Received 24 February 2023, accepted 27 March 2023, date of publication 29 March 2023, date of current version 5 April 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3263196

RESEARCH ARTICLE

Remaining Useful Life Estimation in Prognostics Using Deep Reinforcement Learning

QIANKUN HU, YONGPING ZHAO, YUQIANG WANG, PEI PENG, AND LIHUA REN
College of Energy and Power Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China

Corresponding author: Yongping Zhao (y.p.zhao@163.com)

This work was supported in part by the National Science and Technology Major Project under Grant J2019-I-0010-0010, in part by the Fundamental Research Funds for the Central Universities under Grant NS2022027, and in part by the Science Center for Gas Turbine Project under Grant P2022-B-V-002-001.

ABSTRACT In modern industrial systems, condition-based maintenance (CBM) has been wildly adopted as an efficient maintenance strategy. Prognostics, as a key enabler of CBM, involves the kernel task of estimating the remaining useful life (RUL) for engineered systems. Much research in recent years has focused on developing new machine learning (ML) based approaches for RUL estimation. A variety of ML algorithms have been employed in these approaches. However, there was no research on applying deep reinforcement learning (DRL) to RUL estimation. To fill this research gap, a novel DRL based prognostic approach is proposed for RUL estimation in this paper. In the proposed approach, the conventional RUL estimation task is first formulated into a Markov decision process (MDP) model. Then an advanced DRL algorithm is employed to learn the optimal RUL estimation policy from this MDP environment. The effectiveness and superiority of the proposed approach are demonstrated through a case study on turbofan engines in C-MAPSS dataset. Compared to other approaches, the proposed approach obtains superior performance on all four sub-datasets of C-MAPSS dataset. What is more, on the most complicated sub-datasets FD002 and FD004, the RMSE metric is improved by 14.4% and 7.81%, and the score metric is improved by 3.7% and 48.79%, respectively.

INDEX TERMS Condition-based maintenance, prognostics, remaining useful life estimation, Markov decision process, deep reinforcement learning.

I. INTRODUCTION

Fatal failures may occur in industrial systems due to aging or unexpected incidents. Hence maintenance management plays a key role in modern industrial activities. As an effective maintenance management strategy, condition-based maintenance (CBM) has been wildly studied and adopted by modern industrial systems. In CBM, prognostic technologies are utilized to analyze the available condition monitoring (CM) data. Through the analysis, the potential failures of the monitored system are predicted in advance, and then the appropriate maintenance can be scheduled based on the prediction. Therefore, CBM helps to avoid catastrophic failures and reduce unnecessary maintenance costs. In CBM, the central issue of prognoses is the remaining useful life (RUL) estimation. RUL represents the amount of time a machine is expected to operate before it requires repair or replacement. Once the RUL is accurately estimated, the failure time can be known beforehand, and then the maintenance plan is adjusted accordingly. Therefore, an accurate RUL estimation contributes to enhancing system reliability, improving maintenance efficiency, and achieving cost savings. Due to its advantages and significance, the RUL estimation has generated considerable research interest.

Generally, the existing approaches for RUL estimation can be categorized into three main groups: model-based approaches [1], [2], [3], [4], [5], [6], data-driven approaches [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27] and hybrid approaches [28], [29]. Model-based approaches

The associate editor coordinating the review of this manuscript and approving it for publication was Yongquan Sun.

can precisely estimate the RUL if the degradation process of physical systems is accurately modeled. However, the modeling requires extensive prior knowledge of physical systems, which is usually unavailable in practice. Thus the unavailability restricts the application of model-based approaches. On the other hand, data-driven approaches aim to learn the mapping relationship between available CM data and the corresponding RUL by training with the historical CM data. The prior knowledge of physical systems is not required in the training process. Hence data-driven approaches are preferable when physical models and prior knowledge of industrial systems are difficult to obtain. Moreover, more and more CM data is available in the industry due to the rapid development of sensor technologies. Data-driven approaches are suitable for utilizing useful CM data and able to achieve accurate RUL estimation. Therefore, data-driven approaches have become increasingly popular in the RUL estimation field.

Much research in recent years has focused on developing new data-driven approaches for RUL estimation. In these approaches, many machine learning (ML) algorithms are utilized to learn the mapping relationship between CM data and the corresponding RUL. These adopted ML algorithms can be roughly divided into two groups: general ML algorithms and deep learning (DL) algorithms. The general ML algorithms include support vector regression (SVR) [10], extreme learning machine (ELM) [8], multi-layer perceptron (MLP) [7], etc. Loutas et al. [10] adopted ε-support vector regression (ε-SVR) to estimate the RUL of rolling element bearings. Javed et al. [8] applied ELM to the RUL estimation of turbofan engines. Huang et al. [7] utilized MLP to model and estimate the RUL of the laboratory-tested bearings. In addition, random forest (RF) [12], gradient boosting (GB) [12] and hidden Markov model (HMM) [9] were also adopted for the RUL estimation. These general ML algorithms require appropriate feature engineering, which relies on the relevant expertise of the system. Feature engineering helps improve the performance of the RUL estimation. However, plentiful relevant expertise is needed for feature engineering, and inappropriate features may cause poor performance [20]. In recent years, DL algorithms have rapidly evolved and addressed this issue. DL algorithms can automatically obtain high-level abstractions from raw data without feature engineering. Hence, DL algorithms have been wildly adopted in the RUL estimation field [11], [12], [13], [14], [15], [16], [17], [18], [19], [20].

In recent research, DL algorithms have been introduced into the field of prognostics for RUL estimation. Ren et al. [21] presented a deep neural network (DNN) approach to predict the RUL of rolling bearing. The results of experiments show the superiority and effectiveness of this DL approach. In [22], an enhanced restricted Boltzmann machine (RBM) approach was developed for the RUL prediction of rotating machines. Zhang et al. [12] proposed a multi-objective deep belief networks (DBN) ensemble method for the RUL estimation. Saxena et al. also This method achieves great results on NASA's turbofan engine degradation problem [30]. Due to its powerful ability in capturing temporal information, long short-term memory (LSTM) networks were also applied to the RUL estimation field. The latest proposed approaches based on LSTM include vanilla LSTM [13], LSTM with bootstrap method (LSTMBS) [14], deep bidirectional LSTM (BiLSTM) [16], bidirectional LSTM (BLSTM) [18], multicellular LSTM [23], LSTM with scoring loss function [26], etc. These LSTM based approaches succeed in learning long-term dependencies of the CM data and achieve encouraging performance on the RUL estimation problem. In addition to LSTM, convolutional neural networks (CNN) is another popular DL algorithm in the RUL estimation field. CNN has a powerful representation learning ability and is able to extract useful local features from data [15]. Babu et al. [11] utilized CNN to predict the RUL of aero-engines. In their study, the time window was used to segment the CM data, and the convolutional operation was implemented alone the time dimension of the CM data. Li et al. [15] proposed a deeper CNN model for the RUL estimation. In their model, CNN layers are followed by fully connected networks (FCN). The CNN layers learn the high-level representations of the CM data, and then FCN uses the learned representations for the final RUL estimation [15]. Besides, some combined DL algorithms have also been developed for the RUL estimation. For example, Li et al. [19] proposed a directed acyclic graph (DAG) network combined with CNN and LSTM for the RUL estimation. Similarly, bi-directional gated recurrent units (BGRU) and CNN were combined to predict the RUL in [20]. Moreover, BGRU was also combined with temporal self-attention mechanism in [24].

Although a variety of ML algorithms have been adopted for the RUL estimation in previous research, they all belong to supervised learning and follow the same learning pattern. In this pattern, a labeled training dataset is firstly prepared, consisting of data/label pairs. Then a model is trained on the dataset by using a specific ML algorithm. During training, the algorithm continuously minimizes a loss function and improves the model's fit to the training dataset. However, the continual improvement runs a risk of overfitting, which incurs large generalization error [31]. Thus the learned model tends to have a poor predictive performance on previously unseen data. To prevent overfitting, some regularization methods such as dropout and early-stopping strategy have been employed in supervised learning. However, these methods make a bias-variance tradeoff, and require a lot of manual intervention to choose appropriate hyperparameters or stopping criteria [31]. Therefore due to the overfitting problem, supervised learning algorithms have a limited exploration ability to find the optimal model that has great performance on both the training and unseen testing dataset. In addition to supervised learning, reinforcement learning (RL) [32] is another subbranch of ML. RL has a different learning pattern from supervised learning. In this pattern, a Markov Decision Process (MDP) [32] environment is firstly constructed, instead of a labeled training dataset. Then an RL agent interacts with the environment by trial and error. Based on historical interactions, RL algorithms train the agent to optimize its policy, which aims to maximize the cumulative reward [33]. While interacting, the agent continues to explore in the environment. The exploration aims to avoid converging to some locally optimal policy and approach the globally optimal policy gradually [29]. Therefore, RL has a better exploration ability than supervised learning. In view of its learning mechanism and exploration ability, RL is suitable to solve the RUL estimation problem. Hence there are some recent research applying RL to the field of prognostics for RUL estimation. In [25], the researcher proposed an entropy-based method that combines RL with DL models for the RUL estimation of lithium-ion batteries. And in [34], an RL-based approach was proposed to construct health indicator (HI) for the RUL prediction task based on multi-sensors. In addition to DL, deep reinforcement learning (DRL) is also introduced to the field of prognostics. Lee and Mitici [27] proposed a framework integrating RUL prognostics into predictive maintenance planning. In this framework, the RUL distribution is firstly estimated by CNN combining with Monte Carlo dropout. Based on the estimate, the maintenance planning task is solved as a DRL problem. Unlike [27], in this work we apply DRL to the RUL estimation rather than the maintenance planning task.

DRL [35] is a combination of RL and DL. Recently, DRL has made exciting achievements in many areas such as video games, robotics, intelligent vehicles [35], [36], [37], [38] and so on. These achievements inspire us to apply DRL to prognostics for RUL estimation. In fact, DRL is ideally suitable for solving the RUL estimation problem due to its learning mechanism. From the RL perspective, the RUL estimation problem can be regarded as a game, which is decomposed into a sequential decision-making process. In each time step, the agent receives an environment state represented by a fragment of CM data, then performs a prediction action about the RUL value of the received state. Next, the agent will receive a reward from the environment, which evaluates the accuracy of RUL prediction. The goal of the agent is to obtain as much more cumulative reward as possible, i.e., to predict the RUL of CM data as accurately as possible. Trained by DRL algorithms, the agent can learn an optimal policy that has great performance on the RUL estimation problem. Therefore, the DRL is introduced into the field of prognostics for RUL estimation in this work. To the best of our knowledge, this is the first attempt to apply DRL to RUL estimation. The major contributions of this paper are summarized as follows:

1) A novel DRL based prognostic approach is proposed for RUL estimation. The proposed approach contains two major parts:
    a) The RUL estimation problem is formulated into an MDP model that is suitable for DRL algorithms' application.
    b) An advanced DRL algorithm, Proximal Policy Optimization (PPO) [39], is adopted to learn the optimal estimation policy in the MDP model of RUL estimation.
2) A case study based on turbofan engines is performed to validate the proposed approach. In the case study, some experiments are conducted on a turbofan engine benchmark dataset from NASA [30]. The experimental results demonstrate the effectiveness of the proposed approach. Besides, compared with other state-of-the-art approaches, the proposed approach is proven to be superior.

