Deep reinforcement learning for predictive aircraft maintenance using probabilistic Remaining-Useful-Life prognostics

Lee, J.; Mitici, M.A.

DOI
10.1016/j.ress.2022.108908

Publication date
2022

Document Version
Final published version

Published in
Reliability Engineering & System Safety

Reliability Engineering and System Safety 230 (2023) 108908

Contents lists available at ScienceDirect

Reliability Engineering and System Safety

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

Deep reinforcement learning for predictive aircraft maintenance using probabilistic Remaining-Useful-Life prognostics

Juseong Lee [a,*], Mihaela Mitici [b]

[a] Faculty of Aerospace Engineering, Delft University of Technology, Kluyverweg 1, 2926 HS, Delft, The Netherlands
[b] Faculty of Science, Utrecht University, Heidelberglaan 8, 3584 CS, Utrecht, The Netherlands

ARTICLE INFO

Keywords:
Predictive maintenance
Remaining useful life
Probabilistic RUL prognostics
Deep reinforcement learning
Turbofan engines

ABSTRACT

The increasing availability of sensor monitoring data has stimulated the development of Remaining-Useful-Life (RUL) prognostics and maintenance planning models. However, existing studies focus either on RUL prognostics only, or propose maintenance planning based on simple assumptions about degradation trends. We propose a framework to integrate data-driven probabilistic RUL prognostics into predictive maintenance planning. We estimate the distribution of RUL using Convolutional Neural Networks with Monte Carlo dropout. These prognostics are updated over time, as more measurements become available. We further pose the maintenance planning problem as a Deep Reinforcement Learning (DRL) problem where maintenance actions are triggered based on the estimates of the RUL distribution. We illustrate our framework for the maintenance of aircraft turbofan engines. Using our DRL approach, the total maintenance cost is reduced by 29.3% compared to the case when engines are replaced at the mean-estimated-RUL. In addition, 95.6% of unscheduled maintenance is prevented, and the wasted life of the engines is limited to only 12.81 cycles. Overall, we propose a roadmap for predictive maintenance from sensor measurements to data-driven probabilistic RUL prognostics, to maintenance planning.

1. Introduction

Modern aircraft are equipped with multiple sensors that generate large volumes of health monitoring measurements for aircraft systems and components. For example, for a Boeing 787, approximately 1000 parameters are continuously monitored for the engine, amounting to a total of 20 terabytes of data per flight hour [1]. Such data are the basis for Remaining-Useful-Life (RUL) estimation [2] and predictive aircraft maintenance planning [3]. In this paper, we are interested in integrating RUL prognostics into predictive aircraft maintenance. Below we discuss relevant, recent studies on RUL prognostics and maintenance planning.

Relevant studies on RUL prognostics

Many studies have focused in the last years on developing RUL prognostics for aircraft components and systems [4]. For example, RUL prognostics for aircraft landing gear brakes are developed using stochastic regression models in [5]. The RUL of aircraft cooling units is estimated using particle filtering in [6]. RUL prognostics for electro-mechanical actuators are obtained using a Gaussian process regression in [7]. RUL prognostics have been developed also for components such as batteries [8,9], and fuel cells [10]. Several studies have developed RUL prognostics for turbofan engines using the C-MAPSS data set [11]. Because the degradation of a turbofan engine is non-linear, many such RUL prognostics models are based on deep learning models, such as convolutional neural networks (CNNs) [12], deep convolutional neural network (DCNN) [13], multi-scale DCNN [14], and CNN with pooling [15,16]. Recently, deep learning models have been improved further by using a hybrid approach of deep learning models and physics-based models [17]. All these studies, however, predict RUL as a point estimate, i.e., a single value of RUL. For predictive maintenance planning, quantifying the uncertainty associated with the estimated RUL is a prerequisite [18].

In this paper, we develop RUL prognostics for the turbofan engines of the C-MAPSS data set [11] using neural networks, together with quantifying the uncertainty of the estimated RUL. In general, Bayesian learning, Gaussian process, deep ensembles, and Monte Carlo dropout are approaches used to estimate the uncertainty of the output of neural networks [19]. In [20], Bayesian learning is applied to quantify the uncertainty of the model parameters of the RUL prognostics. In [21], deep ensemble models are applied to quantify the uncertainty of RUL prediction. However, both deep ensembles and the Bayesian learning are computationally intensive, especially for deep neural networks with

* Corresponding author.
E-mail address: J.Lee-2@tudelft.nl (J. Lee).

https://doi.org/10.1016/j.ress.2022.108908
Received 10 June 2022; Received in revised form 5 September 2022; Accepted 18 October 2022
Available online 22 October 2022

significantly many parameters [19]. Another uncertainty quantification approach is deep Gaussian process learning, which estimates the confidence interval of RUL predictions [22]. Finally, Monte Carlo dropout is an efficient approach for uncertainty quantification [23]. Moreover, Monte Carlo dropout approximates Bayesian learning at a much lower computational cost [24]. Thus, in this paper, we use neural networks with Monte Carlo dropout to estimate the probability distribution of RUL.

Relevant studies on maintenance planning

Maintenance planning optimisation has been extensively studied for various assets [25]. However, most these studies propose advanced maintenance planning models with simple assumptions about the degradation of systems/components [26]. For instance, the degradation process of components is often assumed to follow a Gamma process [5,27], a Wiener process [28], a non-homogeneous Poisson process [29], or a Markov process [30,31].

Only a few studies integrate data-driven RUL prognostics into maintenance planning [32]. In [33], for example, the replacement of aircraft brakes is scheduled taking into account data-driven RUL prognostics. In [12], data-driven RUL prognostics for aircraft engines are obtained. Based on these prognostics, alarms are triggered and maintenance actions are specified. In [34], component inspections are scheduled based on the epistemic uncertainty of the estimated RUL. In [35], the replacement of airframe panels is scheduled based on the crack size predicted by the extended Kalman filter. Even when these studies consider RUL prognostics for maintenance planning, planning is done using fixed degradation thresholds for the degradation of components. For instance, in [12], all engines are replaced as soon as their RUL is estimated to be 44 days or less. In [35], the replacement of airframe panels is triggered by a fixed threshold of 47.4 mm crack size.

Recently, studies have proposed deep reinforcement learning (DRL) for adaptive maintenance planning [32], where no fixed thresholds are needed to schedule maintenance. In [36], DRL is used to schedule the replacement of a component whose degradation is represented by 4 discrete states. In [37], the maintenance of multi-component systems is optimised using DRL, where they assume that the degradation follows a compound Poisson process and a Gamma process. In [38], a DRL approach is applied to the case study on railways maintenance. These studies apply DRL for the cases where the system degradation is modelled as a discrete set of states, or the degradation process is modelled as a stochastic process. In contrast with these studies, we develop a threshold-free DRL approach for maintenance planning that considers the distribution of RUL. Thus, our DRL adaptively considers the uncertainty associated with the degradation level of the assets.

In this paper, we propose a deep reinforcement learning (DRL) approach for predictive maintenance that adaptively schedules maintenance considering probabilistic RUL prognostics. The overview of the proposed framework is shown in Fig. 1. Based on sensor measurements, the probability distribution of RUL is estimated using Convolutional Neural Networks (CNNs) with Monte Carlo dropout. We further develop a DRL approach that uses these probabilistic RUL prognostics to plan maintenance. The estimated distribution of RUL directly specifies the states of the DRL. Using DRL, maintenance actions are planned adaptively, without relying on fixed thresholds for the degradation level of the components. We illustrate our approach for maintenance planning of aircraft turbofan engines.

The main contributions of this paper are as follows:

- An integrated framework for predictive maintenance is proposed, where probabilistic Remaining-Useful-Life (RUL) prognostics and deep reinforcement learning (DRL) are used to plan the maintenance of aircraft engines. Here, probabilistic RUL prognostics (the estimated RUL distribution) are directly used to construct the states of the DRL.

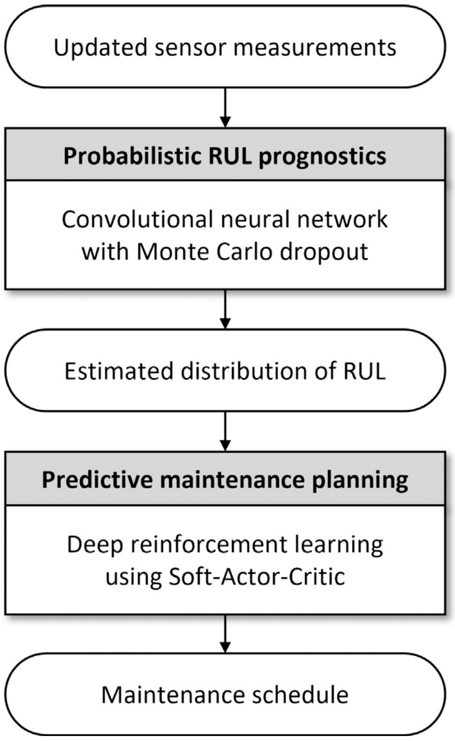

Updated sensor measurements
Probabilistic RUL prognostics
Convolutional neural network with Monte Carlo dropout
Estimated distribution of RUL
Predictive maintenance planning
Deep reinforcement learning using Soft-Actor-Critic
Maintenance schedule

Fig. 1. Overview of the proposed predictive maintenance framework using probabilistic RUL prognostics and DRL.

- Probabilistic RUL prognostics are obtained using Convolutional Neural Networks (CNNs) and Monte Carlo dropout. Using probabilistic RUL prognostics, we show that the number of unscheduled maintenance is lower than when using point-RUL estimates. This shows the benefit of quantifying the uncertainty of RUL estimates for maintenance planning.
- We pose the problem of predictive maintenance planning as a DRL problem. This approach adaptively proposes maintenance actions based on the trends of the estimated RUL prognostics.

The remainder of this paper is organised as follows. In Section 2, we propose probabilistic RUL prognostics using CNN with Monte Carlo dropout, and validate this proposed model for turbofan engines. In Section 3, we formulate a DRL problem for predictive maintenance planning taking into account probabilistic RUL prognostics. In Section 4, we illustrate our DRL approach for the maintenance planning of turbofan engines. In Section 5, we compare our DRL approach against other maintenance strategies. Finally, we provide conclusions in Section 6.

2. Estimating the distribution of RUL using CNN with Monte Carlo dropout

In this section, we propose a multi-channel convolutional neural networks (CNNs) and Monte Carlo dropout for probabilistic RUL prognostics. We validate our model for aircraft engines. These RUL prognostics are updated after every flight cycle as new degradation data become available.

2.1. Data description and pre-processing

We consider the degradation data of aircraft turbofan engines obtained by NASA using the Commercial Modular Aero-Propulsion System Simulation (C-MAPSS) [11]. This data set consists of data subsets FD001, FD002, FD003, and FD004, each considering a specific number

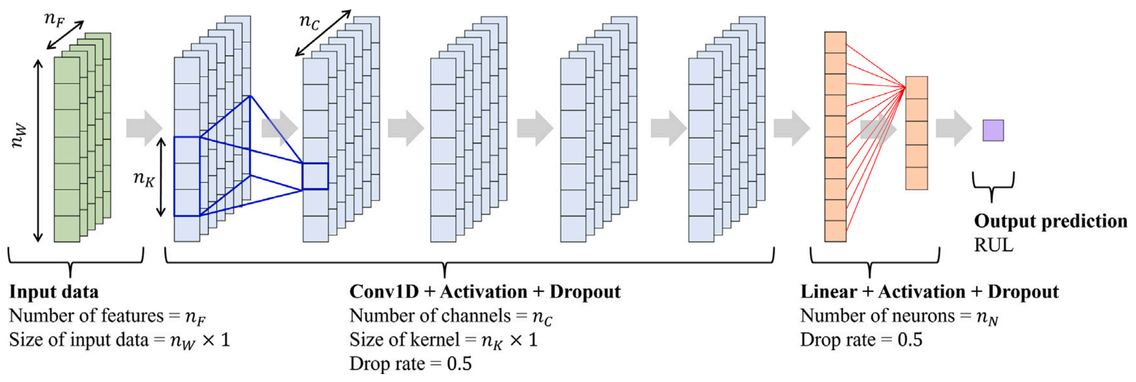

n_F
n_W
n_C
n_K
Output prediction
RUL
Input data
Number of features = n_F
Size of input data = $n_W \times 1$
Conv1D + Activation + Dropout
Number of channels = n_C
Size of kernel = $n_K \times 1$
Drop rate = 0.5
Linear + Activation + Dropout
Number of neurons = n_N
Drop rate = 0.5

Fig. 2. Proposed multi-channel CNN architecture. The blue lines visualise how multiple channels are convoluted into the next layer (see Eq. (3)). The red lines visualise a forward pass of a linear layer (see Eq. (4)).

