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

A B S T R A C T

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
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.

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...
Here, \( n \) denotes the number of cycles that an engine has been operated for under operating condition \( o \), up to \( k \)th flight cycle.

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

\[
m_{s,k} = \frac{2(m_{s,k}^o - m_{s,min}^o)}{m_{s,max}^o - m_{s,min}^o} - 1, \tag{1}
\]

where \( m_{s,k} \) is the normalised measurement of sensor \( s \) at \( k \)th flight cycle, \( m_{s,k}^o \) is the raw measurement generated by sensor \( s \) at \( k \)th flight cycle. This \( k \)th flight cycle is performed under operating condition \( o \). Also, \( m_{s,min}^o \) and \( m_{s,max}^o \) 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 \( h_{s,k} \), 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.,

\[
x = \left[ \begin{array}{cccc} a_1 & h_{1,1} & \ldots & h_{o,k} & m_{1,1} & \ldots & m_{1,14} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ a_{n_W} & h_{1,n_W} & \ldots & h_{o,n_W} & m_{1,n_W} & \ldots & m_{1,14,n_W} \end{array} \right]. \tag{2}
\]

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.

### 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 \( l \)th 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 \( l \)th convolutional layer is obtained as follows:

\[
x^{(l)}_c = g' \left( b'_c + \sum_{c'=1}^{n_C^{(l-1)}} k_{c',c}^l x^{(l-1)}_{c'} \right) \quad \text{for} \ c = 1, \ldots, n_C^l, \tag{3}
\]

where \( g' \) is the convolutional operator, \( k_{c',c}^l \) is the kernel for input channel \( c' \) and output channel \( c \), \( b'_c \) is the bias of output channel \( c \), and \( g' \) 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 defined in Eq. (2), and the number of input channels \( n_C^0 \) 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 2

<table>
<thead>
<tr>
<th>Layer parameters</th>
<th>Type</th>
<th>Layer</th>
<th>n_C</th>
<th>n_K</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Conv1D</td>
<td>n_C = 128,</td>
<td>n_K = 10</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Conv1D</td>
<td>n_C = 64,</td>
<td>n_K = 10</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Conv1D</td>
<td>n_C = 32,</td>
<td>n_K = 10</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Conv1D</td>
<td>n_C = 16,</td>
<td>n_K = 5</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Conv1D</td>
<td>n_C = 8,</td>
<td>n_K = 5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Linear</td>
<td>n_C = 256,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Linear</td>
<td>n_C = 128,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Linear</td>
<td>n_C = 1,</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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.

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

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 $l$th linear layer gets the (flattened) input $x^{(l-1)}$. Then, its output is obtained as follows:

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

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 $k^l_{c,c'}$ 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.
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 $F^{-1}(\zeta|x) = \inf\{R : \zeta \leq F(R|x)\}$. We say that the probabilistic RUL prognostics model is perfectly calibrated if

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

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

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.
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,

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

where $R_t$ is the predicted RUL at the start of decision step $t$. By the definition of RUL, the engine fails at $k$th cycle if $(k-1) < \rho_s \leq k$ when $\rho_s$ is the true RUL since $R_t$ in Eq. (6) is an estimate of $\rho_t$, 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 $\rho_s$ 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 $a_t \in A$ based on a policy $\pi$. Then, reward $r_t$ is obtained based on the hidden state $\rho_s$ 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.
Formally,
\[
s_t = \begin{bmatrix} p_{t,1} & \cdots & p_{t,D} \end{bmatrix},
\]
where \( p_{t,k} \) is the probability that the RUL is less than \( k \) cycles (see Eq. (6)).

Given state \( s_t \), the agent chooses an action \( a_t \): either schedule a replacement of the engine at cycle \( k \) (\( k \in \{1, \ldots, D\} \)), or Do nothing. Formally,
\[
a_t = \begin{cases} k, & 0 < k \leq D \quad \text{Schedule replacement at cycle } k, \\ M, & M > D \quad \text{Do nothing}. \end{cases}
\]

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 \( \rho_t \): (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,
\[
r_t = \begin{cases} -c_{sch}(k), & (k-1) < a_t \leq k \text{ and } \rho_t > k \\ -c_{uns}, & (k-1) < a_t \leq k \text{ and } \rho_t \leq k \\ -c_{uns}, & a_t > D \text{ and } \rho_t \leq D \\ 0, & a_t > D \text{ and } \rho_t > D. \end{cases}
\]

Here, \( c_{sch}(k) \) denotes the cost of a scheduled replacement at cycle \( k \) (\( k \in \{1, \ldots, D\} \)), which is defined as follows:
\[
c_{sch}(k) = c_0 - c_1 k,
\]
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 \( c_{sch}(k) \) 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 \( c_{uns} > c_0 \) 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 \( c_{uns} = 2 \).

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 \( \rho_t \)) in Eq. (9), the DRL agent does not observe the true RUL \( \rho_t \) directly.

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.
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 $\pi(a_t|s_t) : S \times A \rightarrow [0, 1]$, which is the probability to choose action $a_t$ for a given state $s_t$. The optimal policy $\pi^*$ is defined as a policy that maximises the expected reward defined as follows:

$$J(\pi) = \sum_t E_{(s_t,a_t) \sim \rho_{\pi}} \left[ y^t r(s_t, a_t) \right],$$

where $y$ is a discount factor, and $\rho_{\pi}(s_t,a_t)$ is the state-action trajectory distribution induced by a policy $\pi$ [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 $\pi_\theta(a_t|s_t)$ to determine the mean $f^\theta_t(s_t)$ and the standard deviation $\sigma^\theta_t(s_t)$ of an action for a given state $s_t$, where $\theta$ is the trainable parameters of $f^\theta_t$ and $\sigma^\theta_t$. Then, action $a_t$ is chosen as follows:

$$a_t = f^\theta_t(s_t) + \epsilon_t \cdot \sigma^\theta_t(s_t),$$

where $\epsilon_t$ is sampled from a standard Gaussian distribution.

The considered soft objective includes the expected entropy of the policy $\pi_\theta$. Formally,

$$J(\pi) = \sum_t E_{(s_t,a_t) \sim \rho_{\pi}} \left[ \log \pi(a_t|s_t) + H(\pi(.|s_t)) \right],$$

where $a$ 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 $Q : S \times A \rightarrow \mathbb{R}$. The soft Q function is then obtained by iteratively applying the following modified Bellman backup operator $T^\pi$ [44]:

$$T^\pi Q(s_t, a_t) = r_t(s_t, a_t) + \gamma E_{(s_{t+1}) \sim p} [V^\pi(s_{t+1})],$$

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 $V : S \rightarrow \mathbb{R}$ defined as follows:

$$V(s_t) = \sum_{s_{t+1}} Q(s_{t+1}, a_t) - a \log \pi(a_t|s_t).$$

For the SAC algorithm, we train three functions: the policy $\pi$, the soft Q function $Q(\cdot)$, and the soft value function $V(\cdot)$. We model these functions by means of three deep neural networks, $\pi_\theta$, $Q_\phi$, and $V_\psi$, where $\theta$, $\phi$, and $\psi$ are the trainable parameters of each neural network. During training, we collect the replay buffer $D = \{(s_t, a_t, r_t, s_{t+1})\}$ based on the current policy $\pi_\theta$. 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 $\phi$, $\psi$, $\theta_1$, $\theta_2$.
2. for each episode do
3. Initialise observation $s_t$.
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 $D \leftarrow D \cup \{(s_t, a_t, r_t, s_{t+1})\}$
9. for each learning step do
10. Sample mini-batch $D$ from replay buffer $D$
11. $\psi \leftarrow \psi - \lambda_\psi \nabla_\psi J_\psi(\psi)$
12. $\theta_2 \leftarrow \theta_2 - \lambda_2 \nabla_\theta_2 J_\theta_2(\theta_2)$ for $q \in \{1, 2\}$
13. $\phi \leftarrow \phi - \lambda_\phi \nabla_\phi J_\phi(\phi)$
14. $\psi \leftarrow \psi + (1 - \gamma)\psi$
15. end for
16. Update $s_t$
17. end for
18. end for

The policy net $\pi_\theta$ 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:

$$J_\pi(\theta) = \mathbb{E}_{s_t \sim D} \left[ D_{KL} \left( \pi_\theta(.|s_t) \| \frac{\exp \left[ \frac{1}{\alpha} Q(s_t, a_t) \right]}{Z_{\pi_\theta}(s_t)} \right) \right]$$

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

For the value net $V_\psi$, we minimise the residual of the value function calculated based on the critic net $Q_\phi$:

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

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

For the critic net $Q_\phi$, we minimise the modified Bellman residual:

$$J_\phi(\phi) = \mathbb{E}_{s_t \sim D_t} \left[ \frac{1}{2} (Q_\phi(s_t, a_t) - \hat{Q}(s_t, a_t))^2 \right],$$

with $\hat{Q}(s_t, a_t) = r_t(s_t, a_t) + \gamma E_{s_{t+1} \sim p} [V_\psi(s_{t+1})]$.

