Reliability Engineering and System Safety 259 (2025) 110919

Contents lists available at ScienceDirect

Reliability Engineering and System Safety

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

Deep learning-stochastic ensemble for RUL prediction and predictive maintenance with dynamic mission abort policies

Faizanbasha A. [a,b,*], U. Rizwan [a]

[a] Department of Mathematics, Islamiah College, New Town, Vaniyambadi, 635752, Tamil Nadu, India
[b] Thiruvalluvar University, Serkkadu, Vellore, 632115, Tamil Nadu, India

ARTICLE INFO

Keywords:
Predictive maintenance
Prognostics and health management
Smooth semi-martingale
Degradation modeling
Remaining useful life

ABSTRACT

Accurate prediction of Remaining Useful Life (RUL) is crucial for optimizing maintenance strategies in industrial systems. However, existing models often falter under nonlinear and nonstationary degradation conditions with stochastic and abrupt failures, limiting their real-world effectiveness. To address this, we introduce a novel approach that combines advanced deep learning architectures with stochastic modeling and dynamic optimization techniques for more precise RUL prediction. This study has three overarching aims: First, to propose a hybrid ensemble model integrating convolutional neural networks, transformers, long short-term memory networks, and a smooth semi-martingale stochastic layer, a combination not previously explored, to effectively model both deterministic and stochastic degradation processes, thereby enhancing RUL prediction accuracy. Second, to introduce a reinforcement learning-based hyperparameter tuning method that dynamically adjusts model parameters, improving performance and reducing training time, which in turn optimizes the ensemble model's predictive capabilities. Third, to integrate the refined RUL predictions and time-varying thresholds into a multi-stage optimization framework for mission cycle assignment and resource management. This facilitates real-time decision-making and the development of a dynamic mission abort policy, including mission shifting, re-engagement, post-abortion analysis, mission plan adjustments, and maintenance scheduling. Together, these innovations enhance RUL prediction accuracy, model adaptability, and operational efficiency, ensuring reliable and cost-effective maintenance strategies in mission-critical systems. The proposed model, validated using NASA's C-MAPSS dataset, demonstrated superior RUL prediction accuracy over state-of-the-art methods, with sensitivity analyses and ablation studies confirming its stability and effectiveness.

1. Introduction

Prognostics and Health Management (PHM) plays a pivotal role in predicting the future condition of industrial equipment. It enables timely Predictive Maintenance (PdM) decisions to reduce downtime and improve reliability [1]. Among the various metrics in PHM, Remaining Useful Life (RUL) estimation provides a clear and quantifiable measure of the time left before system failure. This makes it essential for industrial applications [2]. In today's globally competitive industrial sectors, organizations are under constant pressure to maximize asset availability, reduce operational risks, and minimize costs [3]. At the same time, maintaining safety and regulatory standards are also equally important. Accurate RUL predictions serve as key indicators for scheduling maintenance activities, optimizing resources, and minimizing unplanned failures. This, in turn, improves operational reliability and safety [4,5]. Advances in deep convolutional neural networks and LSTMs have shown great potential in enhancing the accuracy of data-driven RUL predictions. This is especially true for complex systems where expert models are not yet available [6].

While extensive research has focused on improving the accuracy of data-driven RUL predictions using deep learning methods such as CNNs and LSTMs, several key gaps still remain [7]. Firstly, existing models often fail to account for the stochastic nature of machine degradation, especially under nonlinear and nonstationary operational conditions found in complex industrial systems. Secondly, they struggle to handle abrupt failures and unpredictable shifts in degradation patterns, which are common in real-world applications. Thirdly, there is a lack of integration between advanced deep learning architectures and stochastic modeling techniques for RUL prediction. Lastly, current PdM strategies rarely incorporate dynamic mission abort policies necessary for real-time decision-making in mission-critical operations.

* Corresponding author at: Department of Mathematics, Islamiah College, New Town, Vaniyambadi, 635752, Tamil Nadu, India.
E-mail address: faizan_phdmaths@islamiahcollege.edu.in (Faizanbasha A.).

https://doi.org/10.1016/j.ress.2025.110919
Received 5 October 2024; Received in revised form 12 February 2025; Accepted 14 February 2025
Available online 25 February 2025

Nomenclature

Acronyms & Abbreviations

RUL	Remaining Useful Life
PdM	Predictive Maintenance
CNN	Convolutional Neural Network
LSTM	Long Short-Term Memory network
SSM	Smooth Semi-Martingale
RLHT	Reinforcement Learning-Based Hyperparameter Tuning
MDP	Markov Decision Process
RMSE	Root Mean Squared Error
ReLU	Rectified Linear Unit
TCN	Temporal Convolutional Network
C-MAPSS	Commercial Modular Aero-Propulsion System Simulation
PHM	Prognostics and Health Management
DQN	Deep Q-Network
AGCNN	Attention-Gated Convolutional Neural Network
Bi-LSTM	Bidirectional Long Short-Term Memory network
BiGRU	Bidirectional Gated Recurrent Unit
S-score	Scoring function for RUL prediction
RNN	Recurrent Neural Network
GRU	Gated Recurrent Unit

Notations

$\mathbf{X}$	Raw sensor data
$\mathbf{X}_{\text{cnn}}$	Output of CNN layer
f_{CNN}	CNN function
$\mathbf{F}_{i,j}$	Feature map in CNN
$\mathbf{W}$	Filter weights in CNN
b	Bias term
$\mathbf{A}_{i,j}$	Activation map after ReLU
$\mathbf{X}_{\text{trans}}$	Output of Transformer layer
f_{trans}	Transformer function
$\mathbf{Q}, \mathbf{K}, \mathbf{V}$	Query, Key, Value matrices in Transformer
d_k	Dimensionality of keys and queries
Attention$(\mathbf{Q}, \mathbf{K}, \mathbf{V})$	Attention function
$\mathbf{X}_{\text{lstm}}$	Output of LSTM
f_{LSTM}	LSTM function
$\mathbf{b}_i, \mathbf{b}_f, \mathbf{b}_o, \mathbf{b}_c$	Bias terms in LSTM
$\odot$	Element-wise multiplication
$\mathcal{Z}_t$	State of the system at time t in SSM model
$g(s, \mathcal{Z}_s)$	Nonlinear degradation rate function
$\sigma(s, \mathcal{Z}_s)$	Volatility function
W_s	Standard Brownian motion
$\gamma(s, \mathcal{Z}_{s^-}, z)$	Jump amplitude function
$\tilde{N}(ds, dz)$	Compensated Poisson random measure
M_t	Martingale component
θ	Hyperparameters of the model
$L(\theta)$	Loss function
S_t	State in MDP
A_t	Action in MDP
R_t	Reward function
$P(S_{t+1}\|S_t, A_t)$	Transition probability in MDP
$\pi(S_t)$	Policy in MDP
ρ	Discount factor
$V^*(S_t)$	Optimal value function
$Q(S_t, A_t)$	Action-value function
α	Learning rate
y_i	Target value in DQN
$\mathcal{L}(\theta)$	Loss function in DQN
ζ	Failure threshold
RUL_t	Remaining useful life at time t
τ_t	Dynamic threshold in mission abort policy
δ	Decay factor
ΔRUL_t	Rate of change of RUL
a_t	Alert threshold
A_j	Current aircraft
A_k	Potential replacement aircraft
C_{shift}	Cost of shifting the mission
$C_{\text{cont}}(t)$	Cost of continuing the mission
$\mathcal{H}_t$	Health state of an engine at time t
$r(s)$	Recovery rate
ζ_{re}	Threshold for re-engagement
$C_{\text{re}}(t)$	Re-engagement cost function
C_{maint}	Maintenance cost
$C_{\text{fail}}(t)$	Expected failure cost
t_{opt}	Optimal re-engagement time
$C_{\text{missed}}(t)$	Cost of lost mission opportunities
$U(A_k)$	Utility function for aircraft A_k
$P_{\text{fail}}(A_k, t)$	Probability of failure
δ_{spare}	Threshold for shifting to spare role
$B_{\text{shift}}(t)$	Expected benefit of re-shifting
$S_{\text{re}}(t)$	Success probability if re-shifted
$S_{\text{cont}}(t)$	Success probability if continued
δ_{shift}	Threshold for re-shifting decision
$C_{\text{total}}(t)$	Total expected cost
μ	Mean of a distribution (e.g., in data normalization)
σ_{SD}	Standard deviation of a distribution
ϵ	Risk tolerance
T	Time horizon
Error_i	Prediction error for sample i

To address these gaps, this study proposes a CNN-Transformer-LSTM-SSM ensemble framework designed to capture both deterministic and stochastic degradation processes. Deterministic degradation refers to predictable, gradual wear over time, while stochastic degradation involves random, unpredictable failures or sudden shifts in machinery condition [8]. By effectively modeling both types of degradation, our framework leads to more precise RUL predictions and mission-critical decision-making. The proposed stochastic layer within the ensemble model is based on the Smooth Semi-Martingale (SSM) framework. It captures both gradual wear and sudden degradation shifts [9]. This allows the model to handle unpredictable jumps and falls in degradation processes that traditional and recent deep learning methods often overlook. This approach stabilizes the model performance and results in reduced error during RUL predictions. It enables for more reliable PdM scheduling and reduces the risk of unexpected equipment failures.

A smooth semi-martingale is a special type of stochastic process that effectively models systems exhibiting both predictable trends and random fluctuations [9]. It combines deterministic elements, such as gradual wear, with stochastic components like sudden shocks or failures [8,10]. In the context of our study, machinery degradation is complex and involves both steady wear and abrupt failures due to unforeseen factors. By including an SSM layer in our deep learning ensemble model, we enhance its ability to accurately reflect the complex nature of equipment degradation. This integration improves the model's performance in handling nonlinear and nonstationary degradation patterns, leading to more precise RUL predictions and better maintenance decisions.

1.1. Background and related works

Recent advancements have integrated the data-driven probabilistic RUL prognostics into the maintenance scheduling. This has demonstrated significant reductions in unscheduled maintenance events and

overall maintenance costs for aircraft systems. It shows the potential of deep learning and Monte Carlo-based approaches to improve predictive accuracy and decision-making [11]. The integration of AI-driven PHM systems further enhances predictive capabilities. By utilizing a combined approach of "model-driven" and "data-driven" methods, these systems can detect anomalies and predict RUL in complex engineering systems. This shows great promise in reducing unexpected failures and optimizing maintenance strategies [12]. Additionally, integrating multiple data sources within Industrial IoT (IIoT)-enabled systems can further refine the RUL predictions, improving accuracy and providing more informed decision-making in industrial applications [13].

Dynamic PdM approaches that use probabilistic deep learning models has further improved the accuracy and efficiency of RUL prognostics [14,15]. This is particularly true for multi-component systems, allowing for more precise maintenance scheduling under uncertainty [16]. Recent advancements in "deep learning" methods, such as attention-based "temporal convolutional networks," have significantly improved the accuracy of RUL prediction by efficiently handling large-scale industrial IoT data [17]. Furthermore, the prediction of RUL has advanced through the development of deep learning techniques like the Multicellular LSTM (MGLSTM). This model effectively processes multi-source data for enhanced prediction accuracy in aerospace applications. It has demonstrated superior performance compared to traditional machine learning techniques [18].

In recent years, there has been significant advancement in PdM and RUL prediction using deep learning techniques. For instance, "bidirectional gated recurrent units" combined with "temporal self-attention" mechanisms have been proposed to improve the prediction accuracy of RUL in various industrial systems [19,20]. Approaches incorporating "multi-head dual sparse self-attention" networks [21], spatial–temporal attention-based LSTMs [22], "dual-aspect self-attention" based on Transformers [23], and Temporal Convolutional Networks (TCN) with attention mechanisms have shown improved prognostic performance. This is particularly evident in machinery systems [24,25]. Transformer based architectures have also been utilized by integrating trend augmentation and temporal features with multi-sensor data for RUL prediction [26]. "Bayesian gated-Transformer" models have been developed for "risk-aware" prediction of aero-engine RUL [27]. Additionally, aircraft engine RUL estimation via "double attention-based data-driven architectures" has been explored recently [28]. Moreover, improved Transformer models using feature fusion gates and predictive vector angle minimization have been proposed for aircraft engine RUL prediction, achieving higher accuracy and advanced prediction capabilities [29].

Other studies have focused on dynamic predictive maintenance scheduling using deep learning ensembles [30–32]. Additionally, data-driven approaches combined with sensor-based monitoring have been explored [33]. Hybrid CNN-LSTM models have also been developed for optimizing maintenance and production processes [34]. Recently, advanced techniques such as Bayesian-optimized LSTM for lithium-ion battery health prognostics [35] and soft-thresholding attention-based TCN for machinery prognostics [24] have been extensively studied. Further, multi-objective predictive maintenance scheduling models that integrate RUL estimations into maintenance decisions have also demonstrated significant improvements in maintenance completion time and cost reductions [36]. Lightweight models, such as dual attention LSTMs based on exponential smoothing, have also been proposed for RUL prediction [37]. Dual-task TCN models with multi-channel attention have also been developed to improve RUL predictions and identify the first failure time in complex systems [25]. These developments show the growing importance of integrating deep learning into prognostics to enhance PdM strategies.

Moreover, recent studies have optimized maintenance processes by integrating burn-in, predictive maintenance, and smooth semi-martingale models, effectively reducing downtime and costs while enhancing reliability in manufacturing systems [10]. This demonstrates the effectiveness of combining stochastic processes with PdM strategies, supporting our ensemble framework that incorporates an SSM layer for both deterministic and stochastic degradation.

Based on our review of the literature, we hypothesize that integrating CNNs, Transformers, LSTMs, and an SSM stochastic layer into an ensemble model can effectively capture both deterministic and stochastic degradation processes. This integrated approach aims to address the limitations of existing methods by handling unpredictable shifts in degradation patterns and accounting for the inherent randomness in machine degradation. By doing so, we expect to enhance the accuracy and stability of RUL predictions, leading to more reliable predictive maintenance strategies in industrial applications.

Mission abort policies are critical in preventing catastrophic failures and ensuring mission success, especially in safety-critical operations with uncertain conditions. Furthermore, recent advancements have significantly enhanced the development of dynamic mission abort policies to improve system reliability and safety. Optimal two-stage abort policies have been proposed to balance mission progress with risk mitigation in performance-based missions [38]. Strategies addressing multi-component system failures with interactive failure modes have been developed to enhance system survivability [39]. Joint modeling approaches that integrate loading and mission abort decisions in dynamic environments have also been explored to optimize system performance under varying conditions [40]. Additionally, policies optimizing component activation and mission aborting in multi-attempt missions have demonstrated reductions in expected mission losses [41]. Time-varying and self-adaptive mission aborting policies have been introduced to handle resource constraints and uncertain shock environments effectively [42,43]. Reliability modeling for balanced systems considering mission abort policies has further advanced the understanding of mission success probabilities and system survivability [44]. Moreover, joint optimization of mission abort strategies with system structures has been investigated to address dynamic tasks and enhance mission reliability [45]. Additionally, optimal condition-based abort decisions have been formulated to minimize expected total costs by integrating degradation processes with abort policies [46]. These studies collectively contribute to the robust framework for mission abort policies, ensuring enhanced safety and reliability in diverse operational scenarios.

1.2. Research gap and motivation

Despite the growing adoption of deep learning models in PdM, accurately predicting the RUL under real-world conditions remains a significant challenge [16]. Models like CNNs and LSTM networks capture the spatial and temporal patterns. However, they often fail to account for the stochastic and nonstationary nature of machine degradation, leading to less reliable RUL predictions [47]. Additionally, traditional hyperparameter tuning methods, such as grid search, Bayesian optimization, or manual adjustment, are inefficient and may not yield optimal model performance. There is also a gap in connecting RUL predictions with actionable maintenance strategies, such as dynamic mission abort policies that can make real-time operational decisions based on the predicted RUL.

According to a systematic literature review, existing predictive maintenance models often lack integration of stochastic processes and dynamic mission abort policies, limiting their ability to accurately predict RUL under nonlinear and nonstationary degradation. Nonlinear degradation refers to the irregular, unpredictable wear of machinery components influenced by factors like load, temperature, or operational stress. Nonstationary degradation implies that the statistical properties of the degradation process changes over the time, challenging models that assume consistent wear patterns. In real-world scenarios, such degradation can lead to sudden, unforeseen failures, causing critical downtime and safety risks. These complex degradation behaviors necessitate advanced methodologies capable of adapting to changing

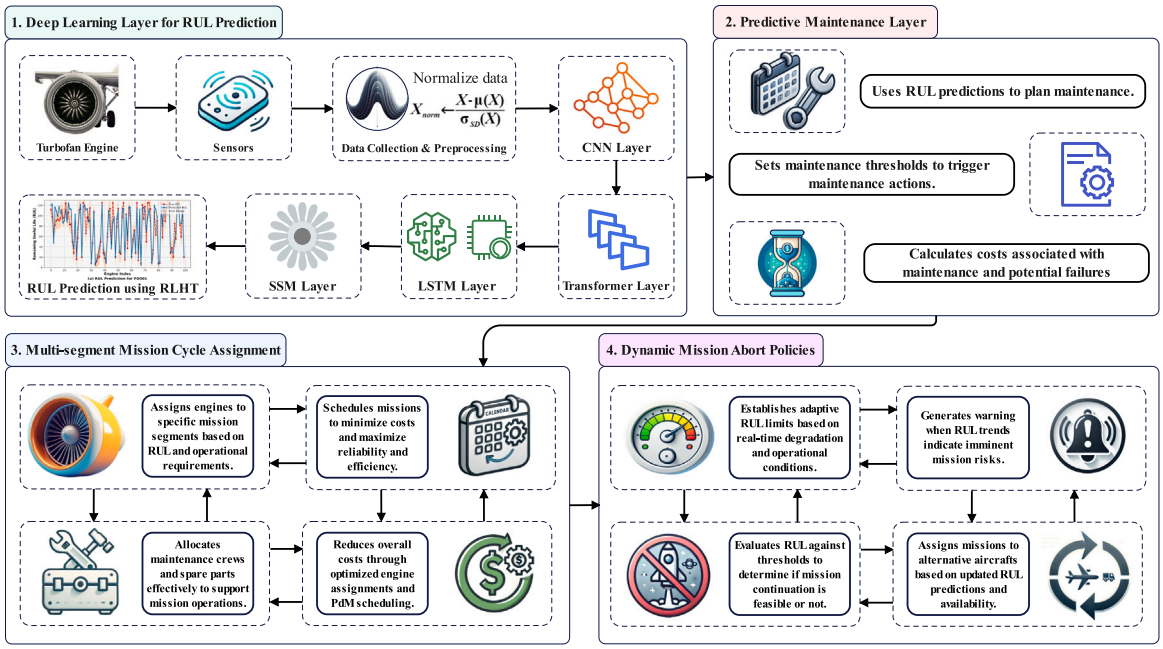


Fig. 1. The proposed framework for RUL prediction, pdm, mission cycle assignment, and dynamic mission abort policies.

conditions. Further, the existing methods often lack reliable uncertainty quantification, a critical aspect in RUL prediction. For instance, a Gaussian process-based framework with active learning has been proposed to address this, but computational inefficiencies still remain a challenge [48].