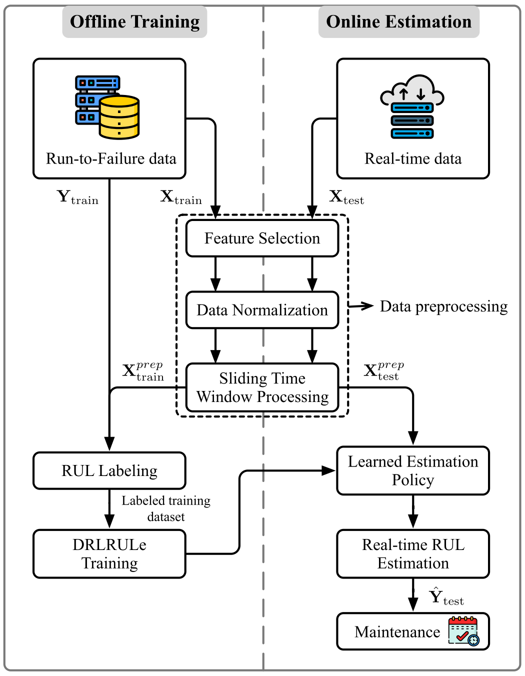

Offline Training
Online Estimation
Run-to-Failure data
Real-time data
$\mathbf{Y}_{\text{train}}$
$\mathbf{X}_{\text{train}}$
$\mathbf{X}_{\text{test}}$
Feature Selection
Data Normalization
Data preprocessing
Sliding Time Window Processing
$\mathbf{X}^{prep}_{\text{train}}$
$\mathbf{X}^{prep}_{\text{test}}$
RUL Labeling
Labeled training dataset
DRLRULe Training
Learned Estimation Policy
Real-time RUL Estimation
$\hat{\mathbf{Y}}_{\text{test}}$
Maintenance

FIGURE 1. The flowchart of the proposed approach.

The rest of this paper is organized as follows. Section II presents the preliminaries and problem statement. Section III elaborates the proposed approach. Subsequently, the proposed approach is experimentally validated in Section IV. At the end, Section V concludes this paper.

II. PRELIMINARIES AND PROBLEM STATEMENT

This paper considers a similar set of engineered systems, whose health conditions are monitored during operation. While operating, the system's CM data are gathered by various sensors in real-time. These sensors consist of a pressure sensor, a speed transducer, and so on. Each kind of sensor data is regarded as an input feature. The gathered CM data are stored in the historical operation database, used for training or estimation (denoted as $\mathbf{X}$). Moreover, the monitoring time is also recorded throughout the life cycle. All monitored engineered systems are categorized into two groups: training equipment and testing equipment. Training equipment refers to a set of failure engineered systems. Their corresponding run-to-failure (RtF) CM data are collected and stored as the training data (denoted as $\mathbf{X}_{\text{train}}$). Besides, the RULs of failure engineered systems are known at every moment and stored as the training labels (denoted as $\mathbf{Y}_{\text{train}}$). On the other hand, testing equipment refers to a set of working engineered systems. Their CM data is gathered in real-time and stored as the testing data (denoted as $\mathbf{X}_{\text{test}}$). Obviously, the RULs of working engineered systems (denoted as $\mathbf{Y}_{\text{test}}$) is unknown at any monitoring time, which need to be estimated.

For this problem, regression algorithms are usually adopted to learn the mapping relationship between CM data and corresponding RULs. This mapping relationship (denoted as function f) is learned based on the training set $(\mathbf{X}_{\text{train}}, \mathbf{Y}_{\text{train}})$. Then the RULs of real-time CM data $\mathbf{X}_{\text{test}}$ can be estimated as

$$\hat{\mathbf{Y}}_{\text{test}} = f\left(\mathbf{X}_{\text{test}}\right). \tag{1}$$

In this paper, the RUL estimation problem is regarded as a sequential decision-making problem, and deep reinforcement learning is employed to learn the optimal estimation policy (i.e., the mapping function f). The proposed approach will be elaborated in the following section.

III. PROPOSED APPROACH

The flowchart of the proposed approach is given in FIGURE 1. As can be seen from FIGURE 1, the proposed approach contains two stages: offline training and online estimation. Besides, these two stages share the common "data preprocessing" procedure. In the offline stage, the training data $\mathbf{X}_{\text{train}}$ is extracted and preprocessed. Then the preprocessed data $\mathbf{X}^{prep}_{\text{train}}$ is labeled with corresponding RULs contained in $\mathbf{Y}_{\text{train}}$. Finally, the labeled training dataset is input into the proposed DRL learning framework for training. Thus an RUL estimation policy can be learned from the labeled training dataset. In the online stage, the real-time data $\mathbf{X}_{\text{test}}$ is also preprocessed. Then the learned estimation policy is employed to estimate real-time RULs based on $\mathbf{X}^{prep}_{\text{test}}$. With the estimated RULs, the maintenance personnel can schedule appropriate maintenance. The proposed DRL learning framework is the kernel of our approach, which is introduced in the following.

In this paper, a DRL based RUL estimation policy learning framework (DRLRULe), is proposed to learn an optimal RUL estimation policy from the labeled training dataset. FIGURE 2 displays the overall structure of DRLRULe. As shown in FIGURE 2, an RUL estimation Markov Decision Process environment (RULeMDP), is firstly constructed base on the labeled training dataset. Then an agent starts to interact with the environment. In every interaction, the agent observes a state s_t, takes an estimation action a_t and finally receives an reward r_t. All historical interactions will be stored in the trajectory τ. At the end, the DRL algorithm is utilized to optimize the agent's estimation policy based on the interaction trajectory. By repeating the above steps, an optimal estimation policy will be gradually learned.

In the following Section III-A, the proposed DRLRULe will be explained in detail. The "data preprocessing" procedure will be introduced in Section III-B.

A. DRL BASED RUL ESTIMATION POLICY LEARNING FRAMEWORK (DRLRULe)

1) REINFORCEMENT LEARNING BACKGROUND

In the context of RL, an agent aims to learn an optimal policy through the interactions with the environment. This process is formally described as Markov Decision Process (MDP) [32]. FIGURE 3 illustrates the agent-environment interaction in an MDP. At the beginning, the very first state of the environment s_0 is randomly sampled from the start-state distribution ρ_0: $s_0 \sim \rho_0(\cdot)$. Then the agent starts to interact with the environment. At time t, the agent takes an action $a_t = \pi\left(s_t\right), a_t \in \mathcal{A}$, based on current state $s_t \in \mathcal{S}$ (π is the control policy, $\mathcal{S}$ is the set of all valid states, $\mathcal{A}$ is the set of all valid actions). Then the environment transfers to next state $s_{t+1} \sim P(\cdot|s_t, a_t)$ (P is the transition probability function). And the agent receives a reward $r_t = R\left(s_t, a_t, s_{t+1}\right), r_t \in \mathbb{R}$ from the environment (R is the reward function). The agent will continually interact with the environment until a terminate state arrives, which is called as an epoch. The sequence of states, actions and rewards obtained in an epoch is denoted as trajectory τ, and $\tau = \{(s_t, a_t, r_t)\}_{t=0}^{T-1}$ where T is the trajectory length. In addition, if the actions available to the agent are finite discrete values, this MDP has a discrete action space. On the other hand, if the actions are real-valued vectors, this MDP has a continuous action space. The goal of the agent is to learn an optimal control policy π^*, which maximizes the expected return $J(\pi) = \underset{\tau\sim\pi}{\mathrm{E}}[R(\tau)]$. In summary, an MDP is represented by $\langle \mathcal{S}, \mathcal{A}, R, P, \rho_0\rangle$. After an MDP is formulated, many DRL algorithms can be applied to it, which train the agent to learn the optimal policy π^*.

2) RUL ESTIMATION MARKOV DECISION PROCESS

The RUL estimation problem is conventionally regarded as a regression problem, and many supervised learning algorithms have been applied to it. Whereas in this paper, we regard this problem as a decision-making problem and adopt DRL algorithms to solve it. In this decision-making problem, the agent receives a CM data sample (i.e., the state s_t) at each time step. Then the agent estimates the RUL of the received CM data (i.e., takes an action a_t). Subsequently, the agent will obtain a reward r_t and the next sample data (i.e., s_{t+1}) from the environment. The reward evaluates the accuracy of RUL estimation. When the agent learns an optimal policy from the interactions with the environment, it will gradually be able to estimate the RUL of CM data as accurately as possible. Therefore, the conventional RUL estimation problem can be formulated into an MDP model, i.e., the RUL estimation Markov Decision Process (RULeMDP). FIGURE 2 contains the overall process of RULeMDP. As shown in FIGURE 2, the labeled training dataset $D = \{(X_i, \text{RUL}_i)\}_{i=1}^{n}$ and contains n data pairs, and each pair consists of a CM data sample X and its corresponding RUL. Before the start of each epoch, m data pairs are randomly sampled from the dataset to compose the states of RULeMDP: $\mathcal{S}_e = \left\{\left(X_j^e, \text{RUL}_j^e\right)\right\}_{j=0}^{m-1}$ where the superscript e denotes epoch number. When the epoch begins,