Table 1
C-MAPSS data sets for turbofan engines [39].

	FD001	FD002	FD003	FD004
Training instances	100	260	100	249
Testing instances	100	259	100	248
Operating conditions	1	6	1	6
Fault mode	1	1	2	2

of fault modes and operating conditions (see Table 1) [39]. The training instances of the subsets have run-to-failure data of sensor measurements, while the testing instances have sensor measurements up to some moment prior to failure. Each instance consists of time-series of 21 sensor measurements per flight cycle. Following [12,13], we select for our analysis 14 non-constant sensor measurements. We discard the remaining 7 constant sensor measurements.

We pre-process the raw data as follows. First, using the clustering of operational settings proposed by [40], 6 operating conditions are identified. Let o_k denote the operating condition of an engine during kth flight cycle, $o_k \in \{1, \dots, 6\}$.

We also consider the history of the operating conditions. Let $h_{o,k}$ denote the number of cycles that an engine has been operated for under operating condition o, up to kth flight cycle.

Next, the measurements of sensor $s \in \{1, \dots, 14\}$ are normalised with respect to operating condition o as follows [12,15]:

$$m_{s,k} = \frac{2(m^o_{s,k} - m^o_{s,\min})}{m^o_{s,\max} - m^o_{s,\min}} - 1, \tag{1}$$

where $m_{s,k}$ is the normalised measurement of sensor s at kth flight cycle, $m^o_{s,k}$ is the raw measurement generated by sensor s at kth flight cycle. This kth flight cycle is performed under operating condition o. Also, $m^o_{s,\min}$ and $m^o_{s,\max}$ are the minimum and maximum measurement of sensor s under operating condition o, respectively. In total, $n_F = 21$ features are considered (the current operating conditions o_k, the history of the 6 operation conditions $h_{o,k}$, and 14 types of sensor measurements $m_{s,k}$).

Finally, n_F features for a time window of n_W flight cycles are considered as the input x of the CNN, i.e.,

$$x = \begin{bmatrix} o_1 & h_{1,1} & \dots & h_{6,1} & m_{1,1} & \dots & m_{14,1} \\ \vdots & \vdots & & \vdots & \vdots & & \vdots \\ o_{n_W} & h_{1,n_W} & \dots & h_{6,n_W} & m_{1,n_W} & \dots & m_{14,n_W}, \end{bmatrix}. \tag{2}$$

Here, n_W is selected based on the number of cycles available for the shortest testing instance in each data subset [12]. We use $n_W = 30$ cycles for FD001 and FD003, $n_W = 21$ for FD002, and $n_W = 19$ for FD004.

Table 2
Architecture of the proposed CNN, where n_C and n_K are the number of output channels and the length of the kernel of Conv1D layers, respectively, and n_N is the number of output neurons of Linear layers. A dropout rate $\rho = 0.5$ is used for all layers .

Layer	Type	Layer parameters	
1	Conv1D	$n_C = 128$,	$n_K = 10$
2	Conv1D	$n_C = 64$,	$n_K = 10$
3	Conv1D	$n_C = 32$,	$n_K = 10$
4	Conv1D	$n_C = 16$,	$n_K = 5$
5	Conv1D	$n_C = 8$,	$n_K = 5$
6	Linear	$n_N = 256$	
7	Linear	$n_N = 128$	
8	Linear	$n_N = 1$	

2.2. Architecture of the multi-channel CNN with Monte Carlo dropout

To obtain probabilistic RUL prognostics, we propose a neural network architecture combining multi-channel convolutional layers, linear layers, and Monte Carlo dropout (see Fig. 2 and Table 2).

In contrast to [12], where a common 1D kernel is applied for all features, we apply one 1D kernel per each time-series of a feature, i.e., each column of the input x is convoluted with different 1D kernels. Such multi-channel 1D convolutional layers are shown to be effective for multi-variate time-series [41], which is also the case of the C-MAPSS data set. Since an independent kernel is used for the time-series of each feature, the convolutional layers are able to learn the patterns of each feature.

A multi-channel 1D convolutional layer is defined by the size (length) n_K of the kernel, and the number of output channels n_C. The lth convolutional layer gets input $x^{(l-1)}$ from $(l-1)$th layer, where $x^{(l-1)}$ has $n_C^{(l-1)}$ channels. Then, the output of channel c of lth convolutional layer is obtained as follows:

$$x^l_c = g^l\left(b^l_c + \sum_{c'=1}^{n_C^{(l-1)}} \kappa^l_{c,c'} * x^{(l-1)}_{c'}\right) \quad \text{for } c \in 1, \dots, n^l_C, \tag{3}$$

where $*$ is the convolutional operator, $\kappa^l_{c,c'}$ is the kernel for input channel c' and output channel c, b^l_c is the bias of output channel c, and $g^l(\cdot)$ is the activation function of the convolutional layer. Here we use the rectified linear unit (ReLU) activation function.

We use 5 convolutional layers. For the first convolutional layer, the input x^0 is x defined in Eq. (2), and the number of input channels n^0_C is equal to the number of features $n_F = 21$. Table 2 shows the number of output channels (n_C) and the size of the kernels (n_K) for all convolutional layers. For all convolutional layers we use zero padding to ensure the same size of the output. These hyper-parameters are selected based on a grid-search, where the hyper-parameters suggested in [12,13] are used as a starting point.

Table 3
RMSE of the RUL predictions of the C-MAPSS data subsets using the proposed architecture of multi-channel CNN with Monte Carlo dropout and the other studies. STD: Standard deviation, where N/A implies that STD is not available in the original papers.

	FD001		FD002		FD003		FD004	
	Mean	STD	Mean	STD	Mean	STD	Mean	STD
Multi-channel CNN with MC dropout (proposed in this study)	11.81	0.29	14.49	0.10	11.69	0.26	17.73	0.21
Single-channel CNN with MC dropout [12]	12.22	N/A	15.07	N/A	12.47	N/A	18.10	N/A
DCNN [13]	12.61	0.19	22.36	0.32	12.64	0.14	23.31	0.39
MS-DCNN [14]	11.44	0.07	19.35	0.08	11.67	0.06	22.22	0.14
CNN with pooling [15]	18.45	N/A	30.29	N/A	19.82	N/A	29.16	N/A
CNN with pyramid pooling [16]	12.64	N/A	25.92	N/A	12.39	N/A	26.84	N/A

After the convolutional layers, we apply two intermediate linear layers, and one output linear layer with a single neuron without activation (see Fig. 2). The lth linear layer gets the (flattened) input $x^{(l-1)}$. Then, its output is obtained as follows:

$$x^l = g^l\left(b^l + w^l x^{(l-1)}\right), \tag{4}$$

where w^l is the weight matrix, b^l is the bias, and $g(\cdot)$ is the ReLU activation function. We denote the number of output neurons of the linear layers as n_N (see Table 2).

Using the Adam optimiser [42], we optimise the kernels $\kappa^l_{c,c'}$ and the bias b^l_c of the convolutional layers, as well as the weights w^l and the bias b^l of the linear layers. The loss function considered here is the mean-squared-error. We train the network using a fixed learning rate of 0.001, a mini-batch of 256 samples, and a maximum of 10^3 training epochs.

Monte Carlo dropout

Typically, Monte Carlo dropout is used only during training to prevent overfitting [23]. In this paper, we use Monte Carlo dropout (i) during training to prevent overfitting of the model, and (ii) during testing to obtain the probability distribution of the RUL [24]. We apply Monte Carlo dropout after each layer. The dropout rate is set to be 0.5 after a grid-search to minimise the test loss of data subset FD002 [12].

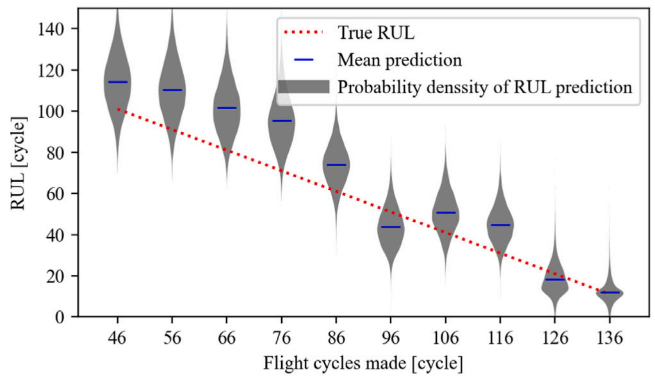
2.3. Probabilistic RUL prognostics for turbofan engines - Validation

We validate our CNN with Monte Carlo dropout for probabilistic RUL prognostics using the C-MAPSS data set for turbofan engines [11].

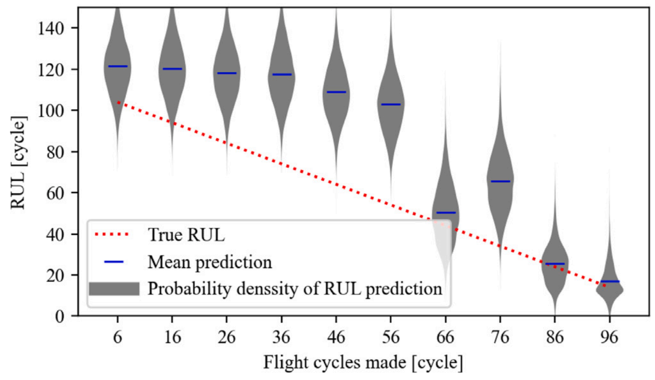
Comparing the RMSE of our estimated RUL against other RUL prognostics models

Our prognostics estimate the distribution of the RUL. We determine the Root Mean Squared Error (RMSE) of our prognostics based on the *mean* of the estimated distribution of RUL and the true RUL of the testing instances of the C-MAPSS data sets (see Table 3). We compare these results against the RUL prognostics models in [12–16]. Since these models estimate RUL as a *point* estimate, the RMSE of the prognostics in [12–16] is determined based on the estimated *point* RUL and the true RUL (see Table 3). For all prognostics models, the RMSE of subset FD002 and FD004 are higher than that of FD001. This is due to the multiple operating conditions considered in FD002 and FD004 (see Table 1). Also, FD002 and FD004 have the shortest time window of the input data compared to FD001 and FD003 (n_W = 21 for FD002 and n_W = 19 for FD004)

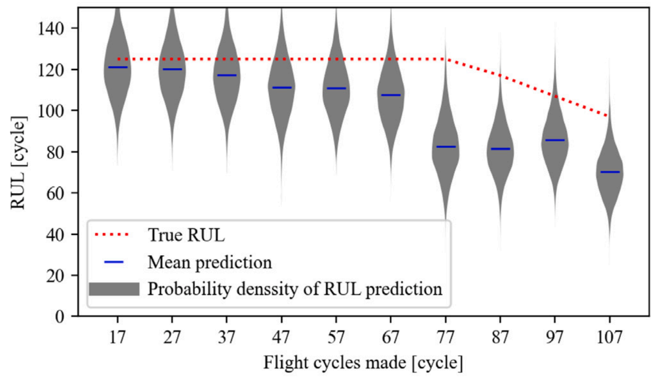
Table 3 shows that our multi-channel CNN with Monte Carlo dropout outperforms several other studies that employ CNNs for RUL prognostics. In fact, we obtain the lowest RMSE for subsets FD002 and FD004. For subsets FD001 and FD003, only MS-DCNN achieves a slightly smaller RMSE compared to our approach [14]. In general, the accuracy of our prognostics is higher or comparable to other existing studies.