To address these challenges, we propose a CNN-Transformer-LSTM-SSM ensemble framework that integrates a stochastic process layer based on the SSM model. This addition enables the model to capture both gradual wear and sudden failures, enhancing performance across varying operational conditions. The SSM layer is particularly effective in capturing unpredictable jumps and random shifts in degradation patterns, the aspects that traditional deep learning models often overlook. Additionally, we propose a new Reinforcement Learning-Based Hyperparameter Tuning (RLHT) method to dynamically adjust key model parameters to further enhance performance and reduce the training time. The proposed approach was validated on the "NASA C-MAPSS" dataset [49]. The results demonstrate better performance in RUL prediction when compared to the recent and traditional "state-of-the-art" methods. Additionally, based on the predicted RUL, we propose a novel mission abort policy that uses dynamic, time-dependent thresholds, facilitating real-time decision-making during mission-critical operations. This policy includes strategies for mission shifting, re-engagement, and post-abortion analysis to ensure operational continuity. To the best of our knowledge, this study is the first to propose such a hybrid framework that introduces an SSM layer within deep learning models. This addition captures both deterministic and random degradation processes, enhancing prediction stability and accuracy. Further, the relationships between the core components of our proposed framework are illustrated in Fig. 1. The framework integrates RUL prediction, PdM, mission cycle assignment, and dynamic mission abort policies. Specifically, RUL predictions from the deep learning layer inform predictive maintenance decisions, optimize mission assignments to improve efficiency, and enable real-time mission abort decisions based on operational conditions and system health.

1.3. Main contributions

The main contributions of this study are:

1. *Proposing a Novel Hybrid Ensemble Model*. We introduce a hybrid CNN-Transformer-LSTM-SSM ensemble model that combines CNNs for spatial feature extraction, Transformers for capturing "long-range temporal dependencies," LSTM networks for sequential modeling, and an SSM stochastic layer for handling stochastic degradation processes. This enhances the accuracy of RUL predictions in complex, nonlinear industrial systems.
2. *Developing an SSM-Based Stochastic Layer*. We develop a stochastic layer using the SSM model to capture both continuous wear and abrupt failure dynamics, addressing limitations of previous deep learning models and improving predictive accuracy under varying operational conditions.
3. *Introducing Reinforcement Learning-Based Hyperparameter Tuning*. We propose a reinforcement learning-based hyperparameter tuning method that dynamically adjusts key parameters during training, enhancing model accuracy and reducing training time compared to traditional methods.
4. *Optimizing Mission Cycles Assignment and Resource Management*. We integrate RUL predictions into a multi-stage optimization framework, enabling dynamic engine allocation, mission plan adjustments, and maintenance scheduling to minimize costs and mitigate failure risks under complex operational constraints.
5. *Implementing a Dynamic Mission Abort Policy*. We introduce a "dynamic mission abort policy" based on time-varying thresholds to allow real-time mission adjustments. We also develop strategies for mission shifting, re-engagement, and post-abortion analysis to effectively manage mission-critical operations.

The remainder of this paper is organized as follows. Section 2 outlines the methodology of the proposed CNN-Transformer-LSTM-SSM ensemble framework. In Section 3, we present the framework for predictive maintenance scheduling and mission cycle optimization based on the proposed ensemble model. Section 4 introduces a dynamic mission abortion policy informed by the predicted RUL, while Section 5 presents an analysis of post-mission abortion and aircraft shifting, focusing on maintenance and optimization considerations. Section 6 provides the experimental analysis, showcasing validation results using the NASA C-MAPSS dataset. Finally, Section 7 concludes the paper with a summary of our findings and suggestions for future research.

2. Methodology

In this section, we explain the methodologies that integrates CNNs, Transformers, LSTM networks, and the smooth semi-martingale model to predict failures and estimate RUL in industrial systems. This approach uses the strengths of deep learning and stochastic modeling to process complex sensor data and predicts machinery degradation dynamics.

2.1. Convolutional neural networks for spatial feature extraction

In the proposed framework, Convolutional Neural Networks (CNNs) are employed to extract informative spatial patterns from the preprocessed sensor data. Initially, the raw sensor signals $\mathbf{X}$ are normalized and segmented into fixed-size windows, $\mathbf{X}_{\text{segment}}$, ensuring that each segment captures localized operational states of the machinery.

The CNN processes these segments through stacked convolutional layers, each applying a set of learnable filters to identify spatial correlations associated with machine health. For each filter $\mathbf{W}$, a feature map $\mathbf{F}$ is generated:

$$\mathbf{F}_{i,j} = \sum_{m=1}^{M}\sum_{n=1}^{N} \mathbf{X}_{i+m-1,j+n-1} \cdot \mathbf{W}_{m,n} + b, \tag{1}$$

where M and N are the dimensions of the filter and b is the bias term. Further, the wear and potential failures of the machine are captured by these feature maps. Nonlinear activation functions (e.g., ReLU) are then applied to enhance the model's capacity to capture complex, nonlinear degradation patterns:

$$\mathbf{A}_{i,j} = \max(0, \mathbf{F}_{i,j}). \tag{2}$$

Subsequent pooling operations reduce dimensionality and focus on the most critical features, effectively condensing large volumes of sensor data into a compact, high-level representation. The resulting output, $\mathbf{X}_{\text{cnn}}$, provides spatially refined features that serve as a robust foundation for downstream temporal modeling stages. By preserving key spatial characteristics of the machinery's health state, the CNN outputs facilitate more accurate RUL estimation and inform the PdM decisions that follow.

2.2. Temporal feature modeling

Following the spatial features extraction by the CNN model, the data is now fed for temporal feature modeling. For this, we use two layers, namely the Transformer layer and the LSTM layer. These layers keeps the most important features from the input time-series data.

1. **Transformers**: The Transformer layer focuses on identifying "temporal features" and "long-term dependencies" via a self-attention mechanism. Unlike approaches reliant on positional information, the vanilla Transformer effectively highlights key time steps and relevant patterns in machinery degradation data. The output of the CNN, $\mathbf{X}_{\text{cnn}}$, serves as input to the Transformer:

   $$\mathbf{X}_{\text{trans}} = f_{\text{trans}}(\mathbf{X}_{\text{cnn}}),$$

   where $\mathbf{X}_{\text{trans}}$ is the Transformer output and f_{trans} denotes the Transformer function. The core of the Transformer's operation is the attention mechanism:

   $$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V}, \tag{3}$$

   where $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ represent the query, key, and value matrices, respectively, and d_k is the dimensionality of the queries and keys. The Fig. 2 details the components of the Transformer architecture.

2. **Long Short-Term Memory (LSTM)**: The LSTM layer processes the Transformer's output to capture longer sequential dependencies. The standard LSTM model is employed due to its reliability, lower computational demands, and broad applicability in sequence tasks. By incorporating LSTMs, both recent and past information is maintained, improving RUL predictions:

   $$\mathbf{X}_{\text{lstm}} = f_{\text{LSTM}}(\mathbf{X}_{\text{trans}}),$$

   where $\mathbf{X}_{\text{lstm}}$ is the LSTM output and f_{LSTM} is the LSTM function. The LSTM's internal operations are defined as:

   $$\mathbf{i}_t = \sigma(\mathbf{W}_{xi}\mathbf{x}_t + \mathbf{W}_{hi}\mathbf{h}_{t-1} + \mathbf{b}_i)$$
   $$\mathbf{f}_t = \sigma(\mathbf{W}_{xf}\mathbf{x}_t + \mathbf{W}_{hf}\mathbf{h}_{t-1} + \mathbf{b}_f)$$
   $$\mathbf{o}_t = \sigma(\mathbf{W}_{xo}\mathbf{x}_t + \mathbf{W}_{ho}\mathbf{h}_{t-1} + \mathbf{b}_o)$$
   $$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tanh(\mathbf{W}_{xc}\mathbf{x}_t + \mathbf{W}_{hc}\mathbf{h}_{t-1} + \mathbf{b}_c)$$
   $$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t),$$

   where $\mathbf{i}_t$, $\mathbf{f}_t$, $\mathbf{o}_t$ are the input, forget, and output gates, $\mathbf{c}_t$ is the cell state, and $\mathbf{h}_t$ is the hidden state at time t. Weight matrices $\mathbf{W}_{xi}, \mathbf{W}_{xf}, \mathbf{W}_{xo}, \mathbf{W}_{xc}$ connect inputs to gates/cell states, and $\mathbf{W}_{hi}, \mathbf{W}_{hf}, \mathbf{W}_{ho}, \mathbf{W}_{hc}$ connect previous hidden states. Biases $\mathbf{b}_i, \mathbf{b}_f, \mathbf{b}_o, \mathbf{b}_c$ adjust these connections. The sigmoid function (σ) controls the gates, tanh updates the cell and hidden states, and the Hadamard product ($\odot$) performs element-wise multiplication. Fig. 3 illustrates the two-layer LSTM architecture, each with 128 units and a dropout rate of 0.2.

While both Transformers and LSTM networks effectively process time series data, integrating them utilizes their complementary strengths in capturing temporal dependencies. Transformers excel at modeling long-range dependencies through self-attention mechanisms, effectively identifying global temporal patterns regardless of their position in the sequence. However, they may not capture local temporal dynamics as precisely as recurrent networks. Conversely, LSTMs are specifically designed to handle sequential data and excel at capturing local patterns through their gated structure, which controls the flow of information over time. By combining Transformers and LSTMs, our model effectively captures both global and local temporal dependencies in the time-series data, enhancing its ability to represent complex degradation processes and leading to more accurate RUL predictions.

2.3. Smooth semi-martingale stochastic modeling

In this section, we present the mathematical framework for the SSM model. This framework is designed to capture the changing behavior of nonlinear and nonstationary degradation processes in industrial machinery. It enables precise and reliable predictions of RUL. As illustrated in Fig. 4, the model captures varying operational conditions and stochastic behaviors. Nonlinear trends, nonstationary variance, and their combination (Fig. 4(a)–(c)) reflect complex real-world degradation processes.

The system state at time t, denoted by $\mathcal{Z}_t$, is modeled as an SSM:

$$\mathcal{Z}_t = \mathcal{Z}_0 + \int_0^t g(s, \mathcal{Z}_s)\, ds + \int_0^t \sigma(s, \mathcal{Z}_s)\, dW_s + \int_0^t \int_{\mathbb{R}} \gamma(s, \mathcal{Z}_{s^-}, z)\, \tilde{N}(ds, dz), \tag{4}$$

where $\mathcal{Z}_0$ is the initial state of the system, $g(s, \mathcal{Z}_s)$ represents a nonlinear degradation rate function, $\sigma(s, \mathcal{Z}_s)$ is the volatility function, modulating the intensity of the stochastic component represented by standard Brownian motion W_s, and $\gamma(s, \mathcal{Z}_{s^-}, z)$ denotes the jump amplitude function, with $\tilde{N}(ds, dz)$ being the "compensated Poisson random measure" for jump events. Further, the martingale component M_t is decomposed into continuous and jump parts:

$$M_t = \int_0^t \sigma(s, \mathcal{Z}_s)\, dW_s + \int_0^t \int_{\mathbb{R}} \gamma(s, \mathcal{Z}_{s^-}, z)\, \tilde{N}(ds, dz).$$

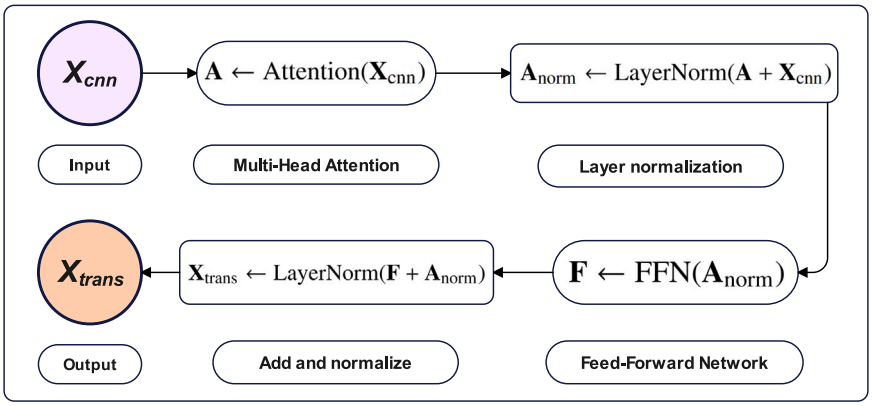


Fig. 2. Transformer architecture.

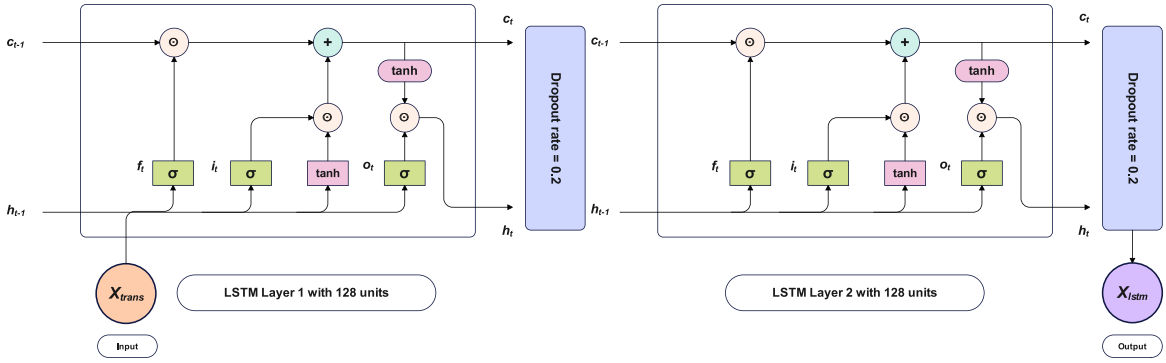


Fig. 3. LSTM structure.

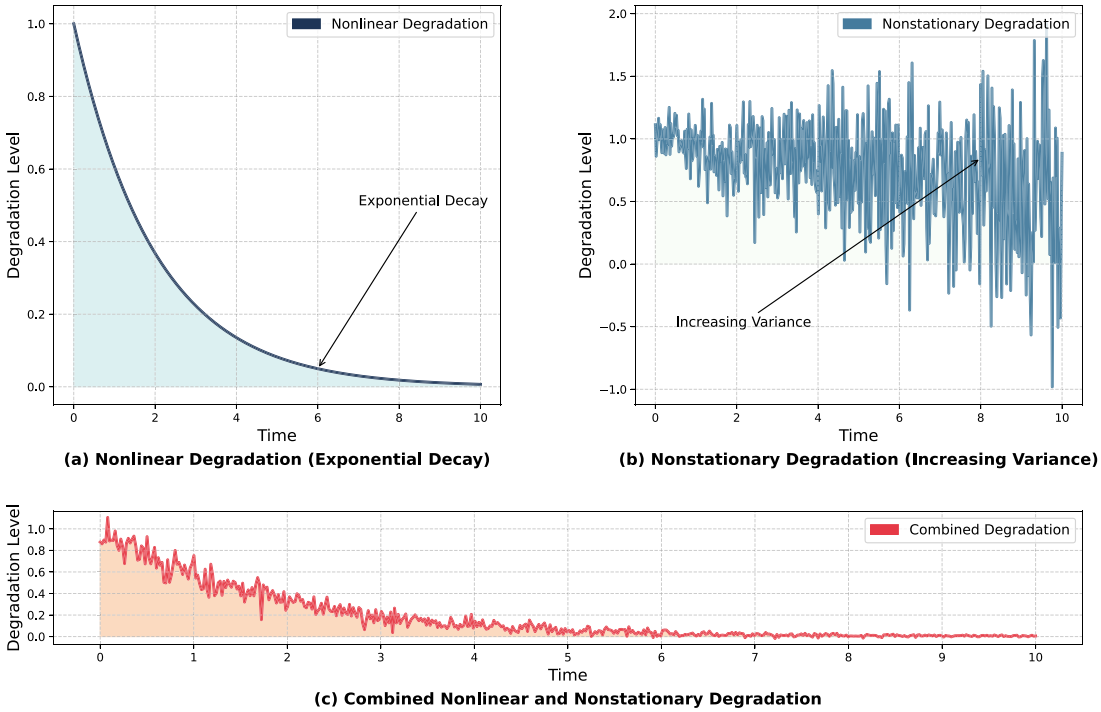


Fig. 4. Nonlinear and Nonstationary degradation processes.

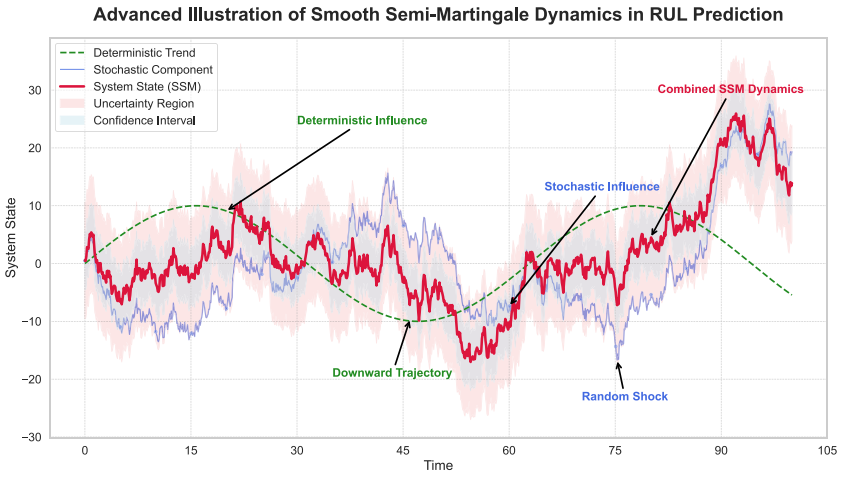


Fig. 5. Dynamic interaction of deterministic and stochastic influences in system state evolution.

The degradation rate function $g(s, \mathcal{Z}_s) = \mathfrak{a}(s) + \mathfrak{b}(s) \cdot f(\mathcal{Z}_s)$ captures environmental and operational effects by employing time-dependent coefficients $\mathfrak{a}(s)$ and $\mathfrak{b}(s)$, along with a nonlinear function $f(\mathcal{Z}_s)$ tailored to machine characteristics. Concurrently, the stochastic component is modeled as a diffusion term driven by standard Brownian motion W_s, with volatility $\sigma(s, \mathcal{Z}_s) = \kappa(s) \cdot \exp(\lambda(s)\mathcal{Z}_s)$. Here, $\kappa(s)$ and $\lambda(s)$ determine the scale and sensitivity of the volatility to the system state, reflecting the unpredictable nature of the degradation. Jumps are modeled by $\gamma(s, \mathcal{Z}_{s^-}, z) = \nu(s, \mathcal{Z}_{s^-}) \cdot h(z)$, linking jump impacts to pre-jump conditions via $\nu(\cdot)$ and modeling jump size distribution through $h(z)$.

The Fig. 5 illustrates how deterministic (green dashed line) and stochastic (blue line) components combine in the SSM model. The deterministic part tracks baseline degradation, while the stochastic component captures variability and abrupt disruptions, such as the random shock observed at $t \approx 75$. Their integration (red line) reflects real-world complexity, including times when the system state drops below acceptable thresholds (e.g., between $t = 45$ and $t = 90$), indicating elevated failure risks. The depicted confidence intervals and uncertainty regions quantify the model's predictive uncertainty, with increasing variability as the system experiences higher stochastic volatility, particularly after $t = 90$. These findings show the SSM model's capability to accurately capture nonlinear, nonstationary, and random influences, which are critical for precise RUL prediction.

