios, the available datasets generally contain only a small number of units and failure modes and are, therefore, not fully representative of all potential future degraded system conditions [12,13]. Moreover, for complex engineered systems subject to continuous and increasing degradation, as well as substantial variability in operating conditions, data-driven approaches struggle to distinguish between the impact of changes in operating conditions and the impact of degradation on the sensor readings in scenarios of limited labeled data [13,14]. Consequently, data-driven

* Corresponding author.
E-mail addresses: manuel.arias@ethz.ch (M. Arias Chao), chetan.s.kulkarni@nasa.gov (C. Kulkarni), kai.goebel@ltu.se (K. Goebel), ofink@ethz.ch (O. Fink).

https://doi.org/10.1016/j.ress.2021.107961
Received 26 October 2020; Received in revised form 25 July 2021; Accepted 16 August 2021
Available online 17 September 2021

methods have difficulties relating the condition monitoring data to the asset failure time, limiting their practical application.

Although data-driven and physics-based methods have limitations when applied in isolation, it is hypothesized here that the combined use of both approaches can potentially lead to performance gains by leveraging the advantages of each. In particular, while physics-based approaches are generally hindered by their limited ability to properly tune the parameters of models with high complexity or model incompleteness, they do not require large amounts of data, retain the interpretability of a model, and provide the opportunity to generate synthetic data. In contrast, data-driven approaches are limited by the representativeness of the training datasets but are simple to implement, and one can use data-driven models to discover complex patterns from large volumes of data. It follows that data-driven solutions can be advantageous in enhancing or replacing inaccurate parts of physics-based models. In addition, physics-based model information can help to reduce the required amount of training data (i.e., overcome the lack of time-to-failure trajectories) by generating synthetic data or providing model parameters that are very informative for data-driven models. In general, the physics-based system models can be used as 'teachers' to guide the discovery of meaningful machine learning models.

A number of approaches have been proposed to combine physics-based and data-driven approaches. Depending on what type of information is processed and how the pieces of information are combined, different types of hybrid architectures have been suggested [15,16]. Some examples of recent hybrid models for prognostics are [17–20]. In particular, hybrid systems combining thermodynamic performance models and data-driven aging models have shown promising results on simpler systems such as lithium batteries [17]. Recently, several physics-guided machine learning approaches have been proposed, whereby physical principles are used to inform the search for a physically meaningful and accurate machine learning model. The architecture proposed in [21], for example, enhances the input space of a data-driven system model with outputs from a physics-based system model. The authors showed how, as a result, the dynamic behavior of the system could be approximated more accurately. In another variation of the physics-guided machine learning idea, a recurrent neural network (RNN) cell was modified to incorporate the information from the system model as an internal state of the RNN. A related idea was applied to a variety of prognostics problems, as in [18–20]. The underlying hypothesis of this method is that the output of the physics-based model is informative with regard to the degradation process and, consequently, the failure time.

In contrast to the hybrid architectures cited above, the framework presented in this paper leverages inferred **unobserved virtual sensors** and **unobservable parameters** of physics-based system models closely related to the system health in order to enhance the input space of deep learning-based prognostics models. For the physics-based system models, we focus on performance models (0D/1D models) that are generally available for the design, control, or performance evaluation of complex systems. Using as inputs the modeled system dynamics along with the sensor readings from the condition monitoring (CM) data, we solve a calibration problem in order to infer model parameters (e.g., efficiency and flow modifiers) and unobserved process properties (e.g., temperatures and pressures) that are informative regarding the health condition and its evolution over time. The model parameters and process properties (i.e., virtual sensors), as well as the calibrated system model responses, are subsequently combined with sensor readings and used as input to a deep neural network to generate a data-driven prognostics model. An overview of the proposed framework is shown in Fig. 1. A calibration-based hybrid framework was earlier applied to a diagnostics problem in [14]. In this paper, we notably extend the framework to address the problem of remaining useful life estimation.

The performance of the proposed hybrid framework is evaluated on a synthetic dataset comprising a small fleet of nine turbofan engines with run-to-failure degradation trajectories exhibiting a high

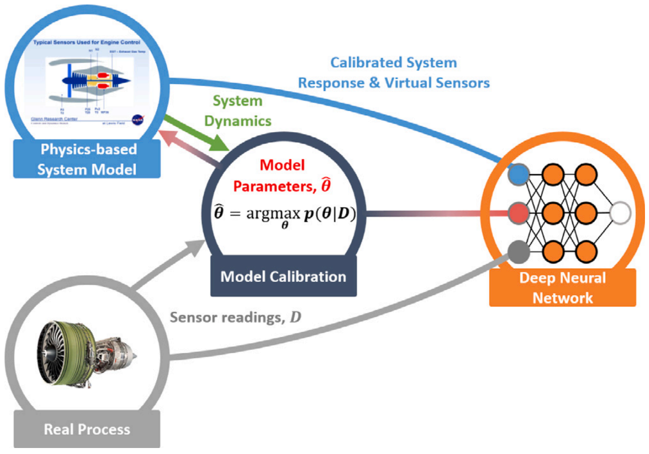

Physics-based System Model
Calibrated System Response & Virtual Sensors
System Dynamics
Model Parameters, $\hat{\theta}$
$\hat{\theta} = \operatorname{argmax}_{\theta} p(\theta|D)$
Model Calibration
Deep Neural Network
Sensor readings, D
Real Process

Fig. 1. Proposed hybrid prognostics framework fusing physics-based and deep learning models. Given the system dynamics and sensor readings, we perform the calibration of the system model to estimate unobservable model parameters $\hat{\theta}$ that encode the health condition of the system components. These parameters are subsequently combined with sensor readings from the condition monitoring (CM) data and calibrated system model responses, as well as unobserved process properties (i.e., virtual sensors), and used as input to a deep neural network to generate the deep learning-based prognostics model.

variability in operating conditions. A scenario reflecting incomplete representation of the test degradation conditions in the training dataset is considered. The dataset was generated with the Commercial Modular Aero-Propulsion System Simulation (CMAPSS) model [22]. Real flight conditions, as recorded on board commercial jets, were used as input to the CMAPSS model [23]. The performance of the approach is compared to an alternative data-driven approach whereby only sensor data is used as input to three types of deep neural networks (a multilayer perceptron feed-forward neural network, a recurrent neural network (RNN), and a convolutional neural network (CNN)). The proposed hybrid method outperforms the equivalent data-driven approaches and provides superior results in RUL estimation under highly variable operating conditions and incomplete representation of the training dataset. Furthermore, the hybrid framework requires less training data as compared to the purely data-driven algorithms.

The remainder of the paper is organized as follows. In Section 2, the background of the solution strategy is provided and the prognostics problem is formally introduced. In Section 3, the proposed framework is described. Section 4 introduces the case study. In Section 5, the results are presented. In Section 6, limitations and future research directions are discussed. Finally, a summary of the work and outlook are given in Section 7.

2. Background

This section briefly introduces the basic concepts and notation related to system performance models and calibration of physics-based models, as they are the building blocks of the proposed framework. In addition, it formally introduces the RUL estimation problem and lists the assumptions the proposed framework is built upon.

2.1. System performance models

The design of complex engineered systems involves modeling the physical principles governing system performance and the thorough validation of those models with field data. Hence, performance models with different fidelity levels are typically available for the control and performance evaluation of complex systems [24,25]. Some examples of system performance models are thermodynamic, electrical, or hydraulic 0D/1D models. These system models typically have a moderate computational load and are yet able to predict measured process variables (e.g., temperatures, pressures, or rotational speeds) as well as global unmeasured system and sub-system performance (e.g., efficiencies and power). Since, in the general case, there is no description given by an explicit formula, the performance models are represented mathematically as coupled systems of nonlinear equations. The inputs of the performance models are divided into scenario-descriptor operating conditions w and unobservable model parameters θ. The unobservable model parameters θ correspond to tuning parameters generally related to the health condition of the sub-components of the system. The outputs of the model are estimates of the measured physical properties $\hat{x}_s$ and unobserved properties $\hat{x}_v$ that are not part of the condition monitoring signals (i.e., *virtual sensors*). Hence, the nonlinear performance model is denoted as:

$$[\hat{x}_s, \hat{x}_v] = F(w, \theta) \tag{1}$$

Besides this compact formulation, system performance models of complex systems, such as the aero-thermodynamic performance model used in this work, often have the topology represented in Fig. 2. The performance model includes two coupled models: (1) a aero-thermodynamic model and (2) a controller. The latter contains the operation and protection logic and allows the simulation of the system response over a wide range of conditions (w) while satisfying the safety and operational limits. The former couples several stand-alone sub-component models. While the integration of the sub-component models resorts to well-understood physical principles (i.e., energy, momentum, and mass conservation), the sub-component models generally resort to a simplified physics (e.g., components maps and empirical correlations). As a result of these simplifications, the performance prediction accuracy of the system model worsens as the degradation of the system increases through operation over time, and the modeled physics of the sub-components does not account for changes in health-related process parameters such as efficiency and capacity loss (i.e., sub-component outputs). Therefore, the aero-thermodynamic model also includes model tuners θ that shift the sub-component outputs to compensate for their missing physics.

2.2. Calibration of physics-based models

Inference of system model parameters from observations x_s is often referred to as calibration [26]. System model calibration is an inverse problem aimed at obtaining the values of the model parameters θ that make the system response follow the observations, i.e., $\hat{x}_s \sim x_s$. Therefore, the problem of calibration of physics-based models corresponds to the problem of modeling a physical process as approximated by a physics-based model. Since both the observations and model parameters are uncertain, model calibration is a stochastic problem. Ideally, the calibration process aims at obtaining the posterior distribution of the calibration factors given the data $p(\theta|w, x_s)$. However, computing the whole distribution is generally computationally expensive. Therefore, in most cases, point value estimations of the parameters are inferred. A typical compromise is to compute the *maximum a posteriori estimation* (MAP), described by

$$\hat{\theta}_{\text{MAP}} = \arg\max_{\theta} p(\theta|w, x_s) \tag{2}$$

Several methods have been proposed to address the problem of dynamic model calibration when the physics-based model structure is well-founded on known physical principles (e.g., aircraft thermodynamic engine models). The majority of the available methods are either probabilistic or estimation approaches developed in the fields of statistics [27] and optimal control [28]. Some examples of popular estimation methods include iterative reweighted least-squares schemes [25], unscented Kalman filters (UKF) [29–31], particle filters [32], and Bayesian inference methods using Markov chain Monte Carlo [25,33]. Recently, deep learning methods based on reinforcement learning and direct mappings with supervised learning have also been proposed in [34].

2.3. Problem formulation

Given are multivariate time series of sensor readings $x_{s_i} = [x_{s_i}^{(1)}, \ldots, x_{s_i}^{(m_i)}]^T$ and their corresponding RUL, i.e., $y_i = [y_i^1, \ldots, y_i^{m_i}]^T$, from a fleet of N units ($i = 1, \ldots, N$). Each observation $x_{s_i}^{(t)} \in R^p$ is a vector of p raw measurements taken under operating conditions $w_i^{(t)} \in R^s$. The length of the sensory signal for the ith unit is given by m_i, which can, in general, differ from unit to unit. The total combined length of the available dataset is $m = \sum_{i=1}^{N} m_i$. More compactly, we denote the available dataset as $\mathcal{D} = \{w_i, x_{s_i}, y_i\}_{i=1}^{N}$. In addition to the condition monitoring (CM) data, i.e., $\{w_i, x_{s_i}\}_{i=1}^{N}$, and the RUL label, i.e., $\{y_i\}_{i=1}^{N}$, we have access to a performance system model $F(w, \theta)$ and a reference θ, i.e., θ_{ref}, which provides the expected dynamic response of a non-degraded reference unit (i.e., $\theta = \theta_{\text{ref}}$). Starting from an unknown initial health condition, the CM data of each unit records the degradation process of the system's components. The system's components experience *normal* (linear) degradation until point in time t_{s_i}, when an *abnormal* (exponential) condition arises, leading to an eventual failure at t_{EOL_i} (end-of-life).

Given this setup, the task is to obtain a predictive model $\mathcal{G}$ that provides a reliable RUL estimate ($\hat{y}$) on a test dataset of M units $\mathcal{D}_{T*} = \{w_{j*}, x_{s_j*}\}_{j=1}^{M}$, where $x_{s_j*} = [x_{s_j*}^1, \ldots, x_{s_j*}^{k_j}]$ are multivariate time series of sensor readings taken under operating conditions $w_{j*}^{(t)}$. The total combined length of the test dataset is $m_* = \sum_{j=1}^{M} k_j$.

2.3.1. Problem assumptions

The problem formulation described above implies assumptions about the availability of the information upon which the proposed framework is built. For clarity, in this section they are explicitly listed as follows:

- Full run-to-failure trajectories of a fleet of assets. In other words, the proposed approach needs recorded degradation trajectories until failure. While data-driven algorithms also require access to a set of run-to-failure trajectories, the number of required trajectories for the proposed approach is smaller, as demonstrated within this research.
- Access to a physics-based performance model or a surrogate model that provides estimates of the measured sensor readings, additional virtual sensor, and health-related parameters while maintaining a moderate computational load. The proposed method excludes lower level representation of the physical process and, in particular, a damage model.
- The fidelity of the performance model in approximating the real process is not a prerequisite. It is assumed that complex physical processes cannot be modeled in full detail with reasonable computational cost. The available performance model, therefore, relies on modeling assumptions and simplifications of the complex physics within the system. As a result of this simplification, the accuracy of the performance prediction of the model worsens as degradation of the system increases with operation over time and the modeled physics does not account for physics of degradation. However, it is assumed that the resulting reality gap can be attributed to shifts in the values of certain model parameters governing the physics of the real process.
- Past operation is sufficient to predict RUL. In other words, it is assumed that future operation will be 'similar' to that previously observed.

3. Proposed framework: Deep learning-based prognostics with physics-inferred inputs

In this work, we propose combining a calibrated, physics-based performance model with deep learning architectures to obtain accurate hybrid prognostics models. The calibration of physics-based performance models provides estimates of the model parameters ($\hat{\theta}$) that

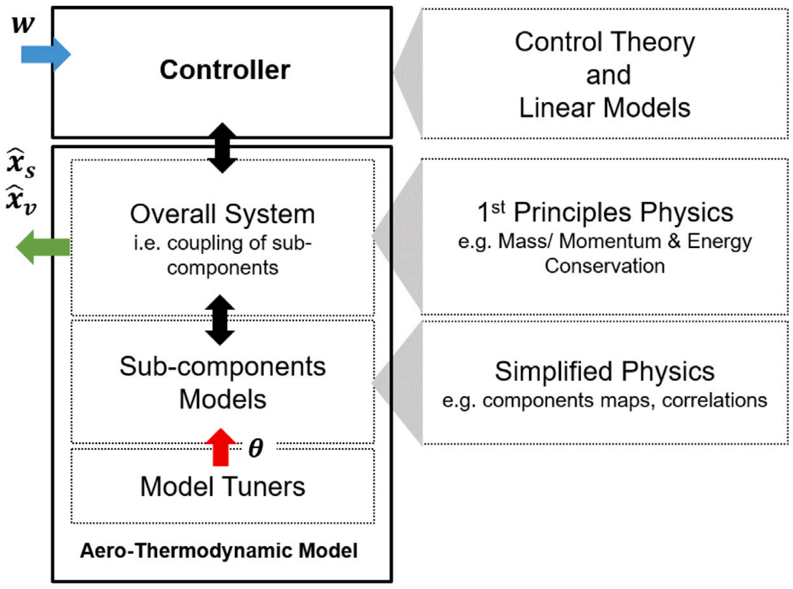


Fig. 2. General topology of an aero-thermodynamic performance model.

encode and explain the deteriorated behavior of their sub-components. The resulting calibrated model, i.e., $F(w, \hat{\theta})$, yields high-confidence estimates of unobserved process variables $\hat{x}_v$ that may be sensitive to fault signatures as well as filtered estimates of the measured process variables $\hat{x}_s$. As a result, model calibration increases the amount of information available for developing a data-driven prognostics model.

Although the physics-inferred information can be combined with the CM data in multiple ways, in this work, we propose enhancing the input feature space of a data-driven prognostics model with the process variables $[\hat{x}_s, \hat{x}_v, \hat{\theta}]$ in order to develop a hybrid prognostics model. In particular, to benefit from the learning ability brought about by recent advances in deep learning, we propose combining the physics-based performance models with deep learning architectures. Fig. 3 shows a block diagram of the proposed calibration-based hybrid prognostics approach that is demonstrated in Section 4. The deep learning prognostics model receives scenario-descriptor operating conditions w, sensor readings x_s, and model variables $[\hat{x}_s, \hat{x}_v, \hat{\theta}]$ as input features. Contrary to a standard data-driven approach aiming to learn a mapping function from the CM signals to the RUL target $\mathbf{y}$ (i.e., $[\mathbf{w}, \mathbf{x}_s] \longmapsto \mathbf{y}$), we first obtain a more informative representation $\mathbf{x}$ with additional health-related features inferred through the calibration of a physics-based system model ($[\mathbf{x}_s, F(\mathbf{w}, \cdot)] \longmapsto \hat{\theta}$). In a subsequent step, we find an optimal mapping $\mathcal{G} : \mathbf{x} \longmapsto \mathbf{y}$ from the enhanced feature space $\mathbf{x} = [w, x_s, \hat{x}_s, \hat{x}_v, \hat{\theta}]$ to perform RUL estimation. The hybrid framework is very flexible and can be combined with any type of calibration method and deep learning architecture.

3.1. Calibration of the system performance model

In this work, instead of focusing on one particular model calibration method, we aim to demonstrate the benefits of combining physics-inferred model parameters, representing sub-model health, with deep learning models in order to generate accurate prognostics models. Furthermore, the goal is to evaluate the impact of different levels of calibration accuracy on the performance of the proposed prognostics framework. Since calibration of the performance model itself is not within the scope of this research, we apply a state-of-the-art approach to calibration: an unscented Kalman filter [29] to infer the values of the model-correcting parameters $\hat{\theta}$. The rationale for this choice is that our models of interest are nonlinear and that UKF provides a good compromise between computational cost and performance. In fact, UKF is widely applied for purposes of aircraft engine health evaluation [14, 28,31]. However, we would like to stress that the proposed framework is adaptable to the chosen model calibration approach.

Model parameter estimation with UKF involves a traditional state-space formulation. In this solution strategy, the state vector comprises the health parameters, which are modeled as a random walk. The measurement equation depends on the states and input signals at the present time step t. The measurement equation is readily available from the system model F. Hence, we consider a nonlinear discrete-time system of the form:

$$\theta^{(t)} = \theta^{(t-1)} + \xi^{(t)} \quad (3)$$

$$x_s^{(t)} = F(w^{(t)}, \theta^{(t)}) + \epsilon^{(t)} \quad (4)$$

where $\xi \sim N(0, Q)$ is a Gaussian noise with covariance Q and $\epsilon \sim N(0, R)$ is a Gaussian noise with covariance R. At each time step, the estimation of the state vector θ and state covariance Q is carried out with the UKF (see Algorithm 2). A more detailed explanation of this problem formulation applied to the monitoring of gas turbine engines can be found in [31].

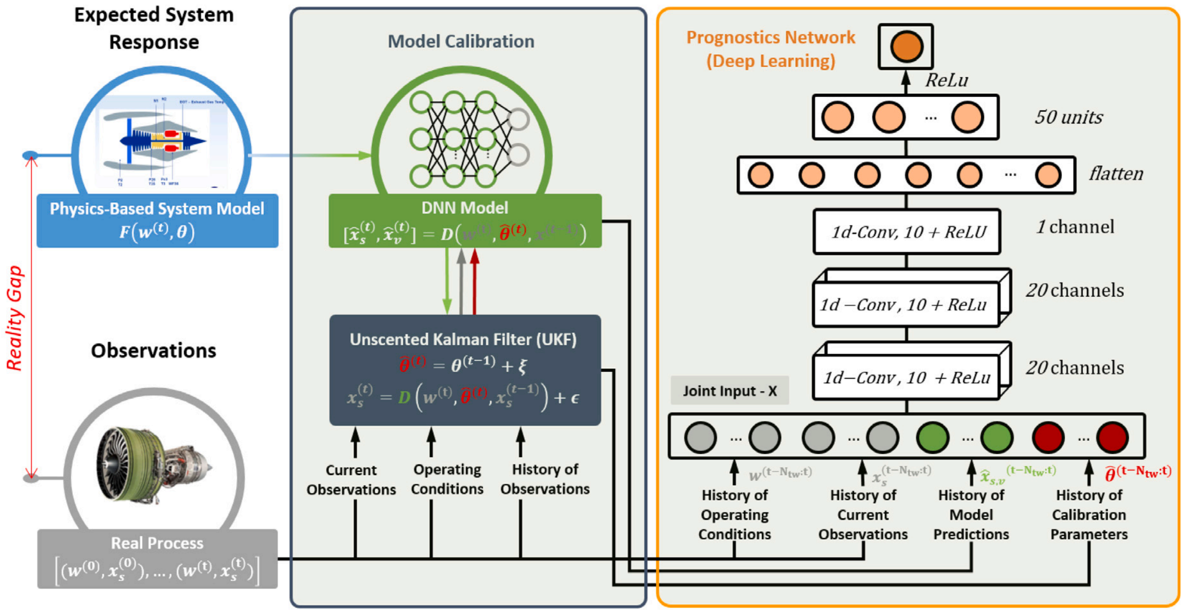
In order to speed up the learning process of the UKF algorithm, a discrete-time counterpart of the physics-based model F in the form of a deep neural network (DNN) is used. The resulting dynamic system D approximates the expected system response given the previous observations $x_s^{(t-1)}$, control variables $w^{(t)}$, and model parameters $\theta^{(t)}$, resulting in:

$$[\hat{x}_s^{(t)}, \hat{x}_v^{(t)}] = D(w^{(t)}, x_s^{(t-1)}, \theta^{(t)}) \quad (5)$$

3.2. Deep learning-based prognostics with physics-inferred inputs

The inferred model parameters $\hat{\theta}$ are informative regarding the health state of the system but do not directly relate to the end-of-life time. Together with the scenario-descriptor operating conditions w, they comprise the ‘expected’ independent factors of variation of sensor readings (x_s). Consequently, the inferred model parameters $\hat{\theta}$ can be useful for disentangling the contribution of system degradation and operating condition change w from the observed system responses. Following this reasoning, the model-correcting parameters are used to complement the raw sensor readings for the generation of data-driven prognostics models.

Since deep learning models have shown an excellent ability to reveal hidden complex functional mapping between inputs and target labels, we choose a deep neural network to discover a mapping $\mathcal{G}$ that relates



Expected System Response
Physics-Based System Model
$F(w^{(t)},\theta)$
Reality Gap
Observations
Real Process
$[(w^{(0)},x_s^{(0)}),\dots,(w^{(t)},x_s^{(t)})]$
Model Calibration
DNN Model
$[\hat{x}_s^{(t)},\hat{x}_v^{(t)}]=D(w^{(t)},\hat{\theta}^{(t)},x_s^{(t-1)})$
Unscented Kalman Filter (UKF)
$\hat{\theta}^{(t)}=\theta^{(t-1)}+\xi$
$x_s^{(t)}=D(w^{(t)},\hat{\theta}^{(t)},x_s^{(t-1)})+\epsilon$
Current Observations
Operating Conditions
History of Observations
Prognostics Network (Deep Learning)
ReLu
50 units
flatten
1d-Conv, 10 + ReLU
1 channel
1d-Conv, 10 + ReLu
20 channels
1d-Conv, 10 + ReLu
20 channels
Joint Input - X
$w^{(t-N_{tw}:t)}$
History of Operating Conditions
$x_s^{(t-N_{tw}:t)}$
History of Current Observations
$\hat{x}_{s,v}^{(t-N_{tw}:t)}$
History of Model Predictions
$\hat{\theta}^{(t-N_{tw}:t)}$
History of Calibration Parameters

Fig. 3. Overview of the hybrid prognostics framework fusing physics-based and deep learning models. The deep learning prognostics model receives as input the scenario-descriptor operating conditions (w) and estimates of the condition monitoring signals ($\hat{x}_s$), as well as the virtual sensors ($\hat{x}_v$) and unobservable model parameters ($\hat{\theta}$). The calibration of the system model (i.e., the estimation of the unobservable model parameters θ) is carried out with a state-space formulation using a UKF. We use a discrete-time counterpart of the physics-based model F to speed up the calibration process. The dynamic system D is modeled by a deep neural network. A convolutional neural network (CNN) architecture is depicted as a deep learning-based prognostics model (see Section 4.4 for further details).

the enhanced input $x = [w, x_s, \hat{x}_s, \hat{x}_v, \hat{\theta}]$ to a target label $\mathbf{y}$ given a training set $S_T \subsetneq D$. Again, multiple learning strategies are possible for this task (supervised or semi-supervised learning). In this research, we chose the standard supervised learning strategy (SL), i.e., a **direct** mapping from input $\mathbf{x}$ to a target label $\mathbf{y}$. The main reasons for this choice are the simplicity and suitability to the problem formulation in Section 2.3. Under this solution strategy, RUL predictions at inference time are obtained with forward past of the prognostics network ($\mathcal{G}_{\mathcal{H}}$) with weights and bias $\mathcal{H}$ given the CM data and the physics-inferred features.

$$\hat{y}^{(j)} \sim \mathcal{G}(w_*^{(t)}, x_{s*}^{(j)}, \hat{x}_{s*}^{(j)}, \hat{x}_{v*}^{(j)}, \hat{\theta}_*^{(j)}; \mathcal{H}) \tag{6}$$

It should be pointed out that under a slightly different problem formulation, a semi-supervised learning [35] or domain adaptation strategy [36] could also be potentially applied. To obtain the mapping function $\mathcal{G}$, a deep fully-connected neural network (FNN), a recurrent neural network (RNN), and a convolutional neural network (CNN) are evaluated within the proposed framework. As already mentioned earlier, different types of architectures could be used within the proposed framework. Section 4.4 provides further details about the architecture.

The entire procedure for the training of $\mathcal{G}$ and RUL prediction is summarized in Algorithm 1. An overview of the individual steps involved in the proposed hybrid approach is presented in Fig. 4.

4. Case study

4.1. A small fleet of turbofan engines

The proposed methodology is demonstrated and evaluated on a synthetic dataset with run-to-failure degradation trajectories of a small fleet comprising nine turbofan engines with unknown and different initial health conditions. The dataset was generated with the Commercial Modular Aero-Propulsion System Simulation (CMAPSS) model [22]. Real flight conditions, as recorded on board commercial jets, were taken as input to the CMAPSS model. Fig. 5 (left) shows the kernel density estimations of the simulated flight envelopes given by the scenario-descriptor variables W: altitude (alt), flight Mach number

Algorithm 1: Deep Learning-Based Prognostics with Physics-Inferred Inputs

```
1  Prognostics Network Training with Physics-Inferred Features - $\mathcal{G}$
     Input  : $\{w^{(i)}, x_s^{(i)}, y^{(i)}\}_{i=1}^{m}$ & D
     Output: $\mathcal{H}$ - Prognostics network weights and bias
2    Phase 1: Obtain physics-inferred features i.e., $\hat{x}_s^{(i)}, \hat{x}_v^{(i)}, \hat{\theta}^{(i)}$
3    for $i = 1 : m$ do
4        $\hat{\theta}^{(i)} \leftarrow \arg\max_\theta p(\theta^{(i)} | w^{(i)}, x_s^{(i)}; D)$
5        $[\hat{x}_s^{(i)}, \hat{x}_v^{(i)}] = D(w^{(i)}, x_s^{(i-1)}, \hat{\theta}^{(i)})$ i.e., Eq. (5)
6    end
7    Phase 2: Obtain prognostics network weights and bias $\mathcal{H}$ given
     $\{w^{(i)}, x_s^{(i)}, \hat{x}_s^{(i)}, \hat{x}_v^{(i)}, \hat{\theta}^{(i)}, y^{(i)}\}_{i=1}^{m_{S_T}} \in S_T$
8    while $i \leq E_s$ (i.e., number of epochs) do
9        repeat
10           $\mathcal{H} \leftarrow$ Update parameters in $\mathcal{G}$ using Stochastic Gradient
11           Descent
12       until convergence of $\mathcal{H}$
13   end
14 end
15 RUL Inference with Prognostics Neural Network
     Input  : $\{w_*^{(j)}, x_{s*}^{(j)}\}_{j=1}^{m_*}$, $\mathcal{H}$ & D
     Output: $\hat{y}^{(j)}$
16   for $j = 1 : m_*$ do
17       $\hat{\theta}_*^{(j)} \leftarrow \arg\max_\theta p(\theta^{(j)} | w_*^{(j)}, x_{s*}^{(j)}; D)$
18       $[\hat{x}_{s*}^{(j)}, \hat{x}_{v*}^{(j)}] = D(w_*^{(j)}, x_{s*}^{(j-1)}, \hat{\theta}_*^{(j)})$
19       $\hat{y}^{(j)} \sim \mathcal{G}(w_*^{(t)}, x_{s*}^{(j)}, \hat{x}_{s*}^{(j)}, \hat{x}_{v*}^{(j)}; \mathcal{H})$ i.e., Eq. (6)
20   end
21 end
```

(XM), throttle-resolver angle (TRA), and total temperature at the fan inlet (T2) for $N = 6$ training units (u = 2, 5, 10, 16, 18 & 20) and

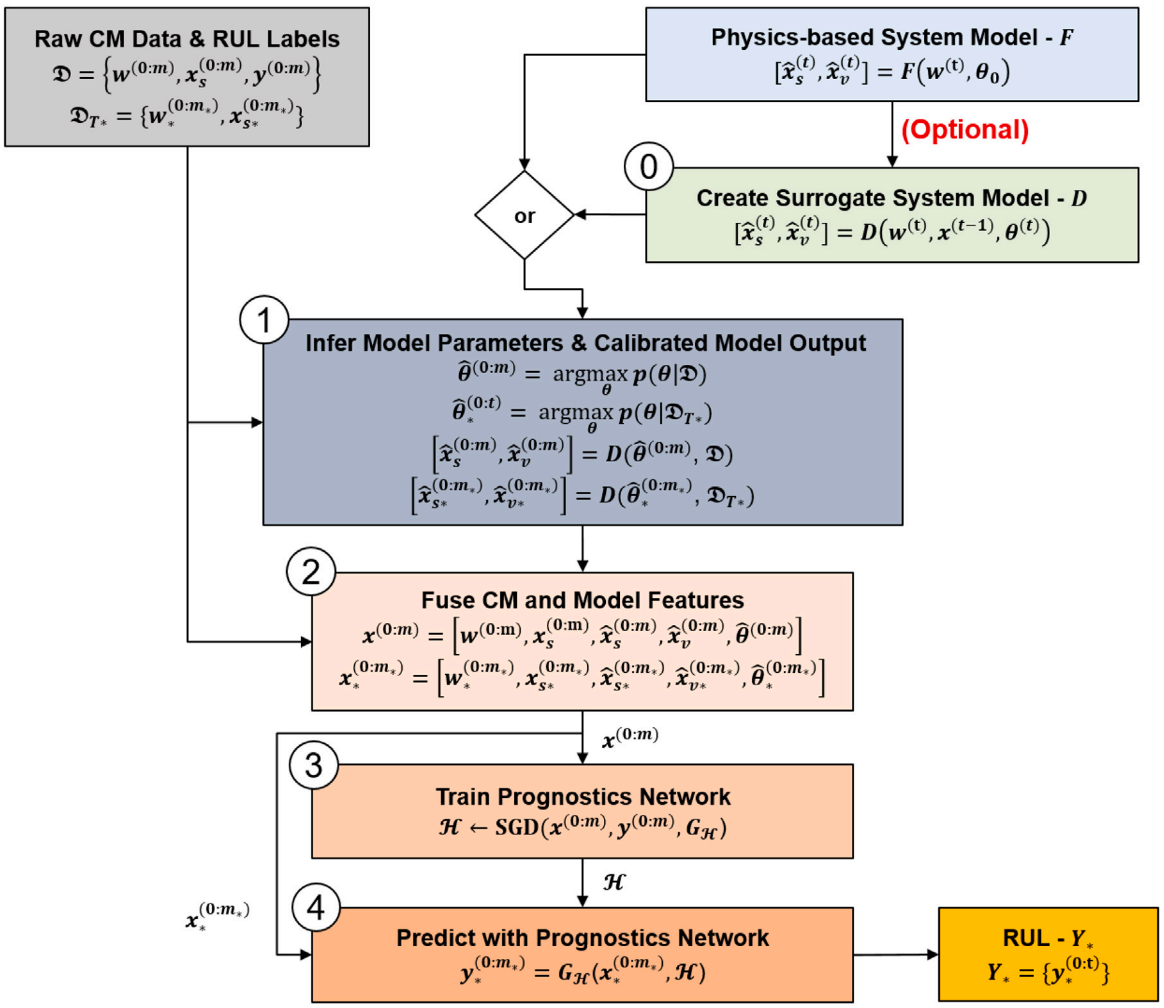

Raw CM Data & RUL Labels
$\mathfrak{D} = \{w^{(0:m)}, x_s^{(0:m)}, y^{(0:m)}\}$
$\mathfrak{D}_{T*} = \{w_*^{(0:m_*)}, x_{s*}^{(0:m_*)}\}$
Physics-based System Model - F
$[\hat{x}_s^{(t)}, \hat{x}_v^{(t)}] = F(w^{(t)}, \theta_0)$
(Optional)
0
Create Surrogate System Model - D
$[\hat{x}_s^{(t)}, \hat{x}_v^{(t)}] = D(w^{(t)}, x^{(t-1)}, \theta^{(t)})$
or
1
Infer Model Parameters & Calibrated Model Output
$\hat{\theta}^{(0:m)} = \operatorname{argmax}_{\theta} p(\theta|\mathfrak{D})$
$\hat{\theta}_*^{(0:t)} = \operatorname{argmax}_{\theta} p(\theta|\mathfrak{D}_{T*})$
$[\hat{x}_s^{(0:m)}, \hat{x}_v^{(0:m)}] = D(\hat{\theta}^{(0:m)}, \mathfrak{D})$
$[\hat{x}_{s*}^{(0:m_*)}, \hat{x}_{v*}^{(0:m_*)}] = D(\hat{\theta}_*^{(0:m_*)}, \mathfrak{D}_{T*})$
2
Fuse CM and Model Features
$x^{(0:m)} = [w^{(0:m)}, x_s^{(0:m)}, \hat{x}_s^{(0:m)}, \hat{x}_v^{(0:m)}, \hat{\theta}^{(0:m)}]$
$x_*^{(0:m_*)} = [w_*^{(0:m_*)}, x_{s*}^{(0:m_*)}, \hat{x}_{s*}^{(0:m_*)}, \hat{x}_{v*}^{(0:m_*)}, \hat{\theta}_*^{(0:m_*)}]$
$x^{(0:m)}$
3
Train Prognostics Network
$\mathcal{H} \leftarrow \mathrm{SGD}(x^{(0:m)}, y^{(0:m)}, G_{\mathcal{H}})$
$\mathcal{H}$
$x_*^{(0:m_*)}$
4
Predict with Prognostics Network
$y_*^{(0:m_*)} = G_{\mathcal{H}}(x_*^{(0:m_*)}, \mathcal{H})$
RUL - Y_*
$Y_* = \{y_*^{(0:t)}\}$

Fig. 4. Overview of the individual steps of the proposed calibration-based hybrid prognostics approach. Both training and inference are represented. The proposed framework uses a physics-based system model (F) - or, optionally, a deep neural network model approximating the complex system dynamics - and a training dataset as inputs (D). The training dataset comprises CM data, i.e., $[w, x_s]$, from run-to-failure degradation trajectories and their corresponding RUL labels (Y). In an initial, optional step, the physics-based system model can be replaced by a surrogate dynamic model in the form of an NN (i.e., D) to reduce the computational load of the framework. **Training**: Step 1 - using a calibration method, the model parameters $\hat{\theta}$ are inferred given the CM data and the modeled system dynamics (F or D) (see Section 3.2). The calibration process also provides the calibrated system model response $\hat{x}_s$ and virtual sensors $\hat{x}_s$. Step 2 - the CM data is fused with the model-derived features i.e., $\hat{x}_s$, $\hat{x}_s$, and $\hat{\theta}$. Step 3 - training of the prognostics network with the increased feature space (x) and the corresponding RUL labels (Y). **Inference**: Following steps 1 to 3, the test dataset (D_*) is processed. Finally, Step 4 performs inference with the trained prognostics network at test conditions $x_*^{(t)}$.

$M = 3$ test units ($u = 11$, 14 & 15). It is worth noting that test units 14 and 15 have an operation distribution significantly different from those of the training units. Concretely, test units 14 and 15 operate shorter and lower altitude flights compared to other units. The training dataset contains, therefore, flight profiles that are not fully representative of the test conditions of these two units. This is a more difficult learning and generalization task. We have chosen this example due to its relevance for practical applications where observed operating conditions of new units may not correspond to the past operating conditions of other units in the fleet. Purely data-driven approaches generally require domain adaptation strategies for this type of setup [37].

Two distinctive failure modes are present in the available dataset (D). Units 2, 5, and 10 have failure modes of an *abnormal* high-pressure turbine (HPT) efficiency degradation. Units 16, 18, and 20 are subject to a more complex failure mode that affects the low-pressure turbine (LPT) efficiency and flow in combination with the high-pressure turbine (HPT) efficiency degradation. Test units are subjected to the same complex failure mode. Fig. 6 shows degradation profiles induced in the nine units of the fleet. The initial deterioration state of each unit is different and corresponds to an engine-to-engine variability equivalent to 10% of the health index. The degradation of the affected system components follows a stochastic process with a linear *normal degradation* followed by a steeper *abnormal degradation*. The degradation rate of each component varies within the fleet. The transition from *normal* to *abnormal* degradation is smooth and occurs at different cycle times for each unit. The transition time (t_s) is dependent on the operating conditions, i.e., flight and degradation profile. It should be noted that although the degradation profiles of individual components show nearly overlapping trajectories, the combined profile (i.e., the profile in three dimensions) is clearly different.

An overview of the transition times t_s, the end-of-life times t_{EOL}, and the number of samples from each unit of the fleet m_i is provided in Table 1. The sampling rate of the data is 0.1 Hz, resulting in a total dataset size of 0.53M samples for model development and 0.12M samples for testing. It is worth noting that while test unit 14 is a short flight engine with the lowest amount of flight time (0.16M seconds), it has the largest number of flight cycles. More details about the generation process and a more recent version of this dataset (i.e., DS02) can be found in [23].

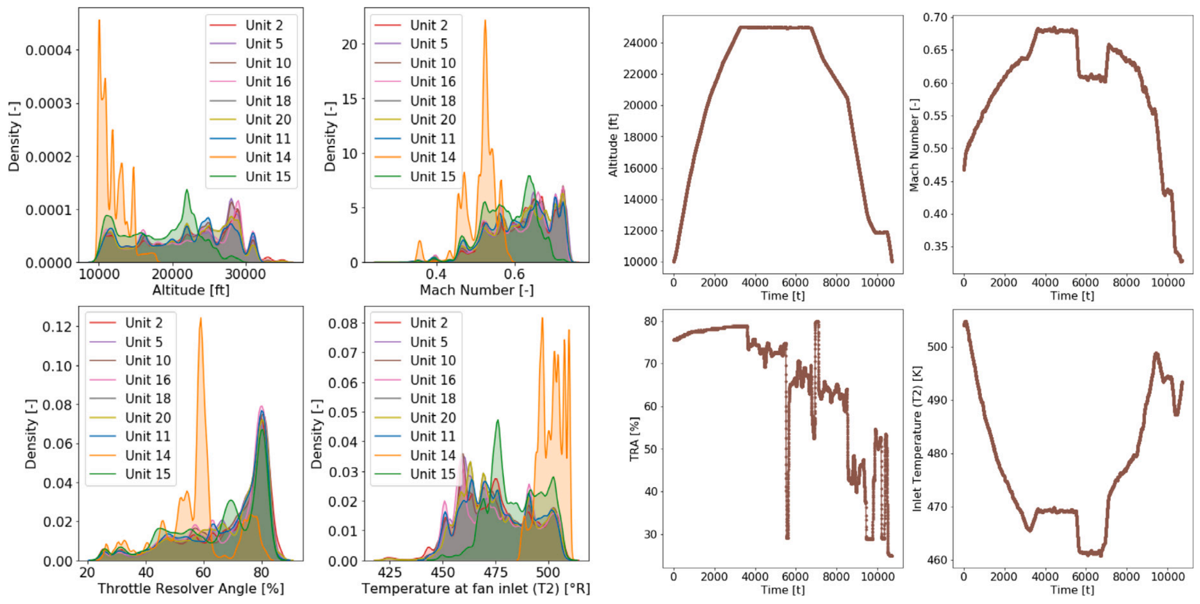


Fig. 5. **Left.** Kernel density estimations of the simulated flight envelopes (i.e., climb, cruise, and descent flight conditions) given by recordings of altitude, flight Mach number, throttle-resolver angle (TRA), and total temperature at the fan inlet (T2) for complete run-to-failure trajectories of six training units (u = 2, 5, 10, 16, 18 & 20) and three test units (u = 11, 14 & 15). **Right.** An example of a typical single flight cycle given by traces of the scenario-descriptor variables for unit 10. Climb, cruise, and descent flight conditions (with alt > 10,000 ft), corresponding to different flight routes taken by the aircraft, are covered.

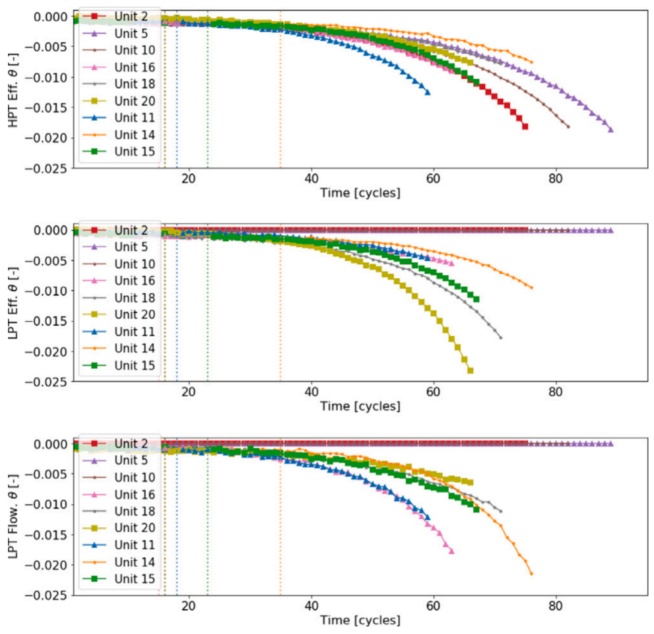


Fig. 6. Traces of the degradation imposed on the high-pressure turbine efficiency (HPT_Eff_mod), low-pressure turbine efficiency (LPT_Eff_mod), and low-pressure turbine flow (LPT_flow_mod) for each unit of the fleet. The onset of the abnormal degradation (i.e., t_s) of each unit is indicated by dashed vertical lines.

Table 1
Size (m_u), the transition cycle time (t_s) and end-of-life time (t_{EOL}) of each unit within the development ($\mathcal{D}$) and test datasets ($\mathcal{D}_{T*}$).

Development dataset - $\mathcal{D}$				
Unit (u)	m_i	t_s	t_{EOL}	Failure mode
2	0.085M	17	75	HPT
5	0.103M	17	89	HPT
10	0.095M	17	82	HPT
16	0.077M	16	63	HPT+LPT
18	0.089M	17	71	HPT+LPT
20	0.077M	17	66	HPT+LPT
Test dataset - $\mathcal{D}_{T*}$				
Unit (u)	m_j	t_s	t_{EOL}	Failure mode
11	0.066M	19	59	HPT+LPT
14	0.016M	36	76	HPT+LPT
15	0.043M	24	67	HPT+LPT

4.2. System performance model

As indicated in Section 2.3.1, in addition to the CM data and the RUL labels from a training dataset, the proposed hybrid approach requires a system performance model $F(w, \theta)$ or a surrogate model thereof. In this case study, a surrogate model (D) was used in the form of a deep neural network (i.e., DNN). Concretely, the surrogate model D approximates the expected system dynamics given the previous observations $x_s^{(t-1)}$, the control variables $w^{(t)}$, and model parameters $\theta^{(t)}$ as described in (5). The development of this surrogate model, given an independent dataset generated with the CMAPSS simulator, is beyond the scope of this case study, but the interested reader can find the implementation details in Appendix B. It is worth pointing out that the dynamic model D is a noisy approximation of the CMAPSS model dynamics. Therefore, predictions of the dynamic model D at operating conditions w given some initial guess $\theta^{(0)}$ deviate from the observed responses (x_s) due to (1) model error of the DNN, (2) the different initial health conditions of each unit, (3) sensor noise, and (4) the increased degradation of each unit over time.