(a) FD002 Testing Engine 148

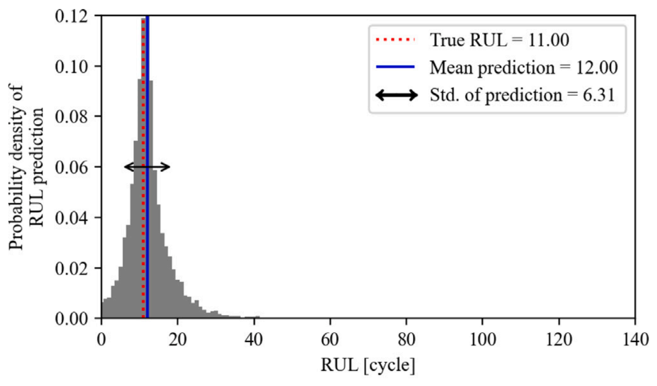


(b) FD002 Testing Engine 173

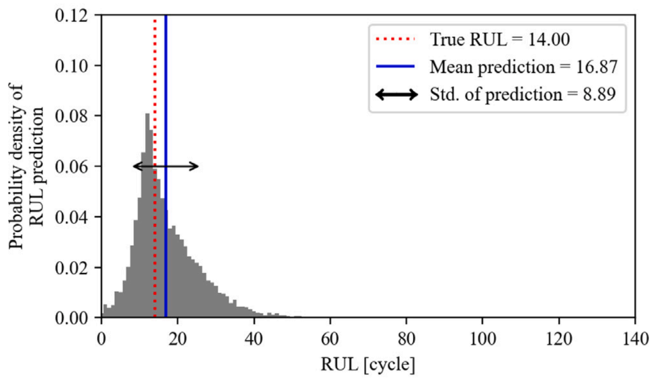


(c) FD002 Testing Engine 021

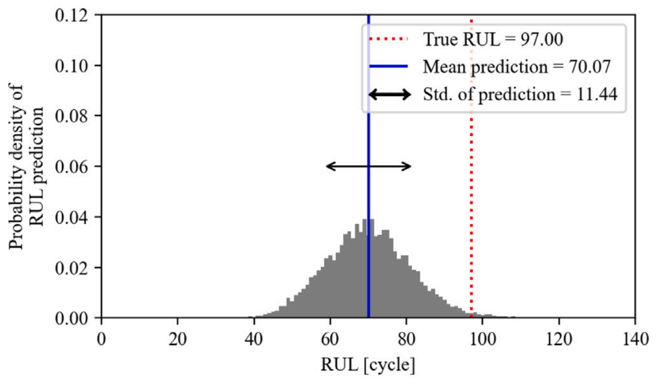
Fig. 3. Evolution of the RUL distribution over time. (a) The mean-estimated RUL gets closer to the true RUL, and the variance decreases. (b) The mean-estimated RUL gets closer to the true RUL, and the variance decreases though it is skewed. (c) Neither the error of the mean-estimated RUL nor the variance decrease.



(a) FD002 Testing Engine 148



(b) FD002 Testing Engine 173



(c) FD002 Testing Engine 021

Fig. 4. Probabilistic RUL prognostics. (a) The error between the mean-estimated RUL and the true RUL is small, and the standard deviation of RUL distribution is small. (b) The mean-estimated RUL is slightly larger than the true RUL, and the RUL distribution is right-skewed. (c) The error between the mean-estimated RUL and the true RUL is large, and the standard deviation of RUL distribution is large.

Estimating the distribution of the RUL

We illustrate the probabilistic RUL prognostics for three turbofan engines in testing instances of subset FD002. Fig. 3 shows the evolution of the estimated RUL distribution over time, as more sensor measurements become available. Fig. 4 shows the estimated RUL distribution of the three engines after they are operated for 136, 96, and 107 flight cycles, respectively. For Engine 148 (Fig. 3(a)), the standard deviation of RUL distribution decreases after 126th cycle. After 136th flight cycle (Fig. 4(a)), the error between the mean-estimated RUL and the true RUL is small (1.00 cycles), and the RUL distribution is concentrated around the true RUL (the standard deviation is 6.31 cycles).

For Engine 173 (Fig. 4(b)), the RUL distribution is right-skewed and the error between the mean-estimated RUL and the true RUL is small (2.87 cycles). Although the error of the estimated *point* (mean) of RUL is small, having the distribution of the RUL provides additional support for maintenance planning. Should we consider only the mean prediction of RUL (16.87 cycles) for Engine 173 to schedule a replacement, then we would be inclined to schedule a replacement at 16th cycle from now. However, this maintenance decision would lead to an engine failure since the true RUL of Engine 173 is 14 cycles. Should we consider the estimated *distribution* of RUL for Engine 173 (Fig. 4(b)), then we would observe the high probability (more than 45%) that Engine 173 fails in less than 14 cycles. In fact, the probability that Engine 173 fails at 12th cycle is highest (8.0%). Observing the RUL distribution we would be inclined to replace the engine at 12th cycle from now, avoiding an engine failure.

For Engine 021 (Fig. 4(c)), the error between the mean RUL prediction and the true RUL is large (26.93 cycles), and the standard deviation of RUL distribution is large (11.44). This is also informative for maintenance decision making, i.e., the accuracy of the RUL prognostic is low. In fact, the variance of RUL distribution is large across a sequence of flight cycles (see Fig. 3(c)), leaving maintenance decision making conservative about the moment of engine replacement.

As shown in Figs. 3 and 4, the distribution of RUL prognostics provides valuable information that can lead to more efficient maintenance decisions. In the next section, we propose a deep reinforcement learning approach to specify the moment of engine replacement based on the estimated distribution of RUL.

Quality of the estimated RUL distribution

We analyse the quality of the estimated RUL distributions using calibration plots [43]. Let $F(R|x)$ be the cumulative distribution function (CDF) of the estimated RUL R, given sensor measurements x. Let $F^{-1}(\zeta|x)$ be the quantile function of R such that $F^{-1}(\zeta|x) = \inf\{R : \zeta \le F(R|x)\}$. We say that the probabilistic RUL prognostics model is perfectly calibrated if

$$P(\rho \le F^{-1}(\zeta|x)) = \zeta \text{ for all } \zeta \in [0,1], \qquad (5)$$

where ρ is the true RUL. For a perfectly calibrated model, the probability that the true RUL is less than or equal to the ζ% quantile of the estimated distribution is ζ%.

Fig. 5 shows the calibration plots of the four C-MAPSS data subsets (FD001, FD002, FD003, FD004). The dashed, black line in Fig. 5 shows a perfectly calibrated model. Fig. 5 shows that our probabilistic RUL prognostics models are well calibrated, i.e., the deviation from the perfectly calibrated model is small. For the case of FD002 (Fig. 5(b)), the probability that the true RUL is less than or equal to the 10%, 50%, and 90% quantiles of the estimated RUL distributions using our prognostics models are 14%, 43%, and 86%, respectively.

3. Planning predictive maintenance using DRL and probabilistic RUL prognostics

In this section, we propose a deep reinforcement learning (DRL) approach for predictive maintenance of turbofan engines taking into account probabilistic RUL prognostics (estimated RUL distribution). These probabilistic RUL prognostics are updated periodically, as more measurements become available.

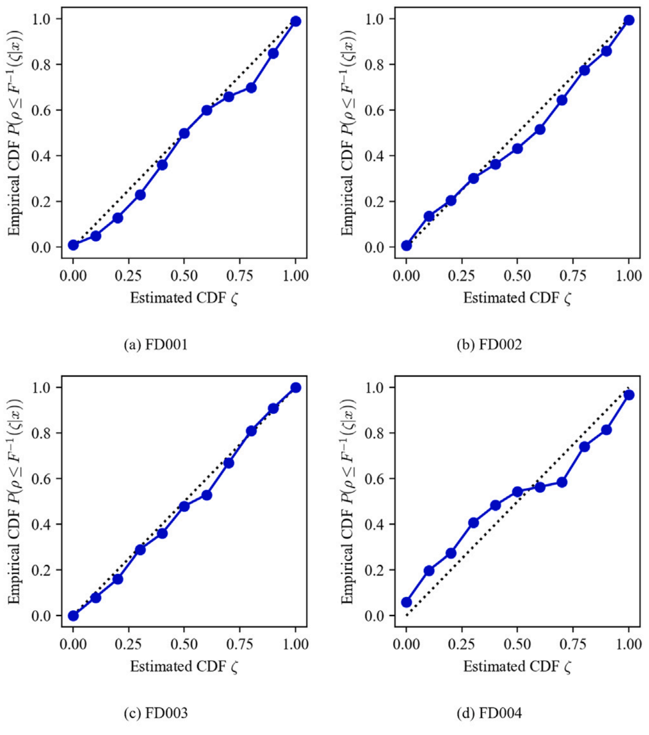


Fig. 5. Calibration plot of the estimated CDF of RUL for four test data sets.

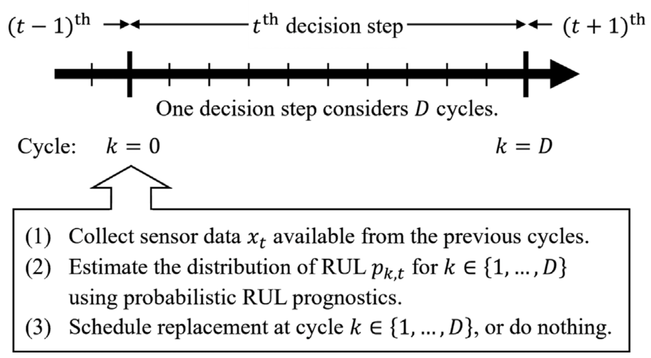

$(t-1)^{\text{th}}$
t^{th} decision step
$(t+1)^{\text{th}}$
One decision step considers D cycles.
Cycle: $k = 0$
$k = D$
(1) Collect sensor data x_t available from the previous cycles.
(2) Estimate the distribution of RUL $p_{k,t}$ for $k \in \{1, \ldots, D\}$ using probabilistic RUL prognostics.
(3) Schedule replacement at cycle $k \in \{1, \ldots, D\}$, or do nothing.

Fig. 6. Maintenance planning at tth decision step.

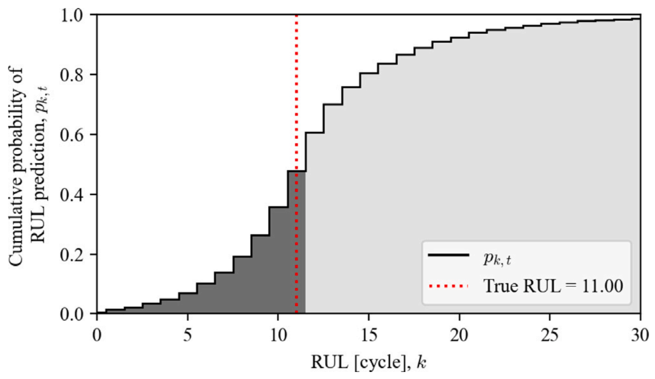


Fig. 7. Estimated cumulative probability $p_{k,t}$ for FD002 Testing Engine 148 after it is operated for 136 cycles.

3.1. Scheduling engine replacements taking into account updated probabilistic RUL prognostics

The maintenance schedule of the engines is updated every D flight cycles. In other words, every D cycles, we need to decide whether to replace an engine during the next D cycles (a *decision step*). Some existing studies assume that maintenance schedules can be updated every 1 cycle/day ($D = 1$) [30]. However, this assumption would be unrealistic for the maintenance of aircraft engines. In practice, several days are needed to prepare the required equipment before replacing an engine [12]. Thus, we assume $D > 1$ and make a maintenance plan for the next D cycles.