Parameter estimation procedure. Estimating the parameters $\Theta = (\mathfrak{a}, \mathfrak{b}, \kappa, \lambda, \nu, f, h)$ is critical for accurately modeling the degradation process. We utilize outputs from the LSTM network to inform initial estimates of $g(s, \mathcal{Z}_s)$ and $\sigma(s, \mathcal{Z}_s)$. Specifically, the LSTM outputs guide the fitting of degradation rates to the functional forms. For statistical estimation, we use Maximum Likelihood Estimation (MLE) on observed trajectories. Due to jump complexities, we discretize time into N intervals, forming increments $\Delta\mathcal{Z}_i = \mathcal{Z}_{t_i} - \mathcal{Z}_{t_{i-1}}$ and a likelihood:

$$\mathcal{L}(\Theta) = \prod_{i=1}^{N} p_{\Delta\mathcal{Z}_i}\left(\Delta\mathcal{Z}_i \mid \mathcal{Z}_{t_{i-1}}; \Theta\right),$$

where $p_{\Delta\mathcal{Z}_i}$ are transition densities approximated by Euler–Maruyama for the continuous part and compound Poisson processes for jumps.

We implement the Expectation-Maximization (EM) algorithm (Algorithm 1) to handle latent variables associated with unobserved jump times and sizes. The E-step computes expected values of these latent variables given current parameter estimates and observed data. The M-step maximizes the expected complete-data log-likelihood with respect to Θ. This iterative process continues until convergence, yielding estimated parameters that best fit the observed data.

Algorithm 1 Expectation-Maximization Algorithm for Parameter Estimation in the SSM Model

Input: Observed states $\{\mathcal{Z}_{t_i}\}_{i=0}^{N}$, initial parameter guess $\Theta^{(0)} = (\mathfrak{a}, \mathfrak{b}, \kappa, \lambda, \nu, f, h)$, convergence threshold ε
Output: Estimated parameters Θ^*
Preprocessing: Discretize the time interval into $\{t_i\}_{i=0}^{N}$; form increments $\Delta\mathcal{Z}_i = \mathcal{Z}_{t_i} - \mathcal{Z}_{t_{i-1}}$.
Initialize $\Theta^{(0)}$ (e.g., via LSTM-informed degradation rates or domain knowledge)
$k \leftarrow 0$
repeat
  E-step: (Expected Sufficient Statistics)
    1. Compute latent jump-related statistics (e.g., expected jump counts, sizes) given $\Theta^{(k)}$.
    2. Evaluate the expected complete-data log-likelihood $\mathbb{E}[\log \mathcal{L}(\Theta \mid \Delta\mathcal{Z}, \text{latent})]$.
  M-step: (Parameter Update)
    1. Maximize the expected complete-data log-likelihood w.r.t. Θ to obtain $\Theta^{(k+1)}$:
    $\Theta^{(k+1)} \leftarrow \arg\max_{\Theta} \mathbb{E}[\log \mathcal{L}(\Theta \mid \Delta\mathcal{Z}, \text{latent})]$.
    2. Update model functions $g(s, \mathcal{Z}_s)$, $\sigma(s, \mathcal{Z}_s)$, and $\gamma(s, \mathcal{Z}_{s^-}, z)$ using new parameter values.
  $k \leftarrow k + 1$
until $\|\Theta^{(k)} - \Theta^{(k-1)}\| < \varepsilon$ or maximum iterations reached
Return: $\Theta^* \leftarrow \Theta^{(k)}$

2.4. Reinforcement learning-based hyperparameter tuning

In this section, we propose a new hyperparameter tuning method based on reinforcement learning. Determining the optimum set of hyperparameters θ gives the best prediction accuracy in deep learning models. Traditional methods, such as grid search or Bayesian optimization, are often static and do not adapt to changing conditions during training. But RLHT adapts to changes during the training and test periods. This method treats the tuning process as a decision problem.

Objective: Minimize the prediction error $L(\theta)$, where L represents metrics like RMSE or MAE.

$$\theta^* = \arg\min_{\theta} L(\theta). \tag{5}$$

In this paper, we model the optimization of hyperparameter tuning problem under RLHT as a "Markov Decision Process (MDP)".

2.4.1. Markov decision process (MDP) formulation

The *State Space* S_t consists of the current hyperparameters $\theta_t = (\theta_{1,t}, \theta_{2,t}, \ldots, \theta_{n,t})$ and performance metrics P_t, defined as $S_t = (\theta_t, P_t)$. The *Action Space* A_t modifies one or more hyperparameters, where $\theta_{i,t}$ is updated as $\theta_{i,t+1} = \theta_{i,t} + \Delta\theta_{i,t}$. The *Reward Function* R_t measures performance improvement; for minimization, $R_t = -\Delta L(\theta_t) = -(L(\theta_{t+1}) - L(\theta_t))$. The *Transition Probability* $P(S_{t+1}|S_t, A_t)$ describes the likelihood of reaching S_{t+1}, influenced by the model's response to updated hyperparameters. Finally, the *Policy* $\pi(S_t)$ determines action selection to find the optimal policy π^* that maximizes cumulative rewards.

Table 1
Performance comparison of Bayesian optimization and RLHT.

Method	MAE (lower is better)	RMSE (lower is better)	Tuning Time (hours)
Bayesian Optimization	12.34	15.67	5.4
RLHT	**10.12**	**13.45**	**3.5**

2.4.2. Reinforcement learning algorithm

The goal of the RL algorithm is to determine the optimal policy π^* that maximizes expected cumulative rewards:

$$G_t = \mathbb{E}\left[\sum_{k=t}^{T} \rho^{k-t} R_k\right], \tag{6}$$

where ρ is the "discount factor" ($0 \leq \rho < 1$), which balances immediate and future rewards.

Bellman Equation. The "optimal value function $V^*(S_t)$" is:

$$V^*(S_t) = \max_{A_t}\left[R_t + \rho \sum_{S_{t+1}} P(S_{t+1}|S_t, A_t) V^*(S_{t+1})\right]. \tag{7}$$

This equation forms the basis of dynamic programming solutions like "Q-Learning" or "Deep Q-Networks (DQN)," which approximate $V^*(S_t)$ to derive the optimal policy.

Q-Learning Algorithm. Q-Learning uses the action-value function $Q(S_t, A_t)$ to estimate expected rewards for choosing action A_t in state S_t and following the optimal policy thereafter. The update rule is:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\left[R_t + \rho \max_{A'} Q(S_{t+1}, A') - Q(S_t, A_t)\right], \tag{8}$$

where α is the learning rate. The optimal policy is:

$$\pi^*(S_t) = \arg\max_{A_t} Q(S_t, A_t) \tag{9}$$

Deep Q-Network (DQN). For large state–action spaces, a DQN approximates $Q(S, A; \theta)$ with a neural network. The loss function for training the network is:

$$\mathcal{L}(\theta) = \mathbb{E}\left[\left(R_t + \rho \max_{A'} Q(S_{t+1}, A'; \theta^-) - Q(S_t, A_t; \theta)\right)^2\right], \tag{10}$$

where θ^- is the target network parameters.

The Algorithm 2 outlines the RLHT process, which iteratively adjusts hyperparameters to optimize model performance.

Algorithm 2 Reinforcement Learning-Based Hyperparameter Tuning (RLHT)

Input: State space S, Action space $\mathcal{A}$, Discount factor ρ, Learning rate α, Initial Q-network parameters θ, Target network parameters θ^-, Replay buffer D, Batch size B
Output: Optimal hyperparameters θ^*
Initialize Q-network with θ and target network $\theta^- \leftarrow \theta$
Initialize replay buffer D
for each episode = 1 to N **do**
  Initialize state S_0
  for each time step $t = 0$ to T **do**
    Select action A_t via ϵ-greedy policy:

$$A_t = \begin{cases} \text{random action} & \text{with probability } \epsilon \\ \arg\max_A Q(S_t, A; \theta) & \text{otherwise} \end{cases}$$

    Execute action A_t to adjust hyperparameters and train the predictive model
    Observe reward R_t and new state S_{t+1}
    Store transition (S_t, A_t, R_t, S_{t+1}) to D
    Sample mini-batch of B transitions (S_i, A_i, R_i, S_{i+1}) from D
    Set target for each mini-batch transition: $y_i = R_i + \rho \max_{A'} Q(S_{i+1}, A'; \theta^-)$
    Perform a gradient descent step on the loss function:

$$\mathcal{L}(\theta) = \frac{1}{B}\sum_{i=1}^{B}\left[y_i - Q(S_i, A_i; \theta)\right]^2$$

    Update $\theta^- \leftarrow \theta$ every C steps
    Update state $S_t \leftarrow S_{t+1}$
  end for
  Decay ϵ to reduce exploration over time
end for
Return Optimal hyperparameters θ^* corresponding to the best observed performance

To evaluate the effectiveness of RLHT, we compare it with Bayesian optimization in terms of MAE, RMSE, and tuning time. Table 1 presents the comparison results of RLHT and Bayesian optimization.

Here, the Bayesian Optimization performed 50 iterations, while RLHT conducted 30 iterations. The different number of iterations is justified by RLHT's faster convergence, allowing it to achieve superior performance with fewer trials. Both methods tested multiple combinations of learning rates ($[0.00001, 0.01]$), dropout rates ($[0.1, 0.5]$), CNN filters ($[32, 128]$), and LSTM units ($[32, 128]$). Exploring a wide range of hyperparameter combinations involves substantial training overhead at each iteration, and our integrated CNN-Transformer-LSTM-SSM framework requires more computational steps per trial to capture both deterministic and stochastic degradation dynamics. Although the reported tuning times, 5.4 h for Bayesian Optimization and 3.5 h for RLHT may appear substantial, they reflect the complexity of testing a comprehensive hyperparameter space and training the integrated CNN-Transformer-LSTM-SSM model repeatedly on the NASA C-MAPSS FD001 dataset. Despite these tuning times, RLHT achieved lower MAE and RMSE compared to Bayesian Optimization, demonstrating that the additional computational effort translates into significantly improved RUL prediction performance. As this process is performed offline, the final deployment of the predictive maintenance framework remains unaffected in real-time operations. Crucially, RLHT's faster convergence and superior accuracy (lower MAE and RMSE) underscore the benefits of this more targeted search approach.

2.5. CNN-transformer-LSTM-SSM ensemble method

This subsection introduces an ensemble approach that integrates CNN, transformers, LSTM networks, and the SSM model. The ensemble method uses the complementary strengths of these individual components to enhance the prediction of machinery failures and to improve the estimation of RUL.

2.5.1. Ensemble architecture

The proposed ensemble model functions as follows: CNN layers initially captures the "spatial features" from the input data, which are then forwarded to a Transformer module for capturing long-range dependencies. The LSTM component further refines temporal patterns, while the SSM layer provides stochastic modeling for more accurate and reliable predictions. The architecture of the proposed ensemble model for RUL prediction and PdM is presented in Fig. 6.

2.5.2. Training process and system configuration

The training of the CNN-Transformer-LSTM-SSM ensemble follows a sequential approach to effectively capture spatial, temporal, and stochastic features. Preprocessing involves dropping irrelevant features (columns 0, 1, 2, 3, 4, 5, 9, 10, 14, 20, 22, 23 in the NASA C-MAPSS dataset) and normalizing the remaining data using Min-Max scaling. Time-series data is segmented into overlapping sequences (30 time steps, shift of 1-step) to preserve temporal dependencies. Each component CNN, Transformer, and LSTM is trained using supervised learning techniques. The CNN uses three Conv1D layers (filters: 64, 128, 64; kernels: 7, 5, 3) to extract localized spatial features. The Transformer's multi-head attention mechanism highlights critical temporal elements, while two LSTM layers (128 units each, dropout 0.2) refine sequential patterns. Outputs are concatenated and passed through fully connected layers with ReLU activations, leading to a final output layer that predicts the RUL using a linear activation function.

The ensemble is trained end-to-end using Adam optimizer (initial learning rate 0.0005) and optimized with RLHT model for parameters like learning rates, dropout, and attention settings. Early stopping with a patience of 10 epochs and learning rate reduction prevent

overfitting. Regularization techniques like L2 weight decay and batch normalization improve stability. The SSM parameters are estimated via MLE to ensure accurate stochastic modeling. The entire training process is implemented using `TensorFlow` and `Keras` in Python 3.7 and executed on a high-performance computing system equipped with an Intel core i3 13th generation processor with 16 GB RAM, enabling efficient handling of large datasets and complex model architectures.

2.6. Remaining useful life prediction under the smooth semi-martingale model

Predicting the RUL of industrial systems with high accuracy enables effective maintenance scheduling, failure prevention, and cost savings. We present an RUL prediction framework utilizing the SSM model, a stochastic process that separates random dynamics into deterministic and stochastic components. This model simulates machinery degradation and estimates RUL as follows:

$$\mathrm{RUL}_t = \inf\{u \geq t : \mathcal{Z}_u \geq \zeta\} - t, \tag{11}$$

where RUL_t is the remaining useful life at time t, $\mathcal{Z}_u$ represents the system state at time u, and ζ is the failure threshold.

The state evolution $\mathcal{Z}_t$ in the SSM model comprises deterministic and stochastic components (Eq. (4)), capturing both continuous degradation and sudden failures. To predict $\mathcal{Z}_u$ for $u > t$, we employ Monte Carlo simulations (Algorithm 3). These simulations generate multiple future paths of $\mathcal{Z}_u$, resulting in a distribution of RUL_t.

Algorithm 3 Monte Carlo Simulation for RUL Prediction under SSM

Input: Current state $\mathcal{Z}_t$, failure threshold ζ, model parameters g, σ, γ, time step Δt, total simulations K, max time $T_{\max}$
Output: Distribution of RUL estimates
Initialize an empty list for RULs: $\mathrm{RUL}_t \leftarrow []$
for $k = 1$ to K **do** ▷ Monte Carlo iterations
  Initialize $u \leftarrow t$, $\mathcal{Z}_u^{(k)} \leftarrow \mathcal{Z}_t$ ▷ Set initial state and time
  while $\mathcal{Z}_u^{(k)} < \zeta$ **and** $u \leq T_{\max}$ **do** ▷ Check threshold or max time
    Compute drift: $\Delta D \leftarrow g(u, \mathcal{Z}_u^{(k)})\Delta t$
    Compute diffusion: $\Delta B \leftarrow \sigma(u, \mathcal{Z}_u^{(k)})\Delta W,\ \Delta W \sim \mathcal{N}(0, \Delta t)$
    Simulate jumps: $N \sim \mathrm{Poisson}(\lambda \Delta t)$
    Compute total jump: $\Delta J \leftarrow \sum_{j=1}^{N} \gamma(u, \mathcal{Z}_u^{(k)}, z_j),\ z_j \sim \nu(z)$
    Update state: $\mathcal{Z}_{u+\Delta t}^{(k)} \leftarrow \mathcal{Z}_u^{(k)} + \Delta D + \Delta B + \Delta J$
    Increment time: $u \leftarrow u + \Delta t$
  end while
  if $\mathcal{Z}_u^{(k)} \geq \zeta$ **then**
    Compute RUL for this simulation: $\mathrm{RUL}_t^{(k)} \leftarrow u - t$
  else
    $\mathrm{RUL}_t^{(k)} \leftarrow T_{\max} - t$ ▷ Max time reached
  end if
  Append $\mathrm{RUL}_t^{(k)}$ to RUL_t
end for
Return: Compute statistics: mean, variance, and confidence intervals of RUL_t

Each simulation updates the system state $\mathcal{Z}_u^{(k)}$ through drift (ΔD), diffusion (ΔB), and jumps (ΔJ). Drift represents deterministic degradation, diffusion captures random fluctuations ($\Delta W \sim \mathcal{N}(0, \Delta t)$), and jumps model sudden failures using Poisson (N) and jump size distribution ($\nu(z)$). The simulation stops when $\mathcal{Z}_u^{(k)}$ exceeds ζ or reaches $T_{\max}$, with RUL as $u - t$, generating a distribution of RUL estimates. Model performance is evaluated using RMSE for RUL predictions and accuracy for failure detection, with cross-validation to prevent overfitting.

In Algorithm 3, each Monte Carlo iteration corresponds to one of the K independent simulations that generate a potential future degradation path for the system state $\mathcal{Z}_t$. By repeatedly simulating the drift, diffusion, and sudden jump components of the SSM model, the algorithm produces multiple prospective trajectories of $\mathcal{Z}_u$. This yields a distribution of RUL estimates, rather than a single deterministic value, thereby quantifying uncertainty in how rapidly the system may degrade. The purpose of this Monte Carlo algorithm is twofold: (1) to capture the inherent variability and randomness in the degradation process, and (2) to provide statistical measures (e.g., mean and confidence intervals) for the RUL prediction. This distribution-driven approach is crucial for more robust decision-making in prognostics, as it enables maintenance strategies to account for worst-case scenarios, sudden failures, or extended lifetimes that deterministic models might overlook. The SSM-based RUL prediction framework is illustrated in Fig. 7.

Representation of uncertainty in RUL predictions. Integrating the SSM model with the deep learning framework explicitly models the uncertainty in RUL predictions. The stochastic components $\sigma(s, \mathcal{Z}_s)\, dW_s$ and $\gamma(s, \mathcal{Z}_{s^-}, z)\, \tilde{N}(ds, dz)$ capture random fluctuations and sudden degradation jumps, respectively. By simulating multiple $\mathcal{Z}_t$ paths using estimated parameters, we derive a distribution of degradation trajectories and corresponding RUL estimates. This probabilistic framework quantifies RUL uncertainty and enables the construction of confidence intervals, supporting risk assessment and decision-making in predictive maintenance.

3. Predictive maintenance scheduling and mission cycle optimization

3.1. Predictive maintenance scheduling

In this section, we present a PdM scheduling framework that uses raw sensor data and our proposed ensemble model to forecast the RUL of machinery.

3.1.1. Predictive model integration for maintenance scheduling

The ensemble model preprocesses sensor data $\mathbf{X}$ through normalization and segmentation, followed by CNNs extracting spatial features: $\mathbf{X}_{\text{feature}} = f_{\text{CNN}}(f_{\text{Preprocess}}(\mathbf{X}))$. Transformers capture long-range temporal dependencies: $\mathbf{X}_{\text{temporal}} = f_{\text{trans}}(\mathbf{X}_{\text{feature}})$, which are further refined by LSTMs: $\mathbf{X}_{\text{lstm}} = f_{\text{LSTM}}(\mathbf{X}_{\text{temporal}})$. The SSM layer models the system state $\mathcal{Z}_t$ with deterministic trends and stochastic fluctuations as:

$$\mathcal{Z}_t = \mathcal{Z}_0 + \int_0^t g(s, \mathcal{Z}_s, \mathbf{X}_{\text{temporal}})\, ds + M_t,$$

where $M_t = \int_0^t \sigma(s, \mathcal{Z}_s, \mathbf{X}_{\text{temporal}})\, dW_s + \int_0^t \int_{\mathbb{R}} \gamma(s, \mathcal{Z}_{s^-}, z, \mathbf{X}_{\text{temporal}})\, \tilde{N}(ds, dz)$. This model provides real-time health assessments, enabling the maintenance scheduling algorithm to optimize maintenance timing based on predicted RUL and failure probabilities.

