RESEARCH ARTICLE

Data-Driven RUL Prediction of CMAPSS Jet Engines: A Swarm Intelligence-Optimized Transformer Approach

Hao Wu[1] **and Tianle Yin**[1,*]

[1] School of Internet of Things Engineering, Jiangnan University, Wuxi 214122, China

Abstract

Remaining useful life (RUL) prediction is a core task in prognostics and health management. While Transformers excel at modeling long-range temporal dependencies, their performance is highly sensitive to hyperparameters, and improper splitting of sliding-window samples can introduce data leakage. We propose a Sparrow Search Algorithm (SSA) optimized Transformer for CMAPSS RUL prediction, adopting an engine-wise split for leakage-aware model selection and using validation RMSE as the fitness function to guide SSA-based hyperparameter optimization. On the FD001 test set, the model achieves RMSE 13.79**, MAE** 10.00**,** $R^2 = 0.88$**, and a NASA score of** 356.26**. The prediction curves and residual diagnostics show stable fitting, with only a few large-error cases.**

Keywords: remaining useful life, prognostics and health management, CMAPSS, hyperparameter optimization.

Submitted: 30 January 2026
Accepted: 05 March 2026
Published: 11 March 2026

Vol. 1, **No.** 2, 2026.
10.62762/AEC.2026.464396

***Corresponding author:**
✉ Tianle Yin
18905616670@163.com

1 Introduction

1.1 Motivation and Background

Predictive maintenance has become a critical enabler for high-reliability industrial assets, where unplanned downtime and safety-critical failures yield substantial operational and economic losses. In this context, prognostics and health management (PHM) leverages condition monitoring signals to infer degradation states and to forecast future health evolution, thereby supporting risk-aware maintenance scheduling. With the rapid progress of deep learning, data-driven PHM has gained strong momentum due to its ability to model nonlinear and high-dimensional sensor streams, yet it simultaneously faces persistent challenges in robustness, generalization, and principled model selection [1–5].

Remaining useful life (RUL) prediction is a central task within PHM, aiming to estimate the number of cycles (or time) remaining before an asset reaches an end-of-life criterion. For fleet-style benchmarks such as the NASA C-MAPSS turbofan engine datasets, each trajectory corresponds to a distinct engine with unknown initial wear and manufacturing variability, and the measurements are contaminated by sensor noise and operational changes. Consequently, degradation signatures can be subtle, nonstationary, and highly coupled with operating regimes, which makes cross-engine generalization significantly more

Citation
Wu, H., & Yin, T. (2026). Data-Driven RUL Prediction of CMAPSS Jet Engines: A Swarm Intelligence-Optimized Transformer Approach. *Aerospace Engineering Communications*, 1(2), 57–67.

difficult than single-trajectory forecasting. These characteristics motivate learning architectures that can extract informative temporal representations while remaining robust under distribution shifts across engines and conditions [2, 3].

1.2 Related Work

In recent years, convolutional and attention-based deep architectures have become dominant paradigms for RUL prediction. Convolutional models can efficiently capture local temporal patterns from multivariate sequences; for example, Yang *et al*. [6] proposed a double-convolutional neural network architecture that improves feature extraction for RUL regression. Beyond pure convolution, attention mechanisms have been introduced to enhance long-range dependency modeling and to focus on degradation-relevant segments; Chen *et al*. [7] developed an attention-based deep learning approach and demonstrated performance gains in machine RUL prediction. To further enhance representation quality, dual-attention designs that combine channel-wise and temporal attention have been explored within temporal convolutional networks, enabling selective emphasis over sensors and time steps [11]. Recently, Transformer-style architectures have attracted increasing interest because self-attention supports parallel sequence modeling and flexible dependency capture across time, which is particularly appealing for long and noisy monitoring sequences [8].

While Transformers provide a strong modelling backbone, achieving robust performance in RUL prediction often requires structural adaptations and training strategies tailored to PHM data. Domain shifts across operating regimes motivate explicit domain adaptation mechanisms; Li *et al*. [8] design a domain-adaptive Transformer framework to mitigate distribution mismatch in RUL prediction. And, spatio-temporal inductive biases and cross-sensor dependency modelling have been integrated to enhance representation learning. For instance, Liang *et al*. [13] enhanced an adaptive Transformer with graph attention to better exploit sensor relationships, Zhang *et al*. [12] introduced a weighted time-embedding Transformer to improve time-awareness in prognostic modeling, and recent turbofan-oriented designs further integrate advanced temporal structure learning, including a parallel spatial–temporal Transformer [9] and a spatio-temporal Koopman dual-branch Transformer [10]. These developments collectively indicate that attention-based architectures can be highly competitive for RUL prediction when properly configured.

However, Transformer performance is notably sensitive to architectural and optimization hyperparameters, including embedding dimension, number of attention heads, feed-forward width, dropout rate, and learning rate. Manual tuning is often inefficient and subjective, whereas exhaustive grid search is computationally prohibitive for deep models. This motivates the use of meta-heuristic optimization as a practical alternative for black-box hyperparameter selection in PHM. The Sparrow Search Algorithm (SSA) is a recent swarm intelligence approach that balances exploration and exploitation through role-based population updates and has shown promise as an efficient optimizer in complex search landscapes [14]. In the RUL context, SSA can be used to search over mixed discrete–continuous hyperparameter spaces with limited human intervention, potentially improving both tuning efficiency and final performance.

Motivated by these considerations, this paper designs an SSA-driven hyperparameter optimization strategy for a Transformer-based RUL predictor on the C-MAPSS benchmark. The proposed pipeline formulates RUL prediction as supervised regression over sliding windows of standardized multivariate sensor sequences and adopts an engine-level group split to obtain a leakage-aware validation estimate under cross-engine generalization. By coupling SSA-based global search with a Transformer backbone, this work aims to provide a reproducible and computationally practical approach for improving attention-based prognostic modeling on fleet-style turbofan datasets [8–10].

1.3 Contribution

The main contributions are summarized as follows:

1) Transformer-based RUL learning under fleet-style variability: We consider multivariate turbofan sensor sequences with cross-engine distribution shifts and employ attention-based architectures to capture long-range temporal dependencies [7, 8].

2) SSA-driven hyperparameter optimization for prognostic Transformers: We model Transformer hyperparameter selection as an optimization problem and adopt SSA to search mixed discrete continuous design spaces efficiently [14].

1.4 Organization

The remainder of this paper is organized as follows. Section 2 introduces the C-MAPSS dataset and the FD001 subset used in this study. Section 3 details the proposed SSA-optimized Transformer framework, including the sliding-window formulation, the Transformer regressor, the leakage-aware engine-wise validation protocol, and the SSA-based hyperparameter optimization scheme. Section 4 reports the simulation results on FD001, presenting both quantitative metrics and qualitative diagnostic visualizations. Finally, Section 5 concludes the paper and discusses potential future directions.

2 Dataset Description

This study evaluates the proposed framework on the C-MAPSS turbofan engine degradation benchmark, which is widely used for data-driven prognostics and RUL estimation. Each subset consists of a fleet of engines operating under nominal conditions at the beginning of the run and gradually developing degradation until failure. The dataset is organized into a training split and a test split. In the training split, each engine trajectory spans from an initially healthy state to a run-to-failure termination, enabling direct construction of cycle-based RUL labels. In the test split, each engine trajectory is truncated before failure, and the objective is to predict the remaining number of operational cycles after the final observed cycle. The measurement streams contain sensor noise and include inter-engine variability due to unknown initial wear and manufacturing differences, which are treated as normal variation rather than fault conditions.

The raw data are provided as space-separated text files. Each row corresponds to one operational cycle of one engine and includes an engine identifier, a cycle index, three operational settings, and a set of sensor measurements. In the standard C-MAPSS format, the columns are structured as: unit number, time in cycles, operational settings $(\text{setting}_1, \text{setting}_2, \text{setting}_3)$, followed by 21 sensor channels. Let u denote the engine index and t denote the cycle index. The observation at cycle t is a vector containing the operational settings and sensor readings, which is subsequently transformed into a standardized feature representation for learning-based modeling.

C-MAPSS provides four subsets with different operating conditions and fault modes. FD001 and FD003 correspond to a single operating condition, whereas FD002 and FD004 contain multiple operating conditions. Moreover, FD001 and FD002 involve a single fault mode, while FD003 and FD004 include two fault modes. In terms of dataset size, FD001 and FD003 each contain 100 training trajectories and 100 test trajectories, while FD002 and FD004 are larger, containing 260/259 and 248/249 training/test trajectories, respectively. Unless otherwise specified, the experimental results in this paper focus on FD001 to isolate the modeling behavior under a single-condition setting and to facilitate controlled analysis, while the proposed methodology is directly applicable to the remaining subsets.

3 SSA-Optimized Transformer for CMAPSS RUL Prediction

This section presents an SSA-optimized Transformer framework for remaining useful life (RUL) prediction on the CMAPSS turbofan benchmark. The overall workflow is illustrated in Figure 1. The task is formulated as supervised regression that maps a fixed-length multivariate sensor window to the scalar RUL at the window end. The proposed pipeline consists of RUL label construction with early-life truncation, feature standardization, sliding-window sample generation, Transformer-based temporal modeling, and automated hyperparameter optimization via the sparrow search algorithm (SSA). To avoid optimistic model selection caused by overlapping windows extracted from the same engine trajectory, all selection and validation steps adopt an engine-wise group split.

3.1 Problem Formulation and Sliding-Window Representation

Each engine trajectory is a multivariate time series indexed by operating cycles. For each engine trajectory let $\mathbf{x}_{u,t} \in \mathbb{R}^F$ denote the standardized feature vector of engine u at cycle t, where F is the number of retained sensor features after feature screening. A sliding window of length L converts each trajectory into supervised samples. For any cycle $t \geq L$, the model input is defined as

$$\mathbf{X}_{u,t} = \left[\mathbf{x}_{u,t-L+1}, \ldots, \mathbf{x}_{u,t}\right] \in \mathbb{R}^{L \times F}. \tag{1}$$

In the training set, the failure time T_u of engine u is known and the cycle-based RUL label is

$$y_{u,t} = T_u - t. \tag{2}$$

Following common practice for CMAPSS, the label is truncated by an upper bound $y_{\max}$ to reduce variance

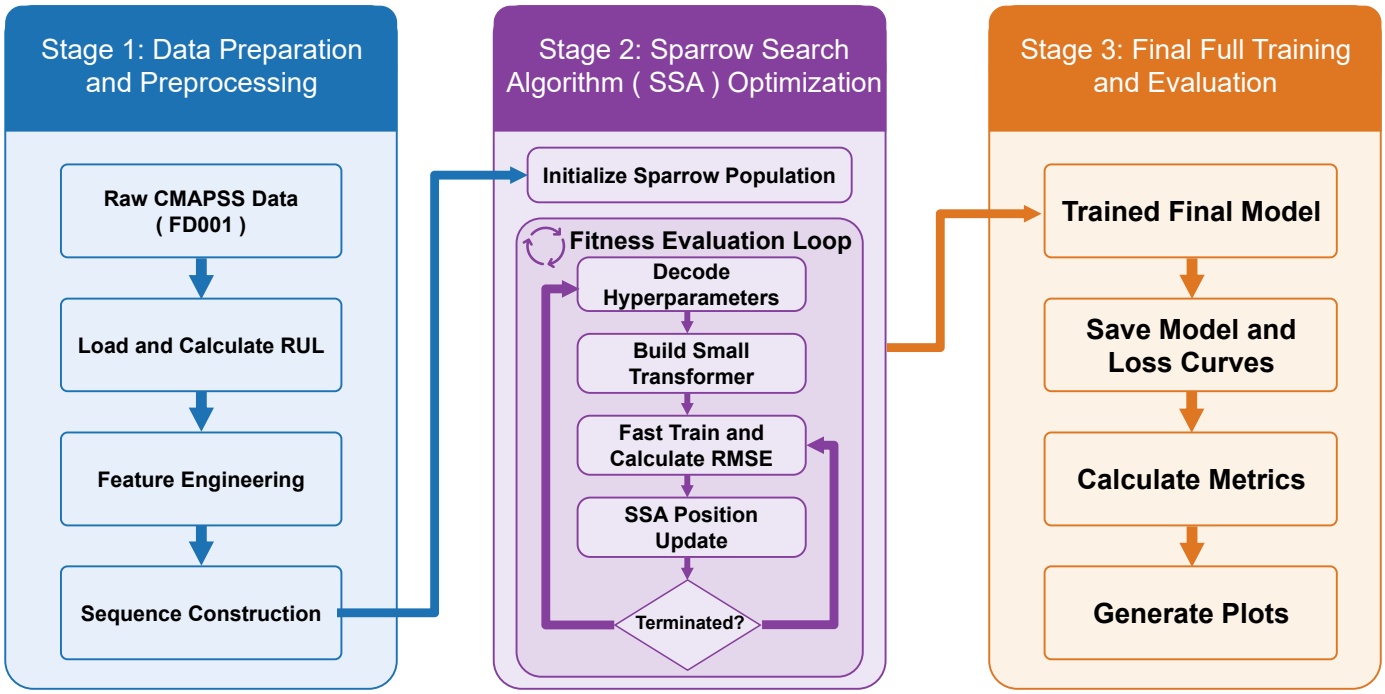

Figure 1. Overall workflow of the SSA-optimized Transformer for CMAPSS jet engine RUL prediction (FD001), including preprocessing, SSA-based hyperparameter search, and final training and evaluation.

in the early healthy stage:

$$\tilde{y}_{u,t} = \min(y_{u,t},\, y_{\max})\,. \tag{3}$$

The regression objective is to learn a mapping that estimates $\hat{y}_{u,t}$ from $\mathbf{X}_{u,t}$. In the test set, where each trajectory ends prior to failure, prediction is performed using the final available window of each engine.