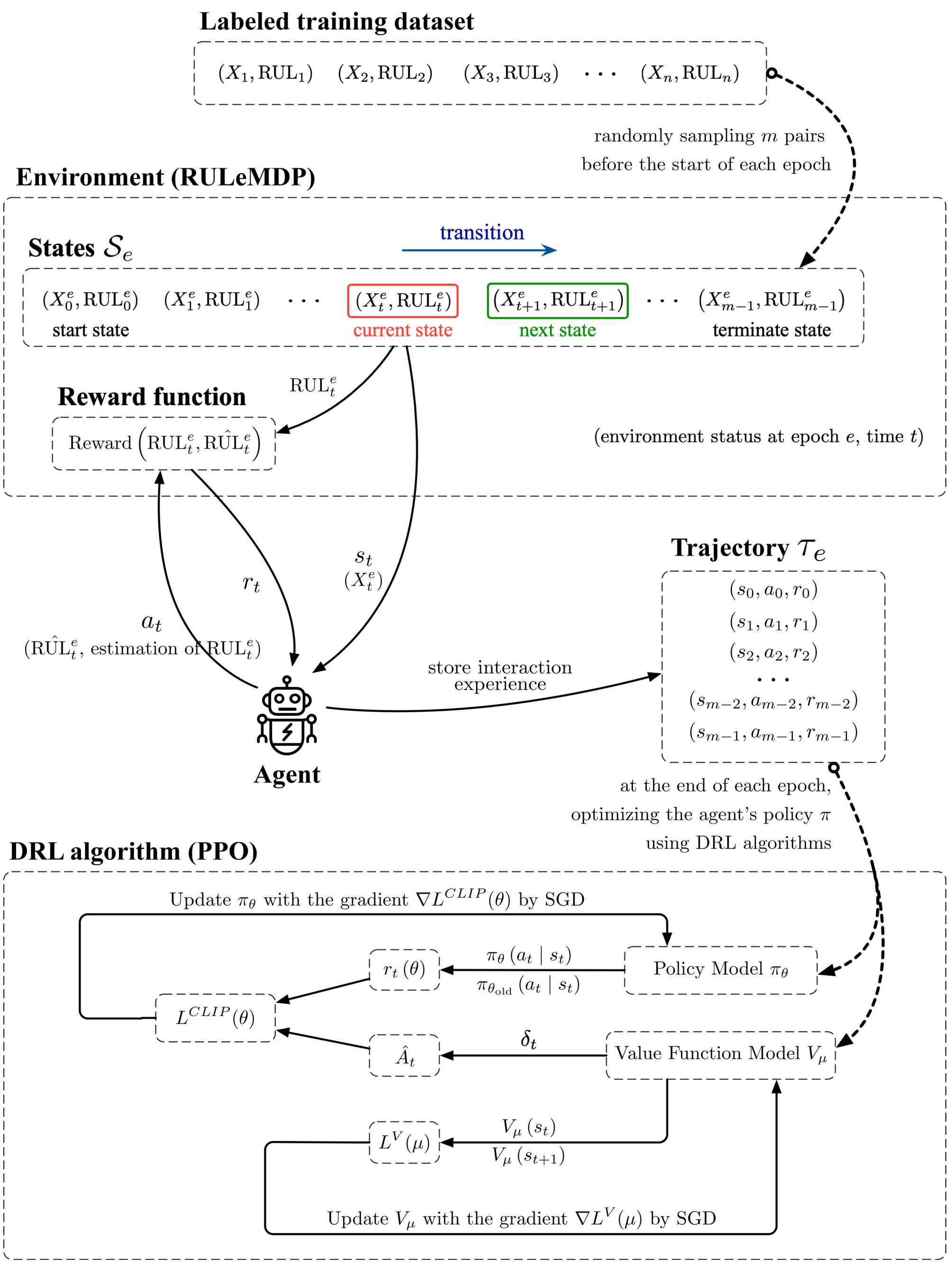

Labeled training dataset
(X_1, RUL_1) (X_2, RUL_2) (X_3, RUL_3) $\cdots$ (X_n, RUL_n)
randomly sampling m pairs before the start of each epoch
Environment (RULeMDP)
States $\mathcal{S}_e$
transition
$(X_0^e, \mathrm{RUL}_0^e)$ start state
$(X_1^e, \mathrm{RUL}_1^e)$
$(X_t^e, \mathrm{RUL}_t^e)$ current state
$(X_{t+1}^e, \mathrm{RUL}_{t+1}^e)$ next state
$(X_{m-1}^e, \mathrm{RUL}_{m-1}^e)$ terminate state
Reward function
RUL_t^e
Reward $\left(\mathrm{RUL}_t^e, \hat{\mathrm{RUL}}_t^e\right)$
(environment status at epoch e, time t)
r_t
s_t (X_t^e)
a_t $(\hat{\mathrm{RUL}}_t^e$, estimation of $\mathrm{RUL}_t^e)$
Agent
store interaction experience
Trajectory $\mathcal{T}_e$
(s_0, a_0, r_0)
(s_1, a_1, r_1)
(s_2, a_2, r_2)
$(s_{m-2}, a_{m-2}, r_{m-2})$
$(s_{m-1}, a_{m-1}, r_{m-1})$
at the end of each epoch, optimizing the agent's policy π using DRL algorithms
DRL algorithm (PPO)
Update π_θ with the gradient $\nabla L^{CLIP}(\theta)$ by SGD
Policy Model π_θ
$r_t(\theta)$
$\pi_\theta(a_t \mid s_t)$
$\pi_{\theta_{\mathrm{old}}}(a_t \mid s_t)$
$L^{CLIP}(\theta)$
$\hat{A}_t$
δ_t
Value Function Model V_μ
$L^V(\mu)$
$V_\mu(s_t)$
$V_\mu(s_{t+1})$
Update V_μ with the gradient $\nabla L^V(\mu)$ by SGD

FIGURE 2. Overall structure of the proposed DRLRULe learning framework.

RULeMDP is at the start state. At each time step, RULeMDP returns the current state (i.e., CM data sample) and a reward, and then transfers to the next state. Once RULeMDP arrives at the terminate state, the current epoch finishes. Then the states will be resampled from D, and next epoch starts.

In order to guide the agent to learn an optimal RUL estimation policy, the reward function for RULeMDP is defined as follows:

$$r_t = R(s_t, a_t, s_{t+1}) = -1 \times \left|a_t - \mathrm{RUL}_t^e\right| \quad (2)$$

where s_t is the current state (i.e., a CM data sample), a_t is the action performed by the agent, RUL_t^e represents the real RUL value of the current state. In Equation 2, a_t represents the prediction action, i.e. the estimated RUL value for the current data sample. The reward function 2 yields reward r_t by calculating the bias between the estimated RUL a_t and real RUL RUL_t^e of the current state s_t. As defined in Equation 2, r_t equals the opposite of absolute estimation bias. Since the agent aims to maximize the cumulative reward in RULeMDP, it is equivalent to minimizing the sum of estimation biases.

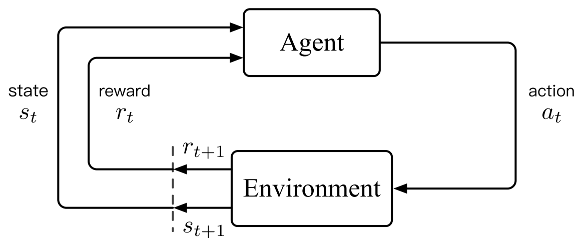


FIGURE 3. The agent–environment interaction in a markov decision process [32].

Therefore, the agent will gradually learn an optimal RUL estimation policy in RULeMDP.

Now we formalize the proposed RULeMDP into a sequential decision-making problem. We propose to train an RUL estimator as an agent evolving in RULeMDP where:

- **State** $\mathcal{S}$: The state of the environment is represented by the CM data sample. The state s_t of environment at each time step corresponds to the CM data sample X_t^e. When a new epoch begins, the states of RULeMDP are resampled from the labeled training dataset D. Thus $\mathcal{S} \subset D$ and $\mathcal{S} \neq \varnothing$.
- **Action** $\mathcal{A}$: The action of the agent is associated with the RULs of the labeled training dataset. The action a_t taken by agent is to estimate the RUL of the CM data sample X_t^e, which is a positive real number. Thus $\mathcal{A} = \mathbb{R}^+$ and RULeMDP has a continuous action space.
- **Reward function** R: A reward r_t is the feedback from the environment through which we measure the accuracy of RUL estimation. It aims to guide the agent to learn an optimal RUL estimation policy. Thus the reward function for RULeMDP has been carefully designed in Equation (2).
- **Transition probability function** P: The transition probability $P(\cdot|s_t, a_t)$ in RULeMDP is deterministic. The environment transfer from the current state s_t to the next state s_{t+1} according to the order of the sampled states at each epoch.
- **Start-state distribution** ρ_0: At each epoch, the first CM data X_0^e is always regarded as the initial state s_0. Therefore, $\rho_0(s_0) = 1$ where $s_0 = X_0^e$.
- **Trajectory** τ: The trajectory τ is a sequence of interactions from the state s_0 to the terminal state s_{m-1}: $\tau = \{(s_t, a_t, r_t)\}_0^{m-1}$, which is used for optimizing the agent's policy.
- **Policy** π_θ: The policy π_θ is a mapping function: $\pi : \mathcal{S} \to \mathcal{A}$, where $\pi_\theta(s_t)$ denotes the action a_t performed by the agent at state s_t. The policy π_θ in RULeMDP can be considered as an RUL estimator, represented by a neural network with parameter θ. The network architecture will be introduced in Section III-A4.

With the definitions and notations above, the RUL estimation problem is formally defined as finding an optimal policy π^*, which maximizes the cumulative rewards in RULeMDP. Therefore, any DRL algorithms that can deal with continuous action space can be employed to find the optimal RUL estimation policy in RULeMDP. In this paper, PPO algorithm is applied to RULeMDP, which is detailed in the following Section III-A3

3) PROXIMAL POLICY OPTIMIZATION ALGORITHM FOR RULeMDP

In this paper, we use an advanced DRL algorithm, i.e., PPO, to train the RUL estimator in RULeMDP. PPO is a model-free, on-policy, and policy gradient algorithm proposed in [39]. It can be used for environments with either discrete or continuous action spaces and has reliable performance. The complete PPO algorithm is given in Appendix B.

Let π_θ denote the agent's policy with parameter θ. At the end of each epoch, PPO optimizes the policy via gradient ascent:

$$\theta_{k+1} = \arg\max_{\theta} \mathop{\mathrm{E}}_{\tau\sim\pi_{\theta_k}} \left[L^{CLIP}(s, a, \theta_k, \theta) \right]. \tag{3}$$

The L^{CLIP} in Equation (3) is given by:

$$\begin{aligned} L^{CLIP}(s, a, \theta_k, \theta) &= \min\Bigg(\frac{\pi_\theta(a \mid s)}{\pi_{\theta_k}(a \mid s)} A^{\pi_{\theta_k}}(s, a), \\ &\quad \text{clip}\left(\frac{\pi_\theta(a \mid s)}{\pi_{\theta_k}(a \mid s)}, 1-\epsilon, 1+\epsilon \right) A^{\pi_{\theta_k}}(s, a) \Bigg) \end{aligned} \tag{4}$$

where $A^{\pi_{\theta_k}}(s, a)$ represents the advantage for taking action a at state s, the clip ratio ϵ represents how far away the new policy is allowed to go from the old, and clip function limits the value of $\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}$ between $1-\epsilon$ and $1+\epsilon$. In PPO, the advantage $A^{\pi_{\theta_k}}(s, a)$ is estimated based on the value function V_μ, which is also represented by neural networks with parameter μ. The value $V_\mu(s_t)$ equals the expected return an agent will obtain if it starts from s_t and then acts according to a particular policy forever after. At the end of an epoch, the value function V_μ is also updated via gradient descent to minimize the mean-squared error:

