The Aeronautical Journal (2025), **129**, pp. 2004–2035
doi:10.1017/aer.2025.25

RESEARCH ARTICLE

Deep learning and similarity-based models for predicting turbofan engine remaining useful life: insights from the CMAPSS dataset

F. Isbilen[1,2], O. Bektas[3] and M. Konar[4]

[1]Graduate School of Natural and Applied Sciences, Erciyes University, Kayseri, Türkiye
[2]Aircraft Technology, Rumeli University, Istanbul, Türkiye
[3]Faculty of Engineering and Natural Sciences, Istanbul Medeniyet University, Istanbul, Türkiye
[4]Faculty of Aeronautics and Astronautics, Erciyes University, Kayseri, Türkiye
Corresponding author: M. Konar; Email: mkonar@erciyes.edu.tr

Received: 10 December 2024; **Revised:** 17 February 2025; **Accepted:** 27 February 2025

Keywords: CMAPPS; predictive maintenance; artificial neural network; similarity base

Abstract

Turbofan engines are having a growing role in modern aircraft maintenance. Due to this increase, estimation of remaining useful life (RUL) of these engines is an important area of study in the field of reliability and maintenance optimisation. In this work, we propose a hybrid approach that combines deep learning models with similarity-based methods for accurate RUL estimation. For a better comparison, we evaluate four architectures: dropout long short-term memory (LSTM), bidirectional LSTM, convolutional neural network 1D (CNN 1D), and multi-layer LSTM. The FD002 subset of NASA's Commercial Modular Aero-Propulsion System Simulation dataset is used in the case study. Root mean square error (RMSE) and mean absolute error (MAE) were used for performance metrics. The main output of the study suggests that the dropout LSTM model achieves the best prediction accuracy with an RMSE score of 26.547 and a MAE score of 18.749. It is worth noting that these are achieved despite requiring higher computational resources compared to multi-layer LSTM. Furthermore, all models had difficulties with smaller test trajectory lengths such as 50–100 due to training data imbalance. Overall, the findings highlight the promise of hybrid deep learning and similarity-based approaches for RUL prediction. However, potential advancements such as hyperparameter optimisation and data augmentation still hold potential for further improvements.

Nomenclature

ANN	Artificial Neural Network
ANNs	Artificial Neural Networks
CNN	Convolutional Neural Network
1D CNN	One-Dimensional Convolutional Neural Network
LSTM	Long Short-Term Memory
RUL	Remaining Useful Life
RMSE	Root Mean Square Error
MAE	Mean Absolute Error
MSE	Mean Squared Error
HI	Health Indicator
CMAPSS	Commercial Modular Aero-Propulsion System Simulation
PHM	Prognostics and Health Management
μ	Mean value of a dataset
σ	Standard deviation x Sensor reading
x'	Transformed sensor reading
y	Predicted Health Indicator value
a, b, c	Coefficients of power-law fitting function

© The Author(s), 2025. Published by Cambridge University Press on behalf of Royal Aeronautical Society.

https://doi.org/10.1017/aer.2025.25 Published online by Cambridge University Press

1.0 Introduction

Turbofan engine is one of the most critical components in modern aircraft systems. When engines become unserviceable, it not only affects flight safety, but also affects maintenance operations negatively and reduces aircraft availability. This kind of unwanted situation may lead to substantial financial losses, diminished customer loyalty and weakened brand reputation for airlines [1]. For this reason, to maintain engine performance within safety limits and meet the demands of intensive flight schedules, regular maintenance is essential [2]. Although scheduled maintenance strategies are well organised within the aviation maintenance, repair and overhaul (MRO) framework [3]), one of the key point is predicting an engine's remaining useful life (RUL). This prediction method decreases unexpected failures risks. On the other hand, prediction increases operation reliability. RUL estimates how much time is left before an engine stops working. It shows the engine's current health. It helps to predict when maintenance is needed [4].

Predicting RUL has some challenges because of machines' complex behaviours. Traditional predictive models cannot handle machines' complex behaviours completely. Artificial Neural Networks (ANNs) have potential to extract meaningful points from the complex behaviours to predict RUL. These potentials are advanced capabilities in processing large, complex datasets and identifying nonlinear degeneration patterns within the data. According to the No Free Lunch (NFL) theorem [5], no single algorithm works best for every problem. This theory always encourages researchers to find better algorithms that suit a specific domain.

In this study, we trained multiple deep learning models — specifically, long short-term memory (LSTM) dropout, bidirectional long short-term memory (bidirectional LSTM) and convolutional neural network 1D (CNN 1D = 1D CNN) — and integrated them with similarity based methods. This hybrid approach leverages both the powerful representational capabilities of neural networks and the unique advantages of similarity measures in identifying subtle patterns within the data. By comparing the performance of these models, we align our methodology with the principles of the NFL theorem, optimising the selection of algorithms to address the specific requirements of RUL prediction and thereby enhance predictive performance.

1.1 Background

Every system requires maintenance at some point to operate properly. In industry, there are three main approaches to optimise system operations: reactive maintenance, preventive maintenance and predictive maintenance. Reactive maintenance is applied once a system failure has occurred, while preventive maintenance is carried out before failure happens. Predictive maintenance, on the other hand, estimates the time to system failure, helping to prevent unexpected downtime and ensuring safe system operation.

In large maintenance operations, unexpected downtime not only causes availability issues but also creates resource management challenges, such as logistics and human resources. Major airlines, with numerous destinations, face additional challenges: storing expensive backup components or maintaining staff at every base is impractical for addressing unexpected major system failures. For this reason, maintenance must be scheduled as much as possible, requiring well-organised operations. RUL prediction is thus a critical approach to ensure the availability and performance of essential components, such as engines.

Traditional RUL prediction methods consist of physical models and statistical models, both of which have certain disadvantages. For example, using a physical model to predict RUL requires domain expertise in jet engine mechanisms, while a statistical model relies on historical data, which can be challenging to collect for new systems and takes time. However, data driven approaches that are powered by machine learning and the growing accumulation of sensor data from engines have become strong alternatives. Among machine learning techniques, neural networks have shown promise in predicting RUL because they can capture complex, nonlinear relationships within the data.

Similarity-based methods have some advantages such as capturing similar relationships and patterns from previous data. These methods have a power to understand subtle degradation trends across similar

https://doi.org/10.1017/aer.2025.25 Published online by Cambridge University Press

components. This ability supports ANNs that may not be able to capture fully nonlinear behaviours. For this reason, combined ANNs models with similarity-based approaches may refine RUL predictions with nuanced patterns of dataset. This hybrid strategy gives the representational power of neural networks. It has advantage from the precision offered by similarity-based analysis. These combined features may improve prediction of model's robust and comprehensive against complex or unseen data. In addition, this approach matches with the principles of the NFL theorem. Combining these methods strengthens the model's accuracy and reliability, helping ensure the availability and operational readiness of critical components, such as engines, in complex maintenance operations.

1.2 Objectives

This paper focuses on developing models to estimate RUL of turbofan engines. We use the FD002 subset of the Commercial Modular Aero-Propulsion System Simulation (CMAPSS) dataset. Instead of applying combined models for feature extraction and prediction, we will train deep learning models like LSTM, Bidirectional LSTM and 1D CNN and then we will apply similarity-based methods to improve accuracy. These techniques help identify degradation patterns and make predictions. This study addresses specific gaps identified in existing RUL prediction approaches, such as the challenges of accurately modeling complex, nonlinear machinery behaviour and the limitations of traditional physical and statistical models. Our work contributes to the field of RUL prediction by presenting a hybrid methodology that combines neural networks with similarity-based methods, providing a more robust framework for accurate RUL estimation. Additionally, by comparing the performance of our models against related works, this study offers insights into the effectiveness of advanced data-driven approaches for predictive maintenance.

1.3 Scope of the study

This study focuses on the application of deep learning and similarity-based methods for predicting the RUL of turbofan engines, specifically using sensor data from the CMAPSS dataset's FD002 subset. The primary emphasis is on observing the performance of similarity-based methods in combination with neural network models, namely LSTM, Bidirectional LSTM and 1D CNN consistent conditions. The study does not aim to optimise hyperparameters; instead, all models will be trained using default parameters to ensure that performance comparisons are centred on the similarity-based method's impact.

While this approach provides insights into the potential of combining neural networks and similarity-based methods for RUL prediction, it has limitations. This research does not address hyperparameter tuning, alternative machine learning models, or applications beyond the chosen dataset and industry focus. Additionally, findings may not directly generalise to other types of equipment or industries, as our scope is limited to a specific dataset and predefined models. By clearly defining these boundaries, the study aims to offer a focused evaluation of similarity-based methods in RUL prediction within a controlled framework.

2.0 Literature review

2.1 Related works

The purpose of prognostics is to improve health management systems by providing better warnings about performance degradation and impending failures [6]. Prognostics methods can be classified as data-driven method, physics-based methods and hybrid methods. The data-driven approach uses historical data to identify degradation patterns and predict future trends. On the other hand, the physics-based approach assumes the existence of a physical model that describes the degradation pattern and combines these physical models with measured data to identify degradation patterns and predict future trends [7]. Hybrid method combines above mentioned methods to improve the prediction performance [8].

Data-driven approaches are generally categorised into two main classes: artificial intelligence-based approaches, which include techniques such as neural networks [9], fuzzy logic [10] and statistical approaches [11], which encompass methods like regression and support vector machines [12].

https://doi.org/10.1017/aer.2025.25 Published online by Cambridge University Press

Physics-based RUL prediction is another method researchers use to forecast how systems degrade over time. It works by gathering real data either from carefully controlled experiments or from sensors that monitor the system in real time [13]. Thanks to modern sensor technology, collecting this data has become more practical, but the setup can still be expensive. A study by Downey and co-authors [14] introduced a new mechanistic approach to battery prognostics that builds on traditional capacity-based models for RUL predictions. They track battery wear with a half-cell setup. After that, they feed the data into a model that uses a non-linear least squares solver. The standout feature of this method is the addition of dynamic bounds to the solver, which adjust model coefficients within set limits as the battery ages. This makes the predictions more stable and reliable. One big advantage of this physics-based approach is its ability to capture multiple causes of battery degradation, leading to more accurate results than older methods. In simulated tests, the new approach delivered a 97.5% improvement over capacity-based models under similar conditions. Thanks to the dynamic bounds that reduce uncertainty, this method holds a lot of promise for future battery management systems (BMSs).

Hybrid methods do not have standardised approaches in the literature, although there are examples of machine learning (ML), neural network, or physics-based models used by researchers for RUL prediction. Hybrid prognostic models can be implemented by combining experience-based models with data-driven models [15], experience-based models with physics-based models [16], or data driven models with physics-based models [17]. Liao et al. conducted an additional experiment employing hybrid methods for battery life prediction. Liao et al.'s research [8] is pivotal in assessing the efficacy of hybrid prediction methodologies for RUL estimation. Hybrid methods offer promising potential by combining the strengths of data-driven, physics-based and experience-based models to address the inherent weaknesses of individual approaches. By blending these techniques, we tap into practical know-how, upkeep insights and up-to-the-minute performance details. This mix can sharpen and steady our forecasts. It also manages tricky setups more effectively than relying on a single method, which might miss the full picture of how things wear down. Blending physics-based insights can help plug gaps left by limited historical data, while data-driven models can quickly adjust to changing operating conditions. Yet, hybrid prognostics methods come with their own challenges, and these can slow or even block real-world adoption. One of the biggest hurdles is figuring out how to combine different data types such as continuous signals, periodic measurements, and rule-based knowledge into one cohesive framework. Developing robust methodologies to address inconsistencies and reconcile missing data is essential but complex. Furthermore, aggregating predictions from competing models remains problematic, as there is no standardised approach to determine which model's output should be prioritised or how to generate a unified prediction. Another significant issue is the dependence on data availability and quality. While data-driven models require large, comprehensive datasets, physics-based models depend on detailed system knowledge and accurate degradation representations, which can be challenging to obtain for complex systems. Additionally, hybrid methods struggle with managing prediction uncertainty, especially when addressing subjective and objective uncertainties across different time horizons. The absence of systematic methodologies to overcome these challenges undermines the reliability and scalability of hybrid prognostic approaches in real-world scenarios.

Isbilen et al. attempted to predict RUL using ML models to evaluate their performance [18]. While their model performances were satisfactory, they still required significant improvements. Additionally, the sensor selection method relied on observing normalised sensor values, which lacked the robustness needed to justify sensor selection effectively. Furthermore, their approach did not provide insights into improving model performance.

In this paper, we concentrate on ANNs, a common data-driven tool in prognostics. Many studies simply predict RUL directly. We do something different. Our model learns the wear patterns over time. During training, we use a health indicator (HI) curve primarily concentrating on spotting engine wear. Neural networks help decrease sensor data noise side effects and make our results more trustworthy. Once the data is cleaned, we look for matching patterns in our training set that resemble the new cases. This matching process helps in getting accurate RUL predictions. By pairing neural networks

https://doi.org/10.1017/aer.2025.25 Published online by Cambridge University Press

with pattern matching, we offer a different combined way to predict remaining useful life and handle noise.