4.3. Pre-processing

The dimension of the input space (i.e., $\mathbf{x} \in R^{m \times n}$) varies depending on the selected solution strategy. The data-driven models are only based on condition monitoring signals ($[w, x_s]$ and have 20 inputs (i.e., n = 20). The proposed hybrid method has 50 inputs (including the addition of the model predictions, calibration parameters, and the virtual sensors i.e., $\mathbf{x} = [w, x_s, \hat{x}_s, \hat{x}_v, \hat{\theta}]$). Tables 2 to 4 provide a detailed overview of the model variables included in the condition monitoring signals $[w, x_s]$, virtual sensors x_v, and model parameters θ. The variable name corresponds to the internal variable name used in the CMAPSS

Table 2
Condition monitoring signals - $[w, x_s]$. The variable symbol corresponds to the internal variable name in CMAPSS. The descriptions and units are reported as in the model documentation [22].

#	Symbol	Description	Units
1	alt	Altitude	ft
2	XM	Flight Mach number	–
3	TRA	Throttle-resolver angle	%
4	Wf	Fuel flow	pps
5	Nf	Physical fan speed	rpm
6	Nc	Physical core speed	rpm
7	T2	Total temperature at fan inlet	°R
8	T24	Total temperature at LPC outlet	°R
9	T30	Total temperature at HPC outlet	°R
10	T40	Total temp. at burner outlet	°R
11	T48	Total temperature at HPT outlet	°R
12	T50	Total temperature at LPT outlet	°R
13	P15	Total pressure in bypass duct	psia
14	P2	Total pressure at fan inlet	psia
15	P21	Total pressure at fan outlet	psia
16	P24	Total pressure at LPC outlet	psia
17	Ps30	Static pressure at HPC outlet	psia
18	P30	Total pressure at HPC outlet	psia
19	P40	Total pressure at burner outlet	psia
20	P50	Total pressure at LPT outlet	psia

Table 3
Virtual sensors - $[x_v]$. The variable symbol corresponds to the internal variable name in CMAPSS. The descriptions and units are reported as in the model documentation [22].

#	Symbol	Description	Units
1	P45	Total pressure at HPT outlet	psia
2	W21	Fan flow	pps
3	W22	Flow out of LPC	lbm/s
4	W25	Flow into HPC	lbm/s
5	W31	HPT coolant bleed	lbm/s
6	W32	HPT coolant bleed	lbm/s
7	W48	Flow out of HPT	lbm/s
8	W50	Flow out of LPT	lbm/s
9	SmFan	Fan stall margin	–
10	SmLPC	LPC stall margin	–
11	SmHPC	HPC stall margin	–

Table 4
Model correcting parameters - $[\theta]$. The variable symbol corresponds to the internal variable name in CMAPSS. The descriptions and units are reported as in the model documentation [22].

#	Symbol	Description	Units
1	HPT_eff_mod	HPT efficiency modifier	–
2	LPT_eff_mod	LPT efficiency modifier	–
3	LPT_flow_mod	LPT flow modifier	–

model. The descriptions and units are reported as provided in the model documentation [22].

The input space **x** to the models is normalized to a range $[-1, 1]$ by a min/max-normalization given the available dataset ($\mathcal{D}$). A validation set $V_T \subsetneq \mathcal{D}$ comprising 10% of the available data was chosen for early stopping of the training process and hyperparameter tuning of the deep learning prognostics model. For RNN and CNN models' requirements, the original dataset was pre-processed with a sliding time window approach of size $N_{tw} = 50$ and stride of 1. The sliding window means that the first input sample to the network takes measurements from timestamps 1–50, the second 2–51, the third 3–52, and so on for each unit of the fleet (i.e., each input has a time length of 500 s).

4.4. Deep learning prognostics model

As mentioned before, the main goal of this research study is to evaluate the capability of the proposed hybrid framework to perform prognostics and to compare this framework to the performance of purely data-driven approaches.

In prognostics, the degradation mechanisms are often mapped to a health index (HI). This health index can be defined as a normalized margin to multiple health-related thresholds evaluated at specified reference conditions. The end-of-life time of a mechanical system corresponds to the point in time at which HI = 0. Under this definition, the health index is time-independent and consequently, a mapping from the system state (represented by CM data and $\hat{\theta}$) to HI exists. While the evolution of the system state is a time-dependent process, the mapping from the system state to the health indicator is not time-dependent. Based on this reasoning, both time-independent and time-dependent deep learning models are considered in this research. In particular, we used three state-of-the-art deep neural networks as prognostics models: a deep multilayer perceptron feed-forward neural network (FNN), a recurrent neural network (RNN), and a convolutional neural network (CNN).

Deep CNN architectures have been proven to provide excellent performance on time series for predictive maintenance in recent works [6, 35,38–40]. Concretely, a solution based on 2D convolutions with 1D filters and no pooling layers was proposed in [6] for the prognostics benchmark problem [11]. Similarly, deep architectures based on 1D convolutions (see Fig. 7) have also achieved excellent results in signal processing applications [41]. Therefore, in light of the good results achieved, we adopted a one-dimensional convolutional neural network architecture as a CNN model.

Deep RNNs based on long short-term memory networks (LSTM) have also shown excellent results with regard to the prognostics problem [7,8,10,36,42]. Therefore, we also adopted a deep LSTM architecture as an RNN model.

To make the comparison for the proposed hybrid framework more challenging, we deliberately provided the purely data-driven approach an unfair advantage by selecting hyperparameters that maximize its prognostics performance while we do not perform hyperparameter tuning for the hybrid approach. We then train the same neural network models with the enhanced inputs space of the hybrid approach. Since the inputs space of the hybrid and purely data-driven approaches have a different dimension, the different sizes of the input also change the number of weights in the first layer and, as a result, such architecture is sub-optimal for the hybrid approach. The network architectures of the deep learning prognostics model are briefly described below.

Feed-forward neural network (FNN). The architecture of the feed-forward neural network used here comprises five fully connected layers ($L = 5$). The input layer has n nodes, where n denotes the size of the input space **x** and varies depending on the considered solution strategy. The first three hidden layers have 200 neurons each. The last hidden layer has 50 neurons. A single linear neuron was used in the output layer. In compact notation, we refer to this architecture as $[n,\ 200,\ 200,\ 200,\ 50,\ 1]$. *ReLU* activation function was used throughout the entire network. It should be noted that RUL estimation is a regression problem. Therefore, the last activation $\sigma^L = I$ is the identity. The network has 95k trainable parameters ($\mathcal{H}$). This final architecture is the result of conducting a grid search wherein the search space over the hyperparameters includes: number of hidden layers [1-4], number of neurons at each hidden layer [50, 100, 200], and activation function type [*tanh*, *relu*].

One-dimensional convolutional neural network (1D CNN). The architecture of the CNN used in this research also comprises five layers ($L = 5$). The network has three initial convolutional layers with filters of size 10. The first two convolutions have ten channels and the last convolution has only one channel. Zero padding is used to keep the feature map through the network. The resulting 2D feature map is flattened and the network ends with a 50-way fully connected layer followed by a linear output neuron. The network uses *ReLu* as the activation function. The network has 24k trainable parameters ($\mathcal{H}$). As with the FNN, the final architecture is the result of conducting a grid

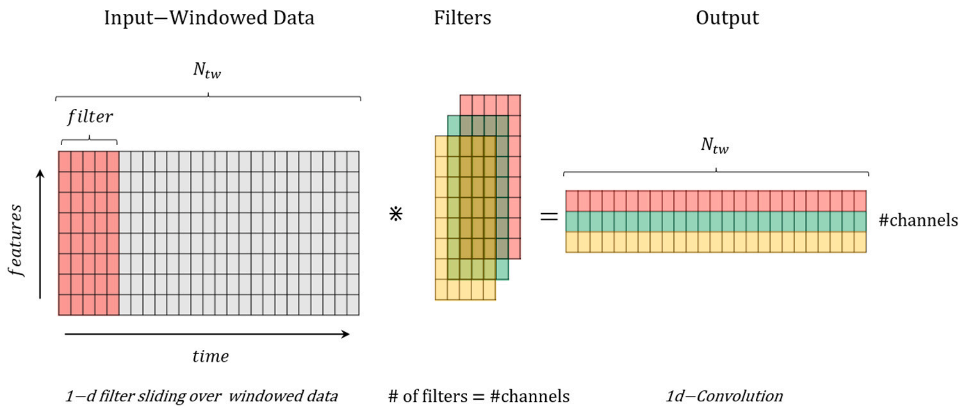


Fig. 7. 1D convolution layer with convolution in the temporal direction.

search over the following hyperparameters: number of hidden layers [1-4], number of channels [10, 20, 30] at each convolutional layer, filter size [10, 20], number of neurons at the fully connected layer [50, 100], activation function type [*tanh*, *relu*], and window size of the sliding window [20, 50, 200].

Long short-term memory recurrent neural network (LSTM). Following the design philosophy from the 1D CNN model, the architecture of the LSTM neural network used in this research also comprises five layers ($L = 5$). The network has three initial LSTM layers followed by a 50-way fully connected layer and ends with a linear output neuron. The network uses *ReLu* as the activation function and a sliding window of size 50. Through a grid search, we determined the optimal number of cells in the hidden layers to be 20. The network has 24k trainable parameters ($\mathcal{H}$).

4.5. Training set-up

The optimization of the network's weights is carried out with mini-batch stochastic gradient descent (SGD) and with the *AMSgrad* algorithm [43]. The *Xavier* initializer [44] is used for the weight initializations. The batch size is set to 1024 and the learning rate to 0.001. The maximum number of epochs (E_s) was set to 60 for the FNN model and 30 for the CNN model. Early stopping with a patience of 5 epochs is considered.

4.6. Evaluation metrics

The performance of the proposed framework is evaluated and compared to the purely data-driven deep learning models on the selected prognostics task. We apply two common evaluation metrics in CMAPSS prognostics analysis [45]: root-mean-square error (RMSE) and NASA's scoring function (s) [46], which are defined as:

$$s = \sum_{j=1}^{m_*} exp(\alpha|\Delta^{(j)}|) - 1 \tag{7}$$

$$RMSE = \sqrt{\frac{1}{m_*}\sum_{j=1}^{m_*} (\Delta^{(j)})^2} \tag{8}$$

where m_* denotes the total number of test data samples, $\Delta^{(j)}$ is the difference between the estimated and the real RUL of the j sample (i.e., $y^{(j)} - \hat{y}^{(j)}$), and α is $\frac{1}{13}$ if RUL is underestimated and $\frac{1}{10}$ otherwise. The resulting s metric is not symmetric and penalizes overestimation more than underestimation. It is worth noting that at test time, RUL estimations ($\hat{y}^{(j)}$) are obtained at each point in time at which a new sample is available. Therefore, unit-specific pointwise RUL estimation (i.e., $\hat{y}_u^{(j)}$) can show high variability within a flight cycle, indicating that some parts of the flight are more informative for purposes of RUL estimation as compared to others. In order to evaluate this effect, we also define the average RUL estimation at cycle c in unit u ($\hat{y}_u^{[c]}$), which is defined as follows:

$$\hat{y}_u^{[c]} = \frac{1}{m_u^{[c]}} \sum_{j=1}^{m_u^{[c]}} y_u^{(j)} \tag{9}$$

where $m_u^{[c]}$ is the length of the flight cycle c for the uth unit, which is formally defined using the indicator function, i.e., $\mathbf{1}\{.\}$, as:

$$m_u^{[c]} = \sum_{j=1}^{m_*} \mathbf{1}\{U^{(j)} = u \wedge C^{(j)} = c\} \tag{10}$$

where U and C are vectors with unit and cycle labels for each sample of the test dataset.

In addition to these two popular performance metrics of CMAPSS prognostics models, we also consider the prediction horizon with a prediction error below 5 cycles ($H_{|\Delta_y|\leq 5}$) and the inference time per sample.[1]

The prediction horizon with a prediction error below 5 cycles is computed as follows:

$$H_{|\Delta_y|\leq 5} = t_{\text{EOL}} - t_{|\Delta_y|\leq 5} \tag{11}$$

where $t_{|\Delta_y|\leq 5}$ is the cycle time in which the prediction error is below 5 cycles for any future time ($t \geq c$).

5. Experimental results

5.1. RUL estimation

The RUL estimation is performed based on the same neural network models (i.e., FNN, LSTM, and CNN) and the same architectures for both set-ups: the purely data-driven approach and the proposed hybrid approach. In this section, the performance of the two approaches is compared based on different metrics. Table 5 shows the failure time prediction performance of both the proposed hybrid approach ($\mathbf{x} = [w, x_s, \hat{x}_s, \hat{x}_v, \hat{\theta}]$) and the *baseline* approach (purely data-driven with $\mathbf{x} = [w, x_s]$). With a reduction ranging from 16% to 47% in RMSE and 21% to 68% in s-score, depending on the neural network model, the hybrid approach clearly outperforms the *baseline*. Since the s-score penalizes overestimation more than underestimation, and RMSE is a symmetric metric, the greater reduction of s compared to RMSE indicates that the proposed method handles RUL overestimation more effectively. The three neural network architectures evaluated achieve nearly the same performance with the hybrid approach. However, important differences between time-dependent and time-independent models based on a

[1] As computed with an average processor Intel(R) Core(TM) i7-8650U CPU @1.90 GHz, 2112 Mhz, 4 Core(s).

Table 5
Overview of the RMSE, s-score, and inference time metrics with hybrid and baseline approaches for complete degradation trajectories on test set ($\mathcal{D}_{T*}$), comprising data from the three test units. Mean and standard deviation results with FNN, LSTM and CNN models over five runs.

FNN model			
Metric	Data-Driven	Hybrid	rel. Delta
RMSE [cycles]	7.89 ± 0.12	**4.22** ± 0.10	−47%
$s \times 10^5$ [–]	1.39 ± 0.04	**0.44** ± 0.01	−68%
Time [s]	0.20e−04	1.02e−02	–
LSTM model			
Metric	Data-Driven	Hybrid	rel. Delta
RMSE [cycles]	5.02 ± 0.21	**4.40** ± 0.14	−12%
$s \times 10^5$ [–]	0.59 ± 0.06	**0.47** ± 0.03	−20%
Time [s]	0.85e−04	1.03e−02	–
CNN model			
Metric	Data-Driven	Hybrid	rel. Delta
RMSE [cycles]	4.95 ± 0.15	**4.14** ± 0.09	−16%
$s \times 10^5$ [–]	0.56 ± 0.03	**0.44** ± 0.02	−21%
Time [s]	5.41e−04	1.08e−02	–

Table 6
Prediction horizon [cycles] for $\Delta_y \leq 5$ (i.e., $t_{\mathrm{EOL}} - t_{|\Delta_y|\leq 5}$) with data-driven and hybrid approaches with CNN model.

u	Data-Driven	Hybrid	rel.Delta
11	11	**31**	195%
14	15	**43**	197%
15	24	**37**	54%
Fleet Avg.	16	**37**	127%

Table 7
Overview of the RMSE and s-score metrics with hybrid and baseline approaches for complete degradation trajectories on the test set ($\mathcal{D}_{T*}$), comprising data from the three test units. Mean and standard deviation results with CNN model over five runs. The training dataset contains only units 16, 18, and 20.