$$L^V(\mu) = \left(V_\mu(s_t) - \hat{R}_t \right)^2. \tag{5}$$

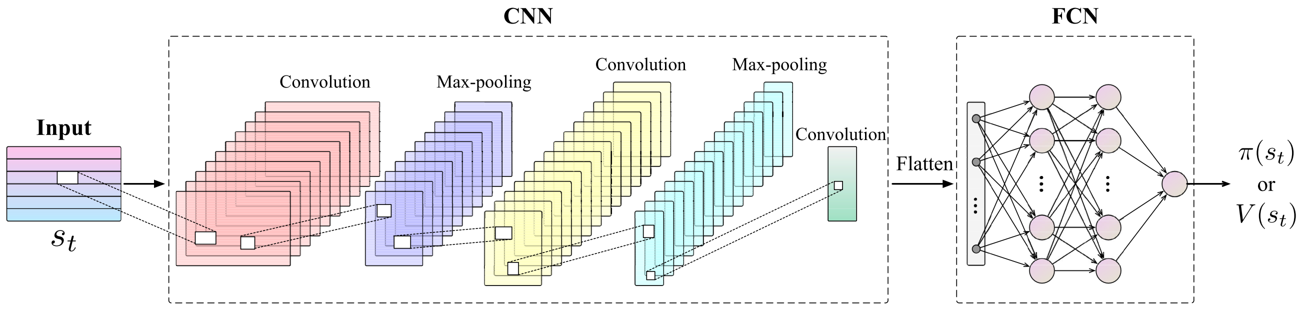
Because RULeMDP has a continuous action space, the diagonal Gaussian policy model is used in PPO. This policy maps from the state s_t to mean action $\mu(s_t)$: $\mu(s_t) = \pi_\theta(s_t)$. At every time step t, the action a_t is generated by

$$a_t = \mu(s_t) + \sigma(s_t) \odot z \tag{6}$$

where $\odot$ denotes the element-wise product of two vectors, $\sigma(s)$ denotes the standard deviation and z is a noise vector from spherical Gaussian (i.e., $z \sim \mathcal{N}(0, I)$). The standard deviation $\sigma(s)$ is usually considered as a constant in PPO implementation.

4) NEURAL NETWORK ARCHITECTURE OF POLICY AND VALUE FUNCTION

In RULeMDP, the state s_t, i.e., the CM data sample, is a 2-dimensional (2D) matrix that contains multiple sensor data



CNN Parameters for each layer

Feature maps	10	10	14	14	1
Kernel size	(1,13)	(1,2)	(1,3)	(1,2)	(1,1)
Padding size	(0,6)	/	(0,1)	/	/
Activation function	tanh	/	tanh	/	tanh

FCN Parameters

Hidden neurons	20	10	1
Activation function	ReLU	ReLU	/

FIGURE 4. The neural network structure for policy π_θ and value function V_μ.

points gathered at T consecutive monitoring cycles:

$$s_t = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_t, \ldots, \boldsymbol{x}_T] \in \mathbb{R}^{F\times T} \tag{7}$$

where $\boldsymbol{x}_t \in \mathbb{R}^F$ denotes the multi-sensory data collected at monitoring cycle t, and F represents the number of features (i.e., sensors). In PPO, s_t is mapped to the mean action $\mu(s_t)$ and the value $V(s_t)$ by the policy π_θ and the value function V_μ. To efficiently map s_t, we design a CNN based neural network for π_θ and V_μ. FIGURE 4 illustrates the network structure and its detailed parameters. As shown in FIGURE 4, this structure consists of two parts: CNN and fully connected network (FCN). Firstly, CNN is used to extract deeper features from the state s_t. Then FCN is employed to map the extracted features to the corresponding π_θ or V_μ value.

CNN is initially proposed for image processing. In recent research, CNN has also been applied to sequential data processing [11], [15], [19], [20], and achieves promising results. In CNN, the convolution operations can extract important local features from input data. And the pooling operations reduce the data size to avoid overfitting. The convolution operation in CNN is defined as

$$z_i = \tanh\left(\boldsymbol{I} * \boldsymbol{f}_i + \boldsymbol{b}_i\right) \tag{8}$$

where $\boldsymbol{I}$ is the input matrix, $*$ denotes convolution operation, $\boldsymbol{f}_i$ represents the i^{th} convolution filter, and $\boldsymbol{b}_i$ is the bias term. The activation function tanh is adopted in this paper. Hence the i^{th} feature map z_i is obtained in Equation 8. If a convolution layer has F filters, its output is expressed as

$$\boldsymbol{Z} = [z_1, z_2, \cdots, z_F]. \tag{9}$$

After the convolution layer, a pooling layer is applied to the output $\boldsymbol{Z}$. Max-pooling is adopted in this paper, which can be expressed as

$$\boldsymbol{g}_i = \text{MaxPooling}\left(\boldsymbol{P}_i\right) \tag{10}$$

where $\boldsymbol{P}_i$ represents the pooling filter matrix in i^{th} feature map, $\boldsymbol{g}_i$ is the pooling output and MaxPooling denotes the operation that selects the maximum element in $\boldsymbol{P}_i$. As shown in FIGURE 4, 2-layer CNN is firstly used to extract local features of s_t. Each CNN layer consists of a convolution and pooling layer. In addition, the padding operation is performed in every convolution layer to keep the data size unchanged. Then a convolution layer is adopted to fuse the features in different feature maps. Next, the fused feature map is flattened into a one-dimensional form. Finally, a 3-layer FCN is employed to map the flatten features to π_θ or V_μ value.

B. DATA PREPROCESSING

1) DATA NORMALIZATION

In practical prognostic applications, the multiple sensors data are first gathered. Generally, different sensor data have different scales. If the raw data are directly used for model training, the unequally weighted input will make the algorithm difficult to converge. Therefore, data normalization is necessary before model training, which converts the raw data into the same scale. In addition, the engineered systems are usually working at different operational conditions in practice. This difference affects the degradation process of engineered systems [18], which can be revealed in CM data. Thus the operational difference should also be considered while performing data normalization. For the above considerations, z-score normalization [40] is adopted in this paper. This normalization method takes the difference in operational conditions into consideration while normalizing. It is defined as follows

$$\boldsymbol{x}_{\text{norm}}^{(m,f)} = \frac{\boldsymbol{x}^{(m,f)} - \boldsymbol{\mu}^{(m,f)}}{\boldsymbol{\sigma}^{(m,f)}} \tag{11}$$

where $\boldsymbol{x}^{(m,f)}$ is the raw data, $\boldsymbol{x}_{\text{norm}}^{(m,f)}$ is the normalized data, m represents one of the M possible operational conditions, f denotes the f^{th} sensor, $\boldsymbol{\mu}^{(m,f)}$ and $\boldsymbol{\sigma}^{(m,f)}$ are the mean and standard deviation from the m^{th} operational condition in terms of the f^{th} sensor. In practice, all $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are calculated from the training dataset (RtF data), and stored

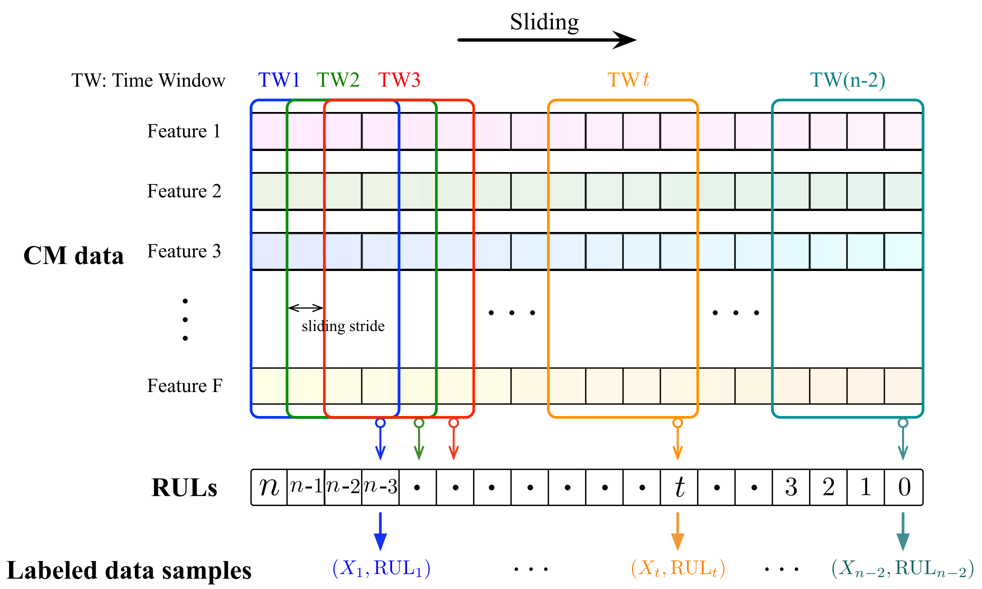


FIGURE 5. Sliding time window processing alone time dimension, where the CM data has F features, the time length is $n + 1$, and for convenience, the window length is set to 4.

for the normalization of the testing dataset (real-time CM data).

2) SLIDING TIME WINDOW PROCESSING

The CM data gathered from each engineered system usually has a different time length. However, the state in RULeMDP is required to have the same data size, i.e., both the same features and time length. To address this issue, sliding time window processing is adopted in this paper. It can convert the raw CM data to fixed-size data samples. FIGURE 5 briefly illustrates this technique. As shown in FIGURE 5, a fixed-length time window is used to enclose multi-feature data points sampled at consecutive monitoring time. The RUL of the last data point in the time window is taken as the RUL of a window. Thus the enclosed CM data and the corresponding RUL form a labeled data sample: (X, RUL), where $X \in \mathbb{R}^{F\times T}$ and $\mathrm{RUL} \in \mathbb{N}$. The time window slides across the time dimension, and the sliding stride is set to 1. Through sliding, a set of labeled data samples is obtained, where every sample has the same data size. Therefore, these samples can be used as the states in RULeMDP.

According to some research [15], [18], the time window with longer length can enclose more important information, which might improve the prognostic performance. However, a longer window will increase the computational load and the complexity of model. In practical applications, the window length should be appropriately chosen according to the raw CM data.

C. SUMMARY OF THE PROPOSED APPROACH

At the end, the algorithmic details of the proposed approach are summarized in Algorithm 1.

IV. CASE STUDY AND DISCUSSION

A. INTRODUCTION TO TURBOFAN ENGINE DATASET

To verify the effectiveness of the proposed approach, a turbofan engine benchmark dataset [30] is used in this paper. This dataset is generated by the Commercial Modular Aero-Propulsion System Simulation (C-MAPSS) program and contains the turbofan engine degradation data. This dataset has been wildly used as the benchmark dataset of RUL estimation. The data in C-MAPSS is gathered from 21 sensors measurements and 3 operational conditions measurements. According to the fault modes and operational conditions, the C-MAPSS dataset is divided into four sub-datasets. Furthermore, each sub-dataset consists of a training dataset and a testing dataset. In the training dataset, the Run-to-Failure sensor data of turbofan engines and the corresponding RULs are offered. On the other hand, the testing dataset provides the real-time sensor data collected before the failure occurs. Hence the RULs of the testing engines need to be estimated. For evaluation, the actual RULs of the testing engines are also offered in the testing dataset. The details about the C-MAPSS benchmark dataset are given in Table 1.