In this paper, we focus on ANNs that is one of the most widely used data driven approaches to prognostics. Unlike many studies in the literature that directly predict RUL as the model's output. On the other hand, our approach trains the model to capture and predict the degeneration patterns in the data. During training, we utilise a [19] (HI) curve specifically designed to identify engine degradation patterns. This approach helps us to take advantage of ANNs to reduce noise in sensor data. It improves the reliability of the predictions. Following this noise reduction step, we apply a similarity-based technique to identify the best match between test patterns and the training data that is estimated as a full degeneration pattern. This helps us to predict RUL estimations with reliable degeneration patterns since the training dataset provides complete degradation patterns, which are essential for accurate comparisons. Although there are a lot of combined usages of ANNs in literature, they use first models to extract features. In this paper, we do not use any pre model to extract features before model training that will predict RUL. Feature extraction is applied in pre pre-processing section and useful sensors are selected by mathematical method. From this aspect, usage of ANNs with a similarity-based approach introduces a new perspective on RUL prediction and handling noise management in sensor data. The following section will explain details about the application of ANNs for RUL prediction.

2.2 Application of artificial neural network in RUL prediction

ANNs have the power that can adapt to new conditions. They can extract meaningful points from complex datasets during training. For this reason ANNs are a very common tool to solve various problems [20–24]. Their capacity to handle complex data provides an advantage in solving significant problems, such as RUL prediction [24–28].

An artificial neural network (ANN) consists of three types of layers: an input layer, one or more hidden layers and an output layer [20]. The network operates as follows: (i) the input layer collects data and forwards it to the first hidden layer; (ii) neurons in the hidden layers perform computations using a weighted linear combination of the inputs, apply an activation function, and transmit the result to the next layer; and (iii) this process continues until the output layer generates the final prediction. This architecture has allowed ANNs to excel in various applications, including RUL prediction.

While ANNs offer significant advantages, they also come with challenges in RUL prediction. First, ANNs can require large, high-quality datasets for effective training. High quality data may not always be available. Noisy or incomplete sensor data can negatively affect the performance of the network. Second, neural networks often operate as '"black-box' models. Black box makes the difficult to explain prediction and explain model decision reasons. This lack of transparency can limit trust in critical applications. Finally, neural networks may struggle with generalisation. This is especially true in dynamic or changing environmental conditions. They learn patterns from training data without built-in adaptability.

A significant gap in current neural network-based RUL prediction lies in handling noisy data and accurately generalising predictions across different scenarios. This study addresses these gaps by employing a HI curve to recognise degradation patterns in engine data. By training the ANN to predict degeneration patterns rather than directly outputting the RUL, this approach reduces the impact of noisy data and improves the network's ability to generalise. Additionally, the use of a similarity-based method to match test patterns with training data provides a novel perspective for RUL estimation, leveraging the strengths of the ANN while mitigating its challenges. This combined methodology enhances prediction accuracy and reliability, offering a robust solution to the limitations of traditional approaches and existing ANN models.

The estimation of RUL using ANNs has broad applications across industries, helping to avoid unexpected maintenance operations, production losses, shipping failures, and poor customer satisfaction [29]. For example, battery lifetime estimation [30] ensures the safe and reliable operation of battery systems, while bearing RUL estimation is critical for diagnosing faults in rotating machinery [31]. Furthermore, RUL estimation in multi-rotor unmanned aircraft systems (UAS) helps mitigate potential propulsion

https://doi.org/10.1017/aer.2025.25 Published online by Cambridge University Press

Table 1. *FD002 sub-dataset*

Dataset	Engine Quantity	Feature Quantity	Trajectory Quantity
FD002 Train	260	28	53,759
FD002 Test	259	28	33,991
FD002 RUL	259	1	

system errors [32]. These examples demonstrate the transformative impact of ANNs in addressing complex predictive maintenance challenges in diverse fields. This work provides deeper insights into model performance reasoning and improvements, which were not discussed in the previous work [18, 21, 22, 33–35].

3.0 Methodology

3.1 Dataset description

The CMAPSS dataset, originally introduced by Saxena and Goebel in 2008 as part of the Prognostics and Health Management (PHM) Challenge, is a widely recognised benchmark in the field of PHM [36, 37]. This dataset provides simulated degradation trajectories for a fleet of large turbofan engines, enabling researchers to develop and evaluate predictive maintenance models [8].

In this study, we specifically used the FD002 sub-dataset from CMAPSS to train machine learning models for RUL prediction. The FD002 subset was chosen because it includes six operating conditions and moderate levels of operational variability, making it a more challenging yet practical dataset for real-world applications.

3.2 Dataset structure and available data

The FD002 training dataset consists of 260 engines, contributing to a total of 53,759 operational cycles. The testing dataset contains 259 engines, spanning 33,991 operational cycles, as detailed in Table 1. Each engine is run until failure, providing valuable degradation patterns essential for RUL estimation.

Each instance in the dataset is represented by 26 columns, categorised as follows:

- Metadata (5 columns):
  - Unit number (identifies the engine)
  - Time in cycles (operational cycle count)
  - Operational setting 1, 2 and 3 (engine operating conditions)
- Sensor readings (21 columns):
  - These correspond to sensor measurements 5 to 25, capturing various engine health indicators.

One of the key challenges of the FD002 dataset is that it does not explicitly define which sensor corresponds to which engine condition. To address this, we analysed sensor variations across different operating conditions and identified condition-relevant sensors for clustering purposes. As shown in Table 2, these sensors were used to differentiate engine health states, further enhancing the accuracy of our RUL predictions.

3.3 Regime normalisation

Figure 1 displays a sample of multi-dimensional trajectory values. The data shown is solely from sensor 6 for illustrative purposes. These sensor measurements align under various operating conditions (regimes) and condition monitoring measurements.

https://doi.org/10.1017/aer.2025.25 Published online by Cambridge University Press

Table 2. *Unique sensor values for each sensor*

Sensor Name	Unique Sensor Values
5	6
6	1,604
7	14,289
8	17,867
9	1
10	14
11	2,067
12	897
13	24,063
14	11
15	836
16	1,672
17	942
18	23,221
19	10,518
20	3
21	67
22	1
23	1
24	533
25	17,867

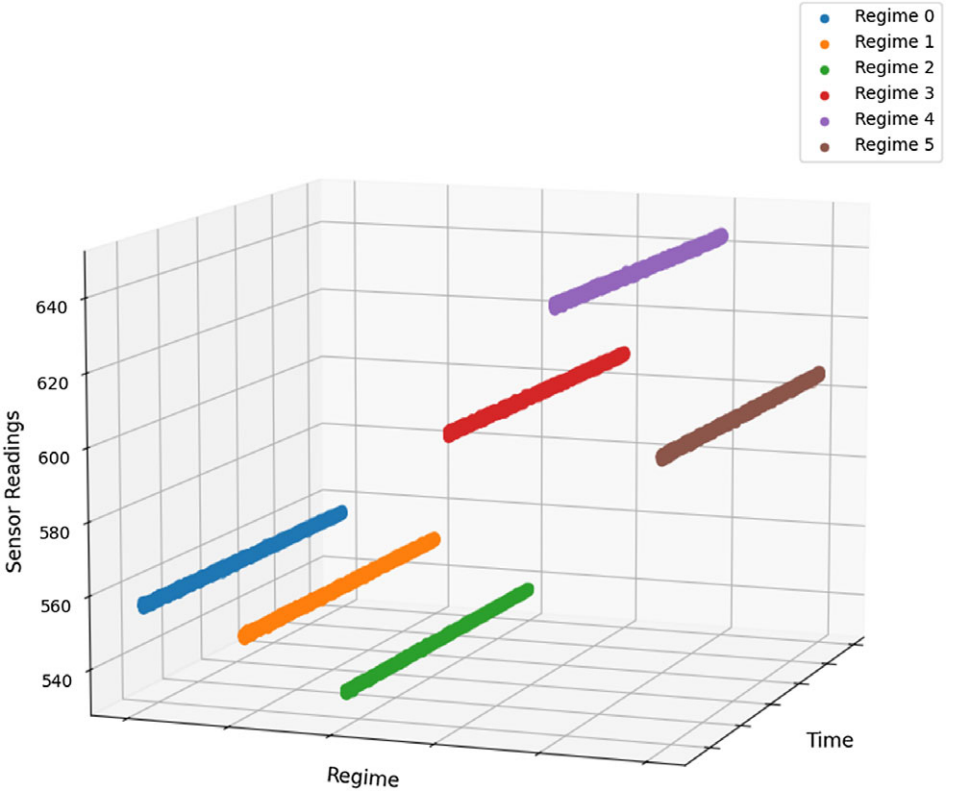


Figure 1. *Multi-dimension sensor values.*

https://doi.org/10.1017/aer.2025.25 Published online by Cambridge University Press

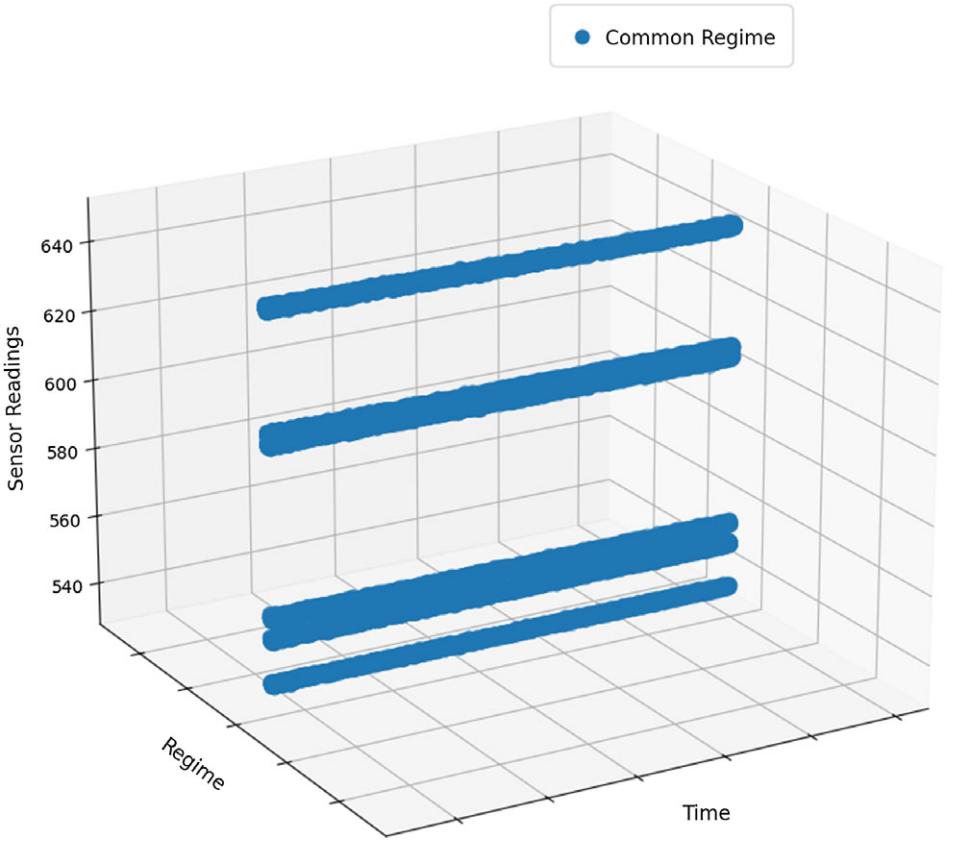


Figure 2. *Single regime sensor values.*

The science of prognosis is based on system aging and monotonic damage accumulation. Therefore, it should be possible to correlate sensor behaviour with signs of aging to estimate a system's RUL [38]. Monotonicity is a crucial feature in prognostics for predicting RUL. The monotonicity approach generally assumes that systems do not self-heal, and their lifespan decreases with use. Thus, the indicator should present non-monotonic values to describe RUL. However, this assumption may not hold for components like batteries, which may undergo some self-repair during short non-use periods [39].

In this paper, since jet engine components are mechanical, we assume monotonicity in predicting RUL. Consequently, one goal of data preprocessing is to transform data into a monotonic form. Nonetheless, multi-dimensional data, as shown in Fig. 1, may not offer useful information for measuring monotonic damage accumulation. Additionally, the monotonicity character will be utilised in the useful sensor selection, as discussed in section 3.4.

In Fig. 2, the standardised sensors are shown after multiple regime normalisation for a single vector ($ns = 1$). The differences between regimes are eliminated, transforming the data into a format that permits only one valid regime for all sensors. This process is applied separately to each regime and the standardised sensors are then combined to form the normalised dataset.