Our aim is to minimise the total maintenance cost while avoiding engine failures and minimising the wasted life of the engines. If an engine is replaced too late and as a result this engine fails before the *scheduled replacement*, then we have to perform a very costly *unscheduled replacement* [33]. On the other hand, if we schedule a replacement too early, we waste the life of this engine. The long-run maintenance cost also increases when engines are replaced too often. Our goal is to propose an approach to optimally schedule engine replacement taking into account probabilistic RUL prognostics (estimates of the RUL distribution).

Fig. 6 illustrates the maintenance planning at decision step t. At the start of decision step t, we use sensor measurements x_t, i.e., the measurements available up to decision step t. Using x_t, we estimate the distribution of the RUL using a CNN with Monte Carlo dropout (see Section 2). Let $p_{k,t}$ denote the estimated cumulative probability that the RUL of the engine is less than or equal to k cycles, given x_t. Formally,

$$p_{k,t} = P(R_t \leq k \mid x_t), \text{ for } k \in \{1, \ldots, D\} \tag{6}$$

where R_t is the predicted RUL at the start of decision step t. By the definition of RUL, the engine fails at kth cycle if $(k-1) < \rho_t \leq k$ when ρ_t denotes the true RUL. Since R_t in Eq. (6) is an estimate of ρ_t, we can interpret $p_{k,t}$ as the estimated probability that the engine fails within k cycles, given x_t. Based on $p_{k,t}$ at decision step t, we decide whether to schedule an engine replacement after k cycles ($k \in \{1, \ldots, D\}$), or do nothing within the next D cycles. If we do not schedule any replacement in the next D cycles, then we only collect further sensor measurements during these cycles. At the beginning of the $(t+1)$th decision step, these measurements are used, together with a CNN, to update the estimated distribution of RUL, $p_{k,(t+1)}$.

For example, at the start of decision step t, we have the estimated $p_{k,t}$ given in Fig. 7. We need to decide when to schedule an engine replacement based on this estimate. Fig. 7 shows the estimated RUL distribution for FD002 Testing Engine 148 of the C-MAPSS data set. This cumulative probability $p_{k,t}$ is estimated after the engine has already been used for 136 cycles. Our prognostics model predicts that this engine fails within 10 and 15 cycles with probability 37% and 82%, respectively. In fact, the true RUL of this engine at this moment is 11 cycles. In the next section, we propose a deep reinforcement learning approach to optimally replace the engine based on this estimated distribution of RUL.

3.2. Predictive maintenance planning as a deep reinforcement learning problem

We formulate the predictive maintenance planning of an engine as a deep reinforcement learning (DRL) problem (see Fig. 8). The *hidden state* ρ_t denotes the true RUL of the engine at decision step t. The *observed state* s_t denotes the RUL distribution estimated based on sensor measurements and CNN. Given s_t, an agent (decision-maker) takes an *action* $a_t \in \mathcal{A}$ based on a *policy* π. Then, *reward* r_t is obtained based on the hidden state ρ_t and action a_t. Finally, the system transits from state s_t to s_{t+1} at the next decision step $(t+1)$. We formalise our DRL problem as follows.

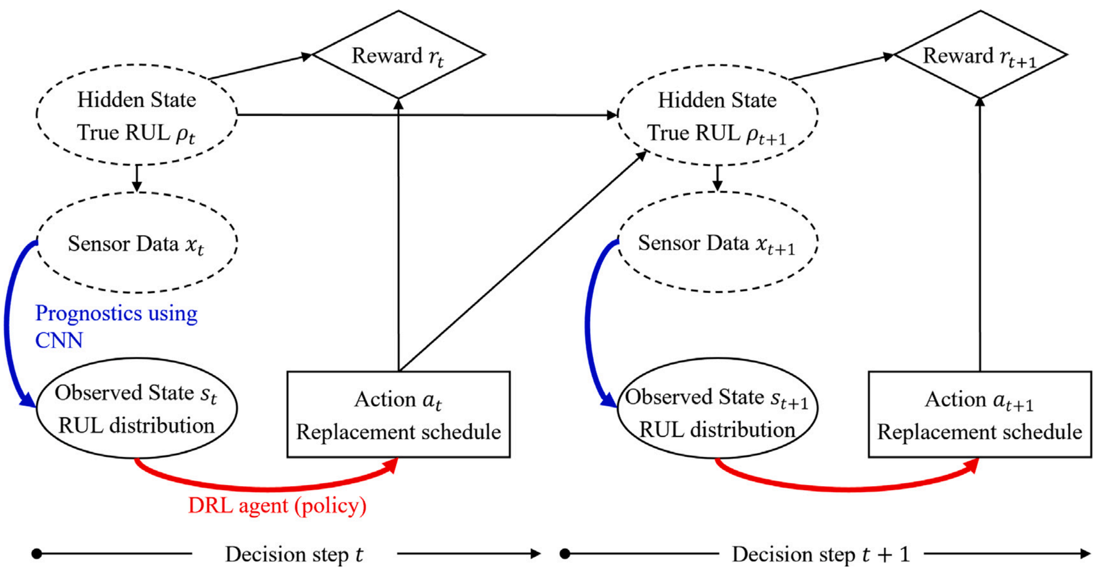

Reward r_t
Hidden State True RUL ρ_t
Sensor Data x_t
Prognostics using CNN
Observed State s_t RUL distribution
Action a_t Replacement schedule
DRL agent (policy)
Decision step t
Reward r_{t+1}
Hidden State True RUL ρ_{t+1}
Sensor Data x_{t+1}
Observed State s_{t+1} RUL distribution
Action a_{t+1} Replacement schedule
Decision step $t+1$

Fig. 8. Illustration of states, actions, rewards and transitions of the DRL problem for predictive maintenance planning.

The observed state s_t is the estimated distribution of the RUL $p_{k,t}$ for the next D cycles, i.e., $k \in \{1, \dots, D\}$. Formally,

$$s_t = [p_{1,t} \quad , \quad \dots \quad , \quad p_{D,t}] , \tag{7}$$

where $p_{k,t}$ is the probability that the RUL is less than k cycles (see Eq. (6)).

Given state s_t, the agent choose an action a_t: either schedule a replacement of the engine at cycle k ($k \in \{1, \dots, D\}$), or Do nothing. Formally,

$$a_t = \begin{cases} k, \quad 0 < k \le D & \text{Schedule replacement at cycle } k, \\ M, \quad M > D & \text{Do nothing.} \end{cases} \tag{8}$$

If $a_t = M > D$, we do not schedule an engine replacement in the next D cycles and postpone the engine replacement to the next decision step $(t+1)$.

The reward r_t obtained at decision step t is defined for 4 cases considering action a_t and the hidden state ρ_t: (1) a replacement is scheduled earlier than the engine failure; (2) a replacement is scheduled later than the engine failure; (3) we decide to do nothing in the next D cycles, but the engine fails within the next D cycles; and (4) we decide to do nothing, and the engine does not fail in the next D cycles. Formally,

$$r_t = \begin{cases} -c_{\text{sch}}(k) & \text{if } (k-1) < a_t \le k \text{ and } \rho_t > k \\ -c_{\text{uns}} & \text{if } (k-1) < a_t \le k \text{ and } \rho_t \le k \\ -c_{\text{uns}} & \text{if } a_t > D \text{ and } \rho_t \le D \\ 0 & \text{if } a_t > D \text{ and } \rho_t > D. \end{cases} \tag{9}$$

Here, $c_{\text{sch}}(k)$ denotes the cost of a scheduled replacement at cycle k ($k \in \{1, \dots, D\}$), which is defined as follows:

$$c_{\text{sch}}(k) = c_0 - c_1 k, \tag{10}$$

where c_0 is a fixed cost of replacement ($c_0 > 0$), and c_1 is a penalty for an early replacement ($c_1 > 0$). We assume that a too early replacement is expensive because we have less time to prepare the required equipment [33]. Also, we assume $c_{\text{sch}}(k)$ is positive for all k, i.e., $c_0 - c_1 D > 0$. In Eq. (9), c_{uns} denotes the cost of an unscheduled replacement. We assume $c_{\text{uns}} > c_0$ since an unscheduled replacement is generally more expensive [33]. Fig. 9 shows the cost model of a scheduled and an unscheduled engine replacement for $c_0 = 1$, $c_1 = 0.01$, and $c_{\text{uns}} = 2$.

The goal of the DRL agent is to choose an optimal moment to schedule a replacement such that the expected reward is maximised (or the expected cost is minimised). When scheduling an engine replacement, the DRL agent considers only the observed state s_t defined in Eq. (7). The training of the DRL agent is based on the observed state s_t, the action taken a_t, the obtained reward r_t, and the observed next state s_{t+1}. Although the reward r_t is calculated using the true RUL (hidden state ρ_t) in Eq. (9), the DRL agent does not observe the true RUL ρ_t directly.

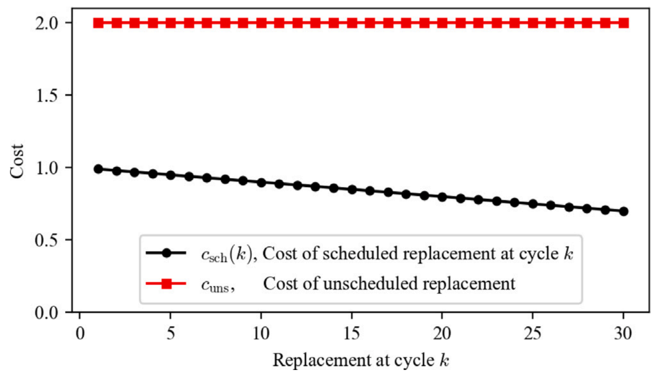


Fig. 9. Cost model of a scheduled and an unscheduled replacement. In this paper, we assume $c_0 = 1$, $c_1 = 0.01$, $c_{\text{uns}} = 2$, and $D = 30$.

In general, it is not trivial to choose an optimal moment to replace an engine, given the estimated RUL distribution (Fig. 7) and the cost model (Fig. 9). As an example, let us consider the cost in Fig. 9 and engine 148 with the estimated RUL distribution shown in Fig. 7. Should we decide to replace engine 148 at 25th cycle, and this engine does not fail by 25th cycle, then the cost of this replacement would be 0.75. Should we decide to replace engine 148 at 25th cycle, but this engine fails before 25th cycle, then an unscheduled replacement would be performed at cost 2.0. For Engine 148, there is a 97% estimated probability that this engine fails before 25th cycle (see Fig. 7). Should we decide to replace the engine at 5th cycle, then the cost would be 0.95. Although a cost of 0.95 is higher than the cost of scheduled replacement at 25th cycle (0.75), this maintenance action reduces the risk of an expensive unscheduled maintenance. For Engine 148, there is a 7% estimated failure probability at 5th cycle, so a low risk of unscheduled maintenance. Overall, deciding at which cycle should engine 148 be performed is non-trivial.

Once the DRL agent chooses an action, the hidden state ρ_{t+1} and the observed state s_{t+1} are updated accordingly. If the engine is replaced at decision step t, then the next decision step considers a new engine from the C-MAPSS data set. Otherwise, we further obtain sensor measurements x_{t+1} during the next D cycles and update the distribution of the RUL (the next state s_{t+1}) by generating new RUL prognostics using a CNN with Monte Carlo dropout.

3.3. Training the DRL agent for predictive maintenance

The DRL agent chooses action a_t (maintenance decision) for a given state s_t (estimated distribution of RUL) based on a *policy* $\pi(a_t|s_t) : \mathcal{S}\times\mathcal{A}\rightarrow[0,1]$, which is the probability to choose action a_t for a given state s_t. The optimal policy π^* is defined as a policy that maximises the expected reward defined as follows:

$$J(\pi)=\sum_t \mathbb{E}_{(s_t,a_t)\sim\rho_\pi}\left[\gamma^t r_t(s_t,a_t)\right], \tag{11}$$

where γ is a discount factor, and $\rho_\pi(s_t,a_t)$ is the state–action trajectory distribution induced by a policy π [44].

Soft–actor–critic algorithm to train the DRL agent for predictive maintenance planning

We train the DRL agent using a Soft–Actor–Critic (SAC) algorithm [44]. The SAC algorithm is an actor–critic algorithm where a policy (actor) is trained to choose actions that maximises the estimated state–action value (critic). Compared to traditional actor–critic algorithms, the SAC uses a stochastic policy and maximises a soft objective to explore new policies.

We consider a stochastic policy $\pi_\phi(a_t|s_t)$ to determine the mean $f^\mu_\phi(s_t)$ and the standard deviation $f^\sigma_\phi(s_t)$ of an action for a given state s_t, where ϕ is the trainable parameters of f^μ_ϕ and f^σ_ϕ. Then, action a_t is chosen as follows:

$$a_t=f^\mu_\phi(s_t)+\epsilon_t\cdot f^\sigma_\phi(s_t), \tag{12}$$

where ϵ_t is sampled from a standard Gaussian distribution.

The considered soft objective includes the expected entropy of the policy π_ϕ. Formally,

$$J(\pi)=\sum_t \mathbb{E}_{(s_t,a_t)\sim\rho_\pi}\gamma^t\left[r_t(s_t,a_t)+\alpha\mathcal{H}\left(\pi(\cdot|s_t)\right)\right], \tag{13}$$

where α is the temperature parameter determining the relative importance between the entropy term and the reward term. Thus, the SAC algorithm simultaneously maximises the expected reward and the entropy of the policy, allowing the exploration of new policies.

Considering the soft objective in Eq. (13), the state–action value (Q function) is modified as the soft Q function $Q:\mathcal{S}\times\mathcal{A}\rightarrow\mathbb{R}$. This soft Q function is then obtained by iteratively applying the following modified Bellman backup operator $\mathcal{T}^\pi$ [44] :

$$\mathcal{T}^\pi Q(s_t,a_t)=r_t(s_t,a_t)+\gamma\mathbb{E}_{s_{t+1}\sim p}[V(s_{t+1})], \tag{14}$$

where p is the distribution of s_{t+1}, given s_t and a_t, and $V(s_t)$ is the soft state value function $V:\mathcal{S}\rightarrow\mathbb{R}$ defined as follows:

$$V(s_t)=\mathbb{E}_{a_t\sim\pi}\left[Q(s_t,a_t)-\alpha\log\pi(a_t|s_t)\right]. \tag{15}$$

For the SAC algorithm, we train three functions: the policy (π), the soft Q function (Q), and the soft value function (V). We model these functions by means of three deep neural networks, π_ϕ, Q_θ, and V_ψ, where ϕ, θ, and ψ are the trainable parameters of each neural network. During training, we collect the replay buffer $\mathcal{D}=\{(s_t,a_t,r_t,s_{t+1})\}$ based on the current policy π_ϕ. Then, we update the trainable parameters to minimise the following loss functions.

Algorithm 1 Soft-Actor-Critic algorithm for predictive maintenance planning.

```
1:  Initialise parameters $\phi$, $\psi$, $\bar{\psi}$, $\theta_1$, $\theta_2$.
2:  for each episode do
3:      Initialise observation $s_0$.
4:      for each decision step $t$ do
5:          Choose action $a_t$ (Eq. (12))
6:          Get reward $r_t$ (Eq. (9))
7:          Get next state $s_{t+1}$ (Eq. (7))
8:          Update replay buffer $\mathcal{D}\leftarrow\mathcal{D}\cup\{(s_t,a_t,r_t,s_{t+1})\}$
9:          for each learning step do
10:             Sample mini-batch $\hat{\mathcal{D}}$ from replay buffer $\mathcal{D}$
11:             $\psi\leftarrow\psi-\lambda_V\nabla_\psi J_V(\psi)$
12:             $\theta_q\leftarrow\theta_q-\lambda_Q\nabla_{\theta_q}J_Q(\theta_q)$ for $q\in\{1,2\}$
13:             $\phi\leftarrow\phi-\lambda_\pi\nabla_\phi J_\pi(\phi)$
14:             $\bar{\psi}\leftarrow\tau\psi+(1-\tau)\bar{\psi}$
15:         end for
16:         Update $s_t$
17:     end for
18: end for
```

The policy net π_ϕ is updated using the Kullback–Leibler (KL) divergence, which guarantees the improvement of the policy in terms of its soft value [44]. We minimise the expected KL divergence as follows:

$$\begin{aligned}J_\pi(\phi)&=\mathbb{E}_{s_t\sim\mathcal{D}}\left[D_{KL}\left(\pi_\phi(\cdot|s_t)\middle\|\frac{\exp(\frac{1}{\alpha}Q_\theta(s_t,\cdot))}{Z_\theta(s_t)}\right)\right]\\&=\mathbb{E}_{s_t\sim\mathcal{D},a_t\sim\pi_\phi}\left[\log\pi_\phi(a_t|s_t)-\frac{1}{\alpha}Q_\theta(s_t,a_t)+\log Z_\theta(s_t)\right],\end{aligned} \tag{16}$$

where $Z_\theta(s_t)$ is the partition function that does not contribute to the gradient with respect to ϕ, and a_t is sampled from the current policy π_ϕ using Eq. (12).

For the value net V_ψ, we minimise the residual of the value function calculated based on the critic net Q_θ:

$$J_V(\psi)=\mathbb{E}_{s_t\sim\mathcal{D}}\left[\frac{1}{2}\left(V_\psi(s_t)-\hat{V}(s_t)\right)^2\right], \tag{17}$$

with

$$\hat{V}(s_t)=\mathbb{E}_{a_t\sim\pi_\phi}\left[Q_\theta(s_t,a_t)-\alpha\log\pi_\phi(a_t|s_t)\right]. \tag{18}$$

For the critic net Q_θ, we minimise the modified Bellman residual:

$$J_Q(\theta)=\mathbb{E}_{(s_t,a_t)\sim\mathcal{D}}\left[\frac{1}{2}\left(Q_\theta(s_t,a_t)-\hat{Q}(s_t,a_t)\right)\right], \tag{19}$$

with

$$\hat{Q}(s_t,a_t)=r_t(s_t,a_t)+\gamma\mathbb{E}_{s_{t+1}\sim p}[V_{\bar{\psi}}(s_{t+1})]. \tag{20}$$

Here, we use the target value net $V_{\bar{\psi}}$, where $\bar{\psi}$ is an exponentially moving average of the value net parameters [45]. Also, we adopt the double Q-learning approach: we simultaneously train two critic nets (Q_{θ_1} and Q_{θ_2}), and we use $Q_\theta(s_t,a_t)=\min\{Q_{\theta_1}(s_t,a_t),Q_{\theta_2}(s_t,a_t)\}$ [46]. Both the target value net and double Q-learning approach are known to stabilise the training process [45,46]

The gradients of the loss functions in Eqs. (16), (17), and (19) are obtained by backward propagation. Given the gradients of the corresponding objectives, the parameters ϕ, ψ, and θ are updated using the Adam optimiser with learning rates λ_π, λ_V, and λ_Q, respectively.

We train the DRL agent for predictive maintenance for aircraft engines using the SAC algorithm (see Algorithm 1). We first initialise the parameters of the neural network models, ϕ, ψ, $\bar{\psi}$, θ, θ_1, and θ_2 (line 1). We train these networks for n_E episodes (line 2). An episode is initialised with observation s_0, which is the initial distribution of

Table 4
Architecture of the deep neural network models for policy net π_ϕ, value net V_ψ, and critic net Q_θ (see also Fig. 10) .

Policy net π_ϕ		
Layer	Type	Number of neurons n_N
Input (State s_t)	–	D
Shared hidden layer 1	Linear	$n_N = 256$
Shared hidden layer 2	Linear	$n_N = 128$
Hidden layer 1 for μ	Linear	$n_N = 64$
Hidden layer 2 for μ	Linear	$n_N = 32$
Output μ ($f_\phi^\mu(s_t)$)	Linear	1
Hidden layer 1 for σ	Linear	$n_N = 64$
Hidden layer 2 for σ	Linear	$n_N = 32$
Output σ ($f_\phi^\sigma(s_t)$)	Linear	1
Value net V_ψ		
Layer	Type	Number of neurons n_N
Input (State s_t)	–	D
Hidden layer 1	Linear	$n_N = 256$
Hidden layer 2	Linear	$n_N = 128$
Output (State value $V(s_t)$)	Linear	1
Critic net Q_θ		
Layer	Type	Number of neurons n_N
Input (State s_t, action a_t)	–	$D + 1$
Hidden layer 1	Linear	$n_N = 256$
Hidden layer 2	Linear	$n_N = 128$
Output (Q value $Q(s_t, a_t)$)	Linear	1

the engine's RUL sampled from a DRL episode data set (line 3). Here, the RUL distribution is estimated using the CNN that is already trained based on an independent training data set. The episode continues for n_T decision steps (line 4). At each decision step t, for the observed state s_t, we sample the action a_t using the policy net $\pi_\phi(a_t|s_t)$ (line 5). Based on this action, we obtain a reward r_t and the next state s_{t+1} (line 6–7). We add (s_t, a_t, r_t, s_{t+1}) into the replay buffer $\mathcal{D}$ (line 8). Then, for each learning step, we sample mini-batch $\hat{\mathcal{D}}$ from replay buffer $\mathcal{D}$ (line 9–10), and use this to calculate the loss functions in Eqs. (16), (17), and (19). We next update the policy net, the value net, and the critic nets such that the corresponding objectives are minimised (line 11–13). Here λ_π, λ_V, and λ_Q are the learning rates of each network. Also, the parameters of the target value net $\bar{\psi}$ is updated with an exponential moving average of ψ, where τ is a smoothing factor (line 14). For the next decision step, we update the current state (line 16).

Design of the architecture of the neural networks

We design the architectures of the policy net π_ϕ, the value net V_ψ, and the critic net Q_θ as shown in Fig. 10 and Table 4.

The policy net π_ϕ has input s_t, which is a vector of size D, and returns two scalar values corresponding to the mean of action $f_\phi^\mu(s_t)$ and the standard deviation of action $f_\phi^\sigma(s_t)$. These two outputs $f_\phi^\mu(s_t)$ and $f_\phi^\sigma(s_t)$ are used to sample action a_t for given state s_t (see Eq. (12)). We consider hidden layers shared by $f_\phi^\mu(s_t)$ and $f_\phi^\mu(s_t)$ to facilitate learning from the shared features (see Fig. 10(a)). Following these shared hidden layers, we consider separated hidden layers for each of $f_\phi^\mu(s_t)$ and $f_\phi^\sigma(s_t)$.

The value net V_ψ has input s_t and returns a scalar value $V_\psi(s_t)$. We consider two hidden, fully-connected layers. The same architecture is used also for the target value net $V_{\bar{\psi}}$.

The input of critic net Q_θ is a vector of size $(D + 1)$, which is the augmentation of state s_t and action a_t. Its output is a scalar value $Q_\theta(s_t, a_t)$. We consider two hidden, fully connected layers (see Table 4). Since we use a double Q-learning approach, we consider two critic networks Q_{θ_1} and Q_{θ_2} having the same architecture but different parameters θ_1 and θ_2.