3.2 Transformer Regressor

The temporal mapping $f_\theta : \mathbb{R}^{L\times F} \to \mathbb{R}$ is implemented using a Transformer regressor. Given an input window $\mathbf{X} \in \mathbb{R}^{L\times F}$, a linear projection embeds each time step into a d-dimensional latent representation:

$$\mathbf{H}_0 = \mathbf{X}\mathbf{W}_e + \mathbf{b}_e, \qquad \mathbf{H}_0 \in \mathbb{R}^{L\times d}, \tag{4}$$

where $\mathbf{W}_e \in \mathbb{R}^{F\times d}$ and $\mathbf{b}_e \in \mathbb{R}^d$ are learnable parameters, and $\mathbf{b}_e$ is added row-wise (broadcast) to all L time steps. Temporal order is encoded by a learnable positional embedding $\mathbf{P} \in \mathbb{R}^{L\times d}$:

$$\tilde{\mathbf{H}}_0 = \mathbf{H}_0 + \mathbf{P}, \tag{5}$$

The sequence embedding is processed by a stack of B Transformer blocks. For the b-th block, multi-head self-attention is applied to capture global temporal dependencies:

$$\mathbf{A}_b = \mathrm{MHA}\left(\tilde{\mathbf{H}}_{b-1},\, \tilde{\mathbf{H}}_{b-1},\, \tilde{\mathbf{H}}_{b-1}\right). \tag{6}$$

Residual connection and layer normalization yield

$$\mathbf{Z}_b = \mathrm{LN}\left(\tilde{\mathbf{H}}_{b-1} + \mathbf{A}_b\right). \tag{7}$$

A position-wise feed-forward network (FFN) enhances nonlinearity:

$$\mathbf{F}_b = \mathbf{W}_2\,\sigma\left(\mathbf{W}_1\mathbf{Z}_b + \mathbf{1}\mathbf{b}_1^\top\right) + \mathbf{1}\mathbf{b}_2^\top, \tag{8}$$

where $\sigma(\cdot)$ denotes the ReLU activation. The block output is obtained by a second residual connection and normalization:

$$\tilde{\mathbf{H}}_b = \mathrm{LN}(\mathbf{Z}_b + \mathbf{F}_b)\,. \tag{9}$$

After B blocks, temporal information is aggregated by global average pooling:

$$\mathbf{h} = \frac{1}{L}\sum_{t=1}^{L} \tilde{\mathbf{H}}_B(t,:) \in \mathbb{R}^d. \tag{10}$$

Finally, a lightweight regression head maps $\mathbf{h}$ to the scalar RUL estimate:

$$\mathbf{z} = \phi(\mathbf{W}_r\mathbf{h} + \mathbf{b}_r)\,, \tag{11}$$

$$\hat{y} = \mathbf{w}_o^\top\mathbf{z} + b_o, \tag{12}$$

where $\phi(\cdot)$ denotes ReLU and the output layer is linear. Dropout is applied within the Transformer blocks and the regression head to improve generalization.

3.3 Training Objective and Engine-Wise Validation

Model parameters are optimized by minimizing a robust regression loss. To reduce sensitivity to occasional noisy measurements and outliers, the Huber loss is adopted:

$$\mathcal{L}_\delta(y,\hat{y}) = \begin{cases} \frac{1}{2}(y-\hat{y})^2, & |y-\hat{y}| \le \delta, \\ \delta|y-\hat{y}| - \frac{1}{2}\delta^2, & |y-\hat{y}| > \delta, \end{cases} \tag{13}$$

where δ controls the transition from quadratic to linear penalty. The optimizer is AdamW (or Adam when AdamW is unavailable), and gradient clipping is employed to stabilize training.

Because sliding windows extracted from the same engine are strongly correlated, sample-level random splits may leak engine-specific information into the validation set, leading to overly optimistic performance estimates. Therefore, an engine-wise group split is used, ensuring that all windows from a given engine are assigned exclusively to either the training set or the validation set:

$$\mathcal{U}_{\text{train}} \cap \mathcal{U}_{\text{val}} = \emptyset, \tag{14}$$

$$\mathcal{U}_{\text{train}} \cup \mathcal{U}_{\text{val}} = \mathcal{U}, \tag{15}$$

where $\mathcal{U}$ denotes the set of engine IDs. This protocol provides a more reliable estimate of cross-engine generalization during model selection.

3.4 SSA-Based Hyperparameter Optimization

Transformer-based RUL predictors are highly sensitive to architectural and optimization choices, which makes manual tuning inefficient and hard to reproduce. We formulate hyperparameter selection as a black-box minimization problem and solve it using the Sparrow Search Algorithm (SSA) [14]. Each sparrow encodes a candidate configuration in a continuous position vector, which is decoded into a concrete model setting before training and validation.

3.4.1 *Objective and Fitness Definition*

Let $\mathbf{p} \in \mathbb{R}^D$ denote a continuous SSA position. A deterministic decoding function $\mathcal{G}(\cdot)$ maps $\mathbf{p}$ to a feasible hyperparameter configuration:

$$\boldsymbol{\lambda} = \mathcal{G}(\mathbf{p}), \tag{16}$$

where $\boldsymbol{\lambda}$ includes both discrete and continuous choices (e.g., depth, width, dropout, learning rate, and batch size), enforced by the decoding rules and box constraints.

Given an engine-wise train/validation split, a Transformer regressor is trained under $\boldsymbol{\lambda}$ and evaluated on the validation set. The SSA objective minimizes the validation RMSE:

$$\mathbf{p}^\star = \arg\min_{\mathbf{p}\in[\mathbf{lb},\mathbf{ub}]} \mathcal{J}(\mathbf{p}), \tag{17}$$

$$\mathcal{J}(\mathbf{p}) = \sqrt{\frac{1}{N_{\text{val}}}\sum_{i=1}^{N_{\text{val}}} (y_i - \hat{y}_i(\boldsymbol{\lambda}))^2}, \tag{18}$$

where $\hat{y}_i(\boldsymbol{\lambda})$ is the model prediction obtained after training with $\boldsymbol{\lambda} = \mathcal{G}(\mathbf{p})$.

To keep the fitness evaluation comparable across iterations, the engine-wise split is fixed during the entire SSA run, so that all candidates are scored on the same validation protocol without leakage across engines.

3.4.2 *SSA Population Update Rules*

At iteration t, SSA maintains a population matrix $\mathbf{P}^{(t)} = [\mathbf{p}_1^{(t)}; \dots; \mathbf{p}_N^{(t)}] \in \mathbb{R}^{N\times D}$ and fitness values $\{f_i^{(t)}\}_{i=1}^N$ with $f_i^{(t)} = \mathcal{J}(\mathbf{p}_i^{(t)})$. Let $\mathbf{p}_{\text{best}}^{(t)}$ and $\mathbf{p}_{\text{worst}}^{(t)}$ denote the best and worst candidates in the current population (lowest and highest fitness), respectively. SSA updates three roles: discoverers, joiners, and aware sparrows [14].

Discoverers. For discoverer index i (after sorting by fitness), a random scalar $r_2 \sim \mathcal{U}(0,1)$ determines whether the environment is safe. The update used in our implementation follows an exploitation–exploration switch:

$$\mathbf{p}_i^{(t+1)} = \begin{cases} \mathbf{p}_i^{(t)} \exp\Big(-\dfrac{i}{u\,(T+1)}\Big), & r_2 < \text{ST}, \\ \mathbf{p}_i^{(t)} + \boldsymbol{\epsilon}, & r_2 \ge \text{ST}, \end{cases} \tag{19}$$

$$u \sim \mathcal{U}(0,1), \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0},\mathbf{I}), \tag{20}$$

where T is the maximum SSA iteration count and ST is the safety threshold. The first case shrinks the position magnitude to refine the search, while the second case injects Gaussian perturbations to enhance exploration.

Joiners. Joiners are updated by either exploiting the current best solution or escaping from poor regions, depending on their rank:

$$\mathbf{p}_i^{(t+1)} = \begin{cases} \mathbf{p}_{\text{best}}^{(t)} + \left|\mathbf{p}_i^{(t)} - \mathbf{p}_{\text{best}}^{(t)}\right| \odot \mathbf{a}, & i \le \dfrac{N}{2}, \\ \mathbf{z} \odot \exp\Big(\dfrac{\mathbf{p}_{\text{worst}}^{(t)} - \mathbf{p}_i^{(t)}}{i^2+\varepsilon}\Big), & i > \dfrac{N}{2}, \end{cases} \tag{21}$$

$$\mathbf{a} \in \{-1,1\}^D, \qquad \mathbf{z} \sim \mathcal{N}(\mathbf{0},\mathbf{I}), \tag{22}$$

where $\odot$ denotes element-wise multiplication and ε is a small positive constant for numerical stability. The

first case searches around the best solution; the second case biases weak joiners away from the worst region through an exponential escape term.

Aware Sparrows. A small randomly selected subset is designated as aware sparrows to maintain diversity and avoid premature convergence:

$$\mathbf{p}_k^{(t+1)} = \begin{cases} \mathbf{p}_{\text{best}}^{(t)} + \boldsymbol{\beta} \odot \left| \mathbf{p}_k^{(t)} - \mathbf{p}_{\text{best}}^{(t)} \right|, & f_k^{(t)} > f_{\text{best}}^{(t)}, \\ \mathbf{p}_k^{(t)} + \mathbf{q} \odot \dfrac{\left| \mathbf{p}_k^{(t)} - \mathbf{p}_{\text{worst}}^{(t)} \right|}{f_k^{(t)} - f_{\text{worst}}^{(t)} + \varepsilon}, & f_k^{(t)} \le f_{\text{best}}^{(t)}, \end{cases} \tag{23}$$

$$\boldsymbol{\beta} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \qquad \mathbf{q} \sim \mathcal{U}(-1, 1)^D. \tag{24}$$

Intuitively, worse aware sparrows are pulled toward the best region with random scaling, while better ones are perturbed away from the worst region to preserve exploration capacity.

Box Projection and Selection. After role-wise updates, all candidates are projected back to the feasible box defined by the search bounds:

$$\mathbf{p}_i^{(t+1)} = \Pi_{[\mathbf{lb},\mathbf{ub}]}\left(\mathbf{p}_i^{(t+1)}\right), \tag{25}$$

where $\Pi_{[\mathbf{lb},\mathbf{ub}]}(\cdot)$ denotes element-wise clipping. Fitness values are then recomputed via Eq. (18), and the best candidate observed throughout the search is returned as $\mathbf{p}^\star$.

Algorithm 1 provides a concise pseudo-code summary of the SSA implementation used in this study. The table is intentionally compact and delegates the concrete update expressions to Eqs. (19) and (25) for clarity.

After SSA terminates, the Transformer is retrained under $\boldsymbol{\lambda}^\star = \mathcal{G}(\mathbf{p}^\star)$ using the same leakage-aware engine-wise protocol, and the final test predictions are produced by applying the trained model to the last available window of each test engine.

Algorithm 1: SSA-driven hyperparameter optimization procedure (implementation-level pseudo-code).

Input: training windows with engine IDs; fixed engine-wise train/validation split; decoding $\mathcal{G}(\cdot)$; bounds $[\mathbf{lb}, \mathbf{ub}]$; SSA iteration budget T

Output: best configuration $\mathbf{p}^\star$

Initialization:;

sample $\mathbf{p}_i^{(0)} \sim \mathcal{U}(\mathbf{lb}, \mathbf{ub})$, $i = 1, \ldots, N$;

Evaluation:;

for *each* i **do**

  set $\boldsymbol{\lambda}_i = \mathcal{G}(\mathbf{p}_i^{(0)})$;

  compute $f_i^{(0)} = \mathcal{J}(\mathbf{p}_i^{(0)})$ using Eq. (18);

end

Main loop:;

for $t = 0$ **to** $T - 1$ **do**

  sort $\{\mathbf{p}_i^{(t)}, f_i^{(t)}\}$ by $f_i^{(t)}$;

  record $\mathbf{p}_{\text{best}}^{(t)}$ and $\mathbf{p}_{\text{worst}}^{(t)}$;

  Role updates:;

  update discoverers by Eq. (19);

  update joiners by Eq. (21);

  update aware sparrows by Eq. (23);