B. PROGNOSTIC PERFORMANCE METRICS

To evaluate the RUL estimation performance of the proposed approach, two popular metrics are adopted in this paper. The first metric is the root mean squared error (RMSE) in Equation (14), which is wildly employed in regression problems. The other one is the RUL scoring function in Equation (15), which is developed in the 2008 PHM data challenge competition [30]. It is a wildly

Algorithm 1 The Proposed DRLRULe

```
         Input: Hyper-parameters(epochs, learning rate, etc.), the raw training set ($\mathbf{X}_{\text{train}}, \mathbf{Y}_{\text{train}}$).
         Output: An optimal learned policy $\pi^*$ for RUL estimation.
Step 1   Data Preprocessing: In this step, $\mathbf{X}_{\text{train}}$ is preprocessed by Feature Selection, Data Normalization and Sliding
             Time Window Processing, which generates the preprocessed $\mathbf{X}^{prep}_{\text{train}}$.
Step 2   RUL Labeling: Label $\mathbf{X}^{prep}_{\text{train}}$ with the corresponding RULs in $\mathbf{Y}_{\text{train}}$, which yields the labeled training dataset
             $D = \{(X_i, \text{RUL}_i)\}_{i=1}^{n}$.
Step 3   Initializing: Initialize the RULeMDP environment, the policy model $\pi_\theta$ and value function model $V_\mu$.
Step 4   Policy Learning:
         for $e = 0, 1, 2, \ldots, epochs$ do
Step 4.1     Randomly sample $m$ data pairs from $D$ to form the states $\mathcal{S}_e$ of RULeMDP, and $\mathcal{S}_e = \left\{\left(X_j^e, \text{RUL}_j^e\right)\right\}_{j=0}^{m-1}$;
Step 4.2     Interacting: Collect the trajectory $\tau_e$ by running policy $\pi_{\theta_e}$ in the environment.
             for $t = 0, 1, 2, \ldots, m-1$ do
Step 4.2.1       The agent receives a state $s_t = X_t^e$ from the environment;
Step 4.2.2       The agent takes an action $a_t$ according to current policy $\pi_{\theta_e}$ by Equation (6);
Step 4.2.3       The environment returns a reward $r_t$ to the agent by Equation (2);
Step 4.2.4       The agent stores the interaction experience $(s_t, a_t, r_t)$ in $\tau_e$;
             end
Step 4.3     Policy Optimizing: Use PPO algorithm to optimize the policy $\pi_e$ based on the trajectory $\tau_e$.
Step 4.3.1   Compute rewards-to-go $\hat{R}_t = \sum_{t'=t}^{T} r_{t'}$ for every state in $\tau_e$, where $T$ denotes the length of $\tau_e$;
Step 4.3.2   Compute advantage estimates, $\hat{A}_t$ (using GAE method [41]) based on the current value function $V_{\mu_e}$;
Step 4.3.3   Update the policy by maximizing the PPO objective:
                 $$\theta_{e+1} = \arg\max_{\theta} \frac{1}{T}\sum_{t=0}^{T} \min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_e}(a_t|s_t)} A^{\pi_{\theta_e}}(s_t, a_t),\quad \text{clip}\left(\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_e}(a_t \mid s_t)}, 1-\epsilon, 1+\epsilon\right) A^{\pi_{\theta_e}}(s_t, a_t)\right), \quad (12)$$
               via stochastic gradient ascent with Adam algorithm;
Step 4.3.4   Fit value function by regression on mean-squared error:
                 $$\phi_{e+1} = \arg\min_{\mu} \frac{1}{T}\sum_{t=0}^{T}\left(V_\mu(s_t) - \hat{R}_t\right)^2, \quad (13)$$
               via stochastic gradient descent with Adam algorithm;
         end
```

TABLE 1. Details of the C-MAPSS benchmark dataset [30].

Sub-dataset	FD001	FD002	FD003	FD004
Training engines number	100	260	100	248
Testing engines number	100	259	100	248
Operational conditions	1	6	1	6
Fault modes	1	1	2	2

acknowledged metric in prognostics, and a smaller score means a better RUL estimation performance. The scoring function penalizes late estimations (i.e., the estimated RUL is larger than the actual) more than early estimations. By contrast, RMSE treats the late and early estimations equally.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} d_i^2} \qquad (14)$$

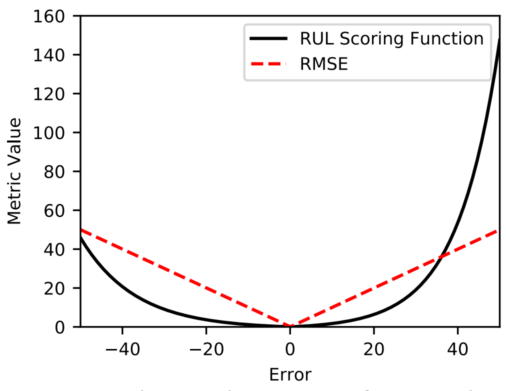


FIGURE 6. Comparison between the RUL scoring function and RMSE.

$$S = \begin{cases} \sum_{i=1}^{n}\left(e^{-\frac{d_i}{13}} - 1\right), & d_i < 0 \\ \sum_{i=1}^{n}\left(e^{\frac{d_i}{10}} - 1\right), & d_i \geq 0 \end{cases} \qquad (15)$$

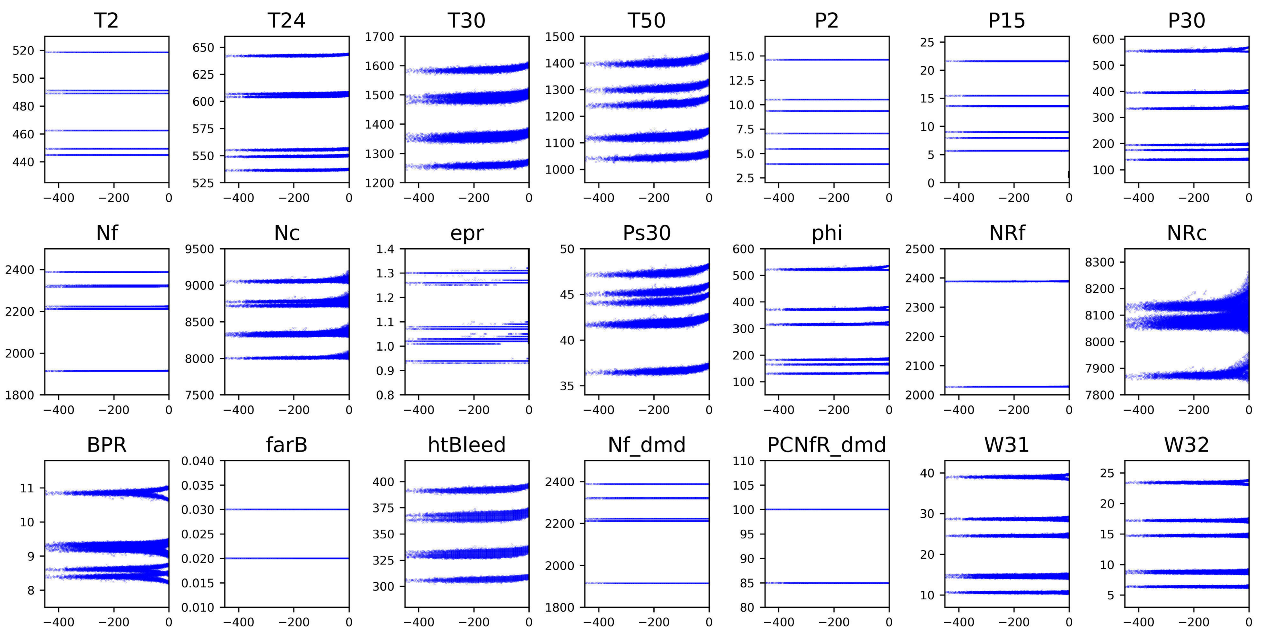


FIGURE 7. All 21 sensors measurements from the sub-dataset FD004. The abscissa of these figures represents the index of monitoring time cycle. The last time cycle (i.e., the failure time) corresponds to index 0. Hence all the previous monitoring time cycles have negative indexes.

where n is the number of testing data samples and $d_i = \hat{\mathrm{RUL}}_i - \mathrm{RUL}_i$ is the error benchmark the estimated RUL and the actual RUL of the i^{th} data sample. The difference between the two metrics is illustrated in FIGURE 6.

TABLE 2. Details of time window processing.

Sub-dataset	FD001	FD002	FD003	FD004
Min. time length of training data	128	128	145	128
Min. time length of testing data	31	21	38	19
Time window length	30	20	30	18
Training data samples	17731	48819	21820	57016
Testing data samples	10196	29070	13696	36998

C. DATA PREPROCESSING FOR CASE STUDY

As mentioned earlier, the first step of data preprocessing is to perform feature selection. In order to select the most informative features, FIGURE 7 displays the degradation measurements of all 21 sensors from the sub-dataset FD004. The different clusters in the figures of FIGURE 7 represent turbofan engines work under different operational conditions. As shown in FIGURE 7, there are some sensor measurements remaining constant, which cannot provide useful degradation information. In this paper, those constant measurements from 7 sensors (i.e., sensor T2, P2, P15, epr, farB, Nf_dmd, and PCNfR_dmd) are discarded. Hence the remaining 14 sensor measurements are selected as the raw input of the proposed approach.

After feature selection, normalization is performed on the selected sensor data. FIGURE 8 shows the sensory data of a single turbofan engine before and after the normalization. As shown in FIGURE 8, the scales of the raw data vary significantly with each other. After the normalization, these scales are converted into a normalized range. Besides, the normalized sensory data displays a clear degradation trend of turbofan engines.

Finally, sliding time window processing is employed on the normalized sensor data. For the C-MAPSS dataset, the time window length is determined according to the time lengths of the sensor data in the training and testing datasets. Table 2 gives the minimum time lengths of each sub-dataset. Thus the time window length should be shorter than them. In this paper, we choose the longest possible time window length to improve the estimation performance. Besides, according to [15], when the time window length is longer than 30, it will not remarkably improve the estimation performance. Instead, it will increase the computing load. Considering the above, the determined time window lengths are given in Table 2. Table 2 also contains the numbers of data samples generated through time window processing.

In addition the above steps, the piece-wise linear RUL target function [42] is also adopted in this paper. By convention, RUL is considered to decrease linearly with time. However, the degradation of machines is not obvious at the early stage of the entire life. Hence at the start period, the RUL can be considered as constant, and when the degradation occurs, the RUL can be considered as a linearly decreasing value. Thus the piece-wise linear RUL target function has been proposed in [42]. It limits the maximum value of the RULs, as shown in FIGURE 9. This function is adopted in this paper to obtain the RUL labels.