3.4 Z score normalisation

Normalisation is a frequently used preprocessing method before analysis [40]. It's applied when a dataset values has a broad range of values, aiming to standardise all values within a common scale. In this paper,

https://doi.org/10.1017/aer.2025.25 Published online by Cambridge University Press

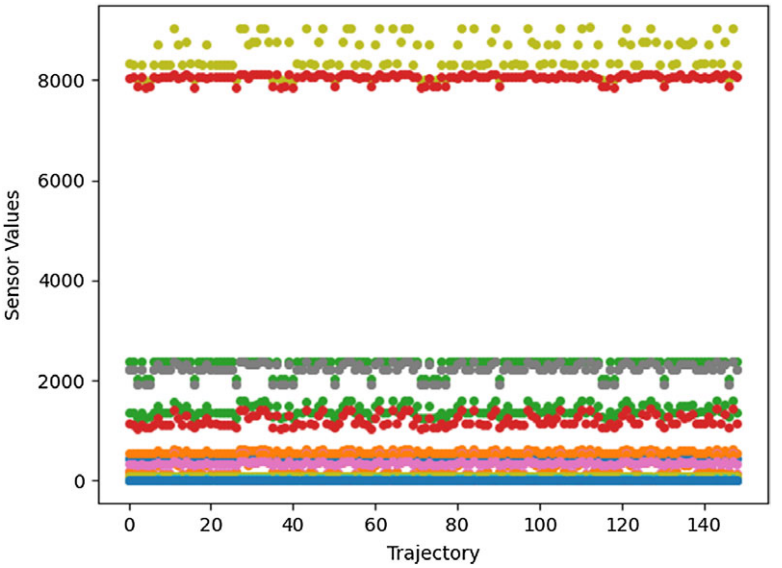


Figure 3. *Train engine 1 raw sensor values.*

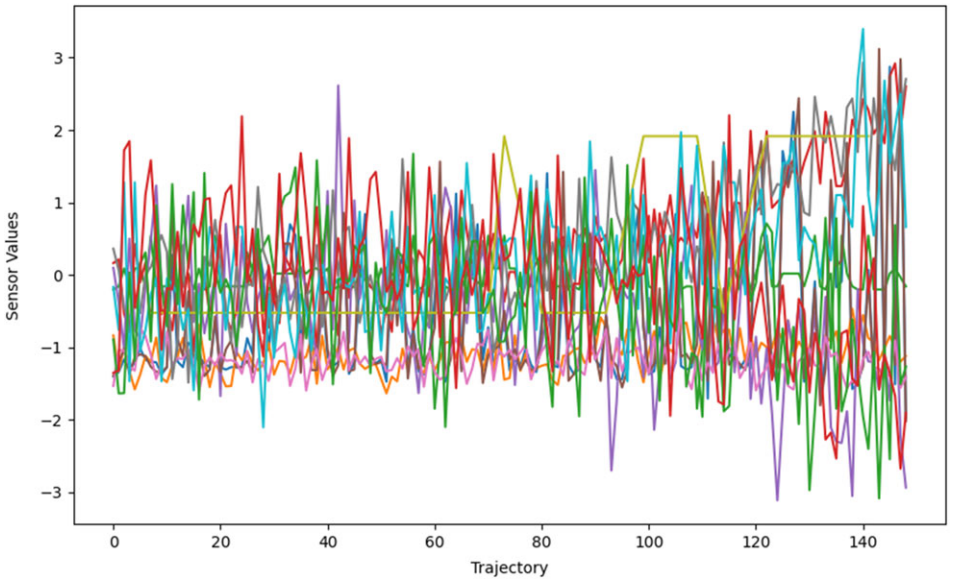


Figure 4. *Train engine 1 normalised sensor values.*

we implemented the z-score normalisation technique, as elaborated in the Equation (1):

$$z = \frac{x - \mu}{\sigma} \tag{1}$$

where x is the raw score, μ is the mean of the population and σ is the standard deviation of the population.

As shown in Fig. 3, there are large gaps between sensor values. To bring these values into a common scale, we applied the z-score normalisation technique to all train sensor values after regime clustering. The normalised sensor values for train engine 1 are depicted in Fig. 4. After applying z-score normalisation, the sensor values across the train dataset share a common range.

https://doi.org/10.1017/aer.2025.25 Published online by Cambridge University Press

3.5 Sensor selection

To effectively characterise the degradation model, it is essential to reduce data dimensionality while selecting the most informative sensors. In this section, we systematically evaluate sensor data using established metrics in PHM [41]. Specifically, we assess monotonicity, prognosability and trendability, which serve as key indicators in identifying suitable sensors for degradation modeling.

The methodologies employed for sensor selection are based on the formalisations proposed by Coble and Hines [42]. Below, we provide the mathematical framework for each metric, followed by a discussion of the ranking process.

3.5.1 Mathematical framework

Monotonicity. Monotonicity quantifies the consistency of an increasing or decreasing trend in sensor measurements over time. A higher monotonicity score (closer to 1) suggests that the sensor data exhibits a strong, unidirectional trend, which is desirable for degradation modeling. The metric is computed as follows:

$$monotonicity = \frac{1}{M}\sum_{j=1}^{M}\left(\frac{1}{N_j - 1}\sum_{k=1}^{N_j-1} sgn\left(x_j\left(k+1\right) - x_j\left(k\right)\right)\right. \tag{2}$$

where:

- M = total number of units
- Nj = number of measurements for unit j
- xj(k) = measurement at time step k for unit j
- sgn = sign function, which determines whether consecutive measurements are increasing or decreasing

A monotonicity score closer to 1 indicates a strictly increasing or decreasing trend, whereas values closer to 0 imply inconsistent patterns.

Prognosability. Prognosability assesses the ability of a sensor to distinguish between healthy and failure states. A sensor with high prognosability has distinct failure and healthy state distributions, facilitating reliable predictions. The metric is defined as:

$$Prognosability = exp\ exp\ -\ \frac{\sigma failure}{\sigma\,\mu failure - \mu healhy}\Bigg) \tag{3}$$

where:

- μfailure = mean of sensor measurements in the failure state
- μhealthy = mean of sensor measurements in the healthy state
- σfailure = standard deviation of sensor measurements in the failure state

This formulation ensures that:

- A larger separation between failure and healthy states (μfailure - μhealthy) improves prognosability.
- Higher variation in failure states (σfailure) reduces prognosability.

A prognosability score closer to 1 indicates a clear distinction between failure and healthy states, making the sensor more reliable for degradation tracking.

https://doi.org/10.1017/aer.2025.25 Published online by Cambridge University Press

Table 3. *Sensor data analysis*

Sensor	Monotonicity	Prognosability	Trendability	Overall Score
15	1.000000	0.869538	0.925333	2.794870
8	1.000000	0.841880	0.896758	2.738638
19	1.000000	0.753498	0.913214	2.666712
21	1.000000	0.770301	0.872485	2.642786
7	1.000000	0.742665	0.818189	2.560854
6	1.000000	0.760700	0.737973	2.498673
25	1.000000	0.618865	0.648781	2.267646
16	1.000000	0.577912	0.658732	2.236644
11	1.000000	0.560245	0.637888	2.198133
24	1.000000	0.610382	0.295425	1.905807
10	0.999359	0.576114	0.262231	1.837704
12	0.984666	0.430482	0.000050	1.415198
17	0.986346	0.424010	0.000475	1.410831
5	0.992436	0.413297	0.000041	1.405774
13	1.000000	0.315761	0.000068	1.315828
18	1.000000	0.293027	0.000135	1.293162
9	0.000000	NaN	NaN	NaN
14	0.935795	NaN	NaN	NaN
20	0.858324	NaN	NaN	NaN
22	0.000000	NaN	NaN	NaN
23	0.000000	NaN	NaN	NaN

Trendability. Trendability evaluates whether different units follow a consistent degradation pattern that can be represented by a common function. Due to sensor noise, we use a correlation-based approach:

$$\text{Trendability} = \min(|corrcoef_{ij} \tag{4}$$

where:

- $corrcoef_{ij}$ = linear correlation coefficient between sensor trajectories of units i and j
- min ($|corrcoef_{ij}|$) = the smallest absolute correlation value across all unit pairs

A higher trendability value suggests that sensor readings exhibit a consistent degradation pattern across multiple units, making the sensor more suitable for prognostics modeling.

3.4.2 Sensor ranking and selection

The above metrics were computed for each sensor in the dataset. Sensors were ranked based on the sum of their monotonicity, prognosability and trendability scores. The final rankings are presented in Table 3.

Top-performing sensors, based on overall scores, include sensors 15, 8, 19, 21, 7, 6, 25, 16, 11 and 24. These sensors were selected as they provide the most reliable and consistent degradation indicators.

3.4.3 Sensor data normalisation and transformation

To ensure that all selected sensors exhibit a consistent degradation trend, a directional transformation was applied. Specifically, sensors 11, 16, 24 and 25 exhibited negative directional slopes as seen in Fig. 5.

To standardise degradation trends, the following transformation was applied to these sensors:

$$x' = -x.$$

https://doi.org/10.1017/aer.2025.25 Published online by Cambridge University Press

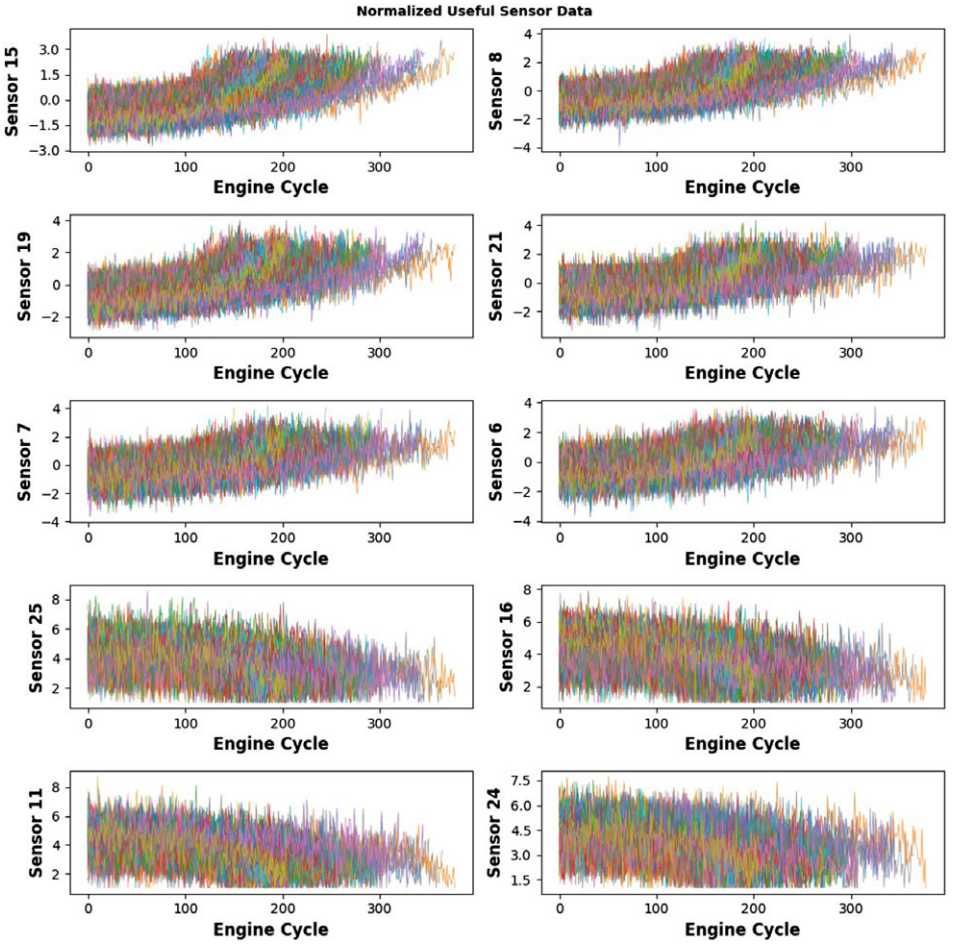


Figure 5. *Useful sensor data sample.*

This adjustment ensures that all useful sensors exhibit a positive slope, facilitating the accurate construction of HI patterns. The transformed sensor values are displayed in Fig. 6.

3.6 Health indicator (HI) pattern determination

HI curve enables the identification of degeneration patterns [19]. Mechanical components typically exhibit a linear HI curve. Consequently, we assume that an engine's HI curve should follow a linear pattern of degeneration. However, this assumption does not apply to components like batteries or others that have self-recovery or recharging capabilities, as their HI curves behave differently and require alternative approaches for RUL prediction [42].

Useful sensor values remain multi-dimensional, as shown in Fig. 6. However, multi-dimensional data is not ideal for model training, as it can negatively affect model performance. Since the HI curve will be used as the target data for training, the training data should be as simplified as possible. For this reason, the dimensions of the multi-dimensional sensor data are reduced by calculating the mean of each sensor across the related trajectory to produce the HI curve. In this way, we generate an HI curve for each engine, as shown in Fig. 7. However, the generated HI curve is still not suitable for model training due to the presence of noise. To address this issue, we apply a power-law curve-fitting approach, which helps smooth the data and reduce the impact of noise. The resulting HI curve, after applying the power-law method, is illustrated in Fig. 8.

https://doi.org/10.1017/aer.2025.25 Published online by Cambridge University Press

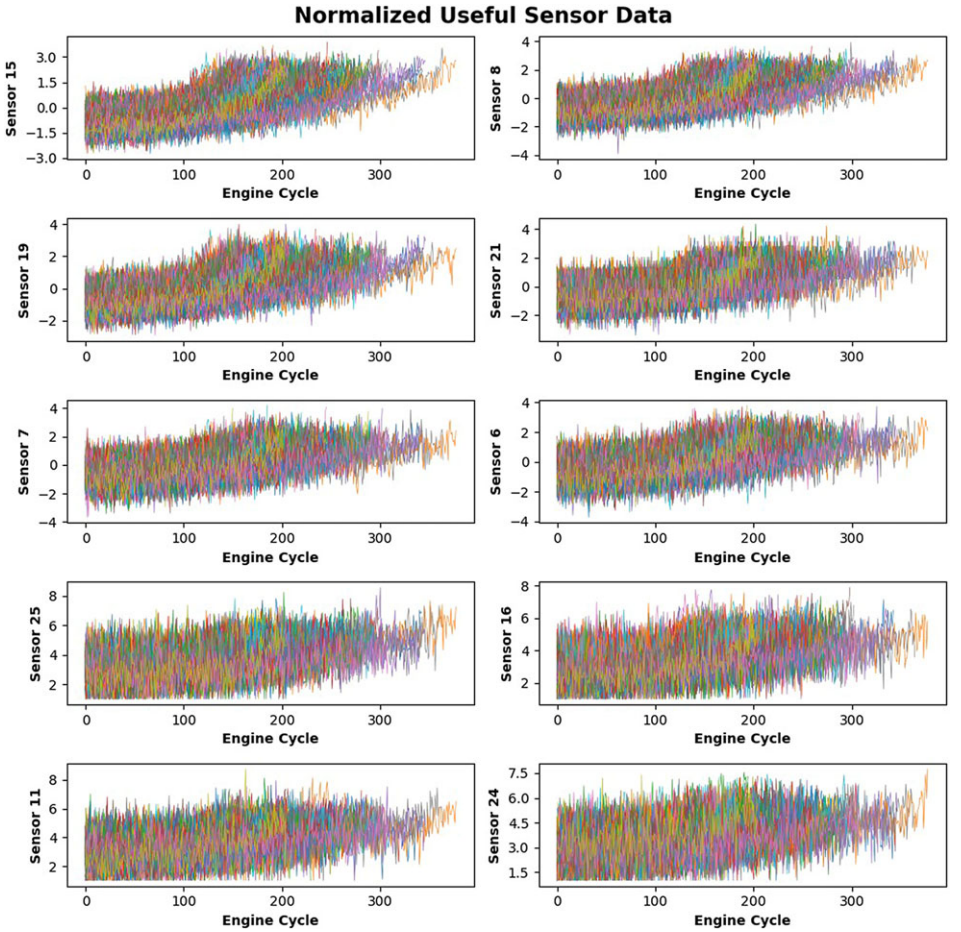


Figure 6. *Useful sensor data after transformation.*

Explanation of the power-law approach

The power-law curve-fitting approach models the relationship between the input x (e.g. time or index) and the output y (e.g. HI values) using a mathematical function of the form:

$$y = a\,x^{b} + c, \tag{5}$$

Here, a is a scaling factor, b is the exponent that determines the growth or decay behaviour, c is a constant offset.