  Projection:;

  apply box projection $\mathbf{p}_i^{(t+1)} = \Pi_{[\mathbf{lb},\mathbf{ub}]}(\mathbf{p}_i^{(t+1)})$ via Eq. (25);

  Re-evaluation:;

  recompute $f_i^{(t+1)} = \mathcal{J}(\mathbf{p}_i^{(t+1)})$ on the same engine-wise validation protocol;

end

Return:;

$\mathbf{p}^\star = \arg\min_{\mathbf{p}} \mathcal{J}(\mathbf{p})$ over all evaluated candidates;

4 Simulation

4.1 Simulation Setup

The CMAPSS FD001 dataset provides multivariate sensor trajectories collected from multiple engines under run-to-failure conditions. Raw signals are standardized using training statistics and then segmented into fixed-length sliding windows to form supervised samples. The RUL target follows the commonly adopted capped-label convention in CMAPSS to stabilize early-life supervision. The SSA optimizer searches the hyperparameter space of the Transformer regressor, and each candidate configuration is evaluated by training a model with early stopping and computing the validation RMSE on the engine-wise split. The best configuration found by SSA is reported in Table 1.

In addition to SSA-Transformer, we include a CNN baseline, a fixed-hyperparameter Transformer baseline, and a PSO-optimized Transformer under the same preprocessing and engine-wise split for fair comparison.

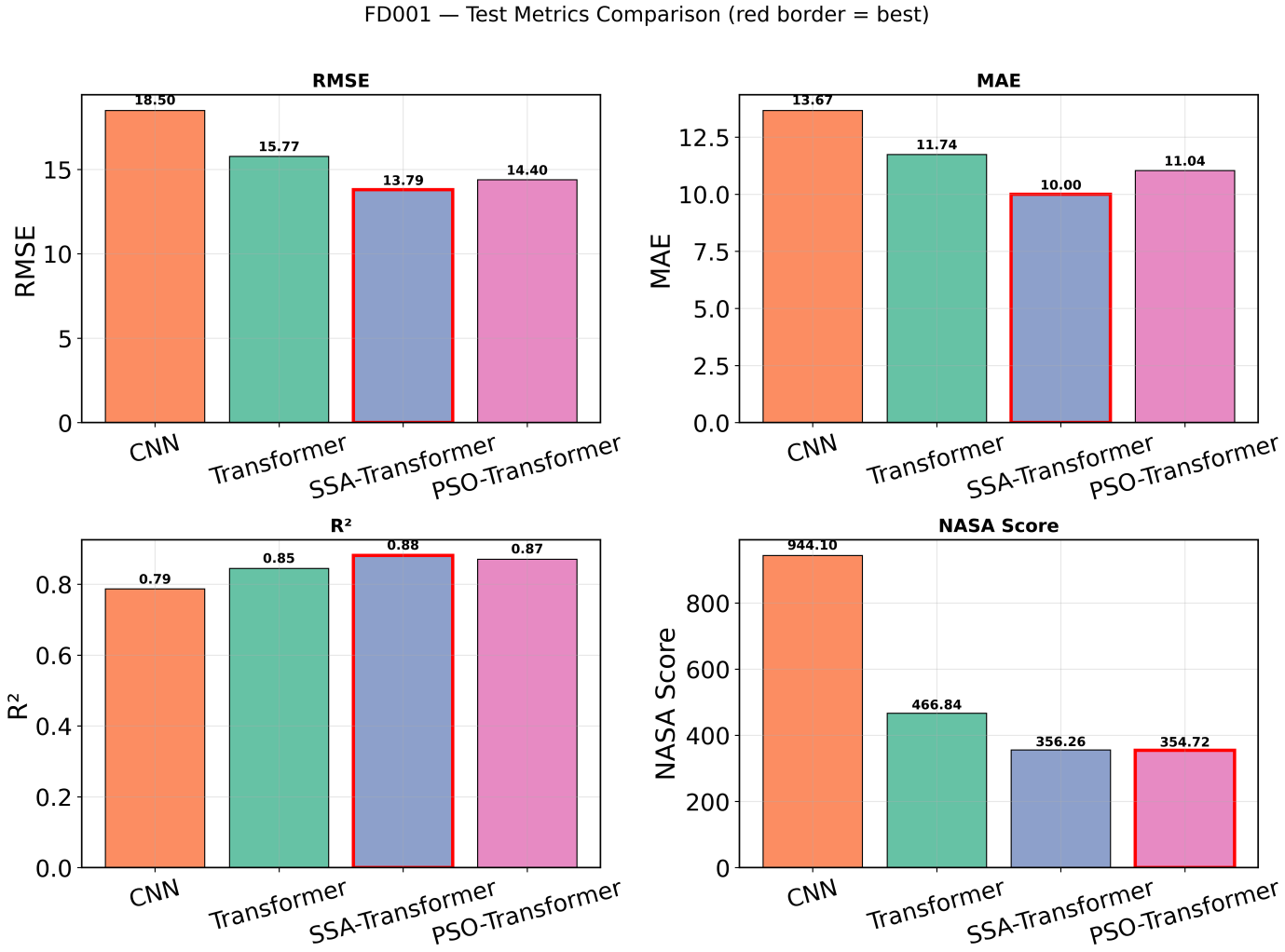

Figure 2. FD001 test metrics comparison among CNN, Transformer (fixed), SSA-Transformer, and PSO-Transformer (red border = best).

Table 1. Best hyperparameter configuration selected by SSA on FD001.

Hyperparameter	Selected value
Model dimension d	32
Number of heads H	4
Number of Transformer blocks B	1
FFN dimension d_{ff}	128
Dropout rate ρ	0.05
Learning rate η	2.512×10^{-3}
Batch size bs	32

4.2 Evaluation Metrics

We report the Root Mean Square Error (RMSE), Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), coefficient of determination (R^2), and the NASA PHM asymmetric score. Let y_i and $\hat{y}_i$ denote the ground-truth and predicted RUL of the i-th test engine, respectively, with N engines in total. The metrics are defined as

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} \tag{26}$$

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i| \tag{27}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2}, \qquad \bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i \tag{28}$$

$$\text{MAPE} = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{y_i - \hat{y}_i}{\max(|y_i|, \epsilon)}\right| \times 100\% \tag{29}$$

where ϵ is a small constant to avoid numerical instability when y_i approaches zero. The NASA PHM

Table 2. Performance comparison on CMAPSS FD001 (engine-wise split).