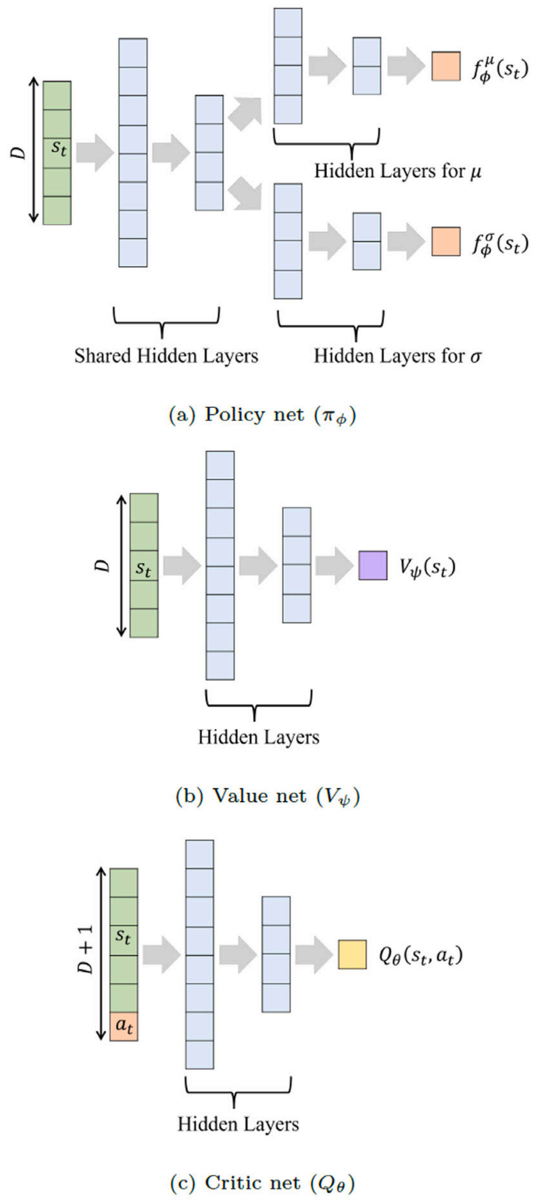


Fig. 10. Architecture of neural network models for DRL. (see also Table 4).

4. Case study: DRL for predictive maintenance of turbofan engines with probabilistic RUL prognostics

This section shows how probabilistic RUL prognostics (Section 2) for turbofan engines are integrated into maintenance planning, i.e., the DRL approach discussed in Section 3.

4.1. Training the probabilistic RUL prognostics

We consider the maintenance of a turbofan engine whose sensor measurements are given in subset FD002 of the C-MAPSS data set (see Table 1) [11]. From the 260 training instances of FD002, we randomly sample 130 engines to obtain data subset FD002-Prog, which is used to train the probabilistic RUL prognostics model (CNN with Monte Carlo dropout, Section 2). The remaining 130 engines, referred to as FD002-DRL, are used to generate episodes of the DRL problem.

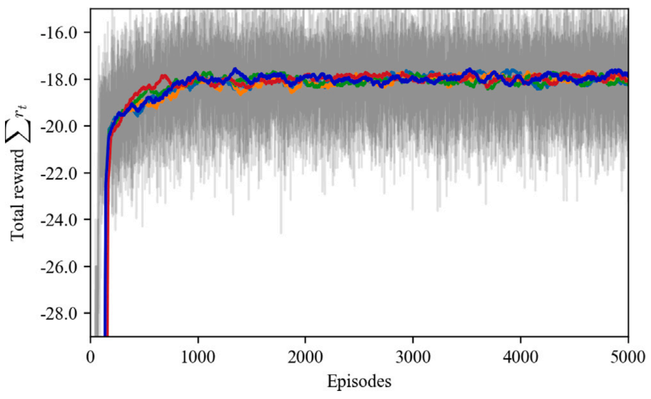


Fig. 11. Learning curve of the DRL approach during 5000 episodes. The thin grey lines are 5 learning curves, and the solid lines are the moving average of 100 episodes of each learning curve.

4.2. Training the DRL agent

The DRL agent considers the maintenance episodes generated based on the data set FD002-DRL. At each episode, we sample engines from FD002-DRL. Using measurements x_t and the trained RUL prognostics model, we generate the RUL distribution ($p_{k,t}$) of the sampled engine. This estimated RUL distribution is state s_t observed by the DRL agent. If the DRL agent decides to do nothing, the sensor measurements of the sampled engine are updated at the next decision step $t+1$. If the DRL agent decides to replace the engine, a new engine from FD002-DRL is sampled.

We train the DRL agent for $n_E = 5000$ episodes using Algorithm 1. Each episode consists of maximum $n_T = 100$ decision steps, and each decision step considers $D = 30$ flight cycles (see Fig. 6 for the definition of a decision step). As a reward (cost) model, we assume $c_{\text{uns}} = 2$, $c_0 = 1$, and $c_1 = 0.01$ for the cost parameters defined in Eqs. (9)–(10) (see also Fig. 9 for the cost model). The hyper-parameters of the SAC algorithms are as follows: discount factor $\gamma = 0.9$, temperature parameter $\alpha = 0.01$, learning rates $\lambda_\phi = 10^{-5}$, $\lambda_\psi = 10^{-4}$, $\lambda_\theta = 10^{-4}$, smoothing factor of target value net $\tau = 10^{-3}$, the maximum size of replay buffer $|\mathcal{D}| = 10^6$, and the size of the mini-batch $|\hat{\mathcal{D}}| = 4096$.

Fig. 11 shows the learning curve of the DRL agent, illustrating the total reward per episode. The total reward rapidly increases during the first 500 episodes and converges to around -18 after 1000 episodes. After 1000 episodes, the total reward of each episode varies because the considered DRL problem is stochastic. However, the moving average of the total reward stabilises after 1000 episodes. Moreover, 5 independent training curves show the same trends. Thus, the training is stopped after 5000 episodes.

4.3. Evaluation of the DRL agent: Predictive maintenance using DRL

Following training, we evaluate the trained DRL agent for 1000 episodes generated by our CNN model and data set FD002-DRL. During evaluation, the DRL agent chooses an action a_t for a given state s_t from the mean action f^{μ}_{ϕ} of the trained policy π_ϕ [44]. Formally,

$$a_t = f^{\mu}_{\phi}(s_t). \tag{21}$$

Below we discuss the benefits of our DRL approach for predictive maintenance, by presenting some decision steps (the estimated RUL distributions and the associated maintenance actions made by the DRL agent).

Maintenance decision based on updated RUL distribution

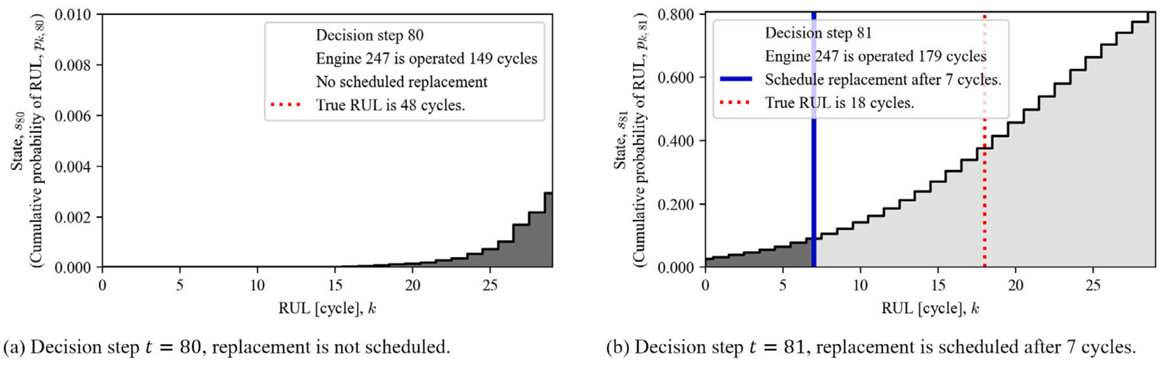
The estimated RUL distributions are updated at every decision step t (D cycles), as more sensor measurements become available. This ensures that the maintenance decision is always based on the most recent RUL prognostics. Fig. 12 shows 2 consecutive decisions steps ($t = 80$ and 81) for the maintenance of Engine 247, FD002 of the C-MAPSS data set. At decision step $t = 80$, the engine has been operated for 149 cycles. Our CNN model predicts the probability that the engine will fail within 30 cycles to be $p_{30,80} = 0.005$. The entire distribution $p_{k,80}$ for $k \in \{0,\dots,30\}$ is given in Fig. 12(a). Such a distribution quantifies the uncertainty of the RUL prognostics, and provides basis for maintenance decisions of the DRL agent. The DRL agent observes the RUL distribution and decides to do nothing, i.e., do not schedule replacement in this decision step $t = 80$. Since no replacement is scheduled for the next $D = 30$ cycles, the engine is operated continuously until the next decision step $t = 81$, and more sensor measurements are collected. Based on the new sensor measurement, we update the distribution of the RUL again using the CNN with Monte Carlo dropout (see Fig. 12(b)). At decision step $t = 81$, the probability that the engine will fail within 30 cycles is estimated to be $p_{30,81} = 0.807$. Given the updated distribution of the RUL, the DRL agent schedule a replacement after 7 cycles (see the blue vertical line in Fig. 12(b)). The probability that the engine will fail within 7 cycles is $p_{7,80} = 0.091$. In fact, the (hidden) true RUL is 18 cycles at decision step $t = 81$, i.e., the DRL agent schedules a replacement 11 cycles before the engine fails.

Adaptive maintenance decision using deep neural network

Using a deep neural network model (policy net), our DRL agent adaptively considers the updated RUL distribution of individual engine, instead of relying on one fixed threshold for all engines. As a result, our DRL agent can identify optimal moment of engine replacement taking into account different trends of RUL distributions. For example, Fig. 13 shows distinctive RUL distributions of three engines, estimated during different episodes. In Fig. 13(a), there is a very high chance that the engine will fail within the next 30 cycles ($p_{30,t} = 0.896$). In this case, the DRL agent schedules a replacement after 5 cycles when the probability that the engine will fail within 5 cycles is estimated to be $p_{5,t} = 0.113$. In Fig. 13(b), the estimated $p_{k,t}$ increases from $p_{1,t} = 0.011$ to $p_{30,t} = 0.748$, and the DRL agent schedules a replacement after 9 cycles when $p_{9,t} = 0.073$. In the last case in Fig. 13(c), the probability that the engine will fail within 30 cycles is smaller compared to the two previous cases ($p_{30,t} < 0.15$), but the trend rapidly increases. In this case, the DRL agent schedules a replacement after 18 cycles when $p_{18,t} = 0.011$, effectively preventing the engine failure.

In contrast to our DRL approach, existing predictive maintenance approaches often consider fixed thresholds for all same-type of components to trigger maintenance. For example, in [12], an alarm is triggered when the estimated RULs of engines are below a threshold (44 days). Similarly, in [35], airframe panels are replaced when the predicted crack size is larger than a threshold (47.4 mm). Since a fixed threshold value is applied for all components, differences between individual RUL prognostics results may not be considered in these traditional approaches.