The fitting process involves the following steps:

1. *Define the power-law function:* The function $y = a\,x^{\mathrm{b}} + c$ is used to represent the relationship between x and y.
2. *Fit the polynomial:* A second-degree polynomial is first fitted to the data to provide a baseline smoothed curve.
3. *Fit the power-law function:* Parameters a, b, and c are determined by minimising the error between the raw data and the power-law function. This step ensures that the power-law model closely approximates the observed HI values while reducing noise.

https://doi.org/10.1017/aer.2025.25 Published online by Cambridge University Press

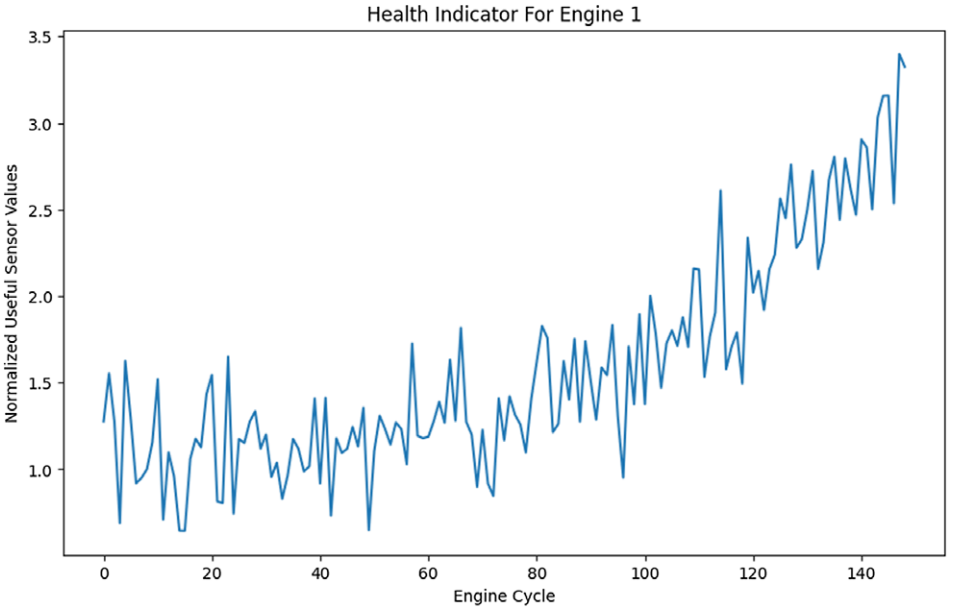


Figure 7. *Health indicator for engine 1.*

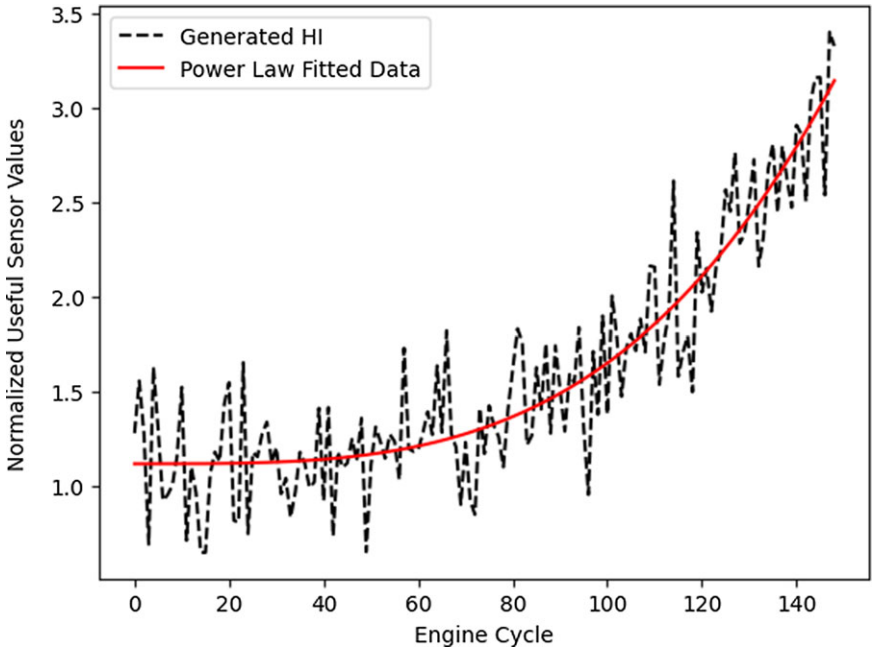


Figure 8. *Power law fitted HI.*

4. *Generate smoothed HI curves:* Using the fitted parameters, the power-law function is applied to create smoothed HI curves that are more robust and better capture the underlying trends in the data.

https://doi.org/10.1017/aer.2025.25 Published online by Cambridge University Press

The power-law fitted HI curves effectively highlight the trends in the normalised sensor values, making them more suitable for model training and subsequent analysis for RUL prediction.

3.7 Artificial neural network model development

This section explains the development of ANN models for HI estimation. We explain each model's structure, unique features and design explanation.

3.7.1 1D convolutional neural network (CNN)

CNN is one of the most famous structures in the ANN era. Its history dates back to the early 1940s [40]. Researchers have drawn inspiration from human neural systems to build CNN models [18, 33, 43–46]. CNN models are used in various applications such as object detection, speech recognition [45], structural health monitoring [46], structural damage detection [44], high-power engine fault monitoring [43], UAS flight damage detection [47] and early arrhythmia detection in electrocardiograms (ECGs) [43]. 2D CNN structures are commonly used for image or video processing. In contrast, 1D CNN structures are used for anomaly detection and time series applications. In this paper, we use a 1D CNN model. It offers advantages for signal processing applications.

The proposed 1D CNN model is designed to efficiently process sequential data by leveraging a layered architecture that captures both local and global features in the input signals. The first layer applies a one-dimensional convolution with 64 filters and a kernel size of 3 to extract low-level temporal features from the input sequence. The rectified linear unit (ReLU) activation function introduces non-linearity, while batch normalisation stabilises the learning process by normalising the output of the convolutional layer. Max pooling layer reduces the feature maps by half, since the max pooling layer's pool size is 2. It keeps important features and lowers the computational load. A 40% dropout layer is added. The dropout layer helps prevent overfitting since drop out layers randomly deactivate some neurons during training.

A second convolutional layer follows. It uses 128 filters with the same kernel size to extract more complex features. The model uses ReLU as activation function and batch normalisation since we use raw training data and max pooling for feature extraction and dropout again for generalisation. During model training, these layers provide more opportunity for the model learning. They also make its predictions more reliable.

To aggregate global information across the entire sequence, a global average pooling layer is incorporated, reducing the feature maps to a single vector while retaining the most critical information. A fully connected dense layer with 50 neurons and ReLU activation is added, serving as a bridge to the final output layer. Another dropout layer with a 40% rate ensures robust generalisation. The output layer comprises a single neuron, providing the final prediction for the target variable.

This architecture is optimised for sequential data processing, balancing feature extraction, model complexity and regularisation to achieve high performance in signal processing tasks. The model is trained using a mean squared error (MSE) loss function to minimise prediction errors while leveraging an adaptive optimisation algorithm to enhance convergence.

3.7.2 LSTM model

LSTM is a special type of recurrent neural network (RNN). This design allows for useful challenges that need temporal sequences and learning long range dependencies in sequential data. This useful feature of LSTM designs makes them popular in many tasks. These tasks are such as language modeling, word embedding, handwriting recognition and speech recognition [46]. Traditional RNNs sometimes have trouble during training. The gradients can become too small or too large, which makes learning difficult. LSTMs solve these issues with a unique design. Their architecture includes memory cells and gates to be useful for temporal sequences and learning long range dependencies in sequential data.

The flowcharts in the LSTM model section Fig. 10 illustrate the implementation of two different LSTM architectures: multi-layer LSTM and dropout LSTM. These models are trained on sensor data to estimate the health index of components. A similarity-based approach is then employed to predict

https://doi.org/10.1017/aer.2025.25 Published online by Cambridge University Press

the RUL by matching the test data with the most similar training instances. The final RUL prediction is derived by averaging the five best-matched sequences. The performance of both models is evaluated using RMSE and MAE, and the best-performing LSTM model is selected accordingly.

Core architecture of LSTMs

An LSTM cell consists of three key gates — input gate, forget gate and output gate — that regulate the flow of information through the network:

- *Input gate*: Determines which new information should be added to the cell state
- *Forget gate:* Controls what information from the previous cell state should be discarded
- *Output gate:* Selects the relevant information from the cell state to be sent as the output or hidden state

The cell state acts as a memory mechanism, allowing LSTMs to retain or forget information over long sequences effectively. This structure enables LSTMs to capture both short-term patterns and long-term dependencies in the data.

LSTM model description

The LSTM model developed here is designed to process sequential data of shape (10, 1), where each input sequence consists of 10 timesteps with a single feature per timestep. The architecture incorporates both LSTM layers and dropout regularisation to enhance the model's performance and generalisability.

Design of the dropout LSTM model

First LSTM layer: First layer of the proposed LSTM model begins with an LSTM layer with 128 Unit. First layer activation function is tanh. Tanh allows the network to learn complex relationships within data. In LSTM layer extracts the features from representations for the next layer.

First dropout layer: 30% drop layer is applied after the first LSTM layer as the first dropout layer of the model. Dropout layer helps to avoid overfitting via randomly removing a fraction of neurons during training. This regularisation increases model generalisation for unseen data since dropout reduces reliance on specific neurons.

Second LSTM layer: The second LSTM layer of the proposed model has 64 units. This layer uses the temporal features extracted by the first layer. The layer squeezes the sequential data into a smaller form. It enables the network to focus on the most important patterns. tanh is applied as an activation function again to handle time-based changes while keeping the system stable.

Dense and dropout layers: The feature vector that is generated by the second LSTM layer of the purposed model converted to a full connected dense layer with 32 neurons. ReLu function is used in dense layer. This dense layer handles non-linearity and allows learning of high-level representations tailored to output. 30% dropout layer is applied again to avoid overfitting in the dense layer.

Output layer: The final dense layer is a single neuron. It is provided as model prediction for the target variable.

Optimisation and loss function

Adam optimiser is used for model compilation. The dynamic learning rate block helps keep training efficient. The MSE loss metric helps prediction errors during model training. It makes the architecture suitable for tasks that involves continuous target variables. This model training configuration helps optimise the model training term while maintaining generalisation and robustness.

Multi-layer LSTM model

Multi-layer LSTM model is built to improve sequential data processing. Advanced regularisation and normalisation techniques are applied belong LSTM layers. This architecture aims to catch

https://doi.org/10.1017/aer.2025.25 Published online by Cambridge University Press

short- and long-term temporal features. In addition, the model aims to keep models robust generalisation and stability during training.

Layered design of the multi-layer LSTM model

First LSTM layer with batch normalisation: The first layer of the multi-layer LSTM model begins with 128 units. First layer activation function is tanh. This layer processes the sequence and outputs a rich representation of temporal patterns for downstream layers. After first layer, batch optimisation is applied. Since we use useful sensor raw data, batch normalisation allows to normalise the output and accelerate convergence. 30% drop layer is also applied to avoid overfitting. Dropout layer removed random neurons during training.

Second LSTM layer with batch normalisation: A second LSTM layer has 64 units in the multi-layer LSTM model. This layer refines the temporal features that are extracted by the first layer. This layer turns sequential data into a simpler. It makes data more compact form. It allows to retain only the most critical information for the output. Like the first layer, it is followed by batch normalisation to standardise its outputs and improve the model's stability . 30% drop layer is applied again to avoid overfitting. Dropout layer removed random neurons during training.