Method	RMSE↓	MAE↓	MAPE(%)↓	R^2 ↑	NASA↓	Time(min)↓
CNN	18.50	13.67	26.72	0.79	944.10	0.28
Transformer (fixed)	15.77	11.74	18.83	0.85	466.84	1.97
SSA-Transformer	13.79	10.00	16.53	0.88	356.26	20.42
PSO-Transformer	14.40	11.04	17.45	0.87	354.72	26.53

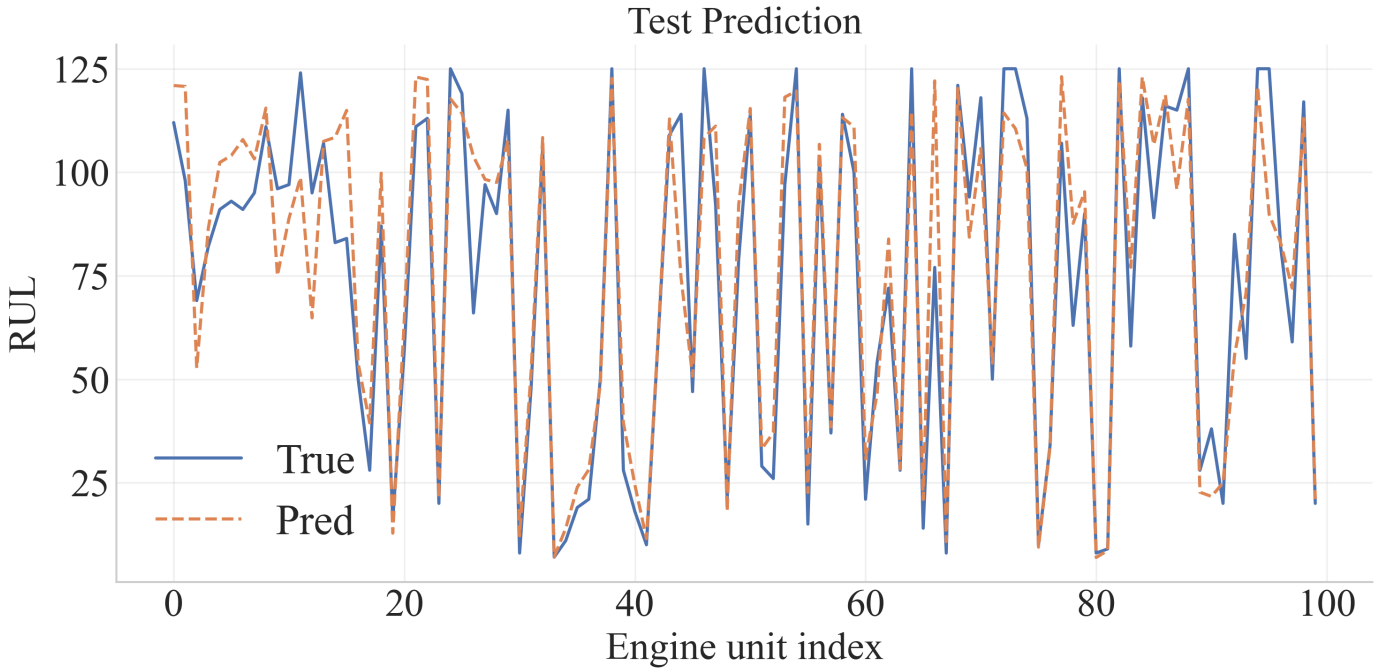

Figure 3. Test-set prediction curve on FD001 (one sample per engine).

score is computed using the standard asymmetric exponential penalty. Let $d_i = \hat{y}_i - y_i$ denote the prediction error. The NASA score is defined as

$$\text{NASA} = \sum_{i=1}^{N} s(d_i), \tag{30}$$

$$s(d) = \begin{cases} \exp\left(-\dfrac{d}{13}\right) - 1, & d < 0, \\ \exp\left(\dfrac{d}{10}\right) - 1, & d \geq 0, \end{cases} \tag{31}$$

where overestimation ($d > 0$, late prediction) is penalized more heavily than underestimation ($d < 0$, early prediction).

4.3 Comparative Results

Table 2 reports the FD001 test-set comparison under the leakage-free engine-wise split, and Figure 2 provides a metric-wise visualization (red border indicates the best method for each metric). Compared with the fixed-hyperparameter Transformer baseline, SSA optimization consistently improves accuracy: RMSE decreases from 15.77 to 13.79, MAE decreases from 11.74 to 10.00, MAPE decreases from 18.83% to 16.53%, and R^2 increases from 0.85 to 0.88. The NASA score also drops from 466.84 to 356.26, indicating fewer high-penalty late predictions under the asymmetric metric. PSO-Transformer achieves a slightly lower NASA score than SSA (354.72 vs. 356.26) while showing slightly worse overall accuracy (RMSE/MAE/MAPE = 14.40/11.04/17.45%), suggesting a small trade-off between global accuracy and the asymmetric late-prediction risk. In terms of efficiency, the CNN and fixed Transformer baselines run much faster, whereas HPO-based methods incur additional cost due to repeated training during search. Figure 4 provides an aggregated view (outer = better): SSA-Transformer forms the outer envelope on RMSE/MAE/MAPE/R^2, while PSO-Transformer is marginally better on the NASA-score dimension.

4.4 Overall Performance of the Selected Model

We use RMSE, MAE, MAPE, and R^2 as the main accuracy metrics. On FD001, the SSA-Transformer

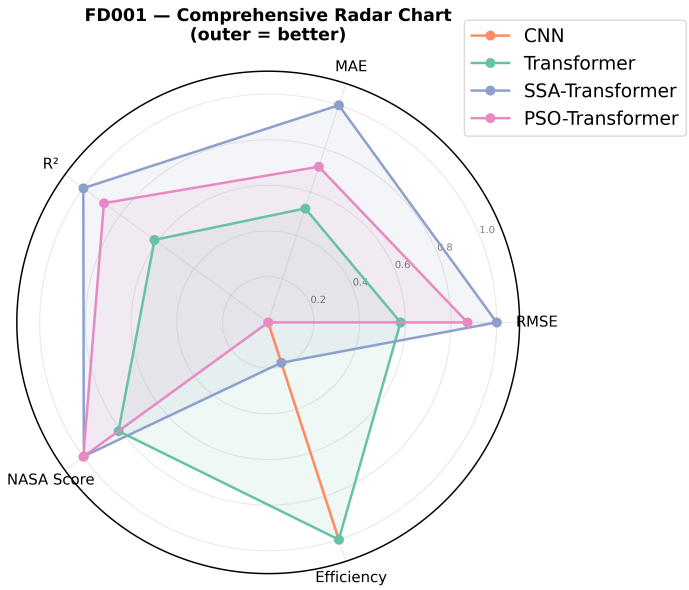

Figure 4. FD001 comprehensive radar chart comparison (outer = better).

achieves RMSE=13.79, MAE=10.00, MAPE=16.53%, and $R^2 = 0.88$ (Table 2). It outperforms the CNN baseline and the fixed-hyperparameter Transformer baseline. PSO-Transformer has a slightly lower NASA score (354.72 vs. 356.26). This suggests slightly fewer high-cost late predictions under the asymmetric exponential penalty. The NASA-score gap is small, and SSA has better overall accuracy. We therefore select SSA-Transformer as the target model. We then analyze the remaining hard cases and the error distribution across the RUL range using prediction and residual diagnostics.