Units 2, 5, 10, 16, 18, 20			
Metric	Data-Driven	Hybrid	rel. Delta
RMSE [cycles]	4.95 ± 0.15	**4.14** ± 0.09	−16%
$s \times 10^5$ [–]	0.56 ± 0.03	**0.44** ± 0.02	−21%
Units 16, 18, 20			
Metric	Data-Driven	Hybrid	rel. Delta
RMSE [cycles]	5.97 ± 0.37	**4.22** ± 0.12	−29%
$s \times 10^5$ [–]	0.61 ± 0.05	**0.43** ± 0.02	−29%
rel. Delta RMSE [%]	17%	2%	
rel. Delta s [%]	8%	−2%	

purely data-driven approach are observed (i.e., −37% in RMSE between FNN and CNN and −36% in RMSE between FNN and LSTM). Overall, the hybrid approach with the CNN model is the best-performing model. Inference with the hybrid approach is 20 to 524 times more time-consuming than with a purely data-driven approach, depending on the network architecture. This increased inference time is mainly driven by the additional calibration step that takes 1.02e−02 s. Inference with prognostics networks based on LSTM architectures is 20 times slower than with those based on FNN. The inference time with the hybrid approach is well below the sampling rate of the data, regardless of the network architecture.

The improvement in prognostics performance is also observed for individual test units. Fig. 8 shows the error between true and predicted *RUL* (i.e., $\Delta_{y_u} = \hat{y}_u^{[c]} - y_u^{[c]}$) of the baseline approach (left) and the proposed hybrid approach (right) for each of the test units with FNN (top), LSTM (middle), and CNN models (bottom). The shaded surface shows the variability of the RUL predictions within each cycle. The upper limit corresponds to $\max(\Delta_{y_u})$ and the lower bound to $\min(\Delta_{y_u})$. *RUL* estimates undertaken with the *baseline* approach at any point in time have a large variability and high bias (specifically, RUL over estimation) as compared to the hybrid approach. We can also observe that the cycle time at which the prediction error is below 5 cycles for any future time ($t \geq c$) decreases for all the test units. We denote this time as $t_{|\Delta_y|\leq 5}$. Table 6 reports the prediction horizon, i.e., $t_{\mathrm{EOL}} - t_{|\Delta_y|\leq 5}$, for each unit and the average value for the whole fleet. Under this metric, the proposed hybrid approach provides an average increase of 127% on the prediction horizon while maintaining a similar prediction accuracy.

5.2. Ablation studies

This section extends the previous analysis with three ablation studies to cover additional realistic prognostics scenarios and provide further insights into the proposed hybrid approach. Ablation study I evaluates both the hybrid and purely data-driven approaches under a variant of the case study with a smaller dataset. Ablation study II evaluates the contribution of the physics-derived features to the overall prediction performance. Finally, in ablation study III, we analyze the impact of the calibration process quality on the hybrid approach's predictive performance.

5.2.1. Ablation study I: Impact of dataset size

The results in the previous section showed that the proposed hybrid approach outperforms the purely data-driven approach regardless of the neural network model used for the prognostics model. Moreover, it is worth noting that the CNN model provided a clear (i.e., 38%) improvement in performance over the FNN model. These results indicate that in presence of abundant representative data, a powerful neural network model can serve as a competitive prognostics model. However, abundant run-to-failure datasets are often unavailable in real applications due to the rarity of fault occurrences in safety-critical systems. In scenarios of non-abundant CM data, we hypothesize that discovering informative and representative features of degradation from raw CM data is more challenging. As a result, the purely data-driven approach will encounter more difficulties in achieving good performance. To test this hypothesis, we consider an alternative dataset with a subset containing 50% of the units. To decouple this analysis from the effect of dataset representatives (i.e., similar or dissimilar degradation trajectories), we selected units 16, 18, and 20, as they are affected by the same failure mode (i.e., HPT and LPT failure). The resulting prognostics performance of the purely data-driven and hybrid CNN models trained with training data from 50% of the units is shown in Table 7. The dataset size reduction has a negligible impact on the prognostics performance (2% increase in RMSE) of the proposed hybrid approach. Contrarily, such a reduction of the dataset size leads to an increase of 17% in the RMSE of the purely data-driven approach. The relative delta between hybrid and purely data-driven increases to 29% in RMSE. Therefore, enhancing the input space with additional features from physical performance models yields a clear advantage in scenarios of non-abundant CM data.

5.2.2. Ablation study II: Impact of physics-derived information

The proposed hybrid framework uses three types of features derived from the calibrated physical model, i.e., $\hat{x}_s, \hat{x}_v, \hat{\theta}$. To analyze the contributions of each of these feature types to the prognostics performance, we evaluated alternative hybrid models that increase the amount of model-derived information used in the prognostics model. In order to have full coverage, we evaluate the following combinations of input signals: the purely data-driven $[w, x_s]$, those with added de-noised sensor readings $[w, x_s, \hat{x}_s]$, those with added virtual sensors $[w, x_s, \hat{x}_s, \hat{x}_v]$, and finally those with added calibration factors $[w, x_s, \hat{x}_s, \hat{x}_v, \hat{\theta}]$.

In order to decouple this analysis from the effect of uncertainty in the calibration process (see next section), we assume that the model information is obtained from a perfect calibration, i.e., $\hat{x}_s, \hat{x}_v, \hat{\theta}$ are

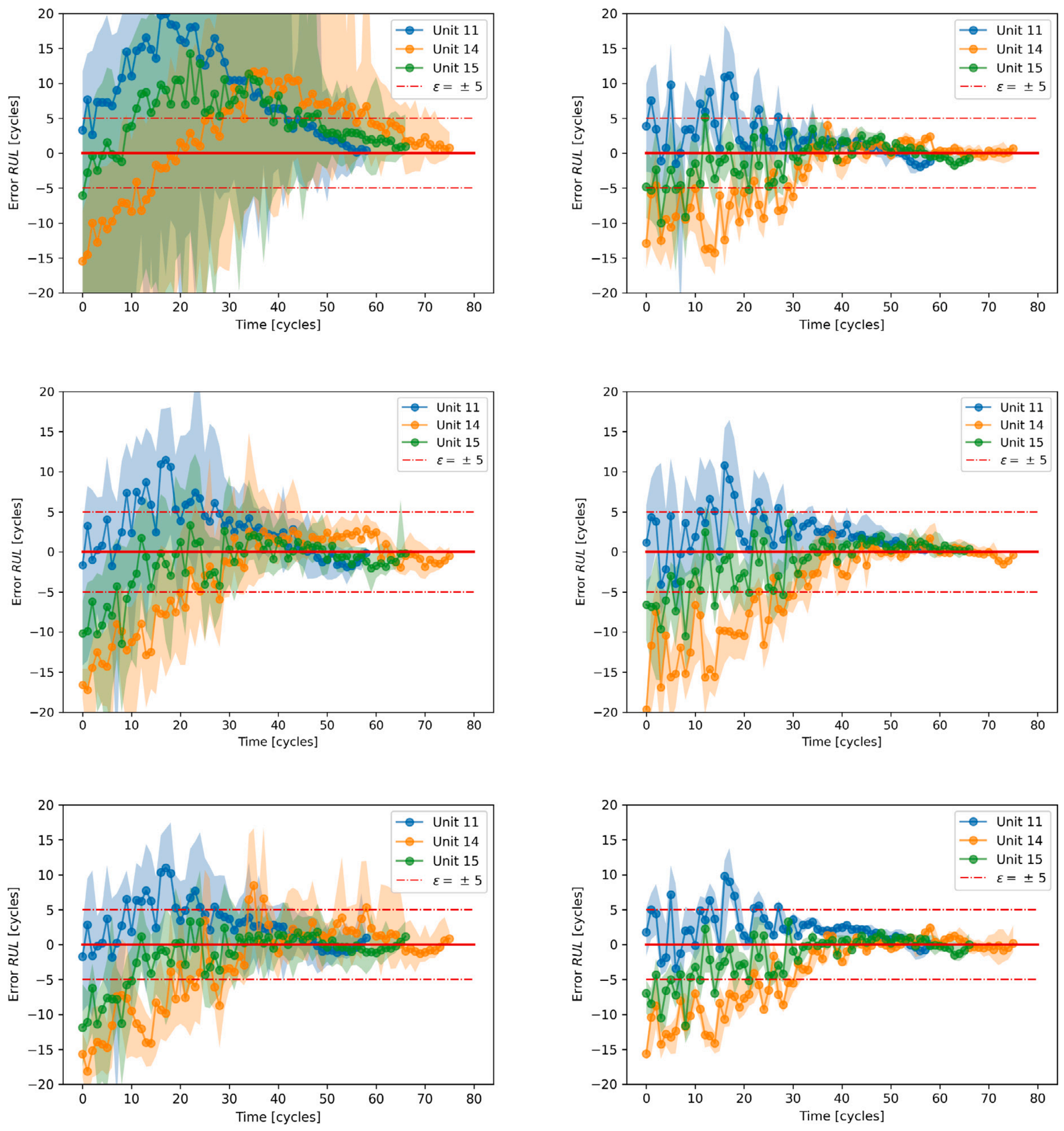


Fig. 8. RUL prediction error for each unit with FNN (top), LSTM (middle), and CNN models (bottom). The dotted lines are the average RUL estimate at each cycle, i.e., $\Delta_{y_u} = \hat{y}_u^{[c]} - y_u^{[c]}$), while the shaded surface represents the uncertainty bounds for RUL predictions within each cycle. The red dashed horizontal lines correspond to ±5 cycles error bars. The three test units are shown: Unit 11 (blue), Unit 14 (orange), and Unit 15 (green).

therefore ground-truth values. The resulting prognostics performances of the new hybrid models are shown in Table 8. We can observe that adding physics-derived features always yields an improvement in the prediction performance. However, optimal performance is only achieved when the calibration factors are included in the prognostics model. This result suggests a hierarchy of information whereby the most informative features for RUL prediction, namely the calibration parameters (i.e., $\hat{\theta}$), are the most informative of the RUL. To verify this hypothesis, Fig. 9 shows the normalized mutual information between input features and the RUL target. The nine most informative input features within the input space $[w, x_s, \hat{x}_s, \hat{x}_v, \hat{\theta}]$ are shown. The calibration parameters ($\hat{\theta}$) are the most informative features for predicting RUL, followed by features representing operating conditions (i.e., w). The de-noised sensor readings P50 and P2 (i.e., the total pressure at LPT outlet and the total pressure at fan inlet, respectively) are the most informative sensors.

Table 8
Overview of the RMSE and s-score results with CNN models given different levels of physics information. Mean ± standard deviation over five runs.

Purely Data-driven		
Model	RMSE [cycles]	$s \times 10^5$
$[w, x_s]$	4.95 ± 0.15	0.56 ± 0.03
Hybrid		
Model	RMSE [cycles]	$s \times 10^5$
$[w, x_s, \hat{x}_s]$	4.90 ± 0.14	0.56 ± 0.03
$[w, x_s, \hat{x}_s, \hat{x}_v]$	4.61 ± 0.18	0.49 ± 0.05
$[w, x_s, \hat{x}_s, \hat{x}_v, \hat{\theta}]$	4.14 ± 0.08	0.44 ± 0.02

5.2.3. Ablation study III: Impact of the model calibration uncertainty

The quality of the calibration process has an impact on the prognostic performance of the proposed hybrid framework. In order to quantify

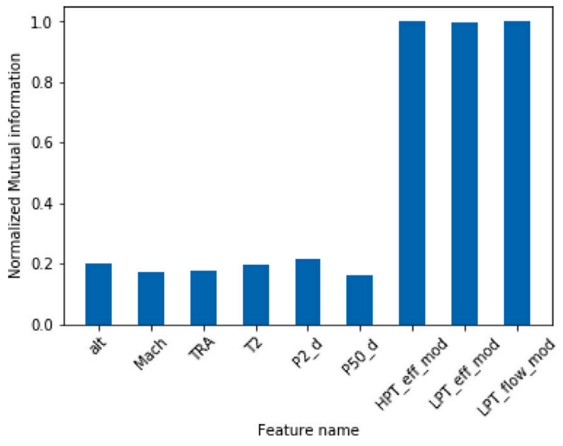


Fig. 9. Top nine features with higher normalized mutual information to the RUL label.

Table 9
Overview of the prognostics performance under model bias and noise with hybrid approaches and 1D CNN model.

Bias		
Intensity	RMSE [cycles]	$s \times 10^5$
$\alpha = +0.5$	4.77 ± 0.24	0.47 ± 0.02
$\alpha = -0.5$	4.66 ± 033	0.59 ± 0.07
Noise		
Intensity	RMSE [cycles]	$s \times 10^5$
$\text{SNR}_{db} = 20$	4.26 ± 0.13	0.47 ± 0.03
$\text{SNR}_{db} = 15$	4.71 ± 0.17	0.52 ± 0.02

this impact, we evaluate the sensitivity of the proposed framework to the calibration performance, i.e., the impact of low-quality estimates of θ on the RUL prediction performance. Concretely, we consider hypothetical situations where $\hat{\theta}$ is noisy (see Fig. 10, left) and also where $\hat{\theta}$ is affected by bias (see Fig. 10, right). To model the noisy calibrations of different quality, white noise with signal-to-noise ratios (SNR_{db}) of 15 and 20 db are imposed on the calibration factors (i.e., *HPT_Eff_mod*, *LPT_Eff_mod* and *LPT_Flow_mod*). To model the bias, we imposed an increasing shift proportional to the nominal calibration factor values ($\theta_\alpha = \theta^{(t)} + \alpha(\theta^{(0)} - \theta^{(t)})$. Two values of α are evaluated (i.e., $\alpha = 0.5$ and $\alpha = -0.5$), representing a ±50% error in the estimation of θ. Since $\hat{\theta}$ is multidimensional and an infinite number of scenarios are possible, we restricted the analysis to cases where the calibration factors are affected equally by noise and bias. Fig. 10 (left) shows the resulting noisy calibration parameters $\hat{\theta}$. The right figure shows the biased calibration factors. Table 9 reports the impact in RMSE and s of the three SNR evaluated values. We can observe a decrease in the accuracy as the noise increases. However, even with a very high SNR of 15, the proposed hybrid framework is still able to achieve a better prognostics performance than the purely data-driven model. Therefore, these results demonstrate the robustness of the proposed hybrid prognostics framework.

5.3. Validation of the proposed framework on a popular benchmark dataset

We validate the proposed framework and compare its performance to other deep neural network architectures on the popular CMAPSS dataset [46]. In particular, we focus on the CMAPSS subset *FD002* since it contains data for six operating set-points and is, therefore, a challenging prognostics dataset.[2]

Table 10
Condition monitoring signals - $[w, x_s]$. The variable symbol corresponds to the internal variable name in the CMAPSS model. The descriptions and units are reported as in the model documentation [22]. The Id corresponds to the column index of each feature in [46].

#	Symbol	Description	Units	Id
1	alt	Altitude	ft	3
2	XM	Flight Mach number	–	4
3	TRA	Throttle-resolver angle	%	5
4	T2	Total temperature at fan inlet	°R	6
5	T24	Total temperature at LPC outlet	°R	7
6	T30	Total temperature at HPC outlet	°R	8
7	T50	Total temperature at LPT outlet	°R	9
8	P2	Total pressure at fan inlet	psia	10
9	P15	Total pressure in bypass duct	psia	11
10	Ps30	Static pressure at HPC outlet	psia	17
11	Nf	Physical fan speed	rpm	13
12	Nc	Physical core speed	rpm	14

Table 11
Virtual sensors - $[x_v]$. The variable symbol corresponds to the internal variable name in the CMAPSS model. The descriptions and units are reported as in the model documentation [22]. The Id corresponds to the column index of each feature in [46] .

#	Symbol	Description	Units	Id
1	epr	Engine pressure ratio (P50/P2)	–	15
2	phi	Ratio of fuel flow to Ps30	–	16
3	P30	Total pressure at HPC outlet	psia	12
4	NfR	Corrected fan speed	rpm	18
5	NcR	Corrected core speed	rpm	19
6	BPR	Bypass ratio	–	20
7	farB	Burner fuel–air ratio	–	21
8	hfBleed	Bleed enthalpy	–	22
9	Nf_{dmd}	Demanded corrected fan speed	rpm	23
10	PCNfRdmd	Percent corrected fan speed	pct	24
11	W31	HPT coolant bleed	lbm/s	25
12	W32	HPT coolant bleed	lbm/s	26

5.3.1. CMAPSS dataset

The CMAPSS dataset *FD002* [46] provides degradation trajectories of 519 turbofan engines with unknown and different initial health conditions for six operating set-points and one failure mode affecting the high-pressure compressor (HPC). The dataset was synthetically generated with the Commercial Modular Aero-Propulsion System Simulation (CMAPSS) dynamical model. The training data contains multivariate sensor readings of the complete run-to-failure trajectories from 260 engines. The records stop at the cycle/time the engine failed. A total number of 54k cycles is available for the training. For the test set, truncated time series of various lengths prior to failure are provided for 259 engines. While it is not specifically stated in [46], the CMAPSS data also provides operating condition (w), sensor readings (x_s), and virtual sensors (x_v). Tables 10 and 11 provide an overview of the signals contained in *FD002* that belong to the measured condition monitoring signals $[w, x_s]$ and virtual sensors x_v. So in fact, the dataset can already be considered as a hybrid dataset containing not only information on measured parameters but also a subset of parameters inferred from the dynamical model. The proposed variable split follows the same logic as in the main case study. The condition monitoring signals are measured signals in turbofan engines, which are therefore used for model calibration. The virtual sensors correspond to modeled variables that are not directly measured.

5.3.2. Model calibration

As in the main study, we follow the framework process flow shown in Fig. 4. However, since the dataset FD002 is ten times smaller than the N-CMAPSS, model calibration (i.e., step 1) is performed with

[2] To make this analysis reproducible, the input data used for training the prognostics networks and the Python script implementing the models are made public. https://gitlab.ethz.ch/arimanue/Fusing-physics-based-and-deep-learning-models-for-prognostics.

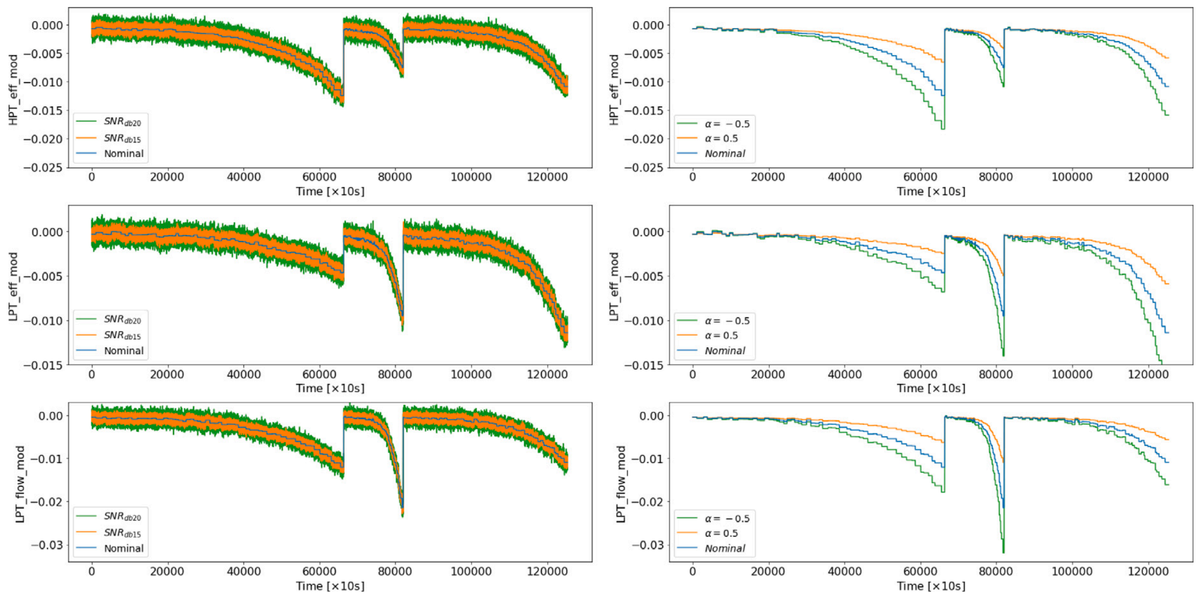


Fig. 10. **(Left)** Noisy calibration factors for a noise level of $\mathrm{SNR}_{db} = 20$ (orange) and $\mathrm{SNR}_{db} = 15$ imposed on the high-pressure turbine (HPT) efficiency, low-pressure turbine (LPT) efficiency, and flow. **(Right)** Biased calibration factors with increasing shift with $\alpha = -0.5$ and $\alpha = 0.5$. Both plots are the θ values for the three units stacked one after the other, generating a single time sequence.

Table 12
Model correcting parameters - [θ]. The variable symbol corresponds to the internal variable name in the CMAPSS model. The descriptions and units are reported as in the model documentation [22].

#	Symbol	Description	Units
1	HPC_eff_mod	HPC efficiency modifier	–
2	HPC_flow_mod	HPC flow modifier	–

the dynamical model F instead of a surrogate model D. Hence, the formulation of the calibration problem corresponds to the state-space formulation according to Eqs. (3) and (4). The estimation of the state vector θ (see Table 12) and state covariance P is carried out with the UKF (see Algorithm 2).

5.3.3. Deep learning prognostics models

Pre-processing. The pre-processing of the input features to the prognostics model follows a three-step process. First, the dataset is normalized with a min/max-normalization to a range $[0, 1]$. Second (only LSTM and CNN models), the dataset is processed with a sliding time window approach of size $N_{tw} = 21$ and stride of 1. We use a window size of 21 cycles/times since the smallest test sample consists of 21 cycles. The RUL label for a sample is then simply the total number of cycles the engine is able to operate minus the cycle in which the window ends. Lastly, the maximum horizon of prediction for RUL, i.e., R_{early}, was limited to 125 cycles following the standard procedure adopted in previous research studies, e.g., [6,47].

Neural network architectures. We tested the same types of deep neural networks as proposed in Section 4.4: a deep feed-forward neural network, a recurrent neural network (LSTM), and a convolutional neural network.

Feed-forward neural network (FNN). The network architecture comprises six fully connected layers. The first four hidden layers have 100 neurons each. The last hidden layer has 50 neurons. A single linear neuron was used in the output layer. *ReLu* activation function was used throughout the entire network. The network has 39k trainable parameters. Motivated by the main study, this final architecture is the result of conducting a grid search with the following search space: number of hidden layers [5-6], number of neurons at each hidden layer [50, 100], and activation function type [*tanh, ReLu*].

One-dimensional convolutional neural network (1D CNN). The network uses five convolutional layers with filters of size three. The first four convolutions have ten channels and the last convolution has only one. Zero padding is used to keep the feature map through the network. The resulting 2D feature map is flattened and the network ends with a 100-way fully connected layer followed by a linear output neuron. The network uses *ReLu* as the activation function. The resulting network has around 4.3k trainable parameters. Motivated by the main study, this final architecture is the result of conducting a grid search with the following search space: number of convolutional hidden layers [5-6], number of channels at each hidden layer [5, 10], filter size [3, 5] and number of neurons at the fully connected layer [50, 100].

Long short-term memory recurrent neural network (LSTM). Following the same architecture as for the 1D CNN network, a five-layer LSTM followed by one fully connected layer with *ReLu* as the activation function was applied. Through grid search, we determined the optimal number of cells in the hidden layers to be 10. The resulting network has 6.4k trainable parameters.

Training set-up. The optimization of the network's weights is carried out with mini-batch stochastic gradient descent and with the *Adam* algorithm [43]. The *Xavier* initializer [44] is used for the weight initializations. The batch size is set to 512 and the learning rate to 0.001. The maximum number of epochs was set to 300. Early stopping with a patience of 30 epochs is applied. A validation set containing 10% of the training data was used for hyper-parameter tuning.

5.3.4. Results

Model Calibration. Fig. 11 shows the inferred unobserved model parameters $\hat{\theta}$ obtained for the 260 units within the training dataset. The state estimates obtained with the UKF unveil the degradation of the engine in the form of performance deficit in the high-pressure compressor (HPC) flow and efficiency. Hence, we observe that the UKF is able to identify and track degradation of the simulated failure mode. The different training units have a different initial health state and different evolution of the degradation over time. The UKF performance on test and training units shows similar results.

Prognostics. Table 13 shows the performance of the FNN, LSTM, and 1D CNN models for purely data-driven and hybrid approaches and compares them to the results reported in the latest, best-performing published papers on this dataset. As in the main case study, the overview

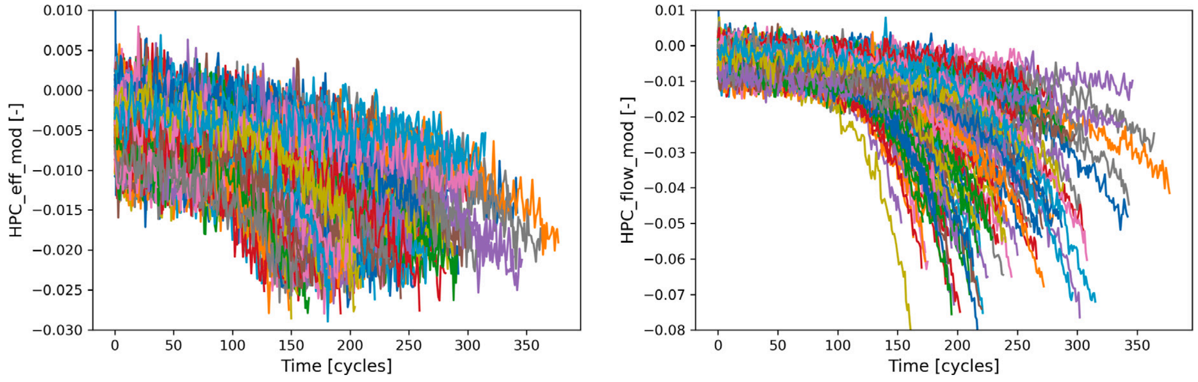


Fig. 11. Traces of HPC efficiency and flow scalers for the run-to-failure trajectories of 260 training units with the UKF.

Table 13
Overview of the RMSE metrics with hybrid, baseline approaches. Mean and standard deviation results with FNN, LSTM, and CNN models over five runs.

Purely Data-driven with Input $[w, x_s]$	
Model	RMSE
FNN model - This work	23.3 ± 0.5
LSTM model - This work	19.6 ± 0.2
CNN model - This work	19.5 ± 0.3
Hybrid with Input $[w, x_s, \hat{x}_v]$	
Dilated CNN [49]	24.5
GCU-Transformer [50]	22.8
Ensemble ResCNN [51]	20.9
GNMR [52]	20.9
RNN autoencoder [9]	19.6
FNN model - This work	18.3 ± 0.3
Attention LSTM [7]	17.7 ± 0.5
LSTM model - This work	17.7 ± 0.5
CNN model - This work	17.6 ± 0.3
CapNet [48]	16.3 ± 0.2
Hybrid with Input $[w, x_s, \hat{x}_v, \hat{x}_s, \hat{\theta}]$	
Model Name	RMSE
FNN model - This work	17.6 ± 0.2
LSTM model - This work	16.9 ± 0.4
CNN model - This work	**16.0** ± 0.2

shown in Table 13 is divided into three sections depending on the input features used in the prognostics network (i.e., purely data-driven $[w, x_s]$, hybrid with virtual sensors $[w, x_s, x_v]$, and the proposed hybrid approach $[w, x_s, x_v, \hat{x}_s, \hat{\theta}]$). It is worth noticing that the variable split considered in the work has not been previously considered in the literature. Previous works assume no difference between measured sensor readings and virtual sensors. The proposed method scores an RMSE value of 16.0, slightly below the best-performing solution in this dataset [48]. With an improvement ranging from 18% to 24%, depending on the applied network architecture, the proposed hybrid approach shows similar improvements to the purely data-driven approach as those seen in the main case study.

Fig. 12 shows the actual (red) and predicted (blue) RUL of the 1D CNN model. It can be observed that the predictions are very accurate for engines close to failure (RUL < 40). The prediction error increases for engines far from failure.

6. Discussion

The implementation of hybrid approaches requires a careful design to avoid inheriting drawbacks from the methods being combined (i.e., data-driven and physics-based methods). As stated in the introduction and demonstrated in the case studies, the proposed approach is designed to minimize each method's individual weaknesses and leverage their advantages. In particular, this includes the selection of a

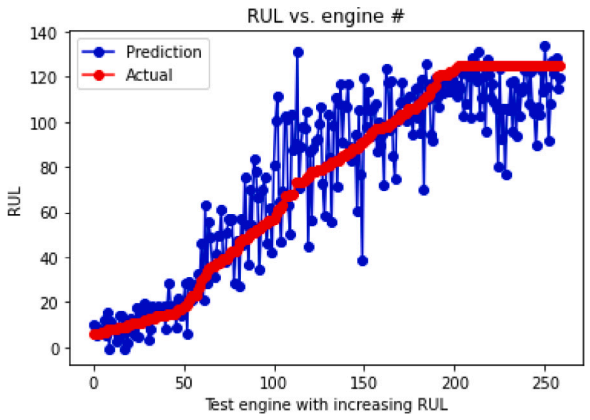


Fig. 12. True RUL (red) and the predicted RUL (blue) with 1D CNN model. The x-axis corresponds to test units with decreasing RUL.

0D/1D system performance model as a physics-based model allows one to infer informative features of the degradation without the need for a detailed model of the physics of failure and a high computational load. Instead, a deep learning model assumes the task of discovering the RUL model from the physics-inferred features and CM data. The inferred features are informative of the degradation and, as a result, increase the accuracy and generalization error of the RUL model compared to a purely data-driven model. In summary, the proposed hybrid methodology provides the following advantages compared to purely data-driven approaches:

- Ability to accurately predict the remaining useful lifetime, even when available datasets to train data-driven approaches are sparse.
- Interpretability of the model and the corresponding outputs in terms of physically meaningful input features.
- Robustness to sensor faults that can be distinguished against faulty conditions. A model calibration step acts as a noise filter of the condition monitoring data. Therefore, it allows for the identification of sensor readings that are inconsistent with others given the prior uncertainty of the measurements and model parameters [25].
- Reduction of the required size of the training dataset while yielding similar or better performance. As shown in the ablation study I, an improved generalization of the hybrid approach makes the RUL model less dependent on the amount of data for accurate RUL prediction.
- Ability to generate additional operating conditions to compensate for the lacking CM data.