Dense layers with regularisation: The feature vector that is generated by the second LSTM layer of the multi-layer LSTM model converted to a full connected dense layer with 64 neurons. ReLu function is used in dense layer. This dense layer handles non-linearity and allows learning of high level representations tailored to output. 30% dropout layer is applied again to avoid overfitting in the dense layer. The next dense layer has 128 neurons. ReLu function is used in the next dense layer. It allows the learn high level features. 30% dropout layer is applied again to avoid overfitting in the next dense layer.

Output layer: The final dense layer is a single neuron. It is provided as model prediction for the target variable.

Optimisation and loss function

This architecture is compiled with the Adam optimiser, leveraging its adaptive learning capabilities to ensure efficient convergence during training. The MSE loss function is employed, making the model ideal for continuous target variables.

3.8 Model training

The training process has the same pipeline for all models that are mentioned in this paper. The pipeline contains data preprocessing, hyperparameter tuning and model evaluation. This pipeline helps to optimise model training.

Pre configuration during the model training is very important. The configuration helps increasing model performance, avoiding overfitting and ensuring efficiency for source management such as making hardware busy during training, time and human source etc. Training settings play a big role in how well the model learns. Important factors include learning rate scheduling, early stopping and model checkpointing. These help the model learn efficiently and perform well on new data.

- Learning rate scheduler: The learning rate determines how much the model's weights are updated during training. A dynamic learning rate schedule helps balance rapid convergence during initial epochs with fine-tuning in later epochs, enhancing the model's performance.
- Early stopping: This mechanism prevents overfitting by halting training when the model's performance on validation data stops improving. It also reduces unnecessary computational effort.
- Model checkpointing: Saving the best-performing model during training ensures that the optimal version of the model is preserved for evaluation and deployment, even if subsequent epochs degrade performance.

https://doi.org/10.1017/aer.2025.25 Published online by Cambridge University Press

By configuring these mechanisms effectively, we ensure a training process that is both efficient and robust.

3.8.1 Configurations applied in this study

The following configurations were applied during the training of the all models:

1. *Learning rate scheduler:* A dynamic learning rate was implemented to adjust during training:
   - Epochs 0–4: $\eta = 0.001$
   - From Epoch 5 onward: $\eta = 0.0001$

This approach helped achieve rapid convergence initially and refined the training process in later epochs.

2. *Early stopping:*
   - Monitored the validation loss (val_loss)
   - Halted training if no improvement was observed for 30 consecutive epochs, preventing overfitting and saving computation time
3. *Model checkpointing:*
   - Retained the best model based on the lowest validation loss
   - Saved with as keras format

These configurations ensured efficient training and optimal model generalisation.

3.8.2 1D CNN model training

Training progress

- Learning rate scheduling: The learning rate was dynamically adjusted using the LearningRateScheduler.
  - For the first 5 epochs: Fixed at 0.001 to facilitate rapid initial learning.
  - From epoch 6 onward: Reduced to 0.0001 for fine-tuning and stability.
- Loss metrics: Both loss (training loss) and val loss (validation loss) were monitored at each epoch to assess model performance. These metrics measure:
  - loss: How well the model fits the training data.
  - val loss: How well the model generalises to unseen validation data.
- Model checkpointing: The best-performing model (based on the lowest val loss) was saved periodically. An improvement trend was observed in val loss, from 0.4471 in epoch 1 to 0.42396 in epoch 3. The final and best checkpoint occurred when val loss reached 0.30254.
- Early stopping and completion: Despite the configuration for early stopping, the model completed the full 150 epochs without being interrupted. The minor changes in val loss during later epochs suggested stabilisation.

Validation results

Post-training, the model's performance was evaluated on the validation set using two widely recognised metrics:

- Root mean square error (rmse): 0.5500
  - Indicates the average magnitude of errors between predicted and actual values, expressed in the same units as the target variable.

https://doi.org/10.1017/aer.2025.25 Published online by Cambridge University Press

Table 4. *Summary of training results for 1D CNN model*

Metric	Value	Notes
Initial learning rate	0.001	Applied for the first 5 epochs
Final learning rate	0.0001	Applied from epoch 6 onward
Best val loss	0.30254	Achieved during the later epochs
RMSE (validation)	0.5500	Indicates average prediction error
MSE (validation)	0.3025	Indicates sensitivity to large errors
Epochs completed	150	Early stopping not triggered

- Mean square error (mse): 0.3025
  - Measures the average squared differences between predicted and actual values. While it is sensitive to larger errors, it complements rmse by providing a broader perspective on the model's performance.

Training highlights

- Model performance trends:
  - The training began with relatively high loss values (e.g. loss: 1.7616 in epoch 1).
  - Significant improvement was observed as training progressed, with loss and val loss generally decreasing in tandem, demonstrating effective learning and good generalisation.
- Best model checkpoint:
  - As shown in Table 4, the best-performing model was saved when the validation loss reached 0.30254, which is consistent with the observed RMSE and MSE metrics after training.
- Learning rate adjustment:
  - The reduction in the learning rate after epoch 5 contributed to the model's stabilisation and enhanced performance in later stages of training.

3.8.3 Dropout LSTM model training

Training progress.

- Learning rate scheduling: The learning rate was dynamically adjusted using the LearningRateScheduler callback.
  - For the first 5 epochs: Fixed at 0.001 to promote rapid initial learning.
  - From epoch 6 onward: Reduced to 0.0001 for fine-tuning and stability.
- Loss metrics: Both loss (training loss) and val loss (validation loss) were monitored at each epoch to assess model performance. These metrics measure:
  - loss: How well the model fits the training data.
  - val loss: How well the model generalises to unseen validation data.
- Model checkpointing: The best-performing model (based on the lowest val loss) was saved periodically.
  - The initial improvement was observed with val loss decreasing from 0.4288 in epoch 1 to 0.42755 in epoch 2.
  - As presented in Table 5, the final checkpoint was saved at the lowest validation loss of 0.05936, representing the most optimal version of the model.
- Early stopping and completion: The training was configured to run for 150 epochs, and it completed without triggering early stopping. Minimal changes in val loss during the later epochs indicated stabilisation.

https://doi.org/10.1017/aer.2025.25 Published online by Cambridge University Press

Table 5. *Summary of training results for LSTM dropout model*

Metric	Value	Notes
Initial learning rate	0.001	Applied for the first 5 epochs
Final learning rate	0.0001	Applied from epoch 6 onward
Best val loss	0.05936	Achieved during later epochs
RMSE (validation)	0.2436	Indicates average prediction error
MSE (validation)	0.05936	Emphasises larger errors
Epochs completed	150	Early stopping not triggered

Validation results

Post-training, the model's performance was evaluated on the validation set. Using the saved best model, predictions were generated for the validation data (X val). The following metrics were calculated to assess model performance:

- RMSE: 0.2436
  - Indicates the average magnitude of errors between predicted and actual values, expressed in the same units as the target variable.
- MSE: 0.05936
  - Measures the average squared differences between predicted and actual values. While it is sensitive to larger errors, it complements RMSE by providing a broader perspective on the model's performance.

Training highlights

- Model performance trends:
  - The training began with relatively high loss values (e.g. 0.5205 in epoch 1).
  - Gradual improvements were observed in val loss, with significant progress occurring in the initial epochs. val loss stabilised near 0.05936 during later stages, demonstrating effective learning.
- Best model checkpoint:
  - The best-performing model was saved when val loss reached 0.05936, aligning with the observed improvements in training performance.
- Learning rate adjustment:
  - The reduction in learning rate after epoch 5 contributed to model stabilisation and better performance during later training stages.
- Evaluation metrics post-training:
  - The model achieved a validation RMSE of 0.2436.
  - The validation MSE was calculated as 0.05936.

3.8.4 Multi-layer LSTM model training
Training Progress

- Learning rate scheduling: The learning rate was dynamically adjusted using the LearningRateScheduler callback.
  - For the first 5 epochs: Fixed at 0.001 to encourage rapid initial learning.
  - From epoch 6 onward: Reduced to 0.0001 for fine-tuning and enhanced stability.

https://doi.org/10.1017/aer.2025.25 Published online by Cambridge University Press

- Loss metrics: Both loss (training loss) and val loss (validation loss) were monitored during training to assess model performance. These metrics measure:
  - loss: How effectively the model fits the training data.
  - val loss: How well the model generalises to unseen validation data.
- Model checkpointing: The best-performing model, based on the lowest val loss, was saved during training.
  - Initial improvement in val loss was observed from 0.9320 in epoch 1 to 0.4605 by epoch 4, with further progress in subsequent epochs.
  - As shown in Table 6, the best checkpoint was saved when the validation loss reached 0.08324.
- Early stopping and completion: Training was halted at epoch 85 due to the EarlyStopping callback, which monitored the val loss. This prevented further training as no improvement was observed for 30 consecutive epochs.

Validation results

After training, the model's performance was evaluated on the validation set. Using the saved best model, predictions were generated for the validation data (X val). The following metrics were calculated to assess model performance:

- RMSE: 0.5233
  - Indicates the average magnitude of errors between the predicted and actual values, expressed in the same units as the target variable.
- MSE: 0.2738
  - Measures the average squared differences between predicted and actual values, with an emphasis on larger errors.

Training highlights

- Model performance trends:
  - The training process exhibited consistent improvements in both loss and val loss, particularly during the initial epochs.
  - Validation performance stabilised near 0.08324, with no significant improvement observed in later epochs.
- Best model checkpoint:
  - The best-performing model was saved when val loss reached 0.08324, aligning with the optimal performance of the training process.
- Learning rate adjustment:
  - The learning rate reduction after epoch 5 contributed significantly to the stabilisation and fine-tuning of the model in later training stages.
- Early stopping:
  - Training was halted at epoch 85 due to the lack of improvement in val loss for 30 consecutive epochs, indicating convergence.
- Evaluation metrics post-training:
  - The model achieved a validation RMSE of 0.5233.
  - The validation MSE was calculated as 0.2738.

https://doi.org/10.1017/aer.2025.25 Published online by Cambridge University Press

Table 6. *Summary of training results for multi-layer LSTM model*

Metric	Value	Notes
Initial learning rate	0.001	Applied for the first 5 epochs
Final learning rate	0.0001	Applied from epoch 6 onward
Best val loss	0.08324	Achieved before early stopping
RMSE (validation)	0.5233	Indicates average prediction error
MSE (validation)	0.2738	Emphasises larger errors
Epochs completed	85	Training stopped early

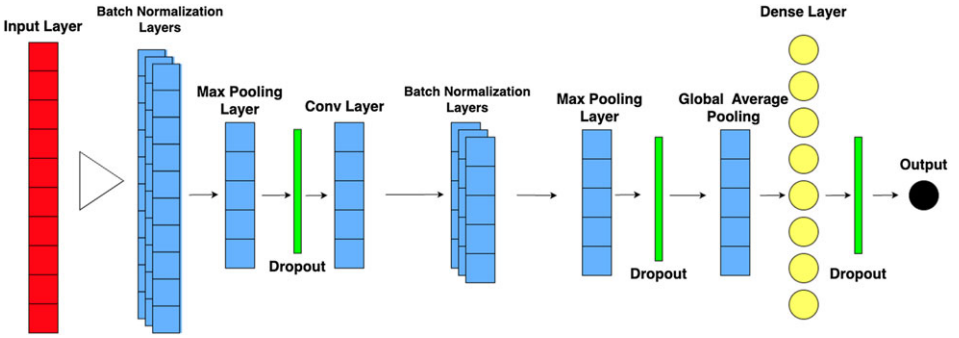


Figure 9. *Schematic of the applied CNN model.*

3.9 Health index estimation with trained models

In this section, both the training and test datasets, containing raw sensor data, were utilised for estimation tasks. The training data underwent a re-estimation process aimed at improving the accuracy of HI calculations. Initially, HI values were manually computed for model training; however, after training, the models were employed to perform automated HI re-estimation. This approach drastically reduced the costs associated with manual HI computation. While we present a smooth curve for clarity, it is crucial to emphasise that such smoothness would not be attainable in practical, real-world scenarios. The noise-free estimations were deliberately designed to minimise potential errors originating from various sources [18].

3.10 Similarity-based approach and RUL prediction

The estimated HI values for both the training and test datasets were organised by engine ID, producing unique trajectories for each engine. Figure 9 depicts the estimated HI values, where the training dataset is represented by a solid black line, and the test dataset is shown as a dashed grey line. To ensure alignment between test and training trajectories, the test engine trajectories were overlaid on the estimated training data. As the test engine trajectories traversed the training data, the Euclidean distances between corresponding points on the test and training trajectories were computed. These distances are visually illustrated in Fig. 9 with red vertical lines connecting points on the test trajectory to the nearest points on the training trajectory.