3.1.2. Maintenance scheduling optimization

Maintenance scheduling aims to minimize the total cost of maintenance and failure using predicted RUL and failure probabilities. Let τ denote the maintenance time. The optimal time τ_{opt} is:

$$\tau_{\text{opt}} = \arg\min_{\tau \geq t} \mathbb{E}\big[C(\tau, \mathcal{Z}_\tau, \mathrm{RUL}_t)\big], \tag{12}$$

where $C(\tau, \mathcal{Z}_\tau, \mathrm{RUL}_t)$ represents the cost associated with both maintenance, urgency of intervention, and failure. In accordance with standard practice in reliability and maintenance theory, a distinct baseline maintenance cost is employed for identical maintenance actions. To explicitly account for the urgency of intervention as RUL decreases, we introduce:

$$C(\tau, \mathcal{Z}_\tau, \mathrm{RUL}_t) = c_{\text{maint}} + c_{\text{scale}} \exp\big(-k\,(\mathrm{RUL}_t - (\tau - t))\big) + c_{\text{fail}}\, \mathbb{I}_{\{\mathcal{Z}_\tau \geq \zeta\}}. \tag{13}$$

Here, c_{maint}, c_{scale}, and c_{fail} represent the baseline maintenance cost, an urgency-based cost scaling term, and a failure penalty, respectively. The exponential term intensifies as the system approaches failure thresholds, signaling higher urgency for maintenance. This dynamic scaling captures real-world conditions, where delayed maintenance triggers emergency labor, secondary damage, and extended downtime, causing exponentially higher repair costs and operational risks. However, this scaling is optional; if $c_{\text{scale}} = 0$, the cost function reduces to a standard cost model in reliability theory. The indicator function $\mathbb{I}_{\{\mathcal{Z}_\tau \geq \zeta\}}$ imposes a penalty when the system's degradation $\mathcal{Z}_\tau$ exceeds the critical limit ζ. The parameter k controls how rapidly the exponential cost rises as RUL decreases. By taking the expectation of $C(\tau, \mathcal{Z}_\tau, \mathrm{RUL}_t)$ over the

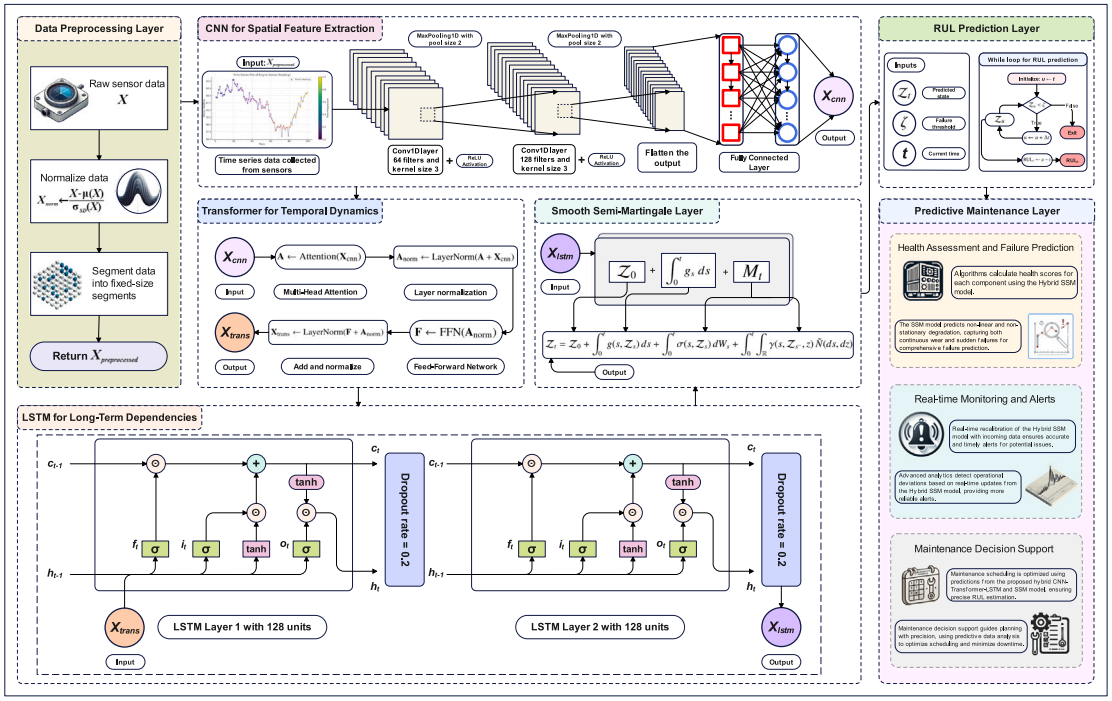


Fig. 6. The proposed ensemble model architecture.

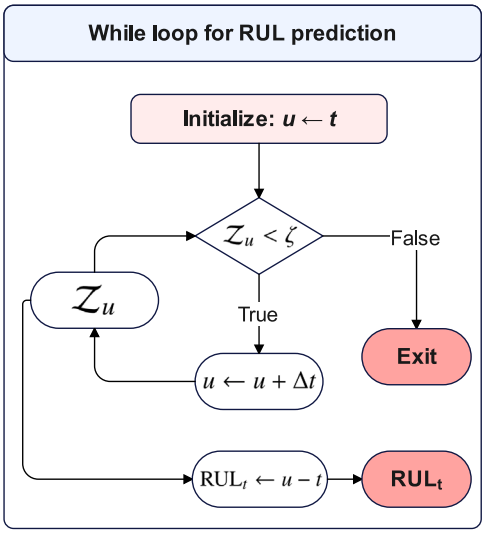


Fig. 7. Remaining useful life prediction using *While Loop* under the SSM model.

stochastic degradation process, we inherently capture the probabilistic nature of system failure. This ensures robust real-time scheduling in complex, uncertain environments.

3.1.3. PdM scheduling under the proposed ensemble model

By continuously updating RUL predictions and expected costs, the PdM scheduling algorithm (Algorithm 4) triggers maintenance when RUL_t/ζ falls below prescribed thresholds, adjusting τ_{opt} as new data emerges. Each iteration acquires the current predicted RUL, evaluates maintenance and failure costs, and selects the τ_{opt} minimizing total expected cost. Computationally, the core complexity arises from RUL prediction (handled efficiently by the trained deep model) and a simple one-dimensional search for τ_{opt}. Thus, the scheduling algorithm remains computationally tractable for real-time implementation.

Algorithm 4 Predictive Maintenance Scheduling Based on Predicted RUL

Input: Predicted RUL_t, degradation state $\mathcal{Z}_t$, failure threshold ζ, time horizon T, baseline cost c_{maint}, scaling parameter c_{scale}, failure penalty c_{fail}, exponential rate k, risk tolerance ϵ, model parameters θ
Output: Optimal maintenance schedule τ_{opt}
Initialization: Initialize RUL_0, $\mathcal{Z}_0$, ζ, θ, and ϵ
procedure PREDICT RUL
  for $t \in [0, T]$ **do**
    Update $\mathrm{RUL}_t = f_{\mathrm{PredictiveModel}}(t; \theta)$ using sensor data.
    Update degradation state $\mathcal{Z}_t$ accordingly.
  end for
end procedure
procedure MAINTENANCE DECISION
  Set maintenance threshold τ_{thresh}.
  if $\mathrm{RUL}_t \le \tau_{\mathrm{thresh}}$ **then**
    Schedule maintenance at time t.
  else
    Compute risk: $\mathrm{Risk} = \frac{\mathrm{RUL}_t}{\zeta}$.
    if Risk $< \epsilon$ **then**
      Trigger maintenance at time t.
    else
      Continue operations.
    end if
  end if
end procedure
procedure COST CALCULATION
  Compute cost at time τ:
  $C(\tau, \mathcal{Z}_\tau, \mathrm{RUL}_t) = c_{\mathrm{maint}} + c_{\mathrm{scale}} \exp\left(-k\left(\mathrm{RUL}_t - (\tau - t)\right)\right) + c_{\mathrm{fail}}\, \mathbb{I}_{\{\mathcal{Z}_\tau \ge \zeta\}}$,
  where $\mathbb{I}_{\{\cdot\}}$ is the indicator function. (Note: Setting $c_{\mathrm{scale}} = 0$ reverts to the standard cost model.)
end procedure
procedure COST OPTIMIZATION
  For $t \in [0, T]$, compute the expected cost: $\hat{C}(t) = \mathbb{E}\left[C(t, \mathcal{Z}_t, \mathrm{RUL}_t)\right]$.
  Determine the optimal maintenance time: $\tau_{\mathrm{opt}} = \arg\min_{t \in [0,T]} \hat{C}(t)$.
end procedure
procedure DYNAMIC ADJUSTMENT
  Continuously update RUL_t and $\mathcal{Z}_t$ with new sensor data.
  Recalculate $\hat{C}(t)$ and adjust τ_{opt} as needed.
  If significant deviations occur, trigger rescheduling.
end procedure
return Optimal maintenance schedule τ_{opt}

Practical applications of the PdM scheduling model. Integrating accurate RUL predictions into maintenance scheduling enables just-in-time actions, minimizing costly downtime and preventing catastrophic failures. In high-stakes sectors such as aerospace, nuclear power, and pharmaceutical production, data-driven maintenance enhances safety, extends equipment life, and ensures operational continuity. In manufacturing, PdM reduces unexpected breakdowns, optimizes production schedules, and improves efficiency. Similarly, energy, oil and gas, transportation, and healthcare benefit from precise strategies that boost system reliability, streamline resource allocation, and ensure compliance with strict safety standards. By reducing downtime, optimizing inventory and manpower use, and improving operational efficiency, PdM delivers a competitive edge in cost-sensitive, risk-intensive global markets.

The feasibility of the proposed PdM model is ensured through: (1) *Computational Efficiency*. The pretrained ensemble-based RUL predictor enables rapid inference, while the one-dimensional optimization for τ_{opt} keeps scheduling computationally lightweight. (2) *Real-Time Implementation.* Incremental sensor updates dynamically refine RUL and cost estimates, while seamless integration with the mission abort strategy ensures synchronized decision-making based on updated risk profiles. (3) *Scalability and Adaptability*. The modular design supports diverse systems by decoupling RUL prediction from scheduling, and adjustable parameters with dynamic recalibration allow customization for specific operational and risk requirements.

3.2. Mission cycles assignment and optimization based on the proposed ensemble model

In mission-critical systems, such as aerospace operations, engines must be dynamically assigned to mission cycles comprising sequential segments with specific time windows, reliability requirements, and varying operational conditions. The proposed ensemble model provides accurate RUL predictions, enabling a scenario-based multi-stage optimization that reallocates engines, adjusts mission plans, and schedules maintenance to minimize overall costs and mitigate mission failure risks.

3.2.1. Mission cycle assignment

Consider a set of missions $\mathcal{M} = \{M_1, \dots, M_m\}$, where each mission M_j consists of L_j segments $(M_{j,1}, \dots, M_{j,L_j})$. Each segment $M_{j,\ell}$ is executed within the time window $[t^{\text{start}}_{j,\ell}, t^{\text{end}}_{j,\ell}]$ and requires a minimum RUL $D_{j,\ell}$ for the assigned engine at $t^{\text{start}}_{j,\ell}$. The set of engines is denoted as $\mathcal{E} = \{E_1, \dots, E_n\}$. Uncertainties in mission loads, environmental factors, engine degradation, and resource availability are modeled through a scenario set Ω, where $\mathbb{P}(\omega)$ is the probability of scenario $\omega \in \Omega$. These scenarios reflect changes such as weather patterns, unplanned route adjustments, or delays in spare parts supply.

We define binary variables $x_{i,j,\ell} \in \{0,1\}$ to indicate if engine E_i is assigned to segment $M_{j,\ell}$, and continuous variables $y^{\omega}_j \in [0,1]$ to represent the fraction of mission M_j completed under scenario ω. The objective is to minimize expected costs, including maintenance, fuel, and failure penalties. Let $C^{\omega}_{i,j,\ell}$ be the scenario-dependent cost of assigning engine E_i to segment $M_{j,\ell}$, and U^{ω}_j denote the penalty for not completing mission M_j. The two-stage stochastic program is formulated as:

$$\min_{x,y} \sum_{\omega\in\Omega} \mathbb{P}(\omega) \left(\sum_{j=1}^{m} \sum_{\ell=1}^{L_j} \sum_{i=1}^{n} C^{\omega}_{i,j,\ell} x_{i,j,\ell} + \sum_{j=1}^{m} U^{\omega}_j (1 - y^{\omega}_j) \right), \tag{14}$$

subject to:

$$\sum_{i=1}^{n} x_{i,j,\ell} = 1 \quad \forall j, \ell, \tag{15}$$

$$\text{RUL}^{\omega}_{i,t_{j,\ell}} \geq D_{j,\ell} x_{i,j,\ell} \quad \forall i, j, \ell, \omega, \tag{16}$$

$$x_{i,j,\ell} \in \{0,1\}, \quad y^{\omega}_j \in [0,1].$$

Additional constraints incorporate mission-specific requirements. If a segment $M_{j,\ell}$ demands high-thrust operations, the RUL must also satisfy an additional buffer $\Delta D^{\omega}_{j,\ell}$:

$$x_{i,j,\ell} = 1 \Rightarrow \text{RUL}^{\omega}_{i,t_{j,\ell}} \geq D_{j,\ell} + \Delta D^{\omega}_{j,\ell}. \tag{17}$$

Engines in abrasive conditions, such as sandstorms, consume extra cycles $h^{\omega}_{i,j,\ell}$, constrained by scenario-dependent limits $\mathbb{L}^{\omega}_i$:

$$\sum_{j=1}^{m} \sum_{\ell=1}^{L_j} h^{\omega}_{i,j,\ell} x_{i,j,\ell} \leq \mathbb{L}^{\omega}_i.$$

If engine E_i lacks required certifications for segment $M_{j,s}$, it cannot be assigned:

$$x_{i,j,\ell} = 0 \quad \text{if } E_i \text{ lacks certification for } M_{j,\ell}. \tag{18}$$

Let Λ^{ω}_t be the maintenance crew capacity at time t under scenario ω, and φ_i the crew workload for engine E_i. Then:

$$\sum_{i=1}^{n} \varphi_i m^{\omega}_{i,t} \leq \Lambda^{\omega}_t, \quad \forall t, \omega, \tag{19}$$

where $m^{\omega}_{i,t} \in \{0,1\}$ indicates if E_i is scheduled for maintenance at time t.

Further, engine assignments must not exceed available spares S^{ω}:

$$\sum_{j=1}^{m} \sum_{\ell=1}^{L_j} x_{i,j,\ell} \leq S^{\omega}, \quad \forall i, \omega. \tag{20}$$

3.2.2. Mission operation and real-time monitoring

During mission execution, the engine health state $\mathcal{H}_t$ evolves according to the SSM model:

$$\mathcal{H}_t = \mathcal{H}_0 + \int_0^t g(s, \mathcal{H}_s)\, ds + M_t, \tag{21}$$

where g represents the deterministic degradation rate, and M_t models stochastic fluctuations and sudden failures. Real-time sensor data updates the RUL estimates $\text{RUL}^{\omega}_{i,t}$ for each engine E_i. If $\mathcal{H}^{\omega}_t$ approaches a critical threshold ζ, the model initiates immediate actions such as aborting a mission segment, rerouting missions, reallocating engine loads, or scheduling immediate maintenance. At each decision epoch t, the optimization problem (Eq. (14)) is re-solved with updated scenarios Ω_t, reflecting current engine health and mission status. By utilizing a receding horizon approach, the framework adapts to evolving conditions, reducing failures and associated costs while acknowledging that not all scenarios guarantee full mission completion.

3.2.3. Optimization and fine-tuning

Let $C^{\omega}_{\text{maint},i,j,\ell}$, $C^{\omega}_{\text{fuel},i,j,\ell}$, and $C^{\omega}_{\text{fail},i,j,\ell}$ represent maintenance, fuel, and failure costs for engine i, segment j, ℓ, and scenario ω, respectively. The total cost, including penalties p for mission aborts, is defined as:

$$C_{\text{total}} = \sum_{j=1}^{m} \sum_{\ell=1}^{L_j} \sum_{i=1}^{n} \left(C^{\omega}_{\text{maint},i,j,\ell} + C^{\omega}_{\text{fuel},i,j,\ell} + C^{\omega}_{\text{fail},i,j,\ell} \right) x_{i,j,\ell} + \sum_{j=1}^{m} U^{\omega}_j (1 - y^{\omega}_j) + p \cdot \mathbb{I}_{\{\text{Failures} > \zeta'\}}, \tag{22}$$

The decision variables $x_{i,j,\ell}$ ensure that only selected actions contribute to the total cost. Operational penalties $\sum_{j=1}^{m} U^{\omega}_j (1 - y^{\omega}_j)$ enforce mission completion, while $p \cdot \mathbb{I}_{\{\text{Failures} > \zeta'\}}$ penalizes excessive failures, with ζ' representing the acceptable failure threshold.

Parameter tuning θ minimizes the expected total cost across all scenarios Ω:

$$\min_{\theta} \sum_{\omega\in\Omega} \mathbb{P}(\omega) \left(C_{\text{total}}(\theta, \omega) + p \cdot \mathbb{I}_{\{\text{Failures}(\theta,\omega) > \zeta'\}} \right). \tag{23}$$

This optimization balances operational costs with reliability and mission success. Techniques such as scenario reduction, robust optimization, and decomposition enhance computational efficiency for large-scale problems.

3.2.4. Minimizing maintenance costs

Maintenance scheduling optimizes the balance between scheduled and unscheduled maintenance to minimize total costs and operational downtime. The maintenance cost function is defined as:

$$C_{\text{maint}} = \sum_{i=1}^{n} \left(c_{\text{sch},i} m_i + c_{\text{unsch},i} (1 - m_i) \right), \tag{24}$$

where $m_i \in \{0, 1\}$ indicates whether engine E_i undergoes scheduled maintenance. Scheduled maintenance incurs a lower cost $c_{\text{sch},i}$, while unscheduled repairs following unexpected failures are more costly, denoted by $c_{\text{unsch},i}$.

To ensure maintenance is performed before critical degradation, we constraint:

$$m_i \geq \frac{x_{i,j,\ell} h^{\omega}_{i,j,\ell}}{H^{\omega}_i} \quad \forall i, j, \ell, \omega, \tag{25}$$

where H^{ω}_i is the maximum allowable wear for engine E_i under scenario ω. This constraint links maintenance decisions m_i to usage-induced degradation $h^{\omega}_{i,j,\ell}$, ensuring maintenance is triggered when the risk of imminent failure increases.

Additionally, to manage spare parts logistics, we impose:

$$\sum_{i=1}^{n} s_i m_i \leq S^{\omega}, \quad \forall t, \omega, \tag{26}$$

where s_i represents the spare parts requirement for engine E_i and S^{ω} denotes the spare parts capacity under scenario ω.