Here, we use the target value net $V_\psi$, which 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 ($Q_\phi$ and $\hat{Q}_\phi$), and we use $Q_\phi(s_t, a_t) = \min \{Q_\phi(s_t, a_t), Q_\phi(s_t, a_t)\}$ [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 $\phi$, $\psi$, and $\theta$ are updated using the Adam optimiser with learning rates $\lambda_\phi$, $\lambda_\psi$, and $\lambda_\theta$, 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, $\phi$, $\psi$, $\theta_1$, $\theta_2$, and $\theta_3$ (line 1). We train these networks for $n_\pi$ episodes (line 2). An episode is initialised with observation $s_0$, which is the initial distribution of
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, for the observed state \( s_t \), we sample the action \( a_t \) using the policy net \( \pi(\theta) \) (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 \( D \) (line 8). Then, for each learning step, we sample mini-batch \( \mathcal{D} \) from replay buffer \( D \) (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 \( \lambda_{\pi} \), \( \lambda_{V} \), and \( \lambda_{Q} \) are the learning rates of each network. Also, the parameters of the target value net \( \psi \) is updated with an exponential moving average of \( \psi \), where \( r \) 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 \( \pi(\theta) \), the value net \( V_{\psi} \), and the critic net \( Q_{\theta} \) as shown in Fig. 10 and Table 4.

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

The value net \( V_{\psi} \) 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 \( V_{\psi} \).

The input of critic net \( Q_{\theta} \) 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 \( Q_{\theta_1} \) and \( Q_{\theta_2} \), having the same architecture but different parameters \( \theta_1 \) and \( \theta_2 \).

### 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.

---

**Table 4**

| Architecture of the deep neural network models for policy net \( \pi(\theta) \), value net \( V_{\psi} \), and critic net \( Q_{\theta} \) (see also Fig. 10). |
|---|---|---|
| **Policy net \( \pi(\theta) \)** | **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 \( \mu \) | Linear | \( n_n = 64 \) |
| Hidden layer 2 for \( \mu \) | Linear | \( n_n = 32 \) |
| Output \( \mu (f^\mu_{\theta}(s_t)) \) | Linear | 1 |
| Hidden layer 1 for \( \sigma \) | Linear | \( n_n = 64 \) |
| Hidden layer 2 for \( \sigma \) | Linear | \( n_n = 32 \) |
| Output \( \sigma (f^\sigma_{\theta}(s_t)) \) | Linear | 1 |

| **Value net \( V_{\psi} \)** | **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 \( Q_{\theta} \)** | **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 |
Reliability Engineering and System Safety 230 (2023) 108908

J. Lee and M. Mitici

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 $x_{k,t}$ and the trained RUL prognostics model, we generate the RUL distribution ($p_{k,t}$) of the sampled engine. This estimated RUL distribution is state $s_{k}$ 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 $c_{\text{data}} = 2, c_1 = 1$, 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 $|D| = 10^6$, and the size of the mini-batch $|D| = 4096$.

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 $f_{\hat{\phi}}^\pi$ of the trained policy $\pi_{\hat{\phi}}$ [44]. Formally, we define

$$a_{t} = f_{\hat{\phi}}^\pi(s_{t}). \tag{21}$$

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 decision 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 $p_{80,30} = 0.005$. The entire distribution $p_{k,t}$ for $k \in \{0, \ldots, 30\}$ 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 $p_{81,30} = 0.807$. Given the updated distribution of the RUL, the DRL agent schedules 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 $p_{81,7} = 0.091$. 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 ($p_{30,30} = 0.896$). 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 $p_{30,5} = 0.113$. In Fig. 13(b), the estimated $p_{30,5}$ increases from $p_{30,5} = 0.011$ to $p_{30,5} = 0.748$, and the DRL agent schedules a replacement after 9 cycles when $p_{30,9} = 0.073$. 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 ($p_{30,30} < 0.15$), but the trend rapidly increases. In this case, the DRL agent schedules a replacement after 18 cycles when $p_{30,18} = 0.011$, 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
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
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.

### Table 5

<table>
<thead>
<tr>
<th>Maintenance Strategy</th>
<th>Total Cost</th>
<th>Number of Unscheduled Replacements</th>
<th>Total Number of Replacements</th>
</tr>
</thead>
<tbody>
<tr>
<td>DRL approach using RUL distribution</td>
<td>17.84 (−36.3%)</td>
<td>0.62 (−95.0%)</td>
<td>14.89 (−6.4%)</td>
</tr>
<tr>
<td>Predictive maintenance at mean-estimated-RUL</td>
<td>25.23 (−9.8%)</td>
<td>10.87 (−22.3%)</td>
<td>14.00 (0.0%)</td>
</tr>
<tr>
<td>Corrective maintenance</td>
<td>27.99 (0.0%)</td>
<td>13.99 (0.0%)</td>
<td>13.99 (0.0%)</td>
</tr>
<tr>
<td>Ideal maintenance at true RUL</td>
<td>16.10 (−42.5%)</td>
<td>0.0 (−100.0%)</td>
<td>13.95 (−0.3%)</td>
</tr>
</tbody>
</table>

### 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 Rejins 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


[27] Juseong Lee: Conceptualisation, Formal analysis, Writing – origina