For each test engine, the five closest distances were identified across all models using this method. In Fig. 9, annotated points such as a and a′, and b and b′, indicate the correspondence between points on the training trajectory (a, b) and their closest points on the test trajectory (a′, b′). Additionally, the maximum trajectory length in the training data was designated as the RUL for each engine. These RUL values served as reference points for estimating the RUL of test engines.

https://doi.org/10.1017/aer.2025.25 Published online by Cambridge University Press

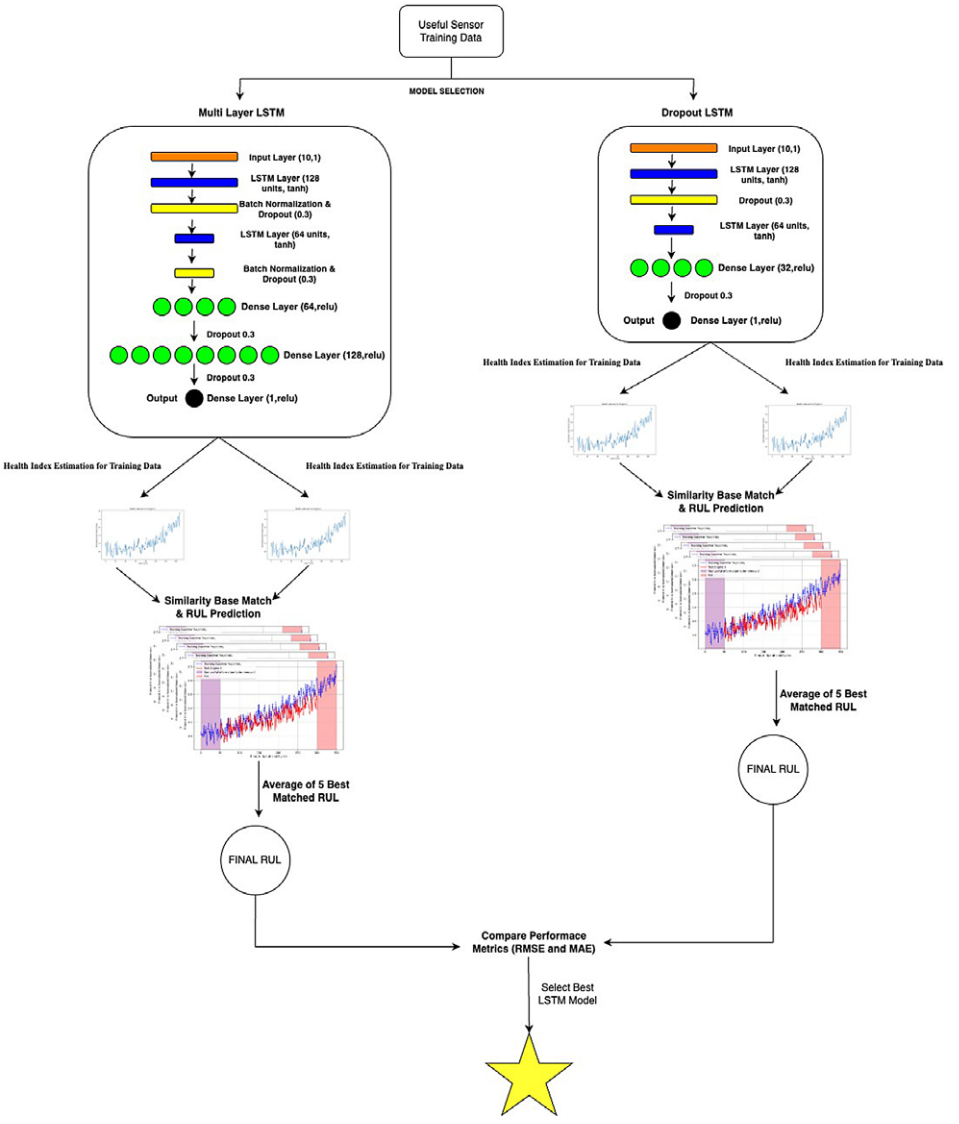


Figure 10. *Flowchart of LSTM model selection.*

To calculate the RUL for a test engine, the best-matched training trajectory was identified, and the RUL was determined using the formula:

$$\text{RUL} = \text{Best Matched Max Length} - (\text{Test Engine Length} + \text{Best Matched Trajectory Value})$$

This calculation was repeated five times for each engine, and the average of the resulting values was taken as the final RUL for the engine. Figure 11 demonstrates the best-matched training trajectory and the RUL estimation for test engine 1. The same methodology was applied to all test engine datasets with each model.

4.0 Results and discussion

The RUL prediction method was tested in this experiment. The RUL of 259 unique test engines was predicted using artificial neural network models and a similarity-based approach. In this experiment, the actual RUL remained unknown, but the measured error score was critical for time series prediction. For

https://doi.org/10.1017/aer.2025.25 Published online by Cambridge University Press

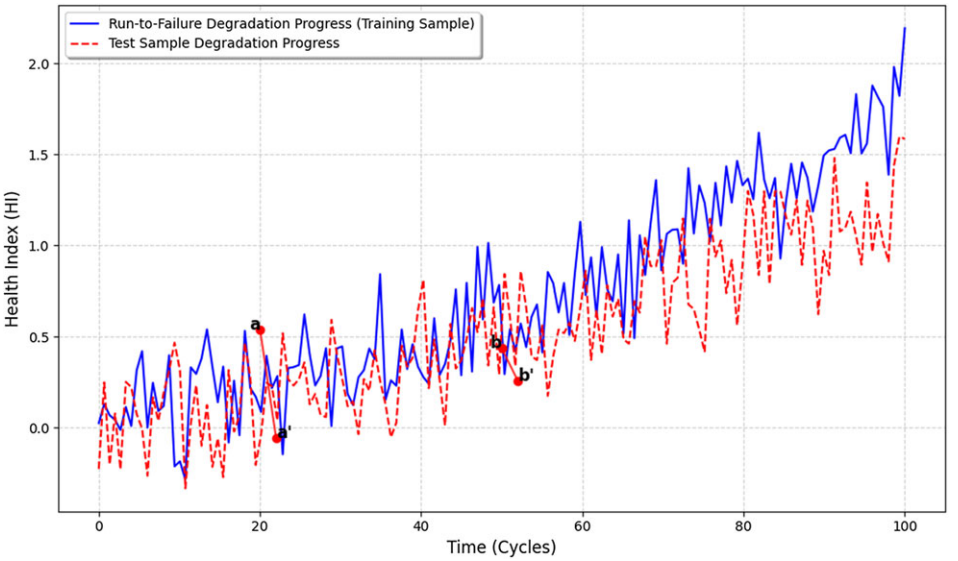


Figure 11. *Similarity methodology.*

this reason, RMSE and MAE were used as performance metrics. While MAE accounts for smaller model errors, RMSE highlights larger prediction errors. Since the test data includes trajectories of varying lengths, RMSE and MAE help identify the weaknesses of the methods.

4.1 Evaluation metrics

There are a lot of evaluation metrics available in the literature. Choosing the effective metrics could be related with sector, sensitivity or tolerance requirements [37]. In this study, we use RMSE and MAE to evaluate models' performances.

Formulas for RMSE and MAE:

- RMSE: RMSE quantifies the average magnitude of prediction errors. It allows us to understand larger errors in model prediction. It is defined as:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_j)^2} \tag{6}$$

where:

– n: Number of predictions,
– $\hat{y}_i$: Predicted value for the i-th observation,
– y_i: Actual value for the i-th observation.

RMSE (Equation (5)) provides insight to identify large prediction errors. The equation is essential metric when outliers could significantly impact engine RUL estimates.

- MAE: MAE represents the average absolute difference between predicted and actual values. It allows us to have insight into the model's prediction accuracy without disproportionately penalising large errors. It is defined as:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|\hat{y}_i - y_i\right| \tag{7}$$

https://doi.org/10.1017/aer.2025.25 Published online by Cambridge University Press

Table 7. *Comparison of model performance with literature*

Model Name	RMSE	MAE
1D CNN	38.806	28.423
Dropout LSTM	26.547	18.749
Multi-layer LSTM	26.707	18.895
Original SBP [48]	25.12	–
LCSS algorithm [49]	20.95	–
Purposed model [50]	32.7	19.5
Second scenario		

where the terms are as defined above. MAE (Equation (6)) provides a straightforward interpretation of the average error magnitude. It allows us to assess the model's overall reliability in engine RUL prediction.

Interpretation of RMSE and MAE for engine RUL prediction:

- RMSE: A lower RMSE, as defined in Equation (5). Equation indicates that the model effectively minimises large errors in predicting RUL. Since engine health estimation can be sensitive to outliers, RMSE helps identify and address significant deviations in predictions. It helps to be ensure the reliability of long-term forecasts.
- MAE: A lower MAE, as described in Equation (6), reflects the model's capability to produce accurate, consistent predictions on average. MAE highlights the typical prediction error magnitude, which is essential for gauging the general robustness of the model's RUL predictions.

4.2 Models performance

In this section, we evaluate the performance of the models applied in this study. Since all models utilise a similarity-based approach for RUL prediction, their individual predictive capabilities are crucial in achieving accurate results.

To provide a broader perspective and address the reviewer's concerns, we have extended our comparison by incorporating relevant studies that employ similarity-based prognostics methods rather than direct RUL prediction. The selected studies offer comparable methodologies and serve as appropriate benchmarks for evaluating model performance.

4.2.1 Comparison with literature

Table 7 presents the performance of our models using RMSE and MAE as evaluation metrics. Our results are compared with recent research on similarity-based RUL prediction from the literature:

- Baharshahi et al. [50] proposed a similarity-based multi-sensor integration approach, achieving an RMSE of 32.7 and MAE of 19.5.
- Xue et al. [48] introduced an improved similarity-based prognostics method with an original SBP model, reporting an RMSE of 25.12.
- Peng et al. [49] utilised an LCSS algorithm for similarity-based spatial-temporal RUL estimation, achieving an RMSE of 20.95.

4.2.2 Model performance evaluation

As shown in Table 8, the dropout LSTM model demonstrated the best performance among the tested architectures, achieving an RMSE of 26.547 and an MAE of 18.749. The multi-layer LSTM model

https://doi.org/10.1017/aer.2025.25 Published online by Cambridge University Press

Table 8. *Comparison of model performance based on RMSE and MAE metrics*

Model name	RMSE	MAE
1D CNN	38.806	28.423
Dropout LSTM	26.547	18.749
Multi-layer LSTM	26.707	18.895

follows closely with an RMSE of 26.707 and an MAE of 18.895. The 1D CNN model exhibited slightly higher error values, with an RMSE of 38.806 and an MAE of 28.423.

4.3 Discussion on predicted RUL

Instead of comparing models with each other, we aim to identify the individual weaknesses of each model in RUL predictions. To pinpoint the weak points in model performance, we analysed engine trajectory lengths where error scores exceeded the individual models' RMSE and MAE. These observations may help improve model performance in the future. The error histograms of the models are provided below.

5.0 Conclusion

In this section, we have analysed the weaknesses of the models, discussed the key findings from the experiments, acknowledged the limitations of this study, and proposed potential methods for improvement to be explored in future work.

5.1 Summary of findings

At the end of this experiment, all models encountered difficulties in accurately predicting the RUL for engines with approximately 50–100 trajectories. The trajectory length appears to be a common weakness for all models, leading to an increase in error scores. These observations are clearly illustrated in Figs 11-18.

The main reason for this issue could be the lack of training data samples with trajectory lengths between 50 and 100. This claim is supported by Fig. 19, which shows that no data samples with 50–100 trajectory lengths are present in the training dataset. Additionally, it can be observed that the models exhibit lower error rates for engines with more trajectory samples in the training data.

However, model performance is not solely determined by metrics; resource consumption is another critical factor during model training. In this experiment, all models were trained under identical conditions, including computational resources, human effort, training time, the number of epochs and learning rate. Under these conditions, the multi-layer LSTM model completed training in the 85th epoch, whereas the dropout LSTM model required 150 epochs. This indicates that the dropout LSTM model consumed more resources compared to the multi-layer LSTM model. Given the negligible performance difference between these two models, the multi-layer LSTM model appears to be the more practical choice in this scenario.

5.2 Limitations

This study provides valuable insights into the application of deep learning and similarity-based methods for predicting the RUL of turbofan engines. However, several limitations should be acknowledged.