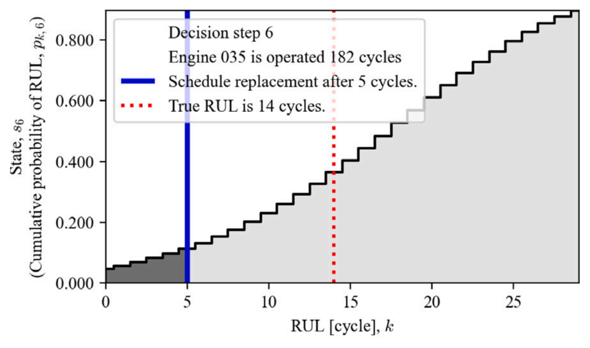
The benefit of our adaptive maintenance planning using deep neural network is evident when trying, unsuccessfully, to find one fixed threshold that is optimal for all three cases in Fig. 13. Let us assume that we use a fixed threshold 0.11 and always schedule an engine replacement after k cycles if $p_{k,t} > 0.11$, irrespective of the RUL distribution. Using such a fixed threshold of 0.11 will effectively prevent the failure for the first case (Fig. 13(a)). Using the same threshold for Figs. 13(b) and 13(c), engine replacements are scheduled after 12 and 28 cycles, respectively. However, in both cases, the engine replacements are later than the true RUL (11 and 21 cycles, respectively), leading to unscheduled replacements at higher cost. In the same line, let us assume that we set a much lower fixed threshold of 0.01, i.e., we always schedule an engine replacement after k cycles if $p_{k,t} > 0.01$. Using this



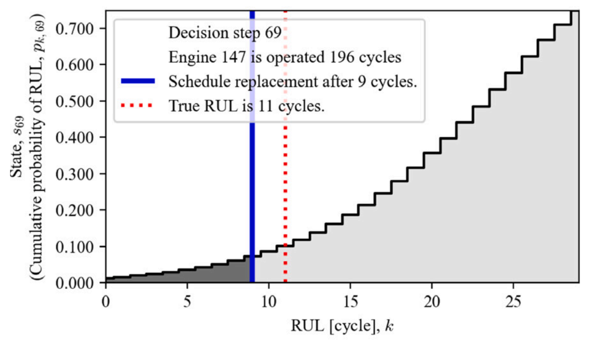
(a) Decision step $t = 80$, replacement is not scheduled.

(b) Decision step $t = 81$, replacement is scheduled after 7 cycles.

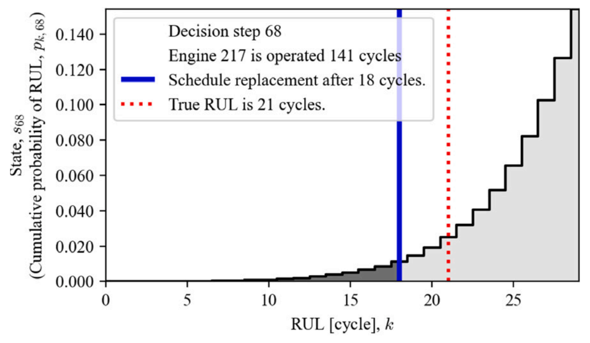
Fig. 12. Estimated RUL distributions and replacement schedules for Engine 247 during 2 consecutive decisions steps ($t = 80$ and 81).



(a) Replacement is scheduled after 5 cycles when $p_{5,6} = 0.113$



(b) Replacement is scheduled after 9 cycles when $p_{9,69} = 0.073$



(c) Replacement is scheduled after 18 cycles when $p_{18,68} = 0.011$

Fig. 13. Three different RUL distributions and adaptive maintenance decisions of the DRL agent.

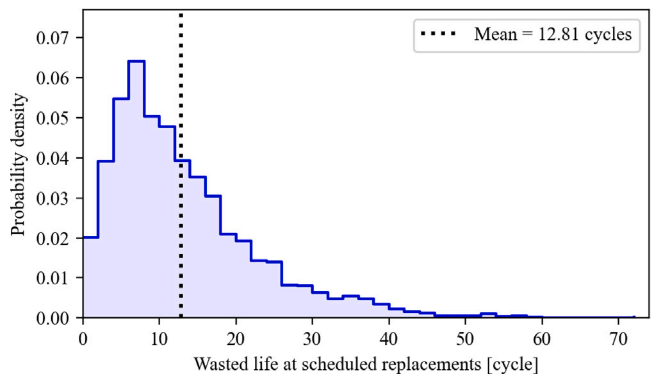


Fig. 14. Wasted life of engines at the moment of replacements under our DRL approach.

threshold will avoid an unscheduled engine replacements in the last case (Fig. 13(c)). However, with such a low threshold, replacements are scheduled too early for the other two cases in Figs. 13(a) and 13(b), wasting the useful life of the engines. This example shows that finding one fixed threshold that is optimal for all cases is challenging. In contrast, our DRL agent adaptively considers the different trends of RUL distributions, without using fixed thresholds. As a result, our approach leads to less unscheduled maintenance.

Scheduling replacements with small wasted life of engines

Using updated RUL distributions and adaptive maintenance decision, our DRL agent schedules engine replacements without wasting useful lives of engines. Fig. 14 shows the distribution of the wasted life of engines at the moment of replacement, when using our DRL approach. Replacements are scheduled when the true RUL of an engine is 12.81 cycles on average. This is only 6% of the average life of the engines in subset FD002. Also, more than 82% of the engines are replaced when their wasted life is less than 20 cycles.

5. Predictive maintenance using DRL vs other maintenance strategies

In this section, we compare the performance of our DRL approach for predictive maintenance against three other traditional maintenance strategies:

(1) *Predictive maintenance at mean-estimated-RUL*: This strategy schedules engine replacements at the mean RUL predicted by the CNN model in Section 2. This strategy uses a point estimate of the RUL, while our DRL approach uses a distribution of the RUL. Considering this strategy, we aim to evaluate the impact of using the distribution

Table 5
Comparison of the proposed DRL approach using RUL distribution, and other maintenance strategies. Percentage in parenthesis indicates the relative ratio to the corrective maintenance .

	Total cost	Number of unscheduled replacements	Total number of replacements
DRL approach using distribution of RUL	17.84 (−36.3%)	0.62 (−95.6%)	14.89 (+6.4%)
Predictive maintenance at mean-estimated-RUL	25.23 (−9.8%)	10.87 (−22.3%)	14.00 (0.0%)
Corrective maintenance	27.99 (0.0%)	13.99 (0.0%)	13.99 (0.0%)
Ideal maintenance at true RUL	16.10 (−42.5%)	0.0 (−100.0%)	13.95 (−0.3%)

of RUL for maintenance planning, rather than just a point estimate of the RUL.

(2) *Corrective maintenance*: This strategy replaces engines as soon as they fail. Under this strategy, we always perform unscheduled replacements, which is the most undesirable case.

(3) *Ideal maintenance at true RUL*: This strategy assumes that the true RUL is known in advance by an Oracle, and engines replacements are scheduled exactly at this true RUL. Under this strategy, there are no unscheduled maintenance tasks and the wasted lives of engines are always zero, i.e., an ideal maintenance strategy.

Table 5 shows the performance of these traditional maintenance strategies vs our DRL approach, using three following performance indicators:

(*i*) The total cost: this is the cost of both scheduled and unscheduled replacements during 3000 cycles of engine operations (i.e., 100 decision steps). The cost (reward) model is given in Eqs. (9)–(10).

(*ii*) The number of unscheduled replacements: this is a direct metric for maintenance reliability. We aim to minimise the number of unscheduled engine replacements.

(*iii*) The total number of replacements: this is the number of both scheduled and unscheduled replacements during 3000 cycles of engine operations. Since we consider a fixed period of cycles, a lower number of total replacements implies that we utilise the engines for a longer duration.

Table 5 shows that our DRL approach using RUL distributions outperforms the other maintenance strategies, especially in terms of the total maintenance cost and the number of unscheduled replacements. Our DRL approach saves 36.3% of the total costs compared to corrective maintenance. Moreover, it also achieves a more reliable maintenance planning by preventing 95.6% of unscheduled replacements. The total number of replacements (both scheduled and unscheduled) is slightly (6.4%) larger for our DRL approach since engines are replaced earlier than their end-of-life to prevent unscheduled replacements. However, this slight increase in the total number of engine replacements is balanced out by a large economic efficiency (large cost savings) and maintenance reliability (lower number of unscheduled replacements) that our DRL approach achieves.

The benefit of using probabilistic RUL prognostics instead of a point estimate of RUL is evident when comparing our DRL approach against predictive maintenance at mean-estimated-RUL (see Table 5). Both strategies make use of RUL prognostics obtained using a CNN (see Section 2). But our DRL approach uses probabilistic RUL prognostics (estimated RUL distribution) to plan engine maintenance. As a result, predictive maintenance based on the mean-estimated-RUL reduces only 9.8% of total costs and 22.3% of unscheduled replacements, while our DRL approach further reduces the total cost (36.3%) and unscheduled replacements (95.6%).

The cost savings obtained by our DRL approach are further explained in Fig. 15. Since we assume 2 times higher costs for unscheduled replacements (see the cost model in Fig. 9), even a small number of unscheduled replacements can take a large portion of the total cost. Due to this reason, all maintenance strategies performed a similar number of total replacements, but the total maintenance costs are significantly different. In the case of predictive maintenance at the mean-estimated-RUL, 85% of the total cost is associated with unscheduled replacements. In contrast, for our DRL approach, only 7% of the total cost is associated with unscheduled replacements.

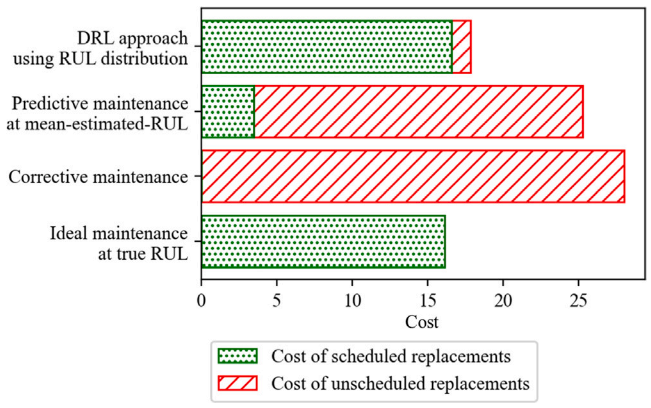


Fig. 15. Cost of scheduled/unscheduled replacements of the proposed DRL approach and other maintenance strategies.

6. Conclusions

In this paper, we propose a deep reinforcement learning (DRL) approach to plan predictive maintenance for aircraft engines. This maintenance planning takes into account the estimated distribution of the engines' Remaining-Useful-Life (RUL).

We first estimate the RUL distribution of engines using Convolutional Neural Networks with Monte Carlo dropout. These estimates are periodically updated, as more sensor measurements become available. Such estimates of the RUL distribution provide useful information about the uncertainty associated with the RUL prognostics and enables more effective maintenance planning.

With the estimated RUL distribution, we schedule maintenance for turbofan engines using DRL. Maintenance actions are specified adaptively, based on the trends of the RUL prognostics. In contrast to existing studies, we do not use fixed thresholds to trigger maintenance actions. Thus, our DRL approach enables adaptive and flexible, threshold-free maintenance planning.

The results show that our DRL approach with probabilistic RUL prognostics leads to lower maintenance costs and fewer unscheduled maintenance events, when compared to several other maintenance strategies. Compared to maintenance planning at mean-estimated-RUL, our DRL approach reduces the total maintenance cost by 29.3%. Moreover, it prevents 95.6% of unscheduled engine replacements. The engines are replaced just before their end-of-life, with an average wasted lives of only 12.8 cycles. Overall, our DRL approach outperforms the several other traditional maintenance strategies in terms of the cost and reliability indicators.

Overall, this study proposes a generic framework to integrate data-driven, probabilistic RUL prognostics into predictive maintenance. This framework is readily applicable for other aircraft components whose health is continuously monitored.

As future works, we plan to expand the proposed DRL approach for predictive maintenance of multiple components. In addition, we consider more realistic inputs and constraints of aircraft maintenance such as limited space of hangar, logistics of spare parts, and dynamic flight conditions.

CRediT authorship contribution statement

Juseong Lee: Conceptualisation, Formal analysis, Writing – original draft, Visualisation, Validation, Software, Methodology, Investigation. **Mihaela Mitici:** Conceptualisation, Formal analysis, Writing – original draft, Writing – review & editing, Validation, Supervision, Resources, Project administration, Methodology, Investigation, 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.

Data availability

Data will be made available on request.

Acknowledgement

We would like to thank MSc student Arthur Reijns and PhD candidate Ingeborg de Pater, supervised by dr. Mihaela Mitici, for sharing the initial code and data for the RUL prognostics.

This research has been partly funded by the European Union's Horizon 2020 research and innovation programme under grant agreement No 769288.