However, to some extent, certain limitations persist. We list below some limitations of the proposed framework and suggest future research directions that might serve to mitigate these shortcomings.

- One anticipated limitation pertains to RUL prediction on degradation trajectories rooted in failure modes that are not covered in the training dataset. For instance, vibration- or oil degradation-related failure modes are not represented in the training dataset of the case study and consequently cannot be accurately predicted. This is a general issue for modes that do not model the physics of failure.
- Another expected limitation applies to the prediction of RUL on test data that involves distant extrapolations on the operative conditions (w) or model parameters (θ) outside the training distribution. This is a general limitation of data-driven models and can potentially be partially mitigated with domain adaptation [36,37], self-learning techniques, tailoring [53], generating synthetic data with the physics-based model, or strong physics-driven inductive bias.
- The interpretability of failure modes is limited to the representation provided by the physics-based model. For instance, in the absence of vibration-specific instrumentation, a degradation mode resulting in high vibrations can only be related to changes in the health parameters θ, virtual sensors x_v, or sensor readings x_s.

The proposed approach assumes that the system model representation is complete. That is, there is no substantial missing physical representation of the dynamic system F that would make the model $F(w, \theta)$ unable to approximate the true system response closely for some unknown $\hat{\theta}$ function. This is a common situation when considering system performance models of critical systems and mature products where the system model has been developed and validated based on multiple field units or testbed units (such as those in which there is sufficient CM data to attempt a data-driven RUL model). Moreover, in mature products, the system performance is also expected to be an accurate representation of the performance of a new/healthy system. In this case, deviations in θ represent the degradation of the sub-components that explains the degraded performance of the system. In contrast, for cases with a certain degree of missing physical representation, the impact of model degradation becomes entangled with the model correction rooted in an inadequate modeling of the physical process. This entanglement has two consequences: (1) the information that $\hat{\theta}$ carries about degradation decreases and (2) the interpretability of $\hat{\theta}$ as the true causal factors of degradation is lost. However, as demonstrated in the ablation study III, the proposed framework is robust to uncertainly in the model calibration step as long as the training dataset is representative of the test conditions (i.e., the same missing physics also applies to the training units).

It is worth pointing out that, for complex system models that have the structure shown in 2, the missing physics is generally only a concern for the sub-components. In other words, since the overall system topology and the physics coupling the sub-components are known, there is no relevant structural uncertainty concerning the overall system. In this case, the discrepancy between the system model response and the true system is rooted in an inaccurate representation of the real physical processes occurring at the level of its sub-components. An approximation of the overall system performance is therefore possible regardless of the fidelity level of the sub-component models. Model calibration with structural model uncertainly could form the basis of a future research question.

7. Conclusions

This paper proposes a hybrid framework fusing information from physics-based performance models with deep learning algorithms for predicting the remaining useful lifetime of complex systems. Health-related model parameters are inferred by solving a calibration problem. Subsequently, this information is combined with sensor readings and used as input to a deep neural network to develop a reliable hybrid prognostics model.

The performance of the hybrid framework was evaluated on a synthetic dataset comprising run-to-failure degradation trajectories from a small fleet of nine turbofan engines operating under real flight conditions. The dataset was generated with the Commercial Modular Aero-Propulsion System Simulation (CMAPSS) dynamic model. The proposed hybrid framework clearly outperforms the alternative data-driven prognostic model in which only sensor data is used as input to the deep neural network. The hybrid framework provides accurate and robust predictions of the failure time, t_{EOL} (i.e., $\epsilon \pm 5$ cycles), while extending the prediction horizon by 127% on average as compared to the pure data-driven methods. More importantly, the proposed framework requires less training data compared to the purely data-driven algorithms. The proposed framework has also been demonstrated on the popular CMAPSS dataset [46] and compared to other deep neural network architectures. The proposed hybrid approach shows similar improvements over the purely data-driven methods as in the main case study and similar predictive performance to the current best-performing solution on this dataset.

As demonstrated in the experiments, the performance of the proposed hybrid prognostics framework is robust to uncertainty imposed by the quality of the model calibration. This research study has also demonstrated the ability of the developed hybrid framework to provide excellent prognostics performance for units that exhibited operating conditions very dissimilar to those of units used for training the algorithms (limited representativeness of the training dataset for the testing dataset). Therefore, the proposed hybrid framework represents a promising direction for further research in PHM applications.

Finally, a potential future research direction is evaluating the transferability of the proposed hybrid approach to other types of problems that fulfill the same criteria: the availability of complete physics-based models and access to sensor readings that provide information about the system state. Furthermore, an additional avenue for future research is developing approaches for which only limited physics-based information is available.

CRediT authorship contribution statement

Manuel Arias Chao: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Writing – original draft preparation, Writing – review & editing, Visualization. **Chetan Kulkarni:** Conceptualization, Writing – review & editing. **Kai Goebel:** Conceptualization, Writing – review & editing, Supervision. **Olga Fink:** Conceptualization, Resources, Writing – review & editing, Supervision, Project administration, Funding acquisition.

Declaration of competing interest

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

Acknowledgments

This research was funded by the Swiss National Science Foundation (SNSF) Grant no. PP00P2 176878. The contributions of Chetan Kulkarni were performed under NASA Ames Research Center, Contract No. 80ARC020D0010.

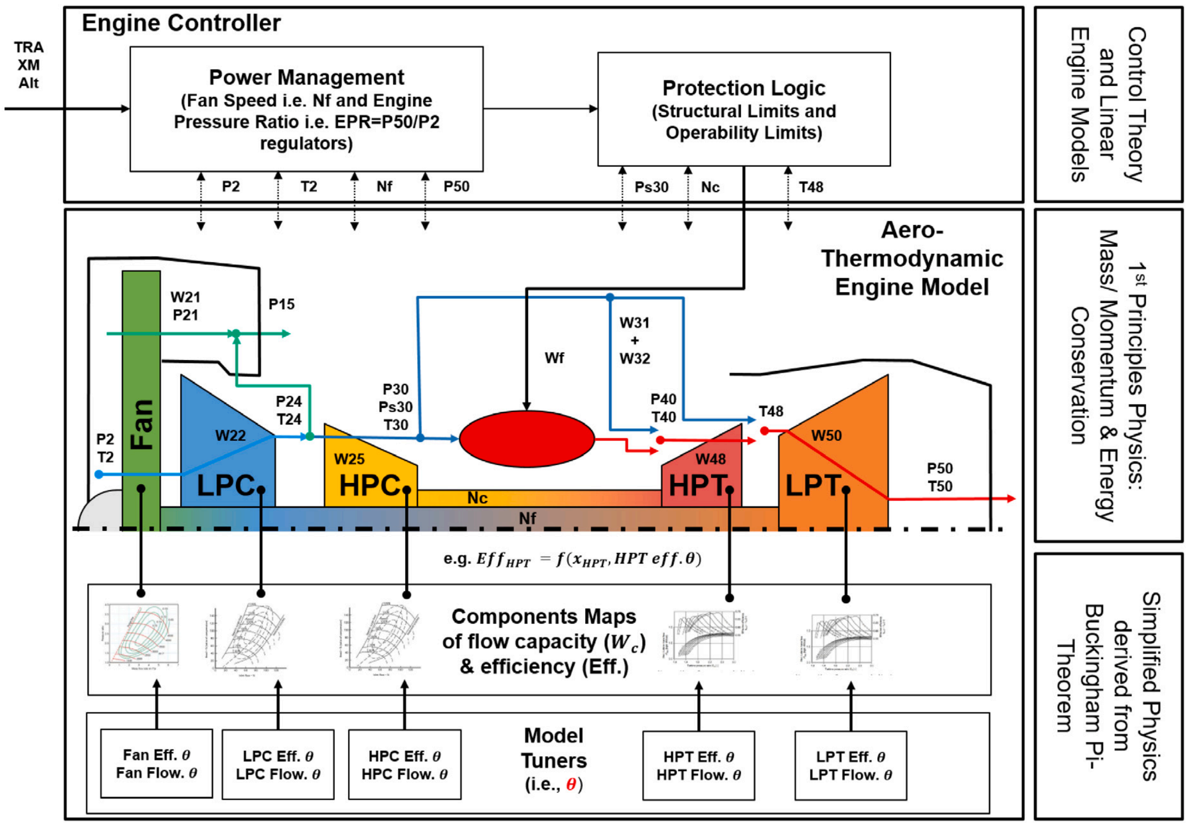

Engine Controller
TRA
XM
Alt
Power Management
(Fan Speed i.e. Nf and Engine Pressure Ratio i.e. EPR=P50/P2 regulators)
Protection Logic
(Structural Limits and Operability Limits)
P2
T2
Nf
P50
Ps30
Nc
T48
Control Theory and Linear Engine Models
Aero-Thermodynamic Engine Model
1st Principles Physics: Mass/ Momentum & Energy Conservation
W21
P21
P15
Fan
P2
T2
W22
LPC
P24
T24
P30
Ps30
T30
W25
HPC
Wf
W31 + W32
P40
T40
T48
W48
HPT
W50
LPT
P50
T50
Nc
Nf
e.g. $Eff_{HPT} = f(x_{HPT}, HPT\ eff.\theta)$
Components Maps of flow capacity (W_c) & efficiency (Eff.)
Model Tuners (i.e., θ)
Fan Eff. θ
Fan Flow. θ
LPC Eff. θ
LPC Flow. θ
HPC Eff. θ
HPC Flow. θ
HPT Eff. θ
HPT Flow. θ
LPT Eff. θ
LPT Flow. θ
Simplified Physics derived from Buckingham Pi-Theorem

Fig. A.13. Schematic representation of the CMAPSS model. The CMAPSS model includes two types of models: (1) a transient thermodynamic performance engine model and (2) a power-management model that allows the engine to be operated over a wide range of thrust levels spanning the full spectrum of flight conditions. The thermodynamic model has the form of a coupled system of nonlinear equations resulting from energy, momentum, and mass conservation throughout the turbofan engine. The resulting model is nonlinear and does not have an explicit formula. The thermodynamic model uses traditional off-design performance modeling approaches, common in gas turbines, that resort to 'component maps' of the rotating components. Refs. [54–56] provide the interested reader with a full description of the physics and an implementation close to that found in CMAPSS. The power-management i.e., control, uses a discrete state-space formulation in combination with linear engine models.

Appendix A. CMAPSS engine model

Although most of the implementation details of the CMAPSS engine model used in this work are not publicly available, the user manual [22] and the documentation of the similar model C-MAPSS40k [57] provide some details about the software and control implementation. In brief, the CMAPSS engine model represents a generic, high-bypass, twin-spool commercial turbofan engine. The engine consists of six main components: fan, low-pressure compressor (LPC), high-pressure compressor (HPC), combustor or burner, high-pressure turbine (HPT), and low-pressure turbine (LPT). The HPC and HPT are connected through the core shaft or high-speed shaft; the fan, LPC, and LPT are all connected to the fan shaft or low-speed shaft [57]. In addition to these turbo-machinery components and the combustor, the engine has an inlet at the front, a nozzle at the rear, a bypass duct, a variable-sized inter-stage bleed valve, a set of variable-angle stator or guide vanes, and a number of cooling bleeds. A schematic of the engine is shown in Fig. A.13. The CMAPSS model includes two types of models: (1) a transient aero-thermodynamic engine model and (2) an engine controller that allows the engine to be operated over a wide range of thrust levels spanning the full spectrum of flight conditions.

Aero-thermodynamic Engine Model. The aero-thermodynamic engine model is a physics-based, component-level model where each of the engine components is represented as an infinitesimally small volume [57]. The engine is then balanced by mass flow rate continuity through the components. The thermodynamic model uses traditional off-design performance modeling approaches, common in gas turbines, that resort to 'component maps' of the rotating components. Thereby, this modeling strategy captures the gross characteristics of each engine component. Refs. [54–56] provide for the interested reader a full description of the physics and an implementation close to that found in CMAPSS.

Engine Controller. The controller converts the throttle command from the pilot into thrust while providing safe operation. The controller has two main components: the power management and protection logic controllers. The power management controller determines an engine pressure ratio (EPR), which is the ratio of turbine exit pressure to inlet pressure (P50/P2), or fan speed (Nf) setpoint based on the PLA, altitude, Mach number, and ambient temperature that, when achieved, results in the desired linear thrust profile. Protection logic aims at providing safe and smooth thrust transitions between distant setpoints. Safe operation implies (1) the protection of physical components by preventing overstress and (2) the avoidance of operability limits (e.g., compressor surge, stall, combustor lean blow-out). This protection is achieved through the use of controller limits on physical variables including limits on the maximum fan speed (Nf), core speed (Nc), and burner pressure (Ps3). The control model resorts to common industry control theory and linear engine models.

Appendix B. Engine dynamic model - *D*

The system dynamics are approximated with an MLP with four layers ($L = 4$). The hidden layers have 100 units ($n^1 = n^2 = n^3 = 100$). The output layer has the dimension of the sensor reading vector (i.e., $n^L = n$). *ReLU* activation function was used throughout the hidden layers. For the output layer, $\sigma^L = I$ is the identity.

Appendix C. The UKF algorithm

The solution of the calibration problem with the UKF requires a state-update formulation where the state equation is modeled as a random walk. However, the estimation of the state (mean vector $\hat{\theta}^{(k)}$ and covariance matrix $P^{(k)}$) at each time step is obtained with the UKF. Concretely, it resorts to the standard UKF update process given by Algorithm 2.

The UKF algorithm also requires the definition of the diagonal covariance matrices Q and R. We assumed the covariance matrices to be diagonal matrices with normalized standard deviation $r = 0.01$ and $q = 0.01$ (i.e., $Q = q^2\mathbf{I_n}$ and $R = r^2\mathbf{I_d}$ where $\mathbf{I_k}$ is the identity matrix of dimension k and d is the dimension of θ).

Algorithm 2: Unscented Kalman filter for health parameter estimation

```
1  Estimation of health parameters
       Input : $\{w^{(k)}, x^{(k)}, \}_{k=1}^{m}$, $\theta_{ref}$, R and $P^{(0)} = Q$ & F
       Output: $\hat{x}^{(k)}, \hat{\theta}^{(k)}, P^{(k)}$
2      for $k = 1 : m$ do
3          $\bar{P}^{(k)} = P^{(k-1)} + Q^{(k)}$
4          $S^{(k-1)} = [\hat{\theta}^{(k-1)}, \hat{\theta}^{(k-1)} + \gamma\sqrt{\bar{P}^{(k)}}, \hat{\theta}^{(k-1)} - \gamma\sqrt{\bar{P}^{(k)}}]$ i.e. Sigma
           points (vectors)
5          $\mathcal{X}_i^{(k)} = F(w^{(k)}, S_i^{(k-1)})$ for $i = 0, 1, \dots, 2d$ Propagate $S$
           through the non-linear function $F$
6          $\hat{x}_s^{(k)} = \sum_{i=0}^{2d} W_i^m \mathcal{X}_i^{(k)}$
7          $r^{(k)} = x_s^{(k)} - \hat{x}_s^{(k)}$
8          $P_y^{(k)} = \sum_{i=0}^{2d} W_i^c (\mathcal{X}_i^{(k)} - \hat{x}_s^{(k)})(\mathcal{X}_i^{(k)} - \hat{x}_s^{(k)})^T + R$
9          $P_{\theta y}^{(k)} = \sum_{i=0}^{2d} W_i^c (S^{(k-1)} - \hat{\theta}^{(k-1)})(\mathcal{X}_i^{(k)} - \hat{x}_s^{(k)})^T$
10         $K^{(k)} = P_{\theta y}^{(k)} P_y^{(k)^{-1}}$
11         $\hat{\theta}^{(k)} = \hat{\theta}^{(k)} - K^{(k)} r^{(k)}$
12         $P^{(k)} = \bar{P}^{(k)} - K^{(k)} P_y^{(k)} K^{(k)^T}$
13     end
14 end
```

The weight coefficients W_0^m, W_0^c and W_i^c and the corresponding scaling parameters are given in Table C.14.