First, the analysis is restricted to the CMAPSS dataset's FD002 subset, and the findings may not generalise to other datasets, types of equipment or industries beyond this specific context. Additionally,

https://doi.org/10.1017/aer.2025.25 Published online by Cambridge University Press

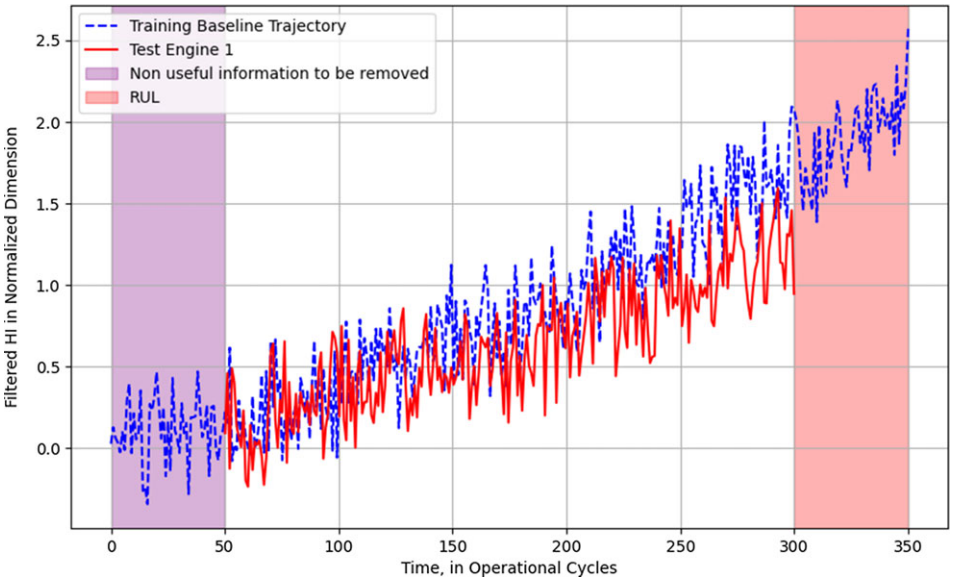


Figure 12. *Similarity-based RUL prediction.*

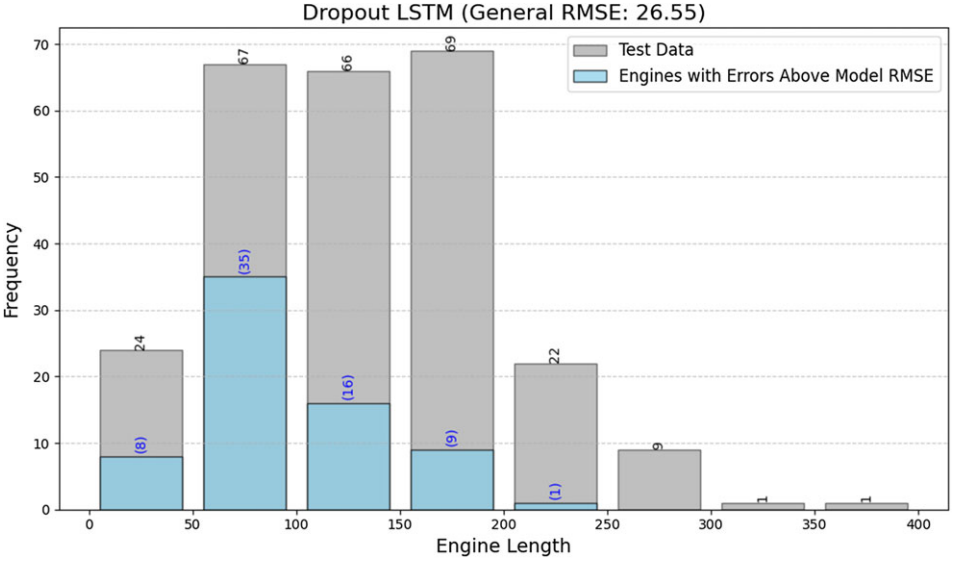


Figure 13. *Dropout LSTM RMSE.*

all neural network models were trained using default parameters to maintain a consistent comparison framework. The lack of hyperparameter tuning likely limited the models' potential performance.

A significant limitation arises from trajectory length. The study revealed that all models struggled with engines having trajectory lengths between 50 and 100. This is attributed to the absence of representative samples within this range in the training data, resulting in increased error scores for these cases, as shown in Fig. 17.

Moreover, the research focused exclusively on deep learning models—LSTM, Bidirectional LSTM and 1D CNN with similarity-based methods for performance comparisons. This study did not compare the performance of these models with results reported in the literature.

https://doi.org/10.1017/aer.2025.25 Published online by Cambridge University Press

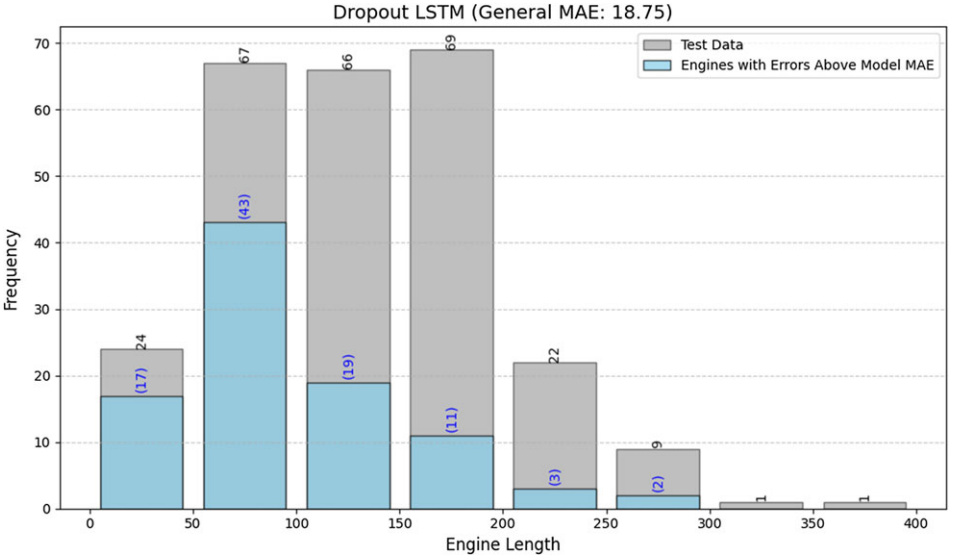


Figure 14. *Dropout LSTM MAE.*

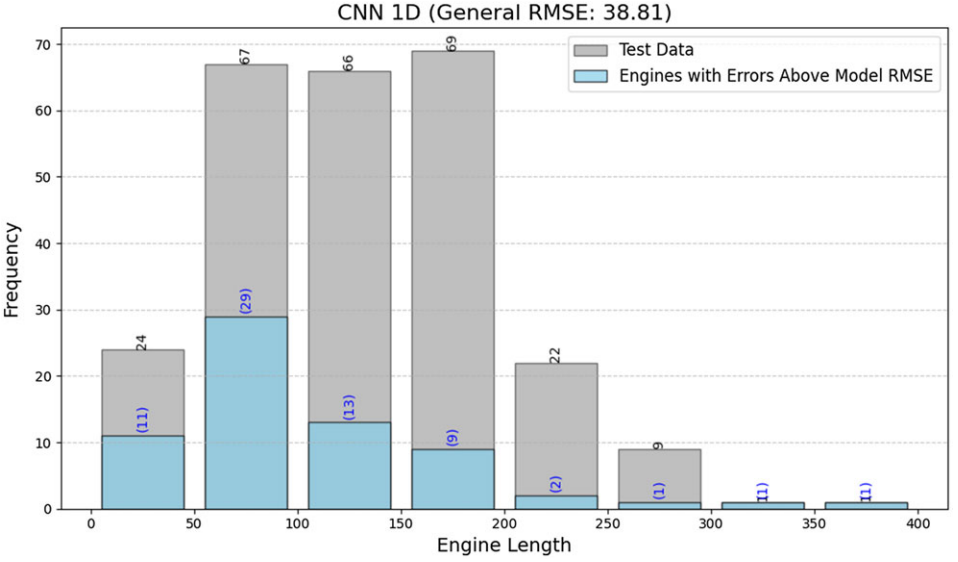


Figure 15. *1D CNN RMSE.*

Finally, the models were trained under identical experimental conditions, including the number of epochs, learning rate and computational resources. While this ensured a fair comparison, the impact of varied training conditions or optimisation techniques was not investigated. Furthermore, the study's findings are limited to the specific use case of turbofan engine RUL prediction.

5.3 Suggestions for future research

Future research can address the limitations outlined in the previous section by implementing several key improvements. Expanding the dataset to include a wider range of engine trajectory lengths, particularly

https://doi.org/10.1017/aer.2025.25 Published online by Cambridge University Press

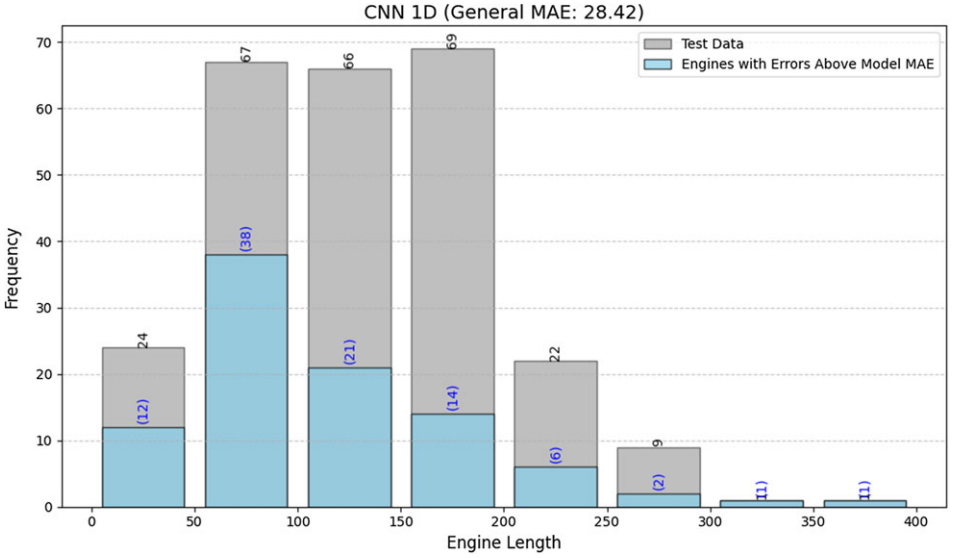


Figure 16. *1D CNN MAE.*

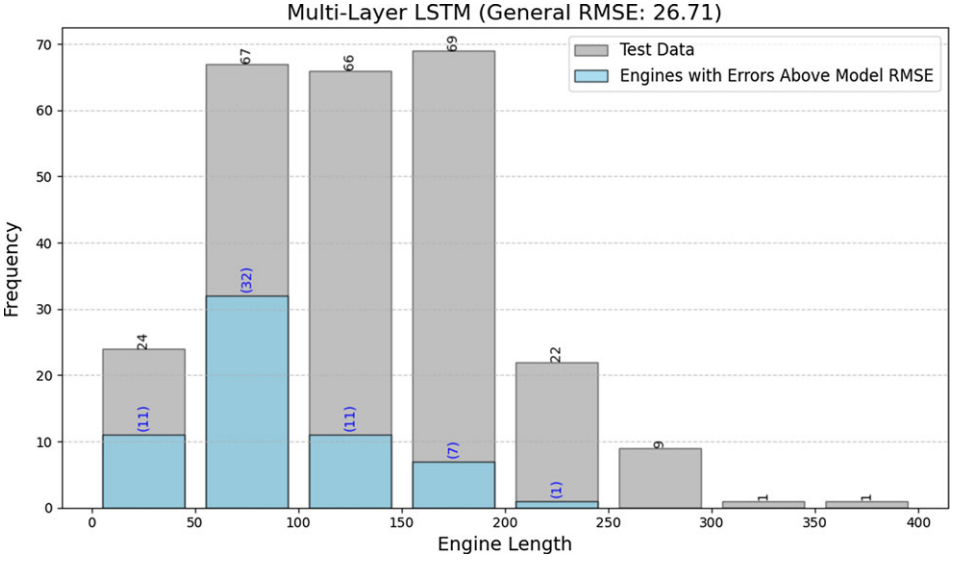


Figure 17. *Multi-layer LSTM RMSE.*

in the 50–100 range, would help mitigate issues caused by data imbalance and improve model accuracy for these cases. Incorporating hyperparameter tuning and advanced optimisation techniques could further enhance model performance by identifying configurations that maximise predictive accuracy.

Finally, we can generate our own dataset to enhance the FD002 subset for the training process. Since the training data provides full trajectory lengths until failure, we could separate long trajectories to generate data samples in the 50–100 range. This method may improve training efficiency and model performance.

https://doi.org/10.1017/aer.2025.25 Published online by Cambridge University Press

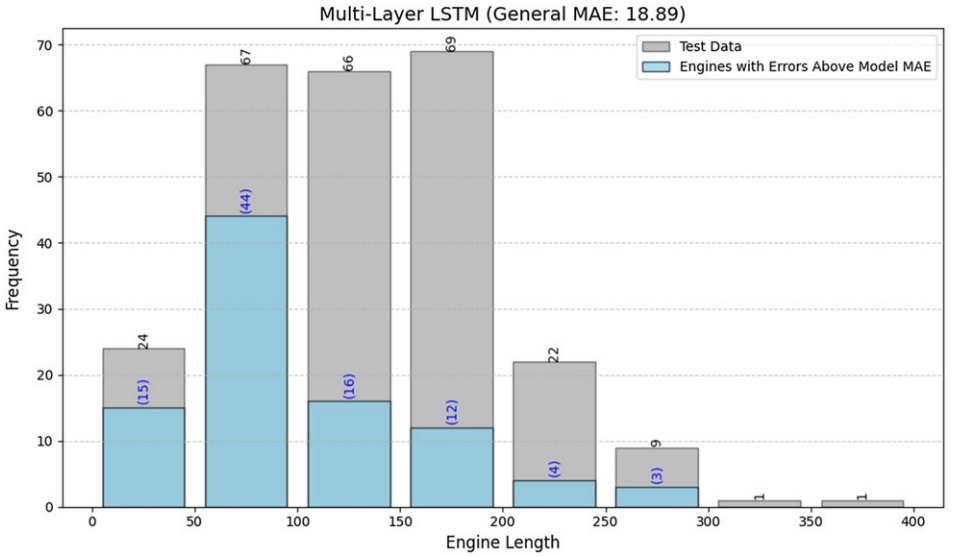


Figure 18. *Multi-layer LSTM MAE.*

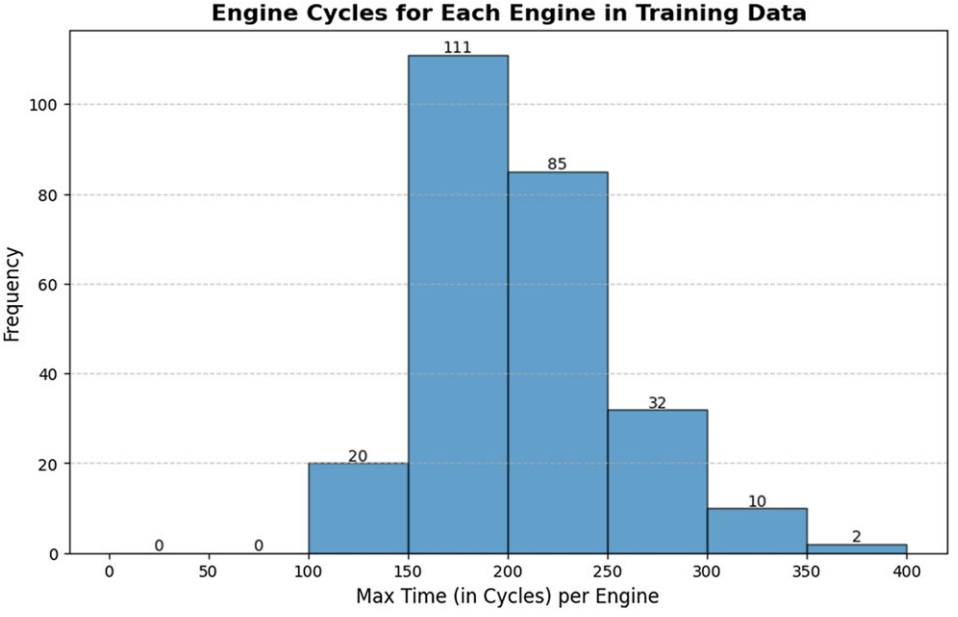


Figure 19. *Train dataset engine trajectory frequency.*

Acknowledgments. The authors gratefully acknowledge the contributions of Oguz Bektas and Mehmet Konar, whose efforts and insights played a significant role in the completion of this research.