4. Dynamic mission abortion policy based on predicted RUL

In mission-critical operations, deciding whether to abort a mission relies on accurate and timely RUL predictions [50,51]. Recent studies have refined mission abort strategies by incorporating multi-component failure interactions [39], adaptive policies for uncertain shock environments [43], and joint optimization with system structure to handle dynamic tasks [45]. The proposed dynamic mission abortion policy uses real-time RUL estimates from the CNN-Transformer-LSTM-SSM ensemble model, enabling informed decisions on whether to continue, adjust, or abort a mission. The RUL, denoted as RUL_t, is updated as:

$$\text{RUL}_t = f_{\text{LSTM}}(f_{\text{trans}}(f_{\text{CNN}}(\mathbf{X}_t))) + M_t,$$

where M_t captures stochastic degradation dynamics.

4.1. Dynamic thresholds based on RUL

Unlike static thresholds, the dynamic threshold τ_t adapts to evolving system health and stochastic degradation patterns [52].

$$\tau_t = \tau_0 \cdot \exp(-\delta \cdot (t - t_0)) + \xi(t), \tag{27}$$

where τ_0 is derived from mission-level reliability requirements, $\delta > 0$ controls sensitivity to time and usage, and $\xi(t)$ represents stochastic variations. As t increases, τ_t decreases, reflecting higher failure risks as components near end-of-life, as modeled by the SSM.

Here, the term $\exp(-\delta(t - t_0))$ aligns with the observed acceleration in failure probabilities for aging machinery, and $\xi(t)$ captures sudden jumps. This integration ensures that τ_t accurately tracks the distribution of future states $\mathcal{Z}_t$, grounding the threshold in the same stochastic mechanics (SSM structure) that govern RUL estimation. The parameters τ_0, δ, and the properties of $\xi(t)$ are optimized using RLHT with historical run-to-failure data (e.g., NASA C-MAPSS). Specifically, δ is tuned to balance false alarms and missed detections, while $\xi(t)$ is modeled as a noise process through MLE to match empirical operational variance. These dynamic thresholds thus adapt to both deterministic drift and stochastic volatility, aligning maintenance actions with real-time risk assessments.

4.2. Continuous monitoring and preemptive alerts

In this framework, continuous monitoring of the RUL enables timely preemptive alerts that prevent unexpected failures. The PdM closely tracks the RUL (RUL_t) and triggers alerts when it nears a dynamic threshold τ_t. The rate of change of RUL, $\Delta\text{RUL}_t = \frac{d(\text{RUL}_t)}{dt}$, serves as an early warning signal. An alert is issued if ΔRUL_t exceeds a time-dependent threshold:

$$a_t = a_0 \cdot \exp(-\kappa \cdot (t - t_0)) + \phi(t), \tag{28}$$

where a_0 is the initial baseline, $\kappa > 0$ controls alert sensitivity over time, and $\phi(t)$ captures mission-specific conditions.

SSM-based RUL predictions capture degradation increments from Brownian motion and Poisson jumps. The rate ΔRUL_t reflects these changes, and comparing it to a_t identifies abnormal surges. When $\Delta\text{RUL}_t > a_t$, it signals significant deviations from expected stochastic patterns, requiring immediate intervention.

Practical implementation. When $\Delta\text{RUL}_t > a_t$, maintenance actions include: (1) Inspecting the engine for issues, (2) Reassigning tasks to healthier engines, and (3) Adjusting operational parameters to reduce degradation. The adaptable a_t minimizes false positives during stable periods while preventing failures that static thresholds overlook. Moreover, the PdM model is integrated with the dynamic mission abort strategy through continuous real-time updates of RUL predictions and risk assessments. As the PdM model refines RUL estimates, it calculates the risk of failure, which is then communicated to the mission abort policy. If the predicted RUL falls below critical thresholds, the mission abort strategy triggers an abort decision to prevent failure. This interaction ensures that the PdM model drives maintenance scheduling, while the abort policy dynamically responds to evolving risk, maintaining system safety and mission success. Further, the Fig. 8 presents the decision process flowchart for the dynamic mission abort policy based on predicted RUL and associated costs.

4.3. Mission reallocation and optimization based on RUL

When the predicted RUL of an aircraft engine is insufficient to safely complete a mission, the policy assesses whether to shift the mission to another aircraft. Let A_j represent the current aircraft and A_k a potential replacement. The reallocation decision minimizes expected total cost, considering both transfer and operational expenses:

$$C_{\text{shift}} = \min_k \left[C_{\text{transfer}}(A_j, A_k) + C_{\text{oper}}(A_k) \cdot \mathbb{I}_{\{\text{RUL}_{k,t} > \tau_t\}} \right],$$

where $C_{\text{transfer}}(A_j, A_k)$ is the transfer cost and $C_{\text{oper}}(A_k)$ the operating cost for the new aircraft.

The mission is reassigned if:

$$C_{\text{shift}} < C_{\text{cont}}(t) + C_{\text{risk}}(t),$$

where $C_{\text{risk}}(t)$ is the expected failure cost of continuing with the original aircraft.

4.4. Cost function and optimization

We define the total expected cost of the mission, $C_{\text{total}}(t)$, to capture the financial implications of continuing or aborting the mission based on the engine's RUL. The cost function is expressed as:

$$C_{\text{total}}(t) = \mathbb{I}_{\{\text{RUL}_t \leq \tau_t\}} \left(C_{\text{abort}}(t) + C_{\text{shift}} \right) + \mathbb{I}_{\{\text{RUL}_t > \tau_t\}} C_{\text{cont}}(t). \tag{29}$$

When $\text{RUL}_t \leq \tau_t$, the mission incurs the costs of aborting $C_{\text{abort}}(t)$ and shifting to another aircraft C_{shift}, reflecting the need to mitigate failure risks. Conversely, if $\text{RUL}_t > \tau_t$, only the cost of continuing the mission $C_{\text{cont}}(t)$ is incurred, representing ongoing operational expenses. By adjusting mission decisions based on the engine's health, this formulation ensures economically optimal outcomes and minimizes overall mission-related costs.

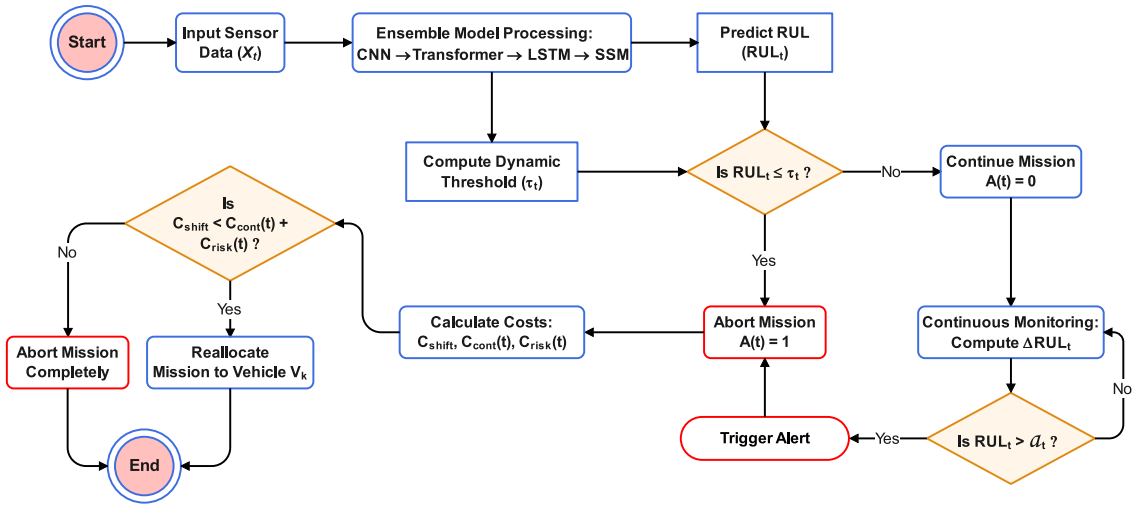


Fig. 8. Dynamic mission abort policy flowchart.

5. Post-mission abortion and aircraft shifting analysis

In mission-critical operations, analyzing the post-abortion phase can improve future cycle assignments, facilitate mission shifts to alternative aircraft, and inform re-engagement strategies. By incorporating maintenance, re-engagement timing, spare aircraft management, and dynamic mission re-shifting, the decision-making framework becomes more resilient and cost-effective.

5.1. Post-abortion maintenance and re-engagement

When an aircraft is aborted due to predicted failure or insufficient RUL, it undergoes maintenance to restore its health. The post-maintenance health state $\mathcal{H}_{\text{maint}}(t)$ evolves as:

$$\mathcal{H}_{\text{maint}}(t) = \mathcal{H}_{\text{init}} - \int_{t_0}^{t} r(s)\, ds + \epsilon(t),$$

where $\mathcal{H}_{\text{init}}$ is the health at abortion, $r(s)$ the recovery rate, and $\epsilon(t)$ captures stochastic delays. The aircraft can be re-engaged once $\mathcal{H}_{\text{maint}}(t) \geq \zeta_{\text{re}}$, where ζ_{re} is the dynamic threshold indicating sufficient recovery to safely rejoin the mission.

5.2. Optimal re-engagement strategy

For re-engaging a previously aborted aircraft to the same mission, we should account for both the risks of re-engagement and the potential benefits of mission completion. We define the re-engagement cost function as:

$$C_{\text{re}}(t) = C_{\text{maint}} + \mathbb{E}\left[C_{\text{fail}}(t) \cdot \mathbb{I}_{\{\mathcal{H}_{\text{maint}}(t) < \zeta_{\text{re}}\}}\right]. \tag{30}$$

Now, the optimal re-engagement time t_{opt} that minimizes costs is:

$$t_{\text{opt}} = \arg\min_{t \geq t_{\text{ready}}} \left\{C_{\text{re}}(t) + C_{\text{missed}}(t)\right\}. \tag{31}$$

Here, t_{ready} is the earliest time when $\mathcal{H}_{\text{maint}}(t) \geq \zeta_{\text{re}}$ and $C_{\text{missed}}(t)$ is the cost of lost mission opportunities during downtime.

5.3. Dynamic aircraft shifting and spare management

When a secondary aircraft replaces the aborted one, decisions about retaining it as the primary asset or assigning it as a spare depend on its utility function:

$$U(A_k) = C_{\text{oper}}(A_k) + (1 - P_{\text{fail}}(A_k, t)),$$

where $P_{\text{fail}}(A_k, t)$ is the failure probability. If $U(A_k) < \delta_{\text{spare}}$, the aircraft is shifted to a spare role to reduce operational load.

5.4. Re-shifting and multi-aircraft optimization

After maintenance, deciding whether to reassign the mission back to the original aircraft or continue with the secondary aircraft depends on the expected benefit of re-shifting:

$$B_{\text{shift}}(t) = \mathbb{E}\left[S_{\text{re}}(t) - S_{\text{cont}}(t)\right],$$

where $S_{\text{re}}(t)$ and $S_{\text{cont}}(t)$ are the success probabilities if re-shifting or continuing, respectively. The re-shift occurs if:

$$B_{\text{shift}}(t) - C_{\text{shift}}(t) > \delta_{\text{shift}},$$

ensuring re-shifting improves mission success and justifies associated costs.

5.5. Cost optimization and final mission analysis

To ensure that all decisions made post-abortion are optimal, the system continuously updates a total cost function $C_{\text{total}}(t)$, which includes all possible costs associated with re-engagement, continued operation, and aircraft re-shifting:

$$C_{\text{total}}(t) = C_{\text{re}}(t) + C_{\text{oper}}(A_k, t) + C_{\text{shift}}(t). \tag{32}$$

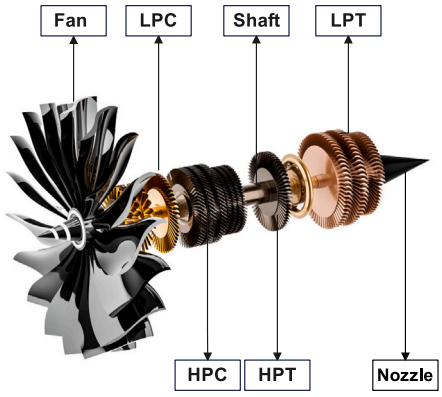
We aim to minimize this cost function to complete the mission assignments in the most optimal and reliable way. Furthermore, it guides managers to make optimal decisions during the post-abortion phase.

6. Experimental analysis

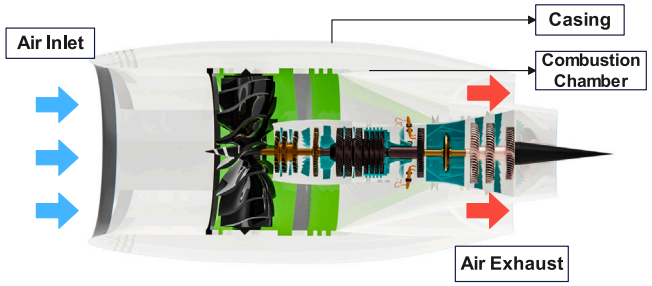
In this section, we evaluate the proposed ensemble model on NASA's C-MAPSS benchmark dataset, focusing on its performance across four subsets (FD001-FD004). We also analyze the proposed mission cycle assignment, dynamic mission abortion policy, and post-abortion maintenance, demonstrating their effectiveness in optimizing operations, minimizing downtime, and reducing costs.

6.1. C-MAPSS dataset overview

For prognostics and RUL prediction, the NASA C-MAPSS dataset is a widely used benchmark. It contains historical run-to-failure turbofan engine data. Each engine operates under three operational settings until failure. The dataset is divided into four subsets: FD001 (single condition, one fault mode), FD002 (multiple conditions, one fault mode), FD003 (one condition, multiple fault modes), and FD004 (multiple conditions and fault modes), adding complexity and realism. In Fig. 9 (a), we illustrate the turbofan engine chassis including the



(a) The Main Chassis of Turbofan Jet Engine



(b) Sideview of Turbofan Jet Engine

Fig. 9. Turbofan rotation section.

fan, Low-Pressure Compressor (LPC), High-Pressure Compressor (HPC), High-Pressure Turbine (HPT), Low-Pressure Turbine (LPT), and nozzle. The HPT extracts energy to drive the HPC, and the LPT extracts energy to drive the fan and LPC, all connected via a rotating shaft. The nozzle directs exhaust gases out of the engine, contributing to thrust. Fig. 9 (b) shows a side view detailing airflow through the inlet, combustion chamber, and exhaust, emphasizing internal rotation [53]. The C-MAPSS dataset, with about 265,256 data points across four subsets, includes training and testing data with true RUL values. It is a key PHM resource enabling development and evaluation of predictive models.

Dataset Division and Model Training. For each C-MAPSS subset (FD001–FD004), the dataset consists of three subfiles: "train_FD001.txt", "test_FD001.txt", and "RUL_FD001.txt" (similarly structured for FD002, FD003, and FD004). Specifically, "train_FD001.txt" and "test_FD001.txt" record various operational and sensor parameters of the turbofan engine as detailed in Table 2, while "RUL_FD001.txt" provides the true RUL corresponding to the engines in "test_FD001.txt". The training data is further divided into 80% for training and 20% for validation to facilitate hyperparameter tuning and ensure robust model performance. During training, we employed the Mean Squared Error (MSE) loss function to optimize the model's predictive accuracy for RUL:

$$\mathcal{L}_{\text{MSE}} = \frac{1}{N}\sum_{i=1}^{N}\left(\text{RUL}_{\text{predicted},i} - \text{RUL}_{\text{true},i}\right)^2, \tag{33}$$

where $\text{RUL}_{\text{true},i}$ denotes the true RUL and $\text{RUL}_{\text{predicted},i}$ is the predicted RUL for the i-th sample. To enhance generalization and mitigate overfitting, an L2 regularization term is incorporated:

$$\mathcal{L} = \mathcal{L}_{\text{MSE}} + \varsigma\sum_{j=1}^{J}\|\theta_j\|^2, \tag{34}$$

where ς is a hyperparameter governing the strength of regularization, θ_j represents the model parameters (e.g., network weights) in the j-th layer, and J is the total number of parameter sets in the model. This combined loss function ensures high predictive accuracy while preventing overfitting, thereby enhancing the model's generalization to unseen data. As summarized in Table 2 and illustrated by the distributions in Fig. 10, each subset's unique conditions and fault modes pose distinct challenges. Despite these complexities, our ensemble model remains robust, achieving low RMSE and S-scores and effectively handles diverse scenarios.

6.1.1. Analysis of nonlinear and nonstationary degradation in the C-MAPSS dataset

To demonstrate nonlinear and nonstationary degradation patterns in the C-MAPSS dataset, we analyze degradation trajectories of multiple FD001 subset engines. Fig. 11 shows normalized degradation metrics over cycles for Engines #15, #27, #46, and #74, highlighting accelerated degradation and increasing variance. The degradation metric, derived from `sensor_11`, is sensitive to engine health. For comparison, readings were normalized to [0, 1] using min–max scaling. As shown in Fig. 11, these engines' degradation processes exhibit nonlinear and nonstationary characteristics:

- **Nonlinear Degradation:** Each engine accelerates degradation after a specific cycle (e.g., cycle 160 for Engine #15), indicating a nonlinear degradation rate. This suggests more aggressive wear mechanisms near failure, consistent with real-world factors like material fatigue and increased friction. Moreover, evidence from [49] supports this behavior, as damage propagation is modeled with exponential fault progression influenced by noise and operational variability, which can produce nonlinear shifts in observed health indices.
- **Nonstationary Variance:** Degradation metric variability increases over time, especially in later operational cycles. This nonstationarity indicates that stochastic fluctuations become more pronounced as engines age, reflecting time-varying degradation dynamics. The C-MAPSS modeling approach incorporates process noise and operational variability, which inherently induce such characteristics, as described in [49].

Accelerated degradation points were identified by detecting significant changes in the local slope of the degradation metric using finite-difference methods and derivative analysis. Similarly, increasing variance regions were marked by monitoring the evolution of the standard deviation of residual fluctuations over consecutive cycles. These observations confirm that the C-MAPSS dataset exhibits nonlinear shifts and nonstationary variance, validating the CNN-Transformer-LSTM-SSM model. The Transformer layer captures temporal dependencies, while the SSM component manages stochastic fluctuations, ensuring accurate and stable RUL predictions. This approach achieves consistently lower RMSE and S-scores, improving prediction accuracy across evolving engine health states.

6.1.2. Performance evaluation metrics: RMSE and S-score

In industrial machinery systems, accurately predicting the RUL can help the PdM to schedule maintenance optimally. Therefore, evaluating the predictions made by the model is important. For this, we consider "two performance metrics," namely the "RMSE" and "S-Score".

Root mean squared error (RMSE). The **RMSE** is a widely used regression metric that measures the average magnitude of prediction errors. It is particularly useful because it gives greater weight to larger errors, making it sensitive to outliers:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\text{RUL}_{\text{predicted},i} - \text{RUL}_{\text{true},i}\right)^2}, \tag{35}$$

where n refers to the total number of predictions. $\text{RUL}_{\text{predicted},i}$ denotes the predicted RUL for the i-th data point, and $\text{RUL}_{\text{true},i}$ is the true RUL for the same i-th data point. The lower RMSE values indicate higher prediction accuracy between predicted and true RUL.

Table 2
Details of the C-MAPSS dataset.