References

[1] Badea VE, Zamfiroiu A, Boncea R. Big Data in the Aerospace Industry. Inf Econ 2018;22(1/2018):17–24. http://dx.doi.org/10.12948/issn14531305/22.1.2018.02.

[2] Ochella S, Shafiee M, Dinmohammadi F. Artificial intelligence in prognostics and health management of engineering systems. Eng Appl Artif Intell 2022;108(May 2019):104552. http://dx.doi.org/10.1016/j.engappai.2021.104552.

[3] Fink O. Data-Driven Intelligent Predictive Maintenance of Industrial Assets. In: Women in industrial and systems engineering. 2020, p. 589–605. http://dx.doi.org/10.1007/978-3-030-11866-2_25.

[4] Sprong JP, Jiang X, Polinder H. A deployment of prognostics to optimize aircraft maintenance - A literature review. In: Proceedings of the annual conference of the prognostics and health management society, PHM, Vol. 11. 2019, p. 1–12. http://dx.doi.org/10.36001/phmconf.2019.v11i1.776.

[5] Lee J, Mitici M. An integrated assessment of safety and efficiency of aircraft maintenance strategies using agent-based modelling and stochastic Petri nets. Reliab Eng Syst Saf 2020;202:107052. http://dx.doi.org/10.1016/j.ress.2020.107052.

[6] Mitici M, de Pater I. Online model-based remaining-useful-life prognostics for aircraft cooling units using time-warping degradation clustering. Aerospace 2021;8(6). http://dx.doi.org/10.3390/aerospace8060168.

[7] Balaban E, Saxena A, Narasimhan S, Roychoudhury I, Koopmans M, Ott C, Goebel K. Prognostic health-management system development for electromechanical actuators. J Aerosp Inf Syst 2015;12(3):329–44. http://dx.doi.org/10.2514/1.I010171.

[8] Hong S, Yue T, Liu H. Vehicle energy system active defense: A health assessment of lithium-ion batteries. Int J Intell Syst 2020;(September):1–19. http://dx.doi.org/10.1002/int.22309.

[9] Ren L, Zhao L, Hong S, Zhao S, Wang H, Zhang L. Remaining Useful Life Prediction for Lithium-Ion Battery: A Deep Learning Approach. IEEE Access 2018;6(c):50587–98. http://dx.doi.org/10.1109/ACCESS.2018.2858856.

[10] Hong S, Sun L, Yin J, Yu T, Wang Y, Zhu W. PEMFC Power Prediction Based on Deep Auto-encoder and LS-SVMR. In: 2018 IEEE 3rd international conference on big data analysis. 2018, p. 391–6. http://dx.doi.org/10.1109/ICBDA.2018.8367714.

[11] Saxena A, Goebel K. Turbofan Engine Degradation Simulation Data Set. 2008.

[12] de Pater I, Reijns A, Mitici M. Alarm-based predictive maintenance scheduling for aircraft engines with imperfect Remaining Useful Life prognostics. Reliab Eng Syst Saf 2022;221:108341. http://dx.doi.org/10.1016/j.ress.2022.108341.

[13] Li X, Ding Q, Sun JQ. Remaining useful life estimation in prognostics using deep convolution neural networks. Reliab Eng Syst Saf 2018;172(June 2017):1–11. http://dx.doi.org/10.1016/j.ress.2017.11.021.

[14] Li H, Zhao W, Zhang Y, Zio E. Remaining useful life prediction using multi-scale deep convolutional neural network. Appl Soft Comput 2020;89:106113. http://dx.doi.org/10.1016/j.asoc.2020.106113.

[15] Babu GS, Zhao P, Li X-L. Deep Convolutional Neural Network Based Regression Approach for Estimation of Remaining Useful Life. In: International conference on database systems for advanced applications, Vol. 9642. Cham; 2016, p. 214–28. http://dx.doi.org/10.1007/978-3-319-32025-0_14.

[16] Song Y, Bliek L, Xia T, Zhang Y. A Temporal Pyramid Pooling-Based Convolutional Neural Network for Remaining Useful Life Prediction. In: Proceedings ofthe 31st European safety and reliability conference. 2021, p. 810–7. http://dx.doi.org/10.3850/978-981-18-2016-8_478-cd.

[17] Arias Chao M, Kulkarni C, Goebel K, Fink O. Fusing physics-based and deep learning models for prognostics. Reliab Eng Syst Saf 2022;217:107961. http://dx.doi.org/10.1016/j.ress.2021.107961, arXiv:2003.00732.

[18] Fink O, Wang Q, Svensén M, Dersin P, Lee WJ, Ducoffe M. Potential, challenges and future directions for deep learning in prognostics and health management applications. Eng Appl Artif Intell 2020;92(January):103678. http://dx.doi.org/10.1016/j.engappai.2020.103678, arXiv:2005.02144.

[19] Abdar M, Pourpanah F, Hussain S, Rezazadegan D, Liu L, Ghavamzadeh M, Fieguth P, Cao X, Khosravi A, Acharya UR, Makarenkov V, Nahavandi S. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. Inf Fusion 2021;76:243–97. http://dx.doi.org/10.1016/j.inffus.2021.05.008, arXiv:2011.06225.

[20] Kraus M, Feuerriegel S. Forecasting remaining useful life: Interpretable deep learning approach via variational Bayesian inferences. Decis Support Syst 2019;125(July):113100. http://dx.doi.org/10.1016/j.dss.2019.113100, arXiv:1907.05146.

[21] Zhang C, Lim P, Qin AK, Tan KC. Multiobjective Deep Belief Networks Ensemble for Remaining Useful Life Estimation in Prognostics. IEEE Trans Neural Netw Learn Syst 2017;28(10):2306–18. http://dx.doi.org/10.1109/TNNLS.2016.2582798.

[22] Biggio L, Wieland A, Chao MA, Kastanis I, Fink O. Uncertainty-Aware Prognosis via Deep Gaussian Process. IEEE Access 2021;9:123517–27. http://dx.doi.org/10.1109/ACCESS.2021.3110049.

[23] Srivastava N, Hinton G, Sutskever A, Krizhevsky I, Salakhutdinov R. Dropout: A simple Way to Prevent Neural networks from Overfitting. J Mach Learn Res 2014;15.

[24] Gal Y, Ghahramani Z. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In: 33rd international conference on machine learning, ICML 2016, Vol. 3. 2016, p. 1651–60, arXiv:1506.02142.

[25] Alaswad S, Xiang Y. A review on condition-based maintenance optimization models for stochastically deteriorating system. Reliab Eng Syst Saf 2017;157:54–63. http://dx.doi.org/10.1016/j.ress.2016.08.009.

[26] van Noortwijk JM. A survey of the application of gamma processes in maintenance. Reliab Eng Syst Saf 2009;94:2–21. http://dx.doi.org/10.1016/j.ress.2007.03.019.

[27] Do P, Assaf R, Scarf P, Iung B. Modelling and application of condition-based maintenance for a two-component system with stochastic and economic dependencies. Reliab Eng Syst Saf 2019;182(October 2018):86–97. http://dx.doi.org/10.1016/j.ress.2018.10.007.

[28] Zhang Z, Si X, Hu C, Lei Y. Degradation data analysis and remaining useful life estimation: A review on Wiener-process-based methods. European J Oper Res 2018;271(3):775– 796. http://dx.doi.org/10.1016/j.ejor.2018.02.033.

[29] Caballé NC, Castro IT, Pérez CJ, Lanza-Gutiérrez JM. A condition-based maintenance of a dependent degradation-threshold-shock model in a system with multiple degradation processes. Reliab Eng Syst Saf 2015;134:98–109. http://dx.doi.org/10.1016/j.ress.2014.09.024.

[30] Zhao Y, Smidts C. Reinforcement learning for adaptive maintenance policy optimization under imperfect knowledge of the system degradation model and partial observability of system states. Reliab Eng Syst Saf 2022;108541. http://dx.doi.org/10.1016/j.ress.2022.108541.

[31] Yang H, Li W, Wang B. Joint optimization of preventive maintenance and production scheduling for multi-state production systems based on reinforcement learning. Reliab Eng Syst Saf 2021;214(February):107713. http://dx.doi.org/10.1016/j.ress.2021.107713.

[32] Hu Y, Miao X, Si Y, Pan E, Zio E. Prognostics and health management: A review from the perspectives of design, development and decision. Reliab Eng Syst Saf 2022;217(May 2021):108063. http://dx.doi.org/10.1016/j.ress.2021.108063.

[33] Lee J, Mitici M. Multi-objective design of aircraft maintenance using Gaussian process learning and adaptive sampling. Reliab Eng Syst Saf 2022;218:108123. http://dx.doi.org/10.1016/j.ress.2021.108123.

[34] Kim S, Choi JH, Kim NH. Inspection schedule for prognostics with uncertainty management. Reliab Eng Syst Saf 2022;222(February):108391. http://dx.doi.org/10.1016/j.ress.2022.108391.

[35] Wang Y, Gogu C, Binaud N, Bes C, Haftka RT, Kim NH. Predictive airframe maintenance strategies using model-based prognostics. Proc Inst Mech Eng O 2018;232(6):690–709. http://dx.doi.org/10.1177/1748006X18757084.

[36] Andriotis CP, Papakonstantinou KG. Deep reinforcement learning driven inspection and maintenance planning under incomplete information and constraints. Reliab Eng Syst Saf 2021;212:107551. http://dx.doi.org/10.1016/j.ress.2021.107551.

[37] Zhang N, Si W. Deep reinforcement learning for condition-based maintenance planning of multi-component systems under dependent competing risks. Reliab Eng Syst Saf 2020;203(June):107094. http://dx.doi.org/10.1016/j.ress.2020.107094.

[38] Mohammadi R, He Q. A deep reinforcement learning approach for rail renewal and maintenance planning. Reliab Eng Syst Saf 2022;225(May):108615. http://dx.doi.org/10.1016/j.ress.2022.108615.

[39] Liu Y, Frederick DK, Decastro JA, Litt JS, Chan WW. User's guide for the commercial modular aero-propulsion system simulation (C-MAPSS). Technical Report March, 2012, p. 1–40.

[40] Peel L. Data Driven Prognostics using a Kalman Filter Ensemble ofNeural Network Models. In: International conference on prognostics and health management. 2008, p. 1–6. http://dx.doi.org/10.1109/PHM.2008.4711423.

[41] Zheng Y, Liu Q, Chen E, Ge Y, Zhao JL. Time series classification using multi-channels deep convolutional neural networks. In: International conference on web-age information management. 2014, p. 298–310. http://dx.doi.org/10.1007/978-3-319-08010-9_33.

[42] Kingma DP, Ba JL. Adam: A method for stochastic optimization. In: 3rd international conference on learning representations, ICLR 2015 - conference track proceedings. 2015, p. 1–15, arXiv:1412.6980.

[43] Kuleshov V, Fenner N, Ermon S. Accurate uncertainties for deep learning using calibrated regression. In: 35th international conference on machine learning, ICML 2018, Vol. 6. 2018, p. 4369–77, arXiv:1807.00263.

[44] Haarnoja T, Zhou A, Abbeel P, Levine S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: 35th international conference on machine learning, ICML 2018, Vol. 5. 2018, p. 2976–89, arXiv:1801.01290.

[45] Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, Graves A, Riedmiller M, Fidjeland AK, Ostrovski G, Petersen S, Beattie C, Sadik A, Antonoglou I, King H, Kumaran D, Wierstra D, Legg S, Hassabis D. Human-level control through deep reinforcement learning. Nature 2015;518(7540):529–33. http://dx.doi.org/10.1038/nature14236.

[46] Haarnoja T, Zhou A, Hartikainen K, Tucker G, Ha S, Tan J, Kumar V, Zhu H, Gupta A, Abbeel P, Levine S. Soft Actor-Critic Algorithms and Applications. 2018, p. 1–17, ArXiv arXiv:1812.05905.