Table C.14
Weight coefficients and scaling parameters

Weight coefficients	
Weights	**Expression**
W_0^m	$\frac{\lambda}{d+\lambda}$
W_0^c	$W_0^m + 1 - \alpha^2 + \beta$
$W_i^c = W_i^m$	$\frac{1}{2(d+\lambda)} \quad \forall i = 1, \dots 2n_\theta$
Scaling parameters	
Parameters	**Values**
γ	$\sqrt{d+\lambda}$
λ	$\alpha^2(d+\kappa) - d$
α	1e−3
β	2
κ	0

References

[1] Bolander N, Qiu H, Eklund N, Hindle E, Rosenfeld T. Physics-based remaining useful life prediction for aircraft engine bearing prognosis. Tech. rep., 2009.

[2] Daigle MJ, Goebel K. A model-based prognostics approach applied to pneumatic valves. Int J Progn Health Manag 2011;2:1–16.

[3] Daigle MJ, Goebel K. Model-based prognostics with concurrent damage progression processes. IEEE Trans Syst Man Cybern A 2013;43(3):535–46. http://dx.doi.org/10.1109/TSMCA.2012.2207109, URL http://ieeexplore.ieee.org/document/6301756/.

[4] Schwabacher M, Goebel K. A survey of artificial intelligence for prognostics. In: Association for the advancement of artificial intelligence AAAI fall symposium 2007. 2007. p. 107–14.

[5] Khan S, Yairi T. A review on the application of deep learning in system health management. Mech Syst Signal Process 2018;107:241–65. http://dx.doi.org/10.1016/j.ymssp.2017.11.024, URL https://www.sciencedirect.com/science/article/pii/S0888327017306064.

[6] Li X, Ding Q, Sun J-Q. Remaining useful life estimation in prognostics using deep convolution neural networks. Reliab Eng Syst Saf 2018;172:1–11. http://dx.doi.org/10.1016/j.ress.2017.11.021, URL https://www.sciencedirect.com/science/article/pii/S0951832017307779.

[7] de Oliveira da Costa PR, Akcay A, Zhang Y, Kaymak U. Attention and long short-term memory network for remaining useful lifetime predictions of turbofan engine degradation. Int J Progn Health Manag 2019;10.

[8] Listou Ellefsen A, Bjørlykhaug E, Æsøy V, Ushakov S, Zhang H. Remaining useful life predictions for turbofan engine degradation using semi-supervised deep architecture. Reliab Eng Syst Saf 2019;183:240–51. http://dx.doi.org/10.1016/j.ress.2018.11.027, URL http://www.sciencedirect.com/science/article/pii/S0951832018307506.

[9] Yu W, Kim IY, Mechefske C. An improved similarity-based prognostic algorithm for RUL estimation using an RNN autoencoder scheme. Reliab Eng Syst Saf 2020;199:106926. http://dx.doi.org/10.1016/j.ress.2020.106926, URL http://www.sciencedirect.com/science/article/pii/S0951832019307902.

[10] Shi Z, Chehade A. A dual-LSTM framework combining change point detection and remaining useful life prediction. Reliab Eng Syst Saf 2021;205:107257. http://dx.doi.org/10.1016/j.ress.2020.107257, URL http://www.sciencedirect.com/science/article/pii/S0951832020307572.

[11] Saxena A, Goebel K. Turbofan engine degradation simulation data set. CA: NASA Ames Research Center, NASA Ames Prognostics Data Repository, Moffett Field; 2008, URL https://ti.arc.nasa.gov/tech/dash/groups/pcoe/prognostic-data-repository/.

[12] Biggio L, Kastanis I. Prognostics and health management of industrial assets: Current progress and road ahead. Front Artif Intell 2020;3:88.

[13] Fink O, Wang Q, Svensén M, Dersin P, Lee W-J, Ducoffe M. Potential, challenges and future directions for deep learning in prognostics and health management applications. Eng Appl Artif Intell 2020;92:103678. http://dx.doi.org/10.1016/j.engappai.2020.103678, URL https://www.sciencedirect.com/science/article/pii/S0952197620301184.

[14] Arias Chao M, Kulkarni CS, Goebel K, Fink O. Hybrid deep fault detection and isolation: Combining deep neural networks and system performance models. Int J Progn Health Manag 2019;10:1–19.

[15] Rai R, Sahu CK. Driven by data or derived through physics? A review of hybrid physics guided machine learning techniques with cyber-physical system (CPS) focus. IEEE Access 2020;8:71050–73. http://dx.doi.org/10.1109/ACCESS.2020.2987324.

[16] Willard J, Jia X, Xu S, Steinbach M, Kumar V. Integrating physics-based modeling with machine learning: A survey. 2020, arXiv:2003.04919.

[17] Zhang D, Dey S, Perez HE, Moura SJ. Remaining useful life estimation of lithium-ion batteries based on thermal dynamics. In: 2017 american control conference. 2017, p. 4042–7. http://dx.doi.org/10.23919/ACC.2017.7963575.

[18] Nascimento RG, Viana FA. Fleet prognosis with physics-informed recurrent neural networks. In: Structural health monitoring 2019 - Proceedings of the 12th international workshop on structural health monitoring, vol. 2. 2019, p. 1740–7, arXiv:1901.05512.

[19] Dourado A, Viana FAC. Physics-informed neural networks for corrosion-fatigue prognosis. In: Annual conference of the PHM society, vol. 11, no. 1. 2019.

[20] Yucesan YA, Viana FAC. Wind turbine main bearing fatigue life estimation with physics-informed neural networks. In: Annual conference of the PHM society, vol. 11, no. 1. p. 1–14. 2019.

[21] Jia X, Karpatne A, Willard J, Steinbach M, Read J, Hanson PC, et al. Physics guided recurrent neural networks for modeling dynamical systems: Application to monitoring water temperature and quality in lakes. Tech. rep., 2018, p. 3, arXiv:1810.02880, URL http://arxiv.org/abs/1810.02880.

[22] Frederick DK, Decastro JA, Litt JS. User's guide for the commercial modular aero-propulsion system simulation (C-MAPSS). Tech. rep., 2007.

[23] Arias Chao M, Kulkarni C, Goebel K, Fink O. Aircraft engine run-to-failure dataset under real flight conditions for prognostics and diagnostics. Data 2021;6(1):5. http://dx.doi.org/10.3390/data6010005, URL https://www.mdpi.com/2306-5729/6/1/5.

[24] Roth BA, Doel DL, Cissell JJ. Probabilistic matching of turbofan engine performance models to test data. In: Proceedings of the ASME turbo expo, vol. 1. American Society of Mechanical Engineers Digital Collection; 2005, p. 541–8. http://dx.doi.org/10.1115/GT2005-68201.

[25] Arias Chao M, Lilley DS, Mathé P, Schloßhauer V. Calibration and uncertainty quantification of gas turbine performance models. In: Proceedings of the ASME turbo expo, vol. 7A. 2015, http://dx.doi.org/10.1115/gt2015-42392, V07AT29A001.

[26] Kennedy MC, O'Hagan A. Bayesian calibration of computer models. J R Stat Soc Ser B Stat Methodol 2001;63(3):425–64. http://dx.doi.org/10.1111/1467-9868.00294, URL http://doi.wiley.com/10.1111/1467-9868.00294.

[27] Sacks J, Welch WJ, Mitchell TJ, Wynn HP. Design and analysis of computer experiments. Statist Sci 1989;4(4):409–23. http://dx.doi.org/10.1214/ss/1177012413.

[28] Crassidis JL, Junkins JL. Optimal estimation of dynamic systems. Applied mathematics & nonlinear science, 2nd ed. Chapman & Hall/CRC; 2011.

[29] Julier SJ, Uhlmann JK. New extension of the Kalman filter to nonlinear systems. In: Kadar I, editor. Signal processing, sensor fusion, and target recognition VI, vol. 3068. International Society for Optics and Photonics, SPIE; 1997, p. 182–93. http://dx.doi.org/10.1117/12.280797.

[30] Turner R, Rasmussen CE. Model based learning of sigma points in unscented Kalman filtering. In: Proceedings of the 2010 IEEE international workshop on machine learning for signal processing. 2010, p. 178–83. http://dx.doi.org/10.1109/MLSP.2010.5589003.

[31] Borguet S. Variations on the Kalman filter for enhanced performance monitoring of gas turbine engines [Ph.D. thesis], Université de Liège; 2012.

[32] Kantas N, Doucet A, Singh SS, Maciejowski J, Chopin N. On particle methods for parameter estimation in state-space models. Statist Sci 2015;30(3):328–51. http://dx.doi.org/10.1214/14-STS511.

[33] Rutter C, Miglioretti D, Savarino J. Bayesian calibration of microsimulation models. J Amer Statist Assoc 2009;104(488):1338–50. http://dx.doi.org/10.1198/jasa.2009.ap07466.

[34] Tian Y, Chao MA, Kulkarni C, Goebel K, Fink O. Real-time model calibration with deep reinforcement learning. 2020, arXiv:2006.04001.

[35] Ellefsen A, Ushakov S, Æsøy V, Zhang H. Validation of data-driven labeling approaches using a novel deep network structure for remaining useful life predictions. IEEE Access 2019;7:71563–75. http://dx.doi.org/10.1109/ACCESS.2019.2920297.

[36] de Oliveira da Costa PR, Akçay A, Zhang Y, Kaymak U. Remaining useful lifetime prediction via deep domain adaptation. Reliab Eng Syst Saf 2020;195:106682. http://dx.doi.org/10.1016/j.ress.2019.106682, URL http://www.sciencedirect.com/science/article/pii/S0951832019304946.

[37] Wang Q, Michau G, Fink O. Missing-class-robust domain adaptation by unilateral alignment. IEEE Trans Ind Electron 2021;68(1):663–71. http://dx.doi.org/10.1109/TIE.2019.2962438.

[38] Li X, Zhang W, Ding Q. Deep learning-based remaining useful life estimation of bearings using multi-scale feature extraction. Reliab Eng Syst Saf 2019;182:208–18. http://dx.doi.org/10.1016/j.ress.2018.11.011, URL https://www.sciencedirect.com/science/article/pii/S0951832018308299.

[39] Yang H, Zhao F, Jiang G, Sun Z, Mei X. A novel deep learning approach for machinery prognostics based on time windows. Appl Sci 2019;9(22):4813. http://dx.doi.org/10.3390/app9224813, URL https://www.mdpi.com/2076-3417/9/22/4813.

[40] Pasa GD, de Medeiros IP, Yoneyama T. Operating condition-invariant neural network-based prognostics methods applied on turbofan aircraft engines. In: Proceedings of the annual conference of the PHM society, vol. 11, no. 1. 2019.

[41] Kiranyaz S, Ince T, Abdeljaber O, Avci O, Gabbouj M. 1-D convolutional neural networks for signal processing applications. In: 2019 IEEE international conference on acoustics, speech and signal processing. 2019, p. 8360–4. http://dx.doi.org/10.1109/ICASSP.2019.8682194.

[42] Ji S, Han X, Hou Y, Song Y, Du Q. Remaining useful life prediction of airplane engine based on PCA–BLSTM. Sensors 2020;20(16). http://dx.doi.org/10.3390/s20164537, URL https://www.mdpi.com/1424-8220/20/16/4537.

[43] Kingma DP, Ba JL. Adam: A method for stochastic optimization. In: 3rd international conference on learning representations - Conference Track Proceedings. 2015, arXiv:1412.6980.

[44] Glorot X, Bengio Y. Understanding the difficulty of training deep feedforward neural networks. Tech. rep., 2010, p. 249–56.

[45] Saxena A, Celaya J, Saha B, Saha S, Goebel K. Metrics for offline evaluation of prognostic performance. Int J Progn Health Manag 2010.

[46] Saxena A, Goebel K, Simon D, Eklund N. Damage propagation modeling for aircraft engine run-to-failure simulation. In: 2008 international conference on prognostics and health management. IEEE; 2008, p. 1–9. http://dx.doi.org/10.1109/PHM.2008.4711414.

[47] Malhotra P, TV V, Ramakrishnan A, Anand G, Vig L, Agarwal P, et al. Multi-sensor prognostics using an unsupervised health index based on LSTM encoder-decoder. In: 1st ACM SIGKDD workshop on machine learning for prognostics and health management. San Francisco: 2016. p. 10. arXiv:1608.06154, URL http://arxiv.org/abs/1608.06154.

[48] Palazuelos AR-T, Droguett EL, Pascual R. A novel deep capsule neural network for remaining useful life estimation. Proc. Inst. Mech. Eng. O 2020;234(1):151–67. http://dx.doi.org/10.1177/1748006X19866546, arXiv:https://doi.org/10.1177/1748006X19866546.

[49] Xu X, Wu Q, Li X, Huang B. Dilated convolution neural network for remaining useful life prediction. J Comput Inf Sci Eng 2020;20(2):021004.

[50] Mo Y, Wu Q, Li X, Huang B. Remaining useful life estimation via transformer encoder enhanced by a gated convolutional unit. J Intell Manuf 2021;1–10.

[51] Wen L, Dong Y, Gao L. A new ensemble residual convolutional neural network for remaining useful life estimation. Math Biosci Eng 2019;16(2):862–80.

[52] Narwariya J, Malhotra P, TV V, Vig L, Shroff G. Graph neural networks for leveraging industrial equipment structure: An application to remaining useful life estimation. 2020, arXiv preprint arXiv:2006.16556.

[53] Alet F, Kawaguchi K, Bauza M, Kuru NG, Lozano-Perez T, Kaelbling LP. Tailoring: encoding inductive biases by optimizing unsupervised objectives at prediction time. 2020, arXiv:2009.10623.

[54] Razak A. 11 - Gas turbine performance modelling, analysis and optimisation. In: Jansohn P, editor. Modern gas turbine systems. Woodhead publishing series in energy, Woodhead Publishing; 2013, p. 423–514. http://dx.doi.org/10.1533/9780857096067.3.423, URL https://www.sciencedirect.com/science/article/pii/B9781845697280500112.

[55] Saravanamuttoo H, Rogers G, Cohen H. Gas turbine theory. Prentice Hall; 2001.

[56] Walsh PP, Fletcher P. Gas Turbine Performance. Wiley; 2004, http://dx.doi.org/10.1002/9780470774533, URL https://onlinelibrary.wiley.com/doi/book/10.1002/9780470774533.

[57] May R, Csank J, Lavelle T, Litt J, Guo T-H. A high-fidelity simulation of a generic commercial aircraft engine and controller. In: 46th AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit. p. 6630.