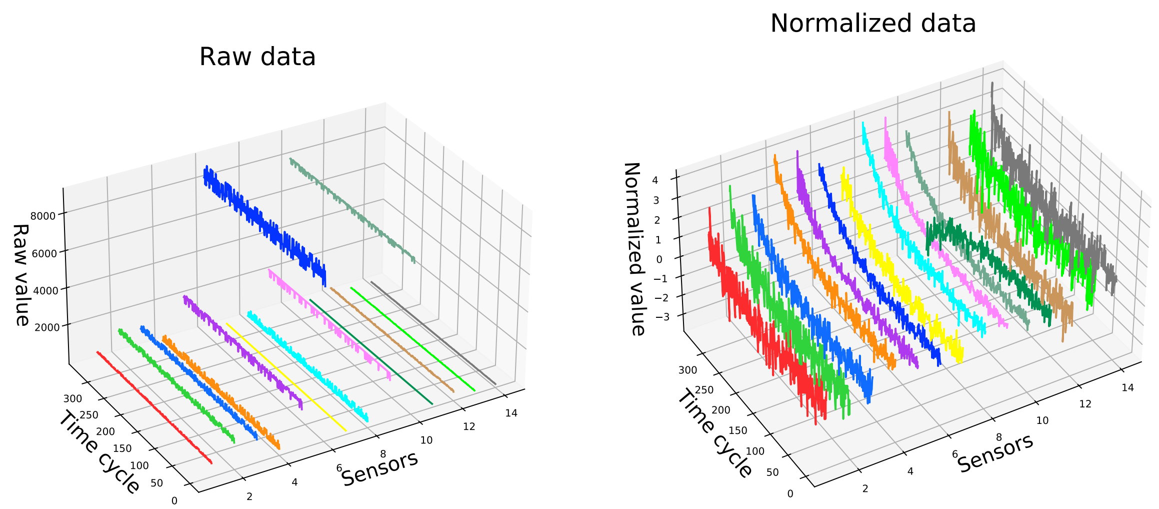


FIGURE 8. Illustration of the normalization.

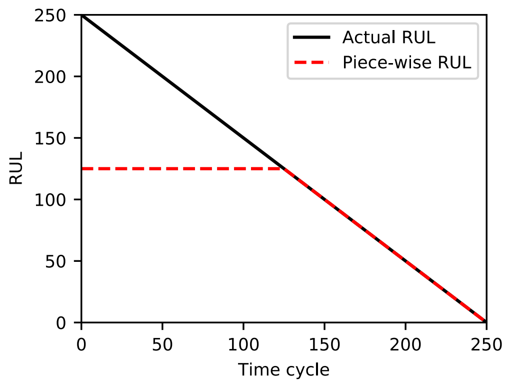


FIGURE 9. The piece-wise linear RUL target function.

D. CASE STUDY RESULTS

In this paper, the experiments are performed on a 64-bit Windows 10 PC equipped with a 32GB RAM, an Intel i5-10400 CPU and an Nvidia RTX2070 GPU. The programming language 'Python 3.8' with the deep learning library 'PyTorch 1.5.0' is utilized to implement the proposed DRL-RULe. Appendix C gives the hyper-parameters of DRLRULe, and the hyper-parameters of CNN model for π and V are given in FIGURE 4 for convenience. In Appendix C, the sampling ratio m/n belongs to (0, 1], and represents the ratio of m sampled states in RULeMDP to n data samples in the labeled training dataset. Since n is fixed before training, a larger m/n means more states in RULeMDP, that is, a longer trajectory τ in every epoch. That will increase the computation load of the gradient descent/ascent steps in Algorithm 1. On the other hand, too small m/n will make the algorithm unstable and difficult to converge. Therefore, the sampling ratio is set to 0.25 in these experiments.

For each sub-dataset in the C-MAPSS dataset, an RUL estimation policy is learned by DRLRULe based on the training engines. Then the learned policy is used to estimate the RULs of the testing engines. In this paper, each experiment is repeated ten times to reduce the effect of randomness, and the results are averaged. Table 3 presents the experimental results of these four sub-datasets. As shown in Table 3, the RMSE and score of FD001 and FD003 are lower than FD002 and FD004. The reason is that FD002 and FD004 have more data samples and are more complicated than FD001 and FD003, which is also revealed in Table 1 and 2.

TABLE 3. Experimental results of the C-MAPSS dataset.

Sub-dataset	FD001	FD002	FD003	FD004
RMSE	12.17	16.28	13.09	18.87
Score	208.06	1436.81	225.50	1725.74

FIGURE 10 illustrates the comparisons between the estimated RUL and the actual RUL values. The light blue region in FIGURE 10 represents the 10% error bound of the RUL estimation. As displayed in FIGURE 10, the learned policy is capable of estimating the RUL values within the error bound. That indicates the proposed approach achieves promising performance in RUL estimation. It can also be observed from FIGURE 10 that in the early period, the estimated RULs remain close to a constant value. Afterwards, the estimated RUL decreases almost linearly with the time until the end of the monitoring time. Although there are some obvious errors between the estimated and actual RUL, the estimation accuracy is high in the late period, where the engines are close to failure. This has industrial value because the late period of engine life-time is very critical in PHM. An accurate RUL estimation in the late period contributes to enhancing system reliability and safety.

E. COMPARISON WITH OTHER APPROACHES

The C-MAPSS dataset has been wildly used as a benchmark dataset in prognostics, and many approaches have been applied to this dataset. To validate the superiority of

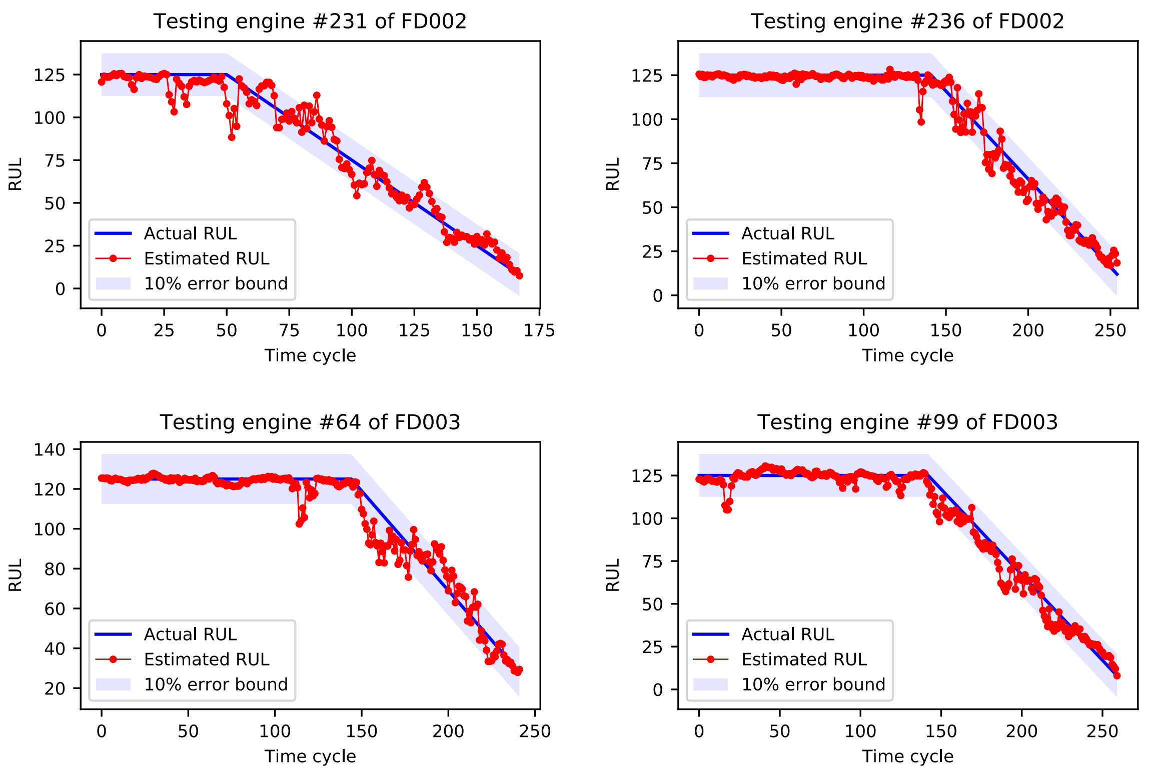


FIGURE 10. Comparisons between the estimated RUL and actual RUL of testing engines.

our approach, we compare the performance of DRLRULe on C-MAPSS dataset with other state-of-the-art approaches. These existing approaches all adopt the supervised learning algorithms to learn the RUL estimation model. Table 4 and Table 5 provide the comparisons of RMSE and score between different approaches. The 1st and 2nd best results are highlighted in red and blue respectively. In addition, the improvement (IMP) of DRLRULe over other approaches is calculated in these tables.

As shown in Table 4, the proposed approach obtains the best results on FD002 and FD004 in terms of RMSE, which are improved by 14.04% and 7.81% each. Although DRLRULe does not achieve improvements on FD001 and FD003, comparable results are obtained. For the score metric, as shown in Table 5, DRLRULe outperforms all other approaches. According to Table 5, the proposed approach obtains the superior performance on all sub-datasets compared with other approaches. Moreover, a significant improvement is achieved on FD004, where the score is reduced by 48.79%. Considering FD004 is the most complicated one in four sub-datasets, this significant improvement reveals the remarkable ability of the proposed approach in dealing with the RUL estimation problem. Although the proposed approach does not obtain best results on sub-datasets FD001 and FD003 in terms of RMSE, it outperforms other approaches on both the score metric on all sub-datasets and the RMSE metric on sub-datasets FD002 and FD004. In prognostic filed, the score metric is more wildly acknowledged than RMSE because it penalizes late estimations more than early ones, as shown in Figure 6. This imbalanced penalty helps improve the condition-based maintenance in modern industrial systems. Therefore, from the prognostic perspective, the proposed DRLRULe still obtains best results on FD001 and FD003.