4.5 Training Convergence

Figure 5 shows the final training and validation loss curves. The training loss decreases steadily, while the validation curve stabilizes after the early stage, suggesting that early stopping effectively prevents further overfitting under the engine-wise split.

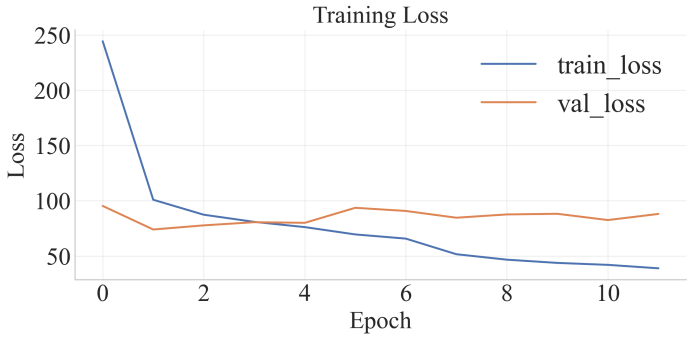

Figure 5. Final training and validation loss curves of the SSA-selected Transformer on FD001.

4.6 Prediction Quality and Error Diagnostics

Figure 3 compares the predicted and ground-truth RUL across test engines (one terminal window per engine). The two curves align closely for most engines, while a subset exhibits larger deviations, reflecting inter-engine variability and local distribution shift.

To assess calibration at the sample level, Figure 6 plots predicted versus true RUL. Most points concentrate around the diagonal, while dispersion increases in the mid-to-high RUL region, implying that early-life operating-condition uncertainty and limited degradation signature may still affect regression stability.

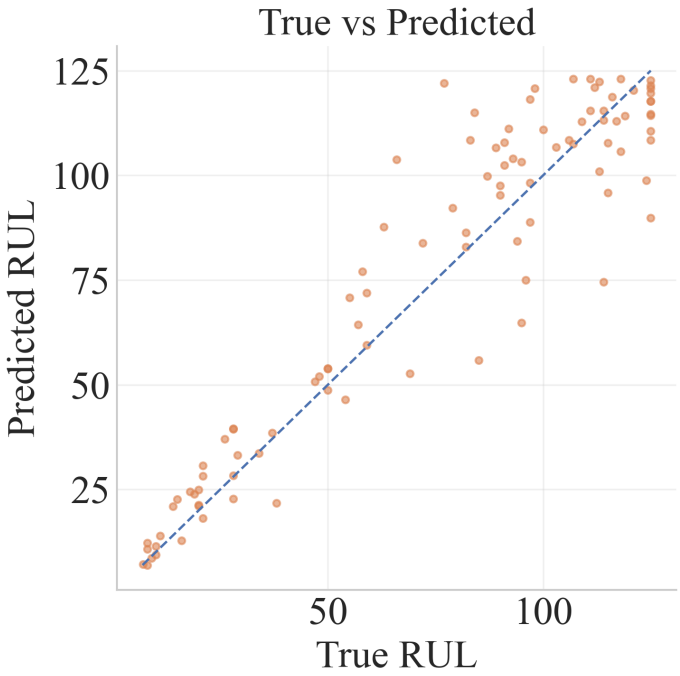

Figure 6. Scatter plot of predicted versus true RUL on FD001. The dashed diagonal indicates perfect prediction.

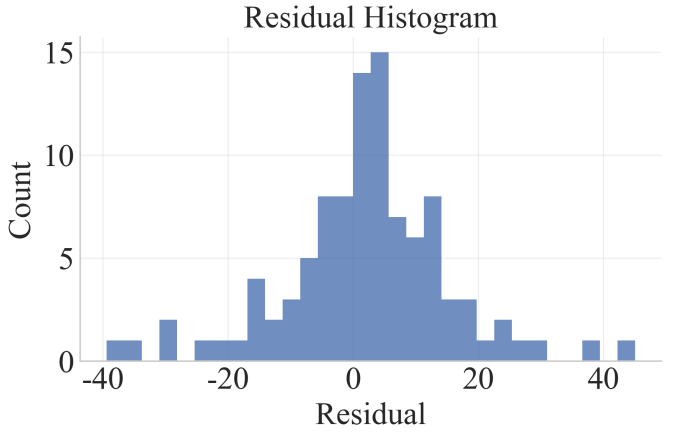

Figure 7. Residual histogram on the FD001 test set.

Figure 7 presents the residual histogram. The residuals

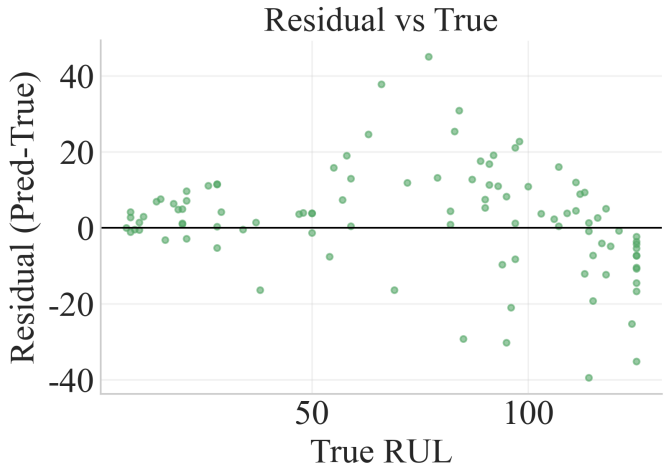

Figure 8. Residuals $(\hat{y} - y)$ versus true RUL on FD001.

are centered near zero with a slight negative mean, indicating mildly conservative predictions on average (i.e., a tendency to under-estimate RUL). Figure 8 further shows residuals versus the true RUL, where the spread suggests heteroscedastic behavior across different health stages.