Feature	FD001	FD002	FD003	FD004
Number of Engines (Training)	100	260	100	249
Number of Engines (Test)	100	259	100	248
Operating Conditions	One (Sea Level)	Six	One (Sea Level)	Six
Fault Modes	One (HPC Degradation)	One (HPC Degradation)	Two (HPC Degradation, Fan Degradation)	Two (HPC Degradation, Fan Degradation)
Training Samples	20,631	53,759	24,720	61,249
Test Samples	13,096	33,991	16,596	41,214
Sensor Drift Patterns	Minimal	Varied across conditions	Minimal	Varied across conditions
Operational Challenges	Uniform	Diverse due to multiple conditions	Uniform	Diverse due to multiple conditions
Typical Failure Modes	Compressor degradation	Compressor & turbine faults	Electronics failure, Compressor degradation	Electronics failure, Turbine faults

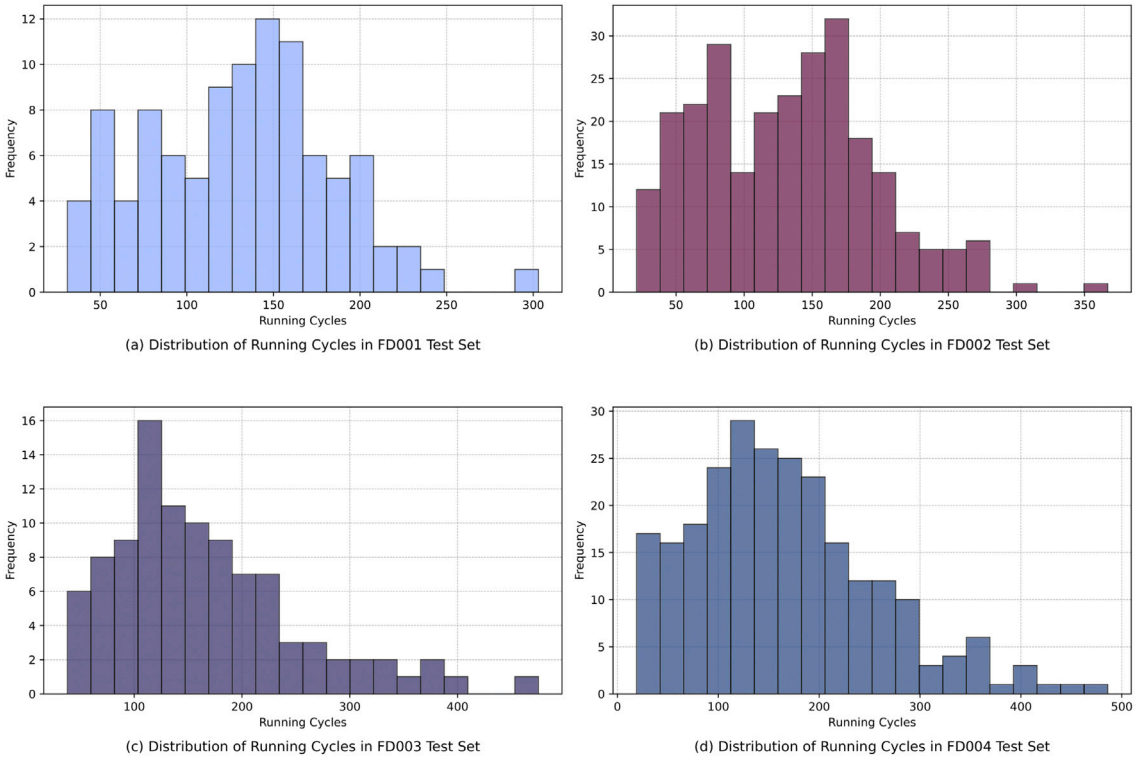


Fig. 10. Distribution of running cycles for subsets (FD001-FD004).

S-score. The **S-score** is a specialized metric designed for predictive maintenance tasks where over-prediction and under-prediction of RUL have different consequences. It penalizes over-prediction more harshly because it can lead to catastrophic failures, while under-prediction, which leads to early maintenance, is less costly.

$$S = \sum_{i=1}^{n} \begin{cases} \exp(-\frac{\text{Error}_i}{13}) - 1 & \text{if Error}_i \geq 0 \\ \exp(\frac{-\text{Error}_i}{10}) - 1 & \text{if Error}_i < 0 \end{cases} \tag{36}$$

Where $\text{Error}_i = \text{RUL}_{\text{predicted},i} - \text{RUL}_{\text{true},i}$.

- **Over-prediction** ($\text{Error}_i \geq 0$): Overestimates of RUL are penalized more harshly, reflecting the critical risk of failures during operation.
- **Under-prediction** ($\text{Error}_i < 0$): The under-prediction of the RUL is less severe. It may waste some of the RUL, leading to some economic loss.

The S-score is a metric that balances these two predictions. That is, lower S-score means the model is more balanced to these predictions, minimizing both premature and late maintenance actions.

Evaluation procedure. The main steps for applying the proposed ensemble model to the CMAPSS dataset are as follows:

1. The ensemble model is trained using the training data contained within each of the subsets (FD001-FD004).
2. The trained model is then applied to the test dataset to estimate the RUL for the engines.
3. Finally, the values $\text{RUL}_{\text{predicted},i}$ and $\text{RUL}_{\text{true},i}$ are substituted in Eqs. (35) and (36) to calculate RMSE and S-score, respectively.

Now, the Table 3 summarizes the optimal hyperparameters for datasets FD001 to FD004.

The Fig. 12 compares the true RUL and predicted RUL across subsets FD001 to FD004, highlighting prediction accuracy and error ranges. The Table 4 compares the RUL prediction performance across different "state-of-the-arts" methods, highlighting the RMSE and S-score for each subset (FD001 to FD004).

6.2. Sensitivity analysis of key parameters

Analyzing the impact of hyperparameters on model performance is important to refine the ensemble model. For this, a sensitivity analysis was conducted on the C-MAPSS FD001 dataset, focusing on window size and learning rate. Using the proposed RLHT algorithm, we varied one parameter while fixing others, identifying hyperparameters that

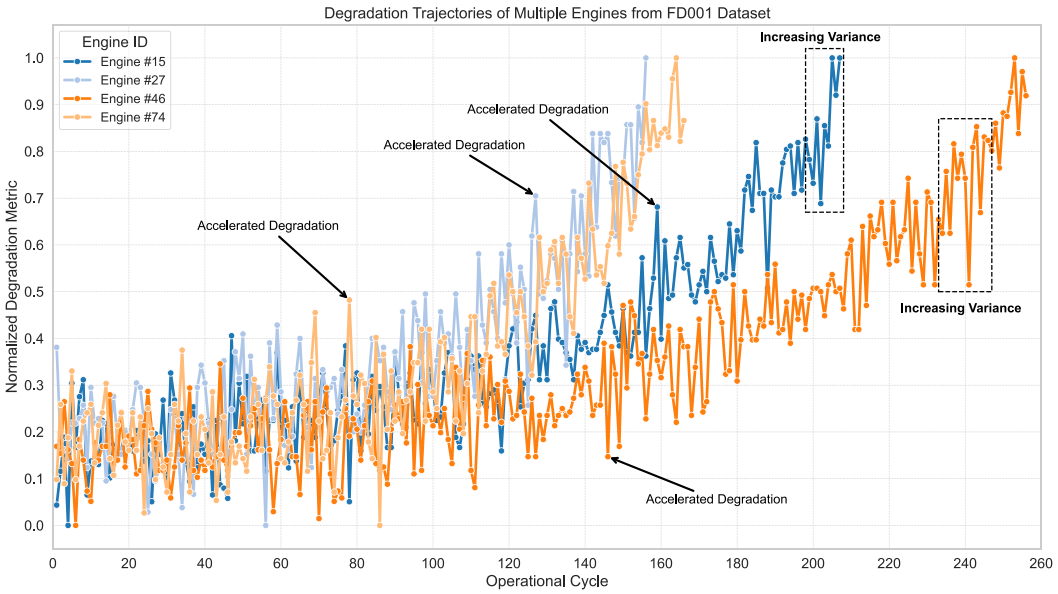


Fig. 11. Degradation trajectories of multiple engines from FD001 dataset, highlighting nonlinear acceleration and nonstationary variance.

Table 3
Optimal hyperparameters used in the model configuration for different datasets.

Hyperparameter	Description	Dataset			
		FD001	FD002	FD003	FD004
Learning Rate	Regulates the magnitude of the steps taken during gradient descent optimization.	1.0852e−05	5.7721e−04	3.9337e−05	4.1333e−04
Batch Size	Number of samples processed before the model is updated.	64	64	64	64
Epochs	Number of times the entire dataset is passed through the model.	65	121	72	114
Dropout Rate	Probability of dropping a neuron during training for regularization.	0.4897	0.2453	0.2796	0.4845
Sequence Length	Length of the input sequence for the model.	30	30	30	30
Window Length	Length of the window used for input data processing.	30	30	30	30
Shift	Number of time steps the window is shifted over the data.	1	1	1	1
Early RUL	Maximum RUL value assigned to engines.	125	125	125	125
CNN Filters (1st Layer)	The total filters used in the first convolutional layer.	114	109	53	119
CNN Filters (2nd Layer)	The total filters used in the second convolutional layer.	155	207	191	110
CNN Filters (3rd Layer)	The total filters used in the third convolutional layer.	123	69	115	75
CNN Kernel Size	Size of the convolutional kernel in each layer.	3, 5, 5	3, 3, 4	6, 5, 4	4, 5, 4
CNN Pool Size	Size of the window for max pooling.	2	2	2	2
CNN Activation Function	Activation function used in CNN layers.	ReLU	ReLU	ReLU	ReLU
LSTM Units (1st Layer)	The total units present in the first LSTM layer.	100	126	100	66
LSTM Units (2nd Layer)	The total units present in the second LSTM layer.	105	37	70	80
LSTM Activation Function	Activation function used in LSTM layers.	ReLU	ReLU	ReLU	ReLU
Dense Layer Activation Function	Activation function used in Dense layers.	ReLU	ReLU	ReLU	ReLU
Output Layer Activation Function	The activation function utilized in the output layer of the model.	Linear	Linear	Linear	Linear
Optimizer	Algorithm used for updating the weights.	Adam	Adam	Adam	Adam
L2 Regularization	Regularization applied to the kernel weights matrix.	9.1007e−06	2.7217e−06	1.3918e−05	1.9915e−06
Early Stopping Patience	The count of epochs without enhancement after which training process is stopped.	10	10	10	10
Reduce LR Patience	The count of epochs without enhancement before reducing the learning rate.	5	5	5	5
Reduce LR Factor	Factor by which the learning rate is reduced.	0.5	0.5	0.5	0.5
Minimum Learning Rate	Minimum value to which the learning rate can be reduced.	1e−6	1e−6	1e−6	1e-6

minimize RMSE and S-score.

1. Window size for data segmentation. The window size w represents the "number of past time steps" used for prediction. We tested window sizes $w \in \{20, 30, 50, 100\}$ and their effects on RMSE and S-score. A window size of $w = 30$ achieved the lowest RMSE (10.67) and S-score (192.14), effectively capturing temporal patterns without excessive noise. Larger sizes, like $w = 100$, increased RMSE, while smaller sizes, such as $w = 20$, also degraded performance (RMSE: 11.95) due to missing critical patterns. Table 5 shows the sensitivity results for FD001.

2. Learning rate. The learning rate η controls the step size during "gradient descent," affecting convergence speed and accuracy. We tested learning rates $\eta \in \{1 \times 10^{-6}, 1.0852 \times 10^{-5}, 1 \times 10^{-4}, 1 \times 10^{-3}\}$ and measured their effects on RMSE, S-score, and convergence time (epochs). The optimal learning rate from the RLHT model was $\eta = 1.0852 \times 10^{-5}$, balancing accuracy and convergence speed.Table 6 presents the sensitivity analysis for FD001.

Using the RLHT model, we identified that a window size of 30 and a learning rate of 1.0852×10^{-5} significantly improved accuracy (RMSE = 10.67) and balanced the S-score for the FD001 dataset. This tuning enhances the model's precision and stability, ensuring reliable

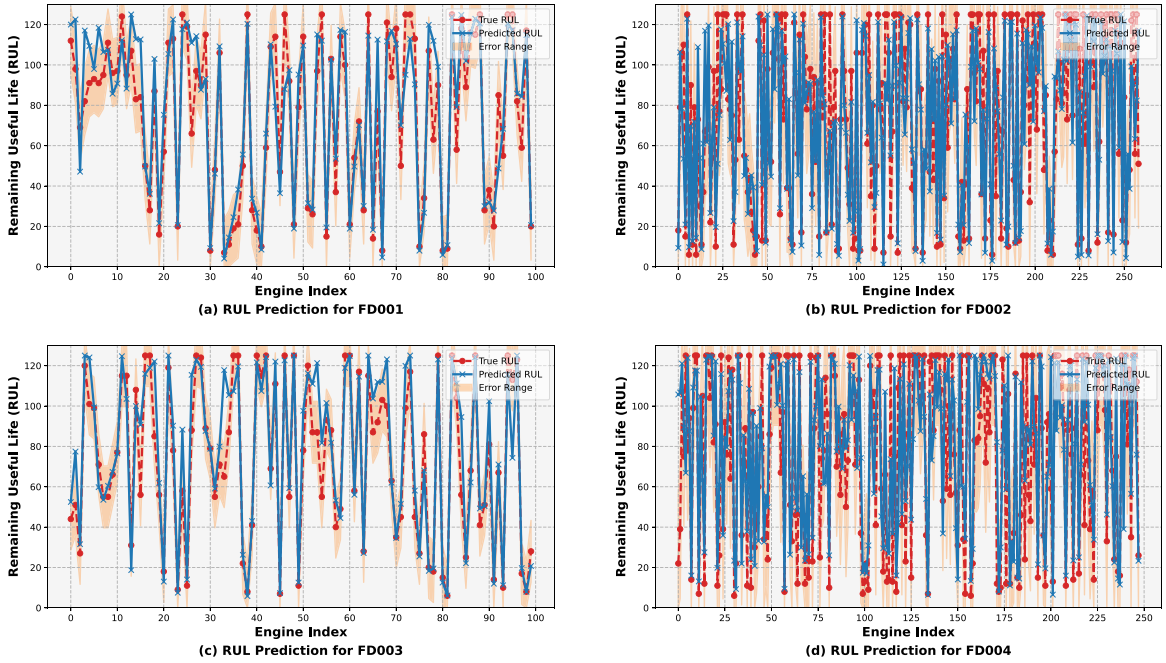


Fig. 12. Comparison of true RUL and predicted RUL for subsets (FD001-FD004).

Table 4
Comparison of RUL prediction performance.

Method	FD001		FD002		FD003		FD004	
	RMSE	S-score	RMSE	S-score	RMSE	S-score	RMSE	S-score
VLSTM [54]	15.23	250.00	22.87	4532.00	14.53	1523.00	26.11	5627.00
Bi-LSTM-ED [55]	12.45	220.07	27.00	3099.09	17.48	574.00	23.49	3022.00
Bi-LSTM [56]	13.65	295.00	23.18	4130.00	17.14	317.00	24.06	5430.00
DCNN [6]	12.63	411.42	25.53	5755.00	13.10	456.00	23.34	6300.00
BiGRU-TSAM [19]	12.56	213.35	18.94	2246.13	14.45	232.86	23.87	3610.34
AGCNN [57]	12.42	225.61	19.38	2412.00	13.30	397.29	22.58	3392.00
BDL [58]	12.19	267.21	18.49	2007.81	16.07	409.99	19.41	2415.71
IMSDSSN [21]	12.14	261.01	17.40	1775.15	12.57	322.49	19.78	2581.83
SCTA-LSTM [22]	12.10	207.00	17.40	1775.00	12.84	248.00	19.41	3100.00
CNN-Bi-LSTM-AM-BO [32]	11.43	201.26	15.69	1214.47	11.28	181.99	18.35	2627.11
Transformer-based Method + DiffRUL [59]	11.71	199.29	15.90	1162.84	11.77	240.37	18.43	1627.37
PVA-FFG Improved Transformer [29]	11.36	173.89	13.24	**714.98**	11.80	215.32	15.16	1049.53
DS-STFN [60]	10.92	161.35	13.77	946.34	**10.01**	150.42	15.53	1079.91
The Proposed Ensemble Model	**10.67**	**157.14**	**12.35**	1201.05	10.32	**147.41**	**14.71**	**1031.86**

Table 5
Sensitivity analysis for window size.

Window size (w)	RMSE (FD001)	S-score (FD001)
20	11.95	215.78
30	**10.67**	**192.14**
50	11.12	198.76
100	12.21	225.43

predictions under various conditions.

6.3. Ablation study

We conducted an ablation study on our CNN-Transformer-LSTM-SSM ensemble model by systematically removing key modules to evaluate performance. This study aimed to quantify the impact of the Transformer, LSTM, and SSM on accurately predicting RUL. The goal is to understand how each module contributes to the overall performance and to assess the trade-offs when certain components are excluded.

Fig. 13 shows the Ablation Study Cases for the CNN-Transformer-LSTM-SSM model. Subfigures (Fig. 13(a)–(f)) display different configurations: (a) complete model, (b) without Transformer, (c) without SSM, (d) without Transformer and SSM, (e) without LSTM, and (f) without LSTM and SSM. These configurations illustrate performance changes when key components are removed. All configurations were evaluated on the "C-MAPSS dataset" using RMSE and S-score.

In Table 7, we present the results of the performed ablation study, with key observations as follows:

- **Without Transformer**: RMSE increased by 15.15% on average, with a 20.6% increase in FD002 and 10.8% in FD004. This highlights the Transformer's self-attention mechanism in capturing temporal dependencies.
- **Without SSM**: RMSE increased by 7.9%, highest in FD003 (8.8%) and least in FD004 (2.7%). This shows the SSM's role in stabilizing performance and capturing nonlinear degradation.
- **Without Transformer and SSM**: RMSE increased by 24.95% across datasets. Without these components, the model relies solely

Table 6
Sensitivity analysis for learning rate.