As evident from Table 4 and Table 5, the RUL estimation performance keeps improving from ML model based approaches [11], [12] to DL model based ones [13], [14], [15], [16], [18], [19], [20]. Compared to DL models, general ML models such as SVR, RF, and GB have limited feature extraction ability. Hence they require appropriate feature engineering in application. And inappropriate features may cause poor performance. These limitations result in large errors in the RUL estimation task. In contrast, DL models such as CNN and LSTM can automatically obtain high-level abstractions from raw data without feature engineering. Thus the RUL estimation performance is improved in DL model based approaches. However, DL models require the manual intervention to choose appropriate hyper-parameters or stopping criteria. Besides, due to the overfitting problem, DL models have a limited exploration ability to find the optimal model that has great performance on both the training and unseen testing dataset. Thus these drawbacks of DL models lead to an inferior RUL estimation performance compared to the proposed DRLRULe method. In supervised learning approaches, an RUL estimation model is built and used to fit the whole training dataset as much as possible. Unlike the

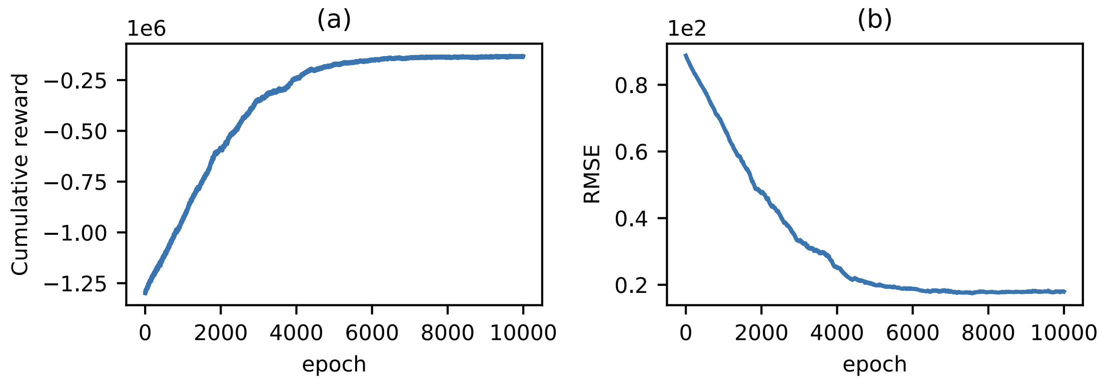


FIGURE 11. The learning curves in one training process of DRLRULe on FD004: (a) the cumulative reward during training; (b) the estimation performance (i.e., RMSE) on testing engines during training.

TABLE 4. Comparison of RMSE on C-MAPSS dataset.

Approaches	RMSE			
	FD001	FD002	FD003	FD004
SVR [11]	20.96	42.00	21.05	45.35
Earlier CNN [11]	18.45	30.29	19.82	29.16
RF [12]	17.91	29.59	20.27	31.12
GB [12]	15.67	29.09	16.84	29.01
ELM [12]	17.27	37.28	18.90	38.43
DBN [12]	15.21	27.12	14.71	29.88
MODBNE [12]	15.04	25.05	12.51	28.66
LSTM [13]	16.14	24.49	16.18	28.17
DCNN [15]	12.61	22.36	12.64	23.31
LSTMBS [14]	14.89	26.86	15.11	27.11
BiLSTM [16]	13.65	23.18	13.74	24.86
BLSTM [18]	/	25.11	/	26.61
DAG [19]	11.96	20.34	12.46	22.43
AGCNN [20]	12.42	19.43	13.39	21.50
MCLSTM [23]	13.71	/	/	23.81
BiGRU-TSAM [24]	12.56	18.94	12.45	20.47
DRLRULe	12.17	16.28	13.09	18.87
IMP	/	14.04%	/	7.81%

TABLE 5. Comparison of score on C-MAPSS dataset.

Approaches	Score			
	FD001	FD002	FD003	FD004
SVR [11]	1381.50	589900.00	1598.30	37114.00
Earlier CNN [11]	1286.70	13570.00	1596.20	7886.40
RF [12]	479.75	70456.86	711.13	46567.63
GB [12]	474.01	87280.06	576.72	17817.92
ELM [12]	523.00	498149.97	573.78	121414.47
DBN [12]	417.59	9031.64	442.43	7954.51
MODBNE [12]	334.23	5585.34	421.91	6557.62
LSTM [13]	338.00	4450.00	852.00	5550.00
DCNN [15]	273.70	10410.00	284.10	12470.00
LSTMBS [14]	481.10	7982.00	493.40	5200.00
BiLSTM [16]	295.00	4130.00	317.00	5430.00
BLSTM [18]	/	4793.00	/	4971.00
DAG [19]	229.00	2730.00	535.00	3370.00
AGCNN [20]	225.51	1492.00	227.09	3392.00
MCLSTM [23]	315	/	/	4826
BiGRU-TSAM [24]	213.35	2264.13	232.86	3610.34
DRLRULe	208.06	1436.81	225.50	1725.74
IMP	2.48%	3.70%	0.7%	48.79%

supervised learning approaches, DRLRULe exposes a part of the training dataset to the agent in every training epoch. Thus the training dataset is partially observed to the agent. This characteristic makes the agent more robust to unseen data, which avoid the overfitting problem to some extent. Besides, the agent aims to learn a policy, which maximizes the expected reward when estimating the RULs for previously unseen data. This optimization intention makes the learned policy have a strong generalization ability. Therefore, the proposed approach can achieve encouraging performance in RUL estimation.

FIGURE 11 displays learning curves in one training process of DRLRULe on FD004. FIGURE 11(a) demonstrates the change of cumulative reward the agent obtains during training, and FIGURE 11(b) shows the estimation performance of the agent's policy on testing engines in every epoch. As can be seen from FIGURE 11, the cumulative reward gradually increases as the training progresses. It indicates the agent is optimizing its policy to maximize the cumulative reward. At the same time, the RMSE on testing engines decreases during training, i.e., the estimation performance is gradually improved. Therefore in DRLRULe, the agent keeps exploring to approach the optimal RUL estimation policy, which maximizes the cumulative reward and enhances the RUL estimation performance at the same time. This learning mechanism ensures the remarkable performance of DRLRULe in RUL estimation.

V. CONCLUSION

In this paper, deep reinforcement learning is introduced into prognostics for the first time. We propose a novel DRL based prognostic framework for the RUL estimation of engineered systems. In this framework, the conventional RUL estimation problem is first formulated into the RULeMDP model, which is suitable for DRL algorithms' application. Then the PPO algorithm is employed to learn an optimal RUL estimation policy in RULeMDP. The effectiveness and superiority of the proposed approach are validated through a case study on the benchmark C-MAPSS dataset. The experimental results indicate that our approach has superior RUL estimation performance to other state-of-the-art approaches.

This paper demonstrates the feasibility of applying DRL to prognostics, and presents a new approach for RUL estimation. In future work, we will employ other advanced DRL algorithms in the RULeMDP environment to further enhance the accuracy of RUL estimation. Besides, we will investigate designing a better reward function for RULeMDP, which can take both RMSE and score metrics into consideration. That reward function may contribute to improving the estimation performance on these two metrics at the same time. Furthermore, in addition to the CNN model in this paper, some other sophisticated models will be developed and used for π and V in future work, which may help the agent learn a better policy for RUL estimation.

APPENDIX A NOTATIONS

$\boldsymbol{b}$	Bias term in convolution output of CNN.
$\boldsymbol{f}$	Convolution filter in CNN.
$\boldsymbol{g}$	Pooling output in CNN.
$\boldsymbol{I}$	Input matrix in CNN.
$\boldsymbol{P}$	Pooling filter matrix in CNN.
$\boldsymbol{x} \in \mathbb{R}^F$	Multi-sensory data value.
$\boldsymbol{z}$	Convolution output in CNN.
$\mathbf{X}$	Monitoring sensor data sample.
$\mathbf{Y}$	Remaining useful life data.
$\mathcal{D}$	Trajectories: $\mathcal{D} = \{\tau_i\}$.
$\mathcal{A}$	Set of all valid actions.
$\mathcal{S}$	Set of all valid states.
$\mathcal{S}_e$	States of RULeMDP, randomly sampled from D.
$\mu(s)$	Mean action function with state s.
ϕ	Mean-squared error.
π	Control policy.
π^*	Optimal control policy.
π_θ	Control policy represented by neural networks with parameter θ.
ρ_0	Start-state distribution.
$\sigma(s)$	Standard deviation function with state s.
τ	Trajectory storing historical interactions.
A^π(s, a)	Advantage for taking action a at state s with policy π.
a_t	Action at time t.
D	Labeled training dataset.
E	Mathematical expectation.
$f(\cdot)$	Mapping function.
$J(\pi)$	Expected return function with policy π.
$L(\cdot)$	Loss function.
$R(s_t, a_t, s_{t+1})$	Reward value with state s_t, action a_t and next state s_{t+1}.
r_t	Reward at time t.
s_t	State at time t.
$V_\mu(s)$	Value function represented by neural networks with parameter μ, with state s.
$\mathbf{X}$	Monitoring sensor data.

APPENDIX B PPO ALGORITHM

```
Input: initial policy parameters $\theta_0$, initial value
    function parameters $\mu_0$
for $k = 0, 1, 2, \dots$ do
    Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by
      running policy $\pi_k = \pi(\theta_k)$ in the
      environment;
    Compute rewards-to-go $\hat{R}_t = \sum_{t'=t}^{T} r_{t'}$ for
      every state in a trajectory $\tau$, where $T$
      denotes the length of $\tau$;
    Compute advantage estimates, $\hat{A}_t$ (using
      Generalized Advantage Estimation (GAE)
      method [41]) based on the current value
      function $V_{\mu_k}$;
    Update the policy by maximizing the PPO
      objective:
      $$\theta_{k+1} = \arg\max_\theta \frac{1}{|\mathcal{D}_k|T} \sum_{\tau\in\mathcal{D}_k} \sum_{t=0}^{T} \min\left(vw, w * \text{clip}\left(v, 1-\epsilon, 1+\epsilon\right)\right)$$
      where $v = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)}$ and $w = A^{\pi_{\theta_k}}(s_t, a_t)$,
      via stochastic gradient ascent with Adam
      algorithm [43];
    Fit value function by regression on
      mean-squared error:
      $$\phi_{k+1} = \arg\min_\mu \frac{1}{|\mathcal{D}_k|T} \sum_{\tau\in\mathcal{D}_k} \sum_{t=0}^{T} \left(V_\mu(s_t) - \hat{R}_t\right)^2,$$
      via stochastic gradient descent with Adam
      algorithm;
end
```

APPENDIX C HYPER-PARAMETERS OF DRLRULe

	Parameter	Value
RULeMDP	Sampling ratio m/n	0.25
PPO	Training epochs	10000
	Discount factor γ	0.1
	Lambda λ of GAE	0.9
	Clip ratio ϵ	0.2
	Policy: optimization algorithm	Adam
	Policy: optimizer learning rate	$5 * 10^{-4}$
	Policy: gradient ascent steps per epoch	80
	Value function: optimization algorithm	Adam
	Value function: optimizer learning rate	10^{-3}
	Value function: gradient descent steps	80
	Adam: betas	(0.9, 0.999)
	Adam: eps	10^{-8}
	Adam: weight decay	0

REFERENCES

[1] M. J. Carr and W. B. Wang, "Modeling failure modes for residual life prediction using stochastic filtering theory," *IEEE Trans. Rel.*, vol. 59, no. 2, pp. 346–355, Jun. 2010.

[2] S. S. H. Zaidi, S. Aviyente, M. Salman, K.-K. Shin, and E. G. Strangas, "Prognosis of gear failures in DC starter motors using hidden Markov models," *IEEE Trans. Ind. Electron.*, vol. 58, no. 5, pp. 1695–1706, May 2011.

[3] J. Z. Sun, H. Zuo, W. Wang, and M. G. Pecht, "Prognostics uncertainty reduction by fusing on-line monitoring data based on a state-space-based degradation model," *Mech. Syst. Signal Process.*, vol. 45, no. 2, pp. 396–407, Apr. 2014.