Finally, Figure 9 reports the empirical CDF of absolute errors, which provides a concise view of reliability under different tolerance thresholds. The curve indicates that a majority of engines fall within moderate error bounds, while a smaller fraction contributes to the long-error tail.

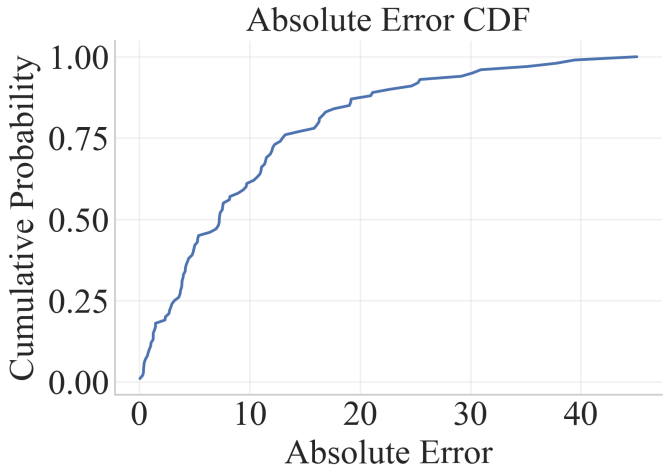

Figure 9. Empirical CDF of absolute prediction error on FD001.

5 Conclusion

This paper studied an SSA-driven hyperparameter optimization strategy for Transformer-based RUL prediction on the CMAPSS benchmark. We minimized validation RMSE during the search and used an engine-wise split for leakage-free model selection. On FD001, the SSA-Transformer improved accuracy over the CNN baseline and the fixed-hyperparameter Transformer, achieving lower RMSE/MAE/MAPE and higher R^2 under the same protocol. We also compared SSA with PSO on the same Transformer backbone: SSA achieved the best RMSE/MAE and the highest R^2, while PSO attained a slightly lower NASA score, indicating a small trade-off between overall accuracy and late-prediction penalty. Two limitations remain. First, we only evaluated FD001, which has a single operating condition. Second, hyperparameter search increases runtime due to repeated training. Future work will test FD002/FD004 and study how operating-condition changes affect the model, explore condition-aware features and domain-robust training, and reduce search cost using early termination and surrogate fitness models.

Data Availability Statement

Data will be made available on request.

Funding

This work was supported without any funding.

Conflicts of Interest

The authors declare no conflicts of interest.

AI Use Statement

The authors declare that no generative AI was used in the preparation of this manuscript.

Ethical Approval and Consent to Participate

Not applicable.

References

[1] Rezaeianjouybari, B., & Shang, Y. (2020). Deep learning for prognostics and health management: State of the art, challenges, and opportunities. *Measurement*, *163*, 107929. [CrossRef]

[2] Fink, O., Wang, Q., Svensen, M., Dersin, P., Lee, W. J., & Ducoffe, M. (2020). Potential, challenges and future directions for deep learning in prognostics and health management applications. *Engineering Applications of Artificial Intelligence*, *92*, 103678. [CrossRef]

[3] Ferreira, C., & Gonçalves, G. (2022). Remaining Useful Life prediction and challenges: A literature review on the use of Machine Learning Methods. *Journal of manufacturing systems*, *63*, 550-562. [CrossRef]

[4] Mo, Y., Wu, Q., Li, X., & Huang, B. (2021). Remaining useful life estimation via transformer encoder enhanced by a gated convolutional unit. *Journal of Intelligent Manufacturing, 32*(7), 1997-2006. [CrossRef]

[5] Xue, F., Jin, G., Tan, L., Zhang, C., & Yu, Y. (2025). Predictive maintenance programs for aircraft engines based on remaining useful life prediction. *Scientific Reports*. [CrossRef]

[6] Yang, B., Liu, R., & Zio, E. (2019). Remaining useful life prediction based on a double-convolutional neural network architecture. *IEEE Transactions on Industrial Electronics, 66*(12), 9521-9530. [CrossRef]

[7] Chen, Z., Wu, M., Zhao, R., Guretno, F., Yan, R., & Li, X. (2020). Machine remaining useful life prediction via an attention-based deep learning approach. *IEEE Transactions on Industrial Electronics, 68*(3), 2521-2531. [CrossRef]

[8] Li, X., Li, J., Zuo, L., Zhu, L., & Shen, H. T. (2022). Domain adaptive remaining useful life prediction with transformer. *IEEE Transactions on Instrumentation and Measurement, 71*, 1-13. [CrossRef]

[9] Fu, S., Jia, Y., Lin, L., Suo, S., Guo, F., Zhang, S., & Liu, Y. (2025). PSTFormer: A novel parallel spatial-temporal transformer for remaining useful life prediction of aeroengine. *Expert Systems with Applications, 265*, 125995. [CrossRef]

[10] Kim, E., Park, S., Lee, H., Ko, S., & Hwang, E. (2025). Enhanced Remaining Useful Life Prediction for Turbofan Engines Using Spatio-Temporal Koopman Dual-branch Transformer. *IEEE Transactions on Instrumentation and Measurement*. [CrossRef]

[11] Lin, L., Wu, J., Fu, S., Zhang, S., Tong, C., & Zu, L. (2024). Channel attention & temporal attention based temporal convolutional network: A dual attention framework for remaining useful life prediction of the aircraft engines. *Advanced Engineering Informatics, 60*, 102372. [CrossRef]

[12] Zhang, M., He, C., Huang, C., & Yang, J. (2024). A weighted time embedding transformer network for remaining useful life prediction of rolling bearing. *Reliability Engineering & System Safety, 251*, 110399. [CrossRef]

[13] Liang, P., Li, Y., Wang, B., Yuan, X., & Zhang, L. (2023). Remaining useful life prediction via a deep adaptive transformer framework enhanced by graph attention network. *International Journal of Fatigue, 174*, 107722. [CrossRef]

[14] Xue, J., & Shen, B. (2020). A novel swarm intelligence optimization approach: sparrow search algorithm. *Systems Science & Control Engineering, 8*(1), 22-34. [CrossRef]

Hao Wu is currently pursuing the B.S. degree in automation with the School of Internet of Things Engineering, Jiangnan University, China. His research interests include deep learning, prognostics and health management (PHM), and autonomous driving systems. (Email: 18251995364@163.com)

Tianle Yin received the B.S. degree in automation from Anhui Polytechnic University, China, in 2023. He is currently pursuing a M.S. degree in control science and engineering at the School of Internet of Things Engineering, Jiangnan University, China. He serves as a reviewer for *Scientific Reports*, and *International Journal of Communication Systems*. His research interests include megaconstellation cluster management and configuration maintenance. (Email: 18905616670@163.com)