Learning rate (η)	RMSE (FD001)	S-score (FD001)	Convergence time (Epochs)
1×10^{-6}	12.43	205.34	65
1.0852×10^{-5}	**10.67**	**192.14**	50
1×10^{-4}	11.82	207.43	40
1×10^{-3}	14.98	365.87	20

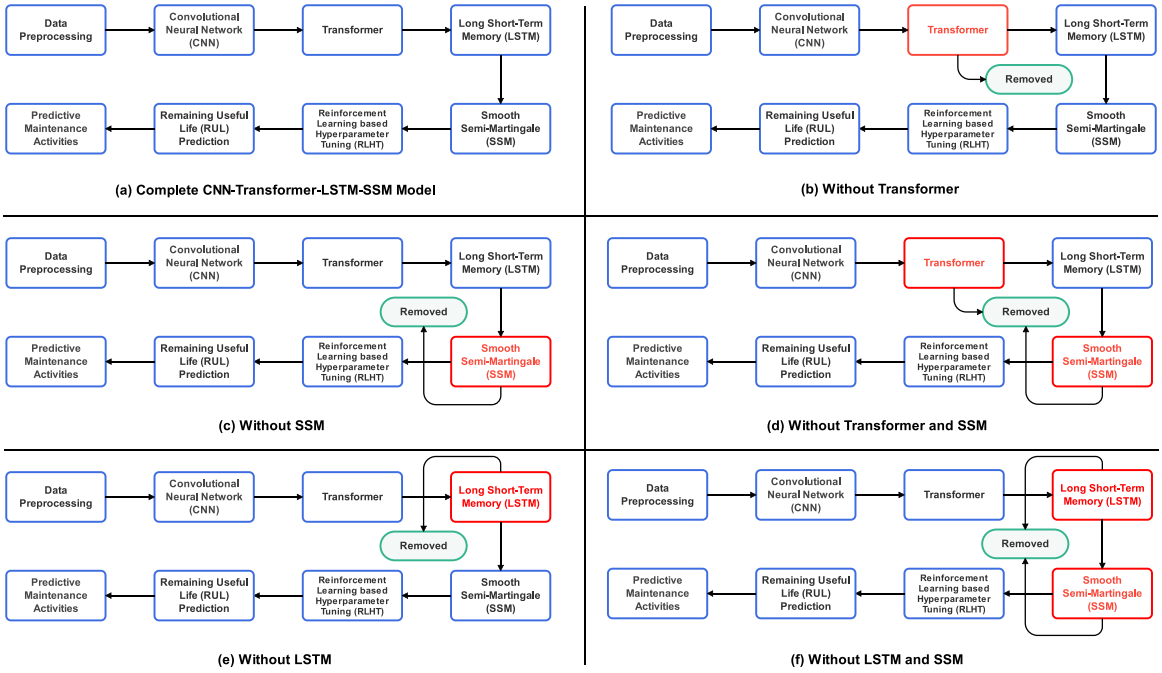


Fig. 13. Ablation study cases.

Table 7
Ablation study results: RMSE and S-score across different configurations.

Model	FD001		FD002		FD003		FD004	
	RMSE	S-score	RMSE	S-score	RMSE	S-score	RMSE	S-score
Full Model (CNN-Transformer-LSTM-SSM)	10.67	157.14	12.35	1201.05	10.32	147.41	14.71	1031.86
Without Transformer	12.10	225.00	14.89	1805.42	11.95	210.23	16.30	1240.35
Without SSM	11.50	194.65	13.87	1452.11	11.23	187.89	15.10	1123.52
Without Transformer and SSM	13.45	290.35	16.45	2007.91	12.50	232.45	17.58	1350.84
Without LSTM	12.85	210.87	14.34	1672.18	11.56	201.22	15.79	1182.79
Without LSTM and SSM	13.50	275.45	15.67	1897.50	12.13	218.54	16.90	1312.67

on the CNN-LSTM framework, neglecting critical temporal and stochastic information, leading to a substantial performance drop.

- **Without LSTM**: RMSE increased by 14.0% on average, indicating the LSTM's importance in modeling sequential dependencies and enhancing the model's ability to predict the RUL. The absence of LSTM results in worse performance, particularly in FD004, where temporal patterns are most critical.
- **Without LSTM and SSM**: RMSE increased by 21.5% on average. The absence of both LSTM and SSM leads to the exclusion of both sequential temporal learning and nonlinear degradation modeling, significantly deteriorating the model's predictive capability.

This ablation study shows the contribution of Transformer, LSTM, and SSM components in improving model performance and reducing the RMSE and S-score.

6.4. Mission cycles: Analysis of complex multi-segment assignments under multiple scenarios

We consider three increasingly complex missions $\mathcal{M} = \{M_1, M_2, M_3\}$. Mission M_1, M_2, and M_3 have 2, 3, and 5 segments, respectively. Here, $M_{j,\ell}$ denotes the ℓ-th segment of mission M_j. Each segment has a time window, baseline RUL requirement, thrust or certification needs, and environmental conditions. We use four turbofan engines from the NASA C-MAPSS FD001 dataset (Engines #15, #27, #46, and #74), each with scenario-dependent RUL, HPC/HPT limits, certification status, and spare parts requirements. Additionally, three scenarios, $\Omega = \{\omega_1, \omega_2, \omega_3\}$, represent nominal, dusty, and severe conditions.

6.4.1. Missions and segments

Table 8 details the missions and segments. Each segment specifies a time window, baseline RUL requirement $D_{j,\ell}$, thrust requirements (normal/high-thrust), and certification or environmental notes. High-thrust segments and those operating under dusty or severe conditions (ω_2, ω_3) require additional RUL and HPC/HPT consumption. Additionally, the N1 certification in the table ensures engines meet performance and safety standards for LPC operations, guaranteeing reliable

Table 8
Missions and segments with baseline requirements.

Mission segment	Time window $[t_{j,\ell}^{\text{start}}, t_{j,\ell}^{\text{end}}]$	$D_{j,\ell}$ (Cycles)	Thrust requirement	Notes
$M_{1,1}$	[0h, 2h]	40	High-Thrust	Normal conditions
$M_{1,2}$	[2h, 5h]	35	Normal	Requires N1 certification
$M_{2,1}$	[0h, 3h]	25	Normal	Normal conditions
$M_{2,2}$	[3h, 6h]	50	High-Thrust	Dusty in ω_2, severe in ω_3
$M_{2,3}$	[6h, 9h]	30	Normal	Normal conditions
$M_{3,1}$	[0h, 1h]	20	Normal	Normal conditions
$M_{3,2}$	[1h, 3h]	45	High-Thrust	High wear rate
$M_{3,3}$	[3h, 6h]	35	Normal	Dusty in ω_2, severe in ω_3
$M_{3,4}$	[6h, 8h]	55	High-Thrust	N1 certification required
$M_{3,5}$	[8h, 10h]	40	Normal	Normal conditions

Table 9
Engine RUL, certifications, and spare parts under scenarios.

Engine ID	RUL (ω_1)	RUL (ω_2)	RUL (ω_3)	$\mathbb{L}_i^{\omega_2}/\mathbb{L}_i^{\omega_3}$	N1 certified	Spare parts s_i
Engine #15	85	70	60	120/90	Yes	$s_{15} = 1$
Engine #27	90	65	55	100/80	No	$s_{27} = 1$
Engine #46	80	60	50	110/75	Yes	$s_{46} = 2$
Engine #74	95	72	58	95/70	Yes	$s_{74} = 1$

Table 10
Dynamic failure thresholds for engines under different scenarios.

Engine ID	Mission(s)	Initial threshold (τ_0)	Failure threshold (ζ') under three scenarios			Rationale
			ω_1 ($\delta = 0.0752$)	ω_2 ($\delta = 0.0953$)	ω_3 ($\delta = 0.1264$)	
Engine #15	$M_{1,1}, M_{3,2}$	90	67	61	54	Reflects dynamic degradation over 4h mission with mild to severe conditions.
Engine #27	$M_{2,1}, M_{2,2}, M_{2,3}$	115	58	49	37	Accounts for 9h mission duration with moderate wear to accelerated degradation.
Engine #46	$M_{1,2}, M_{3,3}, M_{3,4}$	135	74	63	49	High thresholds adjusted for 8h mission with stricter RUL requirements.
Engine #74	$M_{2,2}$	55	44	41	38	Short 3h mission with rapid degradation across conditions.

performance during high-thrust missions.

6.4.2. Engines and scenario effects

The Table 9 summarizes the details of four turbofan engines under nominal (ω_1), dusty (ω_2), and severe (ω_3) conditions. Higher environmental severity reduces RUL, reflecting faster degradation (see Fig. 14(a)). N1-certified engines reliably handle high-thrust segments. Spare parts (s_i, e.g., $s_{46} = 2$ indicates that Engine #46 has two spare parts available) and HPC/HPT limits are critical for maintenance feasibility and engine reassignment across mission segments.

Engine failure threshold. The failure threshold (ζ') for each engine is established using industry reliability standards and empirical analysis of the NASA C-MAPSS FD001 dataset. Examining engines' historical run-to-failure patterns and operational limits, we identify critical RUL values indicating the minimum remaining cycles to safely complete assigned segments. When an engine's predicted RUL falls below ζ', it is deemed unsafe to continue. Thus, ζ' guides maintenance scheduling, mission abort decisions, and engine re-engagement strategies, ensuring alignment with safety constraints and maintenance protocols. The specific failure thresholds for each engine are detailed in Table 10.

The initial threshold (τ_0) is the sum of the cumulative RUL of assigned mission segments and a safety margin, accounting for uncertainties, wear, and unforeseen stress. For instance, for Engine #15's missions ($M_{1,1}, M_{3,2}$), we calculate a combined RUL of 85 cycles and, adding a safety margin of 5 cycles, the initial threshold becomes 90 cycles.

This threshold is dynamically adjusted using degradation rate (δ) based on environmental conditions. From empirical testing, historical data, dust levels, temperature extremes, and operational loads, we set $\delta = 0.0752$ for nominal (ω_1), $\delta = 0.0953$ for dusty (ω_2), and $\delta = 0.1264$ for severe (ω_3). Using $\tau_t = \tau_0 \cdot \exp(-\delta t)$, where t is mission duration, we compute failure thresholds (ζ') dynamically. For instance, Engine #27's thresholds are 58, 49, and 37 cycles under ω_1, ω_2, and ω_3.

Optimal engine-to-segment assignments. We implement a two-stage stochastic optimization model using a Mixed-Integer Linear Programming (MILP) solver to minimize expected costs across scenarios. The objective (Eq. (14)) includes maintenance, fuel, and failure penalties, with constraints (Eq. (15) to Eq. (20)) ensuring feasibility. Binary variables $x_{i,j,\ell}$ assign engines to segments, and continuous variables y_j^{ω} capture scenario-specific outcomes. RUL updates dynamically, modeling degradation under varying conditions. Certification and resource constraints ensure feasible scheduling. Scenario probabilities $\mathbb{P}(\omega)$ ensure robust solutions. The optimization results, presented in Fig. 14(b), showing optimal engine assignments to mission segments.

6.4.3. Scenario increments for high-thrust and harsh conditions

Scenario increments for high-thrust segments are summarized in Table 11. Under ω_1, no increments apply. Under ω_2, each high-thrust segment adds +5 RUL cycles and consumes an additional 10 HPC/HPT cycles. Under ω_3, these increments are doubled, and maintenance resources are limited.

Scheduled maintenance actions. Table 12 highlights maintenance scheduling actions under varying conditions, derived using the scheduling algorithm (Algorithm 4).

6.4.4. Cost analysis

The optimization framework aims to minimize the expected total cost, expressed in currency units, which includes maintenance, fuel consumption, and failure penalties. Table 13 provides a detailed cost breakdown across different scenarios.

As scenarios intensify to ω_3, both maintenance and failure penalties escalate, underscoring increased operational challenges. The optimization effectively prioritizes maintenance resources to mitigate these costs, ensuring mission-critical segments remain operational. For example, Engine #15 is consistently maintained after $M_{1,1}$ across all scenarios, preserving its RUL for high-thrust operations.

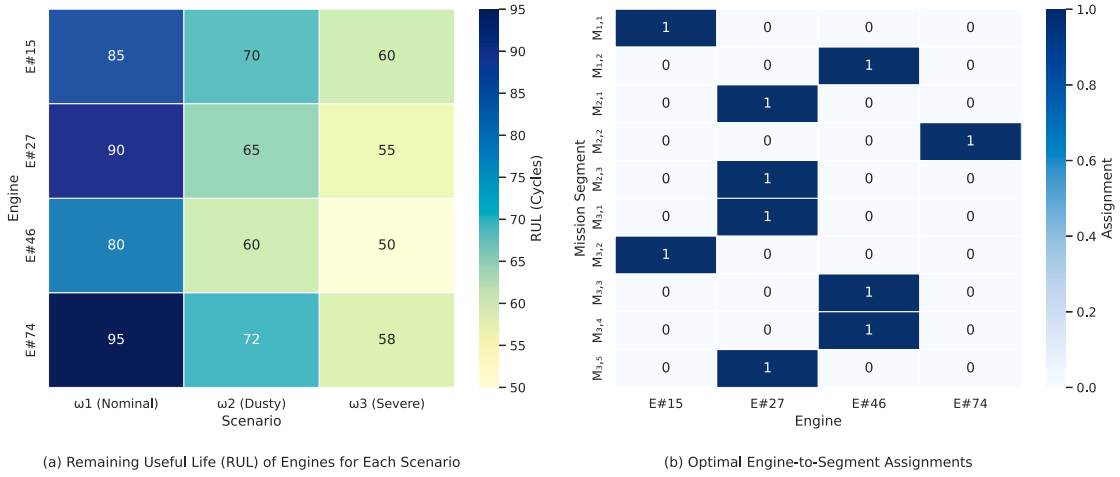


Fig. 14. Heatmaps for RUL of four engines across three scenarios and optimal engine-to-mission assignments.

Table 11
Scenario increments for high-thrust and harsh conditions.

Scenario	Additional RUL	HPC/HPT per High-Thrust	Maintenance constraints
ω_1	+0 cycles	+0 HPC/HPT	Full crew/spares
ω_2	+5 cycles	+10 HPC/HPT	Normal crew/spares
ω_3	+10 cycles	+15 HPC/HPT	Half crew, delayed spares

Table 12
Scheduled maintenance actions by scenario.

Engine ID	Maint. under ω_1/ω_2	Maint. under ω_3	Notes
E#15	Yes, after $M_{1,1}$	Yes, earlier for $M_{3,2}$	Ensures RUL for high-thrust segments
E#27	No or minimal	Maybe partial if M_3 reduced	Handles normal wear segments
E#46	Yes before $M_{1,2}$	Challenging under ω_3	Critical for $M_{3,4}$
E#74	Possibly after $M_{2,2}$	Difficult under ω_3	Adjusts to spares availability

Table 13
Cost breakdown by scenario.

Scenario	Maintenance cost	Fuel cost	Failure penalty	Total cost
ω_1	10,000	50,000	5,000	65,000
ω_2	12,000	55,000	4,000	71,000
ω_3	15,000	60,000	10,000	85,000

6.5. Dynamic mission abort policy: A real-time strategy for preventing failures

The key objective of the dynamic mission abort policy compared to static abort policy is that it uses real-time RUL predictions to abort missions when necessary. It helps in terminating catastrophic in-flight engine failures during missions.

6.5.1. Decision rule for abort policy

If the predicted RUL of an engine is less than the remaining cycles required for the current and upcoming mission segments, aborting the mission is optimal. Consider mission M_2, which consists of three segments $(M_{2,1}, M_{2,2}, M_{2,3})$ requiring a total of $25+50+30 = 105$ cycles. After 40 cycles, suppose mission M_2 completed $M_{2,1}$ (25 cycles) and partially progressed 15 cycles into $M_{2,2}$, leaving 35 cycles of $M_{2,2}$ plus all 30 cycles of $M_{2,3}$. Thus, 65 cycles remain for mission completion.

$$A(t) = \begin{cases} 1 & \text{if RUL}(t) < T_{\text{rem}} \\ 0 & \text{otherwise} \end{cases}$$

For example, let Engine #27 (with a current predicted RUL of 30 cycles) be operating on mission M_2 at $t = 40$ cycles. Since 65 cycles are still needed to complete the remaining segments ($M_{2,2}$ and $M_{2,3}$), and $30 < 65$, aborting the mission prevents potential engine failure.

The Table 14 details the predicted RUL, remaining mission segment cycles, shifting and aborting costs, and total expected costs (calculated using Eq. (29)) for each mission decision point. The analysis reveals that aborting missions dynamically when the RUL is insufficient results in a significantly lower total expected cost (15,000) compared to continuing the mission (32,500).

6.6. Post-mission abort analysis and maintenance: Optimizing post-abortion recovery

After aborting Engine #27 from mission M_2, immediate maintenance can restore engine health. Let ζ_{re} be the threshold for re-engagement. If $\mathcal{H}_{\text{maint}}(t) \geq \zeta_{\text{re}}$, the engine is ready to resume operations on subsequent mission segments.

$$\text{Re-engage}(t) = \begin{cases} 1 & \text{if } \mathcal{H}_{\text{maint}}(t) \geq \zeta_{\text{re}} \\ 0 & \text{otherwise} \end{cases}$$

6.6.1. Mission shifting and re-engagement: Ensuring continuity after aborts

If a mission aborts mid-segment, shifting the remaining segments to another engine with adequate RUL can prevent mission failure costs.

Table 14
Dynamic mission abort policy with integrated cost analysis.

Time (Cycles)	Predicted RUL (Cycles)	Remaining segment cycles	Abort decision	C_{transfer}	C_{abort}	C_{total}
40	30	65	Abort	10,000	5,000	15,000
50	40	55	Continue	–	–	32,500
60	35	45	Continue	–	–	32,500
70	20	35	Abort	10,000	5,000	15,000
80	45	25	Continue	–	–	32,500
90	25	15	Continue	–	–	32,500

Table 15
Remaining engines for re-engagement and their RULs.

Engine ID	Predicted RUL (Cycles)
Engine #15	85
Engine #46	80
Engine #74	95

Strategy for mission shifting. Let $T_{\text{rem}}(M_j)$ be the cycles needed to complete the remaining segments of mission M_j. Another engine E_i can take over if $\text{RUL}_i \geq T_{\text{rem}}(M_j)$.

$$E_i = \arg\max_i \left(\text{RUL}_i \text{ subject to } \text{RUL}_i \geq T_{\text{rem}}(M_j)\right).$$

Shifting missions after abortions. Assume Engine #27 aborts mission M_2 after 40 cycles, leaving 65 cycles of segments remaining. The Table 15 lists the available engines for re-engagement and their predicted RULs.

Engine #74, with 95 RUL cycles, is chosen to complete the remaining mission segments (65 cycles required), ensuring mission continuity. For Engine #74 re-engaging, assume a scheduled maintenance cost of 2,200 and an additional fuel cost of 1,600, totaling:

$$C_{\text{re}} = 2,200 + 1,600 = 3,800.$$

By selecting engines with sufficient RUL and managing re-engagement costs, the overall strategy remains cost-effective, ensuring successful completion of mission segments despite aborts.

Maintenance and re-engagement of engine #27. Setting $\zeta_{\text{re}} = 0.80$, Table 16 shows that after 20 maintenance cycles, Engine #27's health recovers to 0.85, allowing it to re-engage in mission segments if cost-effective. Further, the costs are derived using Eqs. (30) and (32). At lower maintenance cycles (e.g., 10 or 15), no re-engagement occurs, prompting mission shifting and thus higher C_{total}. As maintenance cycles increase and re-engagement becomes feasible, shifting costs are avoided, reducing the overall C_{total}.

The implementation of dynamic mission abort policies, driven by real-time RUL predictions, significantly enhances operational reliability and safety. By proactively aborting missions when engine health is compromised, the strategy prevents catastrophic failures, ensuring mission success and safeguarding critical assets.

Collectively, these experimental results validate the proposed SSM-based ensemble deep learning model's effectiveness in accurately predicting RUL across varied and complex operational scenarios. The reinforcement learning-based hyperparameter tuning enhances model performance and stability, while dynamic mission abort policies and mission shifting strategies demonstrate practical applications of RUL predictions in optimizing operational reliability and reducing maintenance costs. Additionally, the ensemble model effectively informs mission optimization and maintenance strategies, enhancing operational efficiency, reducing costs, and ensuring higher reliability in predictive maintenance applications. Together, these interconnected innovations highlight the potential of our integrated approach to advance predictive maintenance practices in mission-critical systems, offering substantial advancements to the field of prognostics and health management.