[4] Y. Qian and R. Yan, "Remaining useful life prediction of rolling bearings using an enhanced particle filter," *IEEE Trans. Instrum. Meas.*, vol. 64, no. 10, pp. 2696–2707, Oct. 2015.

[5] Z.-X. Zhang, X.-S. Si, and C.-H. Hu, "An age- and state-dependent nonlinear prognostic model for degrading systems," *IEEE Trans. Rel.*, vol. 64, no. 4, pp. 1214–1228, Dec. 2015.

[6] Y. Lei, N. Li, S. Gontarz, J. Lin, S. Radkowski, and J. Dybala, "A model-based method for remaining useful life prediction of machinery," *IEEE Trans. Rel.*, vol. 65, no. 3, pp. 1314–1326, Sep. 2016.

[7] R. Huang, L. Xi, X. Li, C. R. Liu, H. Qiu, and J. Lee, "Residual life predictions for ball bearings based on self-organizing map and back propagation neural network methods," *Mech. Syst. Signal Process.*, vol. 21, no. 1, pp. 193–207, Jan. 2007.

[8] K. Javed, R. Gouriveau, and N. Zerhouni, "A new multivariate approach for prognostics based on extreme learning machine and fuzzy clustering," *IEEE Trans. Cybern.*, vol. 45, no. 12, pp. 2626–2639, Dec. 2015.

[9] K. Zhu and T. Liu, "Online tool wear monitoring via hidden semi-Markov model with dependent durations," *IEEE Trans. Ind. Informat.*, vol. 14, no. 1, pp. 69–78, Jan. 2018.

[10] T. H. Loutas, D. Roulias, and G. Georgoulas, "Remaining useful life estimation in rolling bearings utilizing data-driven probabilistic E-support vectors regression," *IEEE Trans. Rel.*, vol. 62, no. 4, pp. 821–832, Dec. 2013.

[11] G. S. Babu, P. Zhao, and X.-L. Li, "Deep convolutional neural network based regression approach for estimation of remaining useful life," in *Database Systems for Advanced Applications*, vol. 9642, S. B. Navathe, W. Wu, S. Shekhar, X. Du, X. S. Wang, and H. Xiong, Eds. Cham, Switzerland: Springer, 2016, pp. 214–228.

[12] C. Zhang, P. Lim, A. K. Qin, and K. C. Tan, "Multiobjective deep belief networks ensemble for remaining useful life estimation in prognostics," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 10, pp. 2306–2318, Oct. 2017.

[13] S. Zheng, K. Ristovski, A. Farahat, and C. Gupta, "Long short-term memory network for remaining useful life estimation," in *Proc. IEEE Int. Conf. Prognostics Health Manage. (ICPHM)*, Dallas, TX, USA, Jun. 2017, pp. 88–95.

[14] Y. Liao, L. Zhang, and C. Liu, "Uncertainty prediction of remaining useful life using long short-term memory network based on bootstrap method," in *Proc. IEEE Int. Conf. Prognostics Health Manage. (ICPHM)*, Jun. 2018, pp. 1–8.

[15] X. Li, Q. Ding, and J.-Q. Sun, "Remaining useful life estimation in prognostics using deep convolution neural networks," *Rel. Eng. Syst. Saf.*, vol. 172, pp. 1–11, Apr. 2018.

[16] J. Wang, G. Wen, S. Yang, and Y. Liu, "Remaining useful life estimation in prognostics using deep bidirectional LSTM neural network," in *Proc. PHM-Chongqing*, Oct. 2018, pp. 1037–1042.

[17] M. Xia, T. Li, T. Shu, J. Wan, C. W. de Silva, and Z. Wang, "A two-stage approach for the remaining useful life prediction of bearings using deep neural networks," *IEEE Trans. Ind. Informat.*, vol. 15, no. 6, pp. 3703–3711, Jun. 2019.

[18] C. G. Huang, H. Z. Huang, and Y. F. Li, "A bidirectional LSTM prognostics method under multiple operational conditions," *IEEE Trans. Ind. Electron.*, vol. 66, no. 11, pp. 8792–8802, Nov. 2019.

[19] J. Li, X. Li, and D. He, "A directed acyclic graph network combined with CNN and LSTM for remaining useful life prediction," *IEEE Access*, vol. 7, pp. 75464–75475, 2019.

[20] H. Liu, Z. Liu, W. Jia, and X. Lin, "Remaining useful life prediction using a novel feature-attention-based end-to-end approach," *IEEE Trans. Ind. Informat.*, vol. 17, no. 2, pp. 1197–1207, Feb. 2021.

[21] L. Ren, J. Cui, Y. Sun, and X. Cheng, "Multi-bearing remaining useful life collaborative prediction: A deep learning approach," *J. Manuf. Syst.*, vol. 43, pp. 248–256, Apr. 2017.

[22] L. Liao, W. Jin, and R. Pavel, "Enhanced restricted Boltzmann machine with prognosability regularization for prognostics and health assessment," *IEEE Trans. Ind. Electron.*, vol. 63, no. 11, pp. 7076–7083, Nov. 2016.

[23] S. Xiang, Y. Qin, J. Luo, H. Pu, and B. Tang, "Multicellular LSTM-based deep learning model for aero-engine remaining useful life prediction," *Rel. Eng. Syst. Saf.*, vol. 216, Dec. 2021, Art. no. 107927.

[24] J. Zhang, Y. Jiang, S. Wu, X. Li, H. Luo, and S. Yin, "Prediction of remaining useful life based on bidirectional gated recurrent unit with temporal self-attention mechanism," *Rel. Eng. Syst. Saf.*, vol. 221, May 2022, Art. no. 108297.

[25] A. Namdari, M. A. Samani, and T. S. Durrani, "Lithium-ion battery prognostics through reinforcement learning based on entropy measures," *Algorithms*, vol. 15, no. 11, p. 393, Oct. 2022.

[26] L.-H. Ren, Z.-F. Ye, and Y.-P. Zhao, "Long short-term memory neural network with scoring loss function for aero-engine remaining useful life estimation," *Proc. Inst. Mech. Eng., G, J. Aerosp. Eng.*, vol. 237, no. 3, pp. 547–560, Mar. 2023.

[27] J. Lee and M. Mitici, "Deep reinforcement learning for predictive aircraft maintenance using probabilistic remaining-useful-life prognostics," *Rel. Eng. Syst. Saf.*, vol. 230, Feb. 2023, Art. no. 108908.

[28] L. Liao and F. Köttig, "Review of hybrid prognostics approaches for remaining useful life prediction of engineered systems, and an application to battery life prediction," *IEEE Trans. Rel.*, vol. 63, no. 1, pp. 191–207, Mar. 2014.

[29] S. B. Thrun, "Efficient exploration in reinforcement learning," Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep. CMU-CS-92-102, 1992.

[30] A. Saxena, K. Goebel, D. Simon, and N. Eklund, "Damage propagation modeling for aircraft engine run-to-failure simulation," in *Proc. IEEE Int. Conf. Prognostics Health Manage.*, Denver, CO, USA, Oct. 2008, pp. 1–9.

[31] T. Hastie, R. Tibshirani, and J. H. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* (Springer Series in Statistics), 2nd ed. New York, NY, USA: Springer, 2009.

[32] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction* (Adaptive Computation and Machine Learning Series), 2nd ed. Cambridge, MA, USA: MIT Press, 2018.

[33] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *J. Artif. Intell. Res.*, vol. 4, no. 1, pp. 237–285, Jan. 1996.

[34] Z. Peng, X. Huang, D. Tang, and Q. Quan, "Health indicator construction based on multisensors for intelligent remaining useful life prediction: A reinforcement learning approach," *IEEE Trans. Instrum. Meas.*, vol. 72, pp. 1–13, 2023.

[35] V. Mnih, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[36] H. Shi, L. Shi, M. Xu, and K.-S. Hwang, "End-to-end navigation strategy with deep reinforcement learning for mobile robots," *IEEE Trans. Ind. Informat.*, vol. 16, no. 4, pp. 2393–2402, Apr. 2020.

[37] J. Yang, M. Xi, J. Wen, Y. Li, and H. H. Song, "A digital twins enabled underwater intelligent internet vehicle path planning system via reinforcement learning and edge computing," *Digit. Commun. Netw.*, to be published, doi: 10.1016/j.dcan.2022.05.005.

[38] J. Yang, J. Huo, M. Xi, J. He, Z. Li, and H. H. Song, "A time-saving path planning scheme for autonomous underwater vehicles with complex underwater conditions," *IEEE Internet Things J.*, vol. 10, no. 2, pp. 1001–1013, Jan. 2023.

[39] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.

[40] L. Peel, "Data driven prognostics using a Kalman filter ensemble of neural network models," in *Proc. IEEE Int. Conf. Prognostics Health Manage.*, Denver, CO, USA, Oct. 2008, pp. 1–6.

[41] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," 2015, *arXiv:1506.02438*.

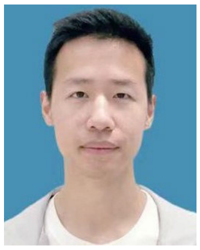
[42] F. O. Heimes, "Recurrent neural networks for remaining useful life estimation," in *Proc. IEEE Int. Conf. Prognostics Health Manage.*, Denver, CO, USA, Oct. 2008, pp. 1–6.

[43] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

QIANKUN HU received the B.E. degree in power engineering of aircraft from the Nanjing University of Aeronautics and Astronautics, in 2018, where he is currently pursuing the Ph.D. degree in aerospace propulsion theory and engineering. His research interests include reinforcement learning and aero-engine fault diagnosis.

YONGPING ZHAO received the B.E. degree in thermal energy and power engineering and the M.S. and Ph.D. degrees from the Nanjing University of Aeronautics and Astronautics, Nanjing, China, in July 2004 and December 2009. He is currently a Professor with the College of Energy and Power Engineering, Nanjing University of Aeronautics and Astronautics. His research interests include aircraft engine modeling, control and fault diagnostics, machine learning, and pattern recognition. He was nominated for the National Excellent Doctoral Dissertation Award of China, in 2013, for the Ph.D. degree.

YUQIANG WANG received the B.E. degree in power engineering of aircraft from the Nanjing University of Aeronautics and Astronautics, in 2020, where he is currently pursuing the Ph.D. degree in aerospace propulsion theory and engineering. His research interests include deep learning, aero-engine control, and fault diagnosis.



PEI PENG received the B.E. degree in power engineering of aircraft from the Nanjing University of Aeronautics and Astronautics, in 2020, where he is currently pursuing the M.S. degree in aerospace propulsion theory and engineering. His research interests include machine learning, aero-engine modeling, and fault diagnosis.

LIHUA REN received the B.E. and Ph.D. degrees from the Nanjing University of Aeronautics and Astronautics. He is currently pursuing the Ph.D. degree. His research interests include machine learning, deep learning, and aero-engine prognostic and health management.