References

[1] Baohui, J., Chunhui, X. and Yaohua, L. Study on optimization method of aircraft maintenance plan based on longest path, *J Appl Sci*, 2013, **13**, (16), pp 3354–3357.

https://doi.org/10.1017/aer.2025.25 Published online by Cambridge University Press

[2] Karaburun, N.N., Hatipoğlu, S.A. and Konar, M. SOC estimation of Li-Po battery using machine learning and deep learning methods, *J Aviat*, 2024, **8**, (1), pp 26–31.
[3] Maulana, F., Starr, A. and Ompusunggu, A.P. Explainable data-driven method combined with bayesian filtering for remaining useful lifetime prediction of aircraft engines using NASA CMAPSS datasets. *Machines*, 2023, **11**, (2), 163.
[4] Xiongzi, C., Jinsong, Y., Diyin, T. and Yingxun, W. Remaining useful life prognostic estimation for aircraft subsystems or components: a review, *IEEE 2011 10th International Conference on Electronic Measurement & Instruments*, Vol. 2, pp 94–98. IEEE, 2011.
[5] Adam, S.P., Alexandropoulos, S.A.N., Pardalos, P.M. and Vrahatis, M.N. No free lunch theorem: a review, Approximation and Optimization: Algorithms, Complexity and Applications, pp 57–82, 2019.
[6] Esteghamati, M.Z. and Flint, M.M. Do all roads lead to Rome? A comparison of knowledge-based, data-driven, and physics-based surrogate models for performance-based early design, *Eng Struct*, 2023, **286**, p 116098.
[7] An, D., Kim, N.H. and Choi, J.H. Practical options for selecting data-driven or physics-based prognostics algorithms with reviews, *Reliab Eng Syst Saf*, 2015, **133**, pp 223–236.
[8] Lei, Y., Li, N., Guo, L., Li, N., Yan, T. and Lin, J. Machinery health prognostics: A systematic review from data acquisition to RUL prediction. *Mech Syst Sig Process*, 2018, **104**, pp 799–834.
[9] Byington, C.S., Watson, M. and Edwards, D. Data-driven neural network methodology to remaining life predictions for aircraft actuator components, In *2004 IEEE aerospace conference proceedings (IEEE Cat. No. 04TH8720)* (Vol. 6, pp 3581–3589). IEEE, 2004.
[10] Volponi, A. *Data fusion for enhanced aircraft engine prognostics and health management* (No. E-15412), 2005.
[11] Watson, M., Byington, C., Edwards, D. and Amin, S. Dynamic modeling and wear-based remaining useful life prediction of high power clutch systems, *Tribol Trans*, 2005, **48**, (2), pp 208–217.
[12] Goebel, K., Saha, B., Saxena, A., Mct, N. and Riacs, N. A comparison of three data-driven techniques for prognostics, 62nd Meeting of the Society for Machinery Failure Prevention Technology (MFPT), pp 119–131, 2008.
[13] Bolander, N., Qiu, H., Eklund, N., Hindle, E. and Rosenfeld, T. Physics-based remaining useful life prediction for aircraft engine bearing prognosis, *Annual Conference of the PHM Society*, Vol. 1, No. 1, 2009.
[14] Downey, A., Lui, Y.H., Hu, C., Laflamme, S. and Hu, S. Physics-based prognostics of lithium-ion battery using non-linear least squares with dynamic bounds, *Reliab Eng Syst Saf*, 2019, **182**, pp 1–12.
[15] Garga, A.K., McClintic, K.T., Campbell, R.L., Yang, C.C., Lebold, M.S., Hay, T.A. and Byington, C.S. Hybrid reasoning for prognostic learning in CBM systems, *2001 IEEE Aerospace Conference Proceedings (Cat. No. 01TH8542)* (Vol. 6, pp 2957–2969). IEEE, 2001.
[16] Swanson, D.C. A general prognostic tracking algorithm for predictive maintenance, *2001 IEEE Aerospace Conference Proceedings (Cat. No. 01th8542)*, Vol. 6, pp 2971–2977. IEEE, 2001.
[17] Celaya, J.R., Saxena, A., Saha, S., & Goebel, K.F. Prognostics of power MOSFETs under thermal stress accelerated aging using data-driven and model-based methodologies. In Annual Conference of the PHM Society Vol. 3, No. 1, 2011.
[18] Isbilen, F., Bektas, O., Avsar, R. and Konar, M. Improved machine learning models with a similarity-based approach for remaining useful life prediction, *Aeronaut J*, 2024, **129**, (1332), pp 1–21.
[19] Wang, T., Yu, J., Siegel, D. and Lee, J. A similarity-based prognostics approach for remaining useful life estimation of engineered systems, *2008 International Conference on Prognostics and Health Management*, pp 1–6. IEEE, 2008.
[20] Hassoun, M.H. *Fundamentals of Artificial Neural Networks*, The MIT Press, 1995.
[21] Kurucan, M., Özbaltan, M., Yetgin, Z. and Alkaya, A. Applications of artificial neural network based battery management systems: a literature review. *Renew Sustain Energy Rev*, 2024, **192**, p 114262.
[22] Barzegar Gerdroodbary, M., *et al.* Optimizing aerodynamic stability in compressible flow around a vibrating cylinder with deep reinforcement learning, Phys Fluids, 2024, 36, p 124143.
[23] Bektas, O., Marshall, J. and Jones, J.A. Comparison of computational prognostic methods for complex systems under dynamic regimes: a review of perspectives. *Arch Comput Methods Eng*, 2020, **27**, (4), pp 999–1011.
[24] Bektas, O., Jones, J.A., Sankararaman, S., Roychoudhury, I. and Goebel, K. A neural network filtering approach for similarity-based remaining useful life estimation. *Int J Adv Manuf Technol*, 2019, **101**, pp 87–103.
[25] Arik, S., Turkmen, I. and Oktay, T. Redesign of morphing UAV for simultaneous improvement of directional stability and maximum lift/drag ratio, *Adv Electric Comput Eng*, 2018, **18**, (4), pp 57–63.
[26] Konar M., Arık Hatipoğlu S. and Akpınar M. Improvement of UAV thrust using the BSO algorithm-based ANFIS model. *Aeronaut J*, 2024, **128**, (1328), pp 2364–2373.
[27] Konar M. and Arık Hatipoğlu S. Maximisation the autonomous flight performance of unmanned helicopter using BSO algorithm, *Aeronaut J*, 2024, **128**, (1329), pp 2656–2667.
[28] Karaburun N.N., Arık Hatipoğlu S. and Konar M. Aircraft takeoff speed prediction with machine learning: parameter analysis and model development. Aeronaut J. Published online 2025, pp 1–16. https://doi.org/10.1017/aer.2024.164
[29] Ma, M., Sun, C. and Chen, X. Deep coupling autoencoder for fault diagnosis with multimodal sensory data, *IEEE Trans Ind Inf*, 2018, **14**, (3), pp 1137–1145.
[30] Razavi-Far, R., Farajzadeh-Zanjani, M., Chakrabarti, S. and Saif, M. Data-driven prognostic techniques for estimation of the remaining useful life of lithium-ion batteries, *2016* IEEE *International Conference on Prognostics and Health Management (ICPHM)*, pp 1–8. IEEE, 2016.
[31] Ma, M. and Mao, Z. Deep-convolution-based LSTM network for remaining useful life prediction, *IEEE Trans Ind Inf*, 2020, **17**, (3), pp 1658–1667.
[32] Ozkat, E.C., Bektas, O., Nielsen, M.J. and la Cour-Harbo, A. A data-driven predictive maintenance model to estimate RUL in a multi-rotor UAS, *Int J Micro Air Veh*, 2023, **15**, p 17568293221150171.

https://doi.org/10.1017/aer.2025.25 Published online by Cambridge University Press

[33] Donahue, J., Anne Hendricks, L., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko, K. and Darrell, T. Long-term recurrent convolutional networks for visual recognition and description, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp 2625–2634, 2015.
[34] Abdeljaber, O., Avci, O., Kiranyaz, M.S., Boashash, B., Sodano, H. and Inman, D.J. 1-D CNNS for structural damage detection: Verification on a structural health monitoring benchmark data, *Neurocomputing*, 2018, **275**, pp 1308–1317.
[35] McCulloch, W.S. and Pitts, W. A logical calculus of the ideas immanent in nervous activity, *Bull Math Biophysics* 1943, **5**, pp 115–133.
[36] NASA. Diagnostics & Prognostics, 2023 [Online accessed 29 March 2025]. https://data.nasa.gov/Aerospace/CMAPSS-Jet-Engine-Simulated-Data/ff5v-kuh6/about_data.
[37] Saxena, A., Celaya, J., Balaban, E., Goebel, K., Saha, B., Saha, S. and Schwabacher, M. Metrics for evaluating performance of prognostic techniques, *2008 International Conference on Prognostics and Health Management*, pp 1–17. IEEE, 2008.
[38] Uckun, S., Goebel, K. and Lucas, P.J. Standardizing research methods for prognostics, *2008 International Conference on Prognostics and Health Management*, pp 1–10. IEEE, 2008.
[39] Coble, J.B. Merging data sources to predict remaining useful life–an automated method to identify prognostic parameters, 2010.
[40] Peel, L. Data driven prognostics using a Kalman filter ensemble of neural network models, *2008 International Conference on Prognostics and Health Management*, pp 1–6. IEEE, 2008.
[41] Lei, Y., Li, N., Guo, L., Li, N., Yan T. and Lin J. Machinery health prognostics: A systematic review from data acquisition to rul prediction. *Mech Syst Sig Process*, 2018, **104**, pp 799–834.
[42] Coble, J. and Hines, J.W. Identifying optimal prognostic parameters from data: a genetic algorithms approach, *Annual Conference of the PHM Society*, Vol. 1, No. 1, 2009.
[43] Kiranyaz, S., Ince, T. and Gabbouj, M. Real-time patient-specific ECG classification by 1-D convolutional neural networks, *IEEE Trans Biomed Eng*, 2015, **63**, (3), pp 664–675.
[44] Ince, T., Kiranyaz, S., Eren, L., Askar, M. and Gabbouj, M. Real-time motor fault detection by 1-D convolutional neural networks, *IEEE Trans Ind Electron*, 2016, **63**, (11), pp 7067–7075.
[45] Jiang, H. Discriminative training of HMMs for automatic speech recognition: A survey, *Comput Speech Lang*, 2010, **24**, (4), pp 589–608.
[46] Dyke, S.J., Bernal, D., Beck, J. and Ventura, C. Experimental phase II of the structural health monitoring benchmark problem, *Proceedings of the 16th ASCE Engineering Mechanics Conference,* 2003.
[47] Bektash, O. and la Cour-Harbo, A. First-order layer in artificial pain pathway, *Neural Process Lett*, 2023, **55**, (1), pp 319–343.
[48] Xue, B., *et al.* Improved similarity based prognostics method for turbine engine degradation with degradation consistency test, *Appl Intell*, 2022, **52**, pp 10181–10201.
[49] Peng, C., Hu, X. and Tang, Z. Turbofan engine health assessment based on spatial–temporal similarity calculation, *Sensors* 2023, **23**, **24**, p 9748.
[50] Baharshahi, M., Seyedhosseini, S.M. and Limon, S.M. Remaining useful life prediction using the similarity-based integrations of multi-sensors data, *Qual Eng* 2024, **36**, (1), pp 36–53.

Cite this article: Isbilen F., Bektas O. and Konar M. (2025). Deep learning and similarity-based models for predicting turbofan engine remaining useful life: insights from the CMAPSS dataset. *The Aeronautical Journal*, **129**, 2004–2035. https://doi.org/10.1017/aer.2025.25

https://doi.org/10.1017/aer.2025.25 Published online by Cambridge University Press