7. Conclusion

In this study, we introduced a hybrid ensemble model that combines CNNs, Transformers, LSTMs, and a Smooth Semi-Martingale (SSM) stochastic layer to improve the prediction of Remaining Useful Life (RUL) in industrial equipment. By modeling both deterministic and stochastic degradation processes, our approach addresses challenges posed by nonlinear, nonstationary, and stochastic failures in complex systems. We further enhanced the model's performance and reduced training time by introducing a new Reinforcement Learning-Based Hyperparameter Tuning (RLHT) method, which adapts to the training process in real-time. Moreover, based on the proposed ensemble model a new predictive maintenance scheduling algorithm has been developed which integrates our model's outputs into maintenance decision-making processes. This algorithm schedules optimal maintenance by considering factors such as maintenance costs, failure risks, and operational thresholds, thereby enhancing operational continuity, safety, and cost-effectiveness. Additionally, we introduced a framework for optimizing multi-segment mission cycle assignments and resource management, integrating RUL predictions into multi-stage optimization for dynamic engine allocation, mission adjustments, and maintenance scheduling to minimize costs and mitigate failure risks under complex constraints. We also developed a dynamic mission abort policy informed by the predicted RUL, enabling real-time decision-making in mission-critical operations through strategies like mission shifting, re-engagement, and post-abortion analysis, which are crucial for ensuring operational continuity, safety, and cost-effectiveness. Validation on the NASA C-MAPSS dataset demonstrates that our proposed model performs better compared to existing state-of-the-art methods, achieving lower RMSE and S-score values across all subsets (FD001–FD004), with RMSE values of 10.67, 12.35, 10.32, and 14.71, respectively. Sensitivity analyses and ablation studies confirm the stability and effectiveness of the proposed framework, highlighting the critical role of important components within the ensemble. By effectively capturing complex degradation patterns through deep learning and stochastic modeling, the proposed model provides more accurate and reliable RUL predictions. The integration of dynamic decision-making policies enhances its utility in mission-critical applications, potentially contributing to increased safety, reduced operational costs, and improved efficiency in industrial operations.

Despite the promising results, the integration of multiple deep learning architectures and the SSM layer increases computational complexity, which may limit real-time deployment in resource-constrained settings. Future work should focus on optimizing the model to reduce computational overhead without compromising accuracy. Expanding the framework to accommodate multi-component systems and exploring alternative stochastic modeling techniques could enhance its applicability and robustness. Implementing the dynamic mission abort policy in real-world industrial environments would provide valuable insights for refining the strategy and assessing its impact on operational decision-making.

CRediT authorship contribution statement

Faizanbasha A.: Writing – review & editing, Writing – original draft, Visualization, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **U. Rizwan:** Writing – review & editing, Visualization, Validation, Supervision, Resources, Methodology, Data curation.

Table 16
Post-abortion maintenance, re-engagement, and cost analysis for engine #27 ($\zeta_{re} = 0.80$).

Time after abort (Cycles)	Engine state $\mathcal{H}_{maint}(t)$	Re-engagement decision	C_{re}	C_{total}
10	0.75	No	2,000	8,800
15	0.78	No	2,150	8,950
20	0.85	Yes	2,200	5,200
25	0.90	Yes	2,350	5,350

Funding

This research was funded by "*University Grants Commission, Bahadurshah Zafar Marg, New Delhi*" under the scheme of *CSIR-UGC JRF* with ref no: 211610000578. The first author thanks "*Department of Higher Education, Ministry of Education, Govt. of India*" for the financial assistance received during this research work.

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.

Acknowledgment

The authors express their sincere gratitude to the reviewers and editors for their time and consideration in evaluating this manuscript, contributing to the advancement of scholarly knowledge in this field.

Data availability

Data will be made available on request.

References

[1] Zio E. Prognostics and health management (PHM): Where are we and where do we (need to) go in theory and practice. Reliab Eng Syst Saf 2022;218:108119. http://dx.doi.org/10.1016/j.ress.2021.108119.
[2] Wu F, Wu Q, Tan Y, Xu X. Remaining useful life prediction based on deep learning: A survey. Sensors 2024;24:3454. http://dx.doi.org/10.3390/s24113454.
[3] Taşcı B, Omar A, Ayvaz S. Remaining useful lifetime prediction for predictive maintenance in manufacturing. Comput Ind Eng 2023;184:109566. http://dx.doi.org/10.1016/j.cie.2023.109566.
[4] Zhan Y, Kong Z, Wang Z, Jin X, Xu Z. Remaining useful life prediction with uncertainty quantification based on multi-distribution fusion structure. Reliab Eng Syst Saf 2024;251:110383. http://dx.doi.org/10.1016/j.ress.2024.110383.
[5] Zhang Q, Liu Q, Ye Q. An attention-based temporal convolutional network method for predicting remaining useful life of aero-engine. Eng Appl Artif Intell 2024;127:107241. http://dx.doi.org/10.1016/j.engappai.2023.107241.
[6] Li X, Ding Q, Sun J-Q. Remaining useful life estimation in prognostics using deep convolution neural networks. Reliab Eng Syst Saf 2018;172:1–11. http://dx.doi.org/10.1016/j.ress.2017.11.021.
[7] Ferreira C, Gonçalves G. Remaining useful life prediction and challenges: A literature review on the use of machine learning methods. J Manuf Syst 2022;63:550–62. http://dx.doi.org/10.1016/j.jmsy.2022.05.010.
[8] Faizanbasha A, Rizwan U. Optimal age replacement time for coherent systems under geometric point process. Comput Ind Eng 2024;190:110047. http://dx.doi.org/10.1016/j.cie.2024.110047.
[9] Faizanbasha A, Rizwan U. Optimizing replacement times and total expected discounted costs in coherent systems using geometric point process. Comput Ind Eng 2025;201:110879. http://dx.doi.org/10.1016/j.cie.2025.110879.
[10] Faizanbasha A, Rizwan U. Optimizing burn-in and predictive maintenance for enhanced reliability in manufacturing systems: A two-unit series system approach. J Manuf Syst 2025;78:244–70. http://dx.doi.org/10.1016/j.jmsy.2024.12.002.
[11] Lee J, Mitici M. Deep reinforcement learning for predictive aircraft maintenance using probabilistic remaining-useful-life prognostics. Reliab Eng Syst Saf 2023;230:108908. http://dx.doi.org/10.1016/j.ress.2022.108908.
[12] Ochella S, Shafiee M, Dinmohammadi F. Artificial intelligence in prognostics and health management of engineering systems. Eng Appl Artif Intell 2022;108:104552. http://dx.doi.org/10.1016/j.engappai.2021.104552.
[13] Wen P, Li Y, Chen S, Zhao S. Remaining useful life prediction of IIoT-enabled complex industrial systems with hybrid fusion of multiple information sources. IEEE Internet Things J 2021;8(11):9045–58. http://dx.doi.org/10.1109/JIOT.2021.3055977.
[14] Nguyen KT, Medjaher K. A new dynamic predictive maintenance framework using deep learning for failure prognostics. Reliab Eng Syst Saf 2019;188:251–62. http://dx.doi.org/10.1016/j.ress.2019.03.018.
[15] Mitici M, de Pater I, Barros A, Zeng Z. Dynamic predictive maintenance for multiple components using data-driven probabilistic RUL prognostics: The case of turbofan engines. Reliab Eng Syst Saf 2023;234:109199. http://dx.doi.org/10.1016/j.ress.2023.109199.
[16] Zeng J, Liang Z. A dynamic predictive maintenance approach using probabilistic deep learning for a fleet of multi-component systems. Reliab Eng Syst Saf 2023;238:109456. http://dx.doi.org/10.1016/j.ress.2023.109456.
[17] Song Y, Gao S, Li Y, Jia L, Li Q, Pang F. Distributed attention-based temporal convolutional network for remaining useful life prediction. IEEE Internet Things J 2021;8(12):9594–602. http://dx.doi.org/10.1109/JIOT.2020.3004452.
[18] Xiang S, Qin Y, Luo J, Pu H, Tang B. Multicellular LSTM-based deep learning model for aero-engine remaining useful life prediction. Reliab Eng Syst Saf 2021;216:107927. http://dx.doi.org/10.1016/j.ress.2021.107927.
[19] Zhang J, Jiang Y, Wu S, Li X, Luo H, Yin S. Prediction of remaining useful life based on bidirectional gated recurrent unit with temporal self-attention mechanism. Reliab Eng Syst Saf 2022;221:108297. http://dx.doi.org/10.1016/j.ress.2021.108297.
[20] Yang L, Liao Y, Duan R, Kang T, Xue J. A bidirectional recursive gated dual attention unit based RUL prediction approach. Eng Appl Artif Intell 2023;120:105885. http://dx.doi.org/10.1016/j.engappai.2023.105885.
[21] Zhang J, Li X, Tian J, Luo H, Yin S. An integrated multi-head dual sparse self-attention network for remaining useful life prediction. Reliab Eng Syst Saf 2023;233:109096. http://dx.doi.org/10.1016/j.ress.2023.109096.
[22] Tian H, Yang L, Ju B. Spatial correlation and temporal attention-based LSTM for remaining useful life prediction of turbofan engine. Measurement 2023;214:112816. http://dx.doi.org/10.1016/j.measurement.2023.112816.
[23] Zhang Z, Song W, Li Q. Dual-aspect self-attention based on transformer for remaining useful life prediction. IEEE Trans Instrum Meas 2022;71:1–11. http://dx.doi.org/10.1109/TIM.2022.3160561.
[24] Wang Y, Deng L, Zheng L, Gao RX. Temporal convolutional network with soft thresholding and attention mechanism for machinery prognostics. J Manuf Syst 2021;60:512–26. http://dx.doi.org/10.1016/j.jmsy.2021.07.008.
[25] Fu S, Lin L, Wang Y, Guo F, Zhao M, Zhong B, Zhong S. MCA-DTCN: A novel dual-task temporal convolutional network with multi-channel attention for first prediction time detection and remaining useful life prediction. Reliab Eng Syst Saf 2024;241:109696. http://dx.doi.org/10.1016/j.ress.2023.109696.
[26] Zhang Y, Su C, Wu J, Liu H, Xie M. Trend-augmented and temporal-featured transformer network with multi-sensor signals for remaining useful life prediction. Reliab Eng Syst Saf 2024;241:109662. http://dx.doi.org/10.1016/j.ress.2023.109662.
[27] Xiang F, Zhang Y, Zhang S, Wang Z, Qiu L, Choi JH. Bayesian gated-transformer model for risk-aware prediction of aero-engine remaining useful life. Expert Syst Appl 2024;238:121859. http://dx.doi.org/10.1016/j.eswa.2023.121859.
[28] Liu L, Song X, Zhou Z. Aircraft engine remaining useful life estimation via a double attention-based data-driven architecture. Reliab Eng Syst Saf 2022;221:108330. http://dx.doi.org/10.1016/j.ress.2022.108330.
[29] Zhou Z, Long Z, Wang R, Bai M, Liu J, Yu D. An aircraft engine remaining useful life prediction method based on predictive vector angle minimization and feature fusion gate improved transformer model. J Manuf Syst 2024;76:567–84. http://dx.doi.org/10.1016/j.jmsy.2024.08.025.
[30] Chen C, Zhu ZH, Shi J, Lu N, Jiang B. Dynamic predictive maintenance scheduling using deep learning ensemble for system health prognostics. IEEE Sensors J 2021;21(23):26878–91. http://dx.doi.org/10.1109/JSEN.2021.3119553.
[31] Wang L, Zhu Z, Zhao X. Dynamic predictive maintenance strategy for system remaining useful life prediction via deep learning ensemble method. Reliab Eng Syst Saf 2024;245:110012. http://dx.doi.org/10.1016/j.ress.2024.110012.
[32] Wang L, Chen Y, Zhao X, Xiang J. Predictive maintenance scheduling for aircraft engines based on remaining useful life prediction. IEEE Internet Things J 2024;11(13):23020–31. http://dx.doi.org/10.1109/JIOT.2024.3376715.
[33] Lu B, Chen Z, Zhao X. Data-driven dynamic predictive maintenance for a manufacturing system with quality deterioration and online sensors. Reliab Eng Syst Saf 2021;212:107628. http://dx.doi.org/10.1016/j.ress.2021.107628.

[34] Shoorkand HD, Nourelfath M, Hajji A. A hybrid CNN-LSTM model for joint optimization of production and imperfect predictive maintenance planning. Reliab Eng Syst Saf 2024;241:109707. http://dx.doi.org/10.1016/j.ress.2023.109707.
[35] Meng H, Geng M, Han T. Long short-term memory network with Bayesian optimization for health prognostics of lithium-ion batteries based on partial incremental capacity analysis. Reliab Eng Syst Saf 2023;236:109288. http://dx.doi.org/10.1016/j.ress.2023.109288.
[36] Wang L, Li B, Zhao X. Multi-objective predictive maintenance scheduling models integrating remaining useful life prediction and maintenance decisions. Comput Ind Eng 2024;197:110581. http://dx.doi.org/10.1016/j.cie.2024.110581.
[37] Shi J, Zhong J, Zhang Y, Xiao B, Xiao L, Zheng Y. A dual attention LSTM lightweight model based on exponential smoothing for remaining useful life prediction. Reliab Eng Syst Saf 2024;243:109821. http://dx.doi.org/10.1016/j.ress.2023.109821.
[38] Chen J, Gao H, Fang C. Optimal two-stage abort policy considering performance-based missions. Reliab Eng Syst Saf 2025;257:110803. http://dx.doi.org/10.1016/j.ress.2025.110803.
[39] Cheng G, Shen J, Wang F, Li L, Yang N. Optimal mission abort policy for a multi-component system with failure interaction. Reliab Eng Syst Saf 2024;242:109791. http://dx.doi.org/10.1016/j.ress.2023.109791.
[40] Zhao X, Li R, Cao S, Qiu Q. Joint modeling of loading and mission abort policies for systems operating in dynamic environments. Reliab Eng Syst Saf 2023;230:108948. http://dx.doi.org/10.1016/j.ress.2022.108948.
[41] Meng S, Xing L, Levitin G. Optimizing component activation and operation aborting in missions with consecutive attempts and common abort command. Reliab Eng Syst Saf 2024;243:109842. http://dx.doi.org/10.1016/j.ress.2023.109842.
[42] Levitin G, Xing L, Dai Y. Optimizing time-varying performance and mission aborting policy in resource constrained missions. Reliab Eng Syst Saf 2024;245:110011. http://dx.doi.org/10.1016/j.ress.2024.110011.
[43] Levitin G, Xing L, Dai Y. A new self-adaptive mission aborting policy for systems operating in uncertain random shock environment. Reliab Eng Syst Saf 2024;248:110184. http://dx.doi.org/10.1016/j.ress.2024.110184.
[44] Fang C, Chen J, Qiu D. Reliability modeling for balanced systems considering mission abort policies. Reliab Eng Syst Saf 2024;243:109853. http://dx.doi.org/10.1016/j.ress.2023.109853.
[45] Zhao X, Liu H, Wu Y, Qiu Q. Joint optimization of mission abort and system structure considering dynamic tasks. Reliab Eng Syst Saf 2023;234:109128. http://dx.doi.org/10.1016/j.ress.2023.109128.
[46] Qiu Q, Maillart LM, Prokopyev OA, Cui L. Optimal condition-based mission abort decisions. IEEE Trans Reliab 2023;72(1):408–25. http://dx.doi.org/10.1109/TR.2022.3172377.
[47] Pan J, Sun B, Wu Z, Yi Z, Feng Q, Ren Y, Wang Z. Probabilistic remaining useful life prediction without lifetime labels: A Bayesian deep learning and stochastic process fusion method. Reliab Eng Syst Saf 2024;250:110313. http://dx.doi.org/10.1016/j.ress.2024.110313.
[48] Lin YH, Ding ZQ, Li YF. Similarity based remaining useful life prediction based on Gaussian process with active learning. Reliab Eng Syst Saf 2023;238:109461. http://dx.doi.org/10.1016/j.ress.2023.109461.
[49] Saxena A, Goebel K, Simon D, Eklund N. Damage propagation modeling for aircraft engine run-to-failure simulation. 2008 Int Conf Progn Heal Manag 2008;1–9. http://dx.doi.org/10.1109/PHM.2008.4711414.
[50] Qiu Q, Cui L, Wu B. Dynamic mission abort policy for systems operating in a controllable environment with self-healing mechanism. Reliab Eng Syst Saf 2020;203:107069. http://dx.doi.org/10.1016/j.ress.2020.107069.
[51] Liu L, Yang J. A dynamic mission abort policy for the swarm executing missions and its solution method by tailored deep reinforcement learning. Reliab Eng Syst Saf 2023;234:109149. http://dx.doi.org/10.1016/j.ress.2023.109149.
[52] Liu L, Yang J, Yan B. A dynamic mission abort policy for transportation systems with stochastic dependence by deep reinforcement learning. Reliab Eng Syst Saf 2024;241:109682. http://dx.doi.org/10.1016/j.ress.2023.109682.
[53] Sobhi JS. Jet engine turbofan design modelling CATIA V5. 2021, Available from: https://grabcad.com/library/jet-engine-turbofan-design-modelling-catia-v5-1.
[54] ElDali M, Kumar KD. Fault diagnosis and prognosis of aerospace systems using growing recurrent neural networks and LSTM. In: 2021 IEEE Aerospace Conference (50100). 2021, p. 1–20. http://dx.doi.org/10.1109/AERO50100.2021.9438432.
[55] Yu W, Kim IY, Mechefske C. Remaining useful life estimation using a bidirectional recurrent neural network based autoencoder scheme. Mech Syst Signal Process 2019;129:764–80. http://dx.doi.org/10.1016/j.ymssp.2019.05.005.
[56] Wang J, Wen G, Yang S, Liu Y. Remaining useful life estimation in prognostics using deep bidirectional LSTM neural network. In: 2018 Prognostics and System Health Management Conference. 2018, p. 1037–42. http://dx.doi.org/10.1109/PHM-Chongqing.2018.00184.
[57] Liu H, Liu Z, Jia W, Lin X. Remaining useful life prediction using a novel feature-attention-based end-to-end approach. IEEE Trans Ind Informatics 2021;17(2):1197–207. http://dx.doi.org/10.1109/TII.2020.2983760.
[58] Kim M, Liu K. A Bayesian deep learning framework for interval estimation of remaining useful life in complex systems by incorporating general degradation characteristics. IISE Trans 2021;53(3):326–40. http://dx.doi.org/10.1080/24725854.2020.1766729.
[59] Wang W, Song H, Si S, Lu W, Cai Z. Data augmentation based on diffusion probabilistic model for remaining useful life estimation of aero-engines. Reliab Eng Syst Saf 2024;252:110394. http://dx.doi.org/10.1016/j.ress.2024.110394.
[60] Zhang Q, Yang P, Liu Q. A dual-stream spatio-temporal fusion network with multi-sensor signals for remaining useful life prediction. J Manuf Syst 2024;76:43–58. http://dx.doi.org/10.1016/j.jmsy.2024.07.004.