Received: 14 June 2022 | Revised: 24 November 2022 | Accepted: 1 February 2023

DOI: 10.1002/qre.3288

RESEARCH ARTICLE

WILEY

Three-stage feature selection approach for deep learning-based RUL prediction methods

Youdao Wang | **Yifan Zhao**

School of Aerospace, Transport and Manufacturing, Cranfield University, Cranfield, Bedford, UK

Correspondence
Yifan Zhao, School of Aerospace, Transport and Manufacturing, Cranfield University, Cranfield, Bedford MK43 0AL, UK.
Email: yifan.zhao@cranfield.ac.uk

Abstract

The remaining useful life (RUL) prediction plays an increasingly important role in predictive maintenance. With the development of big data and the Internet-of-Things (IoT), deep learning (DL) techniques have been widely adopted for RUL prediction. Addressing the limitation of the current methods for data under multiple operating conditions, this paper proposes a three-stage feature selection approach for DL-based RUL prediction models. The k-medoids cluster is initially used to sort raw data based on different operating conditions. In the first stage of feature selection, an operational-based normalisation approach is applied to reconstruct the data. Afterwards, Spearman's rank and pair-wise Pearson correlation coefficients are used to eliminate irrelevant and redundant features in the second and third stages, respectively. A case study using NASA's Commercial Modular Aero-Propulsion System Simulation (C-MAPSS) dataset is presented to quantitatively evaluate the influence of the proposed feature selection method using the Recurrent Neural Network (RNN) and its' variants, enhanced by an optimised activation function and optimiser. The results confirm that the proposed method can improve the stability of DL models and achieve about a 7.3% average improvement in the RUL prediction for popular and state-of-the-art DL models.

KEYWORDS

deep learning, feature engineering, long short-term memory, remaining useful life prediction

1 | INTRODUCTION

The economic cost caused by the failure of highly complex and automated machines has become increasingly expensive. Thus, a more effective maintenance strategy is crucial to improve machines' overall reliability and productivity. With the rapid development of Internet-of-Things (IoT) and cyber manufacturing techniques in recent years, the manufacturing industry has adopted condition-based maintenance (CBM). The rapid growth of modern sensor technology and condition monitoring systems enhances the possibilities of utilising real-time and historical data to comprehensively manage maintenance.[1] The remaining useful life (RUL) prediction of components or systems is the core and major challenging role in the CBM of machines.[2]

RUL prediction generally refers to the study of predicting the specific time length from the current time to the end of the useful life of an asset or system.[3] Approaches of RUL prediction can be catalogued into model-based, data-driven, and hybrid methods. Model-based approaches, also called physics-based approaches, evaluate the health condition of a system by building mathematical models based on the failure mechanisms or the first principle of damage.[4] In data-driven approaches, RUL is computed through statistical and probabilistic methods by utilising historical information and routinely monitored data of the system.[5] Complex and noisy working conditions often impede the construction of a physical model. Meanwhile, it is often difficult to determine the parameters of the physical model. In contrast, the requirements for data-driven methods to model the degradation and predict RUL are much easier to be satisfied, especially with the fast development of big data and IoT. Therefore, data-driven approaches are more widely used in RUL prediction at present. Hybrid approaches, combining model-based approaches and data-driven approaches, aim to leverage the advantages of both categories. An *et al.*[6] further classified the data-driven approaches into artificial intelligence (AI) approaches and statistical approaches. AI approaches intend to learn the degradation patterns from available observations directly, which can deal with prognostic issues of complex mechanical systems.[7] Statistical approaches predict the RUL by fitting the empirical model close to the collected data and extrapolating the fitted curve to failure criteria.

Deep learning (DL), a subset of AI approaches attracting significant investigations in the last few years in RUL prediction, can be used to extract multilevel features from data.[8] As an end-to-end machine learning method, it can automatically process original signals, identify discriminative and trend features in the input data layer by layer, and then directly output classification or regression results. Because of its strength in self-learning features, DL has already achieved great success in applications in the manufacturing industry.[9] There are mainly four representatives of DL-based architectures for RUL prediction, including Auto-encoder (AE), Deep Belief Network (DBN), Convolutional Neural Network (CNN), and Recurrent Neural Network (RNN).[10] AE and DBN are often used for pre-training networks, while CNN and RNN are generally used as predictive models. For example, Lu *et al.*[11] proposed a stacked denoising autoencoder (SDA) fed with signals containing ambient noise and working condition fluctuations to conduct the fault diagnosis of rotary machinery components. Gan *et al.*[12] developed a hierarchical diagnosis network to identify the bearing fault location and severity by combining a wavelet packet transform and DBN. It has been proven that both CNN and RNN outperform the traditional prognosis algorithms in RUL prediction, while CNN-based approaches are used more in fault diagnosis and surface integration inspection.[13] RNN, on the other hand, gained much more attention and achievements in the field of RUL prediction because it is good at dealing with time sequence data.[14] RNNs can be trained by backpropagation through time for supervised tasks with sequential input data and target outputs. The limitation of a vanilla RNN is that it may not capture long-term dependencies during the backpropagation for model training, resulting from the vanishing gradient or exploding gradient problem. Therefore, long-short term memory (LSTM), a modified structure of the recurrent cell that incorporates the standard recurrent layer along with additional ‘memory’ control gates, was developed. Besides, gated recurrent unit (GRU), bi-directional long-short term memory (Bi-LSTM), and many other variants of RNN are also designed and adopted in the manufacturing field in the last few years. Zheng *et al.*[14] presented a multi-layer LSTM approach to investigate the hidden patterns from sensors and operational data with multiple operating conditions, fault and degradation models. Chen *et al.*[15] utilised a GRU network to establish the nonlinear deterioration model for RUL prediction. Elsheikh *et al.*[16] built a bidirectional handshaking LSTM (BHLSTM) network for RUL prediction, where short sequences of monitored signals were provided with random initial wear.

The main challenge of degradation monitoring is to extract representative features from the collected raw data.[17] Time-domain and frequency-domain-based features are commonly adopted in signal processing, and these features usually consider the degradation model stationary until the occurrence of a fault. The limitation is that some of these features may be only effective for a specific operational stage.[18] Moreover, in modern manufacturing, machines often work under complex environments, such as multiple operating conditions, making it even harder to extract representative features. Different operating conditions may lead to different degradation models for the same machine, often reflected by the dynamic of the data collected from the sensors, which challenges the applicability of the existing stationary features. The most popular solution to address this challenge nowadays is adopting DL-based methods to extract features from data automatically. Although DL methods are claimed to be able to find out the hidden patterns of the dynamic data, it remains a black box and lacks the transparency required for further system improvement. Zhu *et al.*[19] developed a novel transfer learning method based on multiple layer perceptron (MLP) to solve the distribution discrepancy problem caused by multiple working conditions. The domain adaptation modules are used to learn the time-invariant features, which have been proven effective and advantageous in RUL prediction. Another solution is to utilise feature engineering methods to address data degradation patterns. The predictive model can extract more representative features to accommodate different operating conditions with the degradation patterns addressed. Kundu *et al.*[20] propose a Weibull accelerated failure time regression (WAFTR) model to extract representative frequency domain features of the rolling element bearings that operate at different operating conditions. When the operating condition data is not considered in the RUL

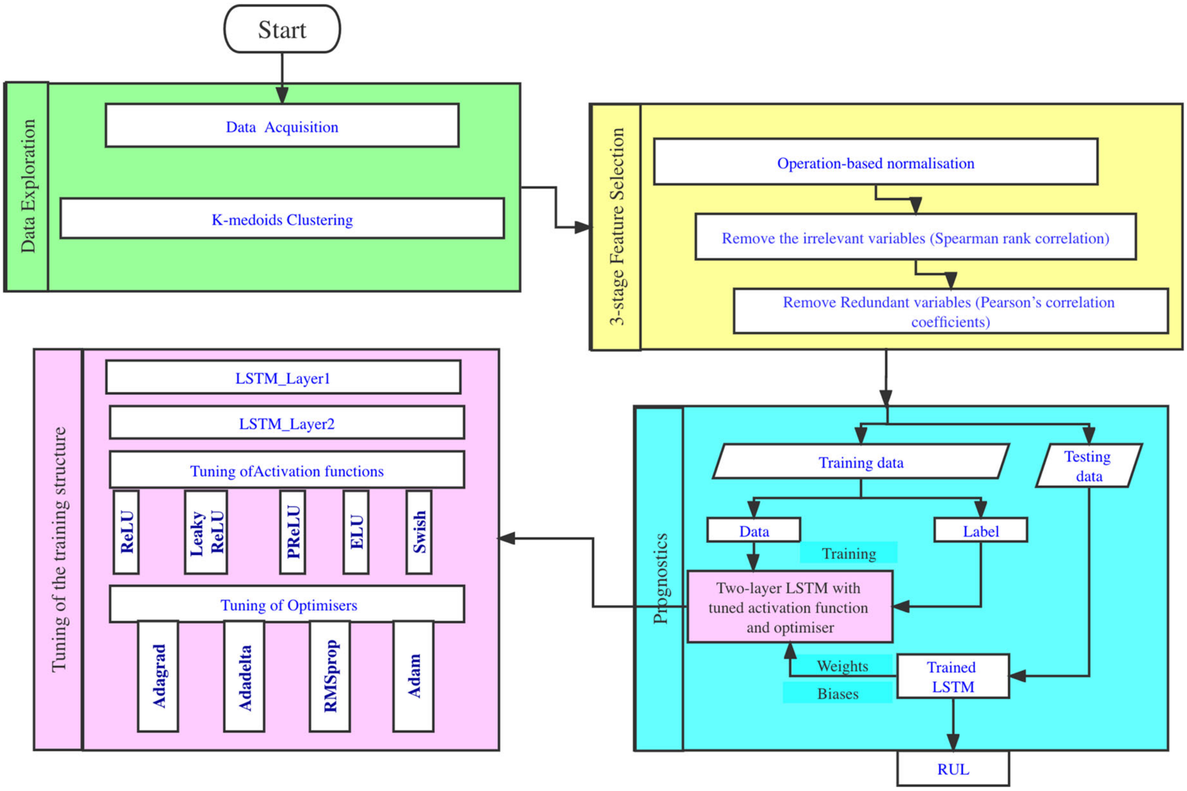

Start
Data Exploration
Data Acquisition
K-medoids Clustering
3-stage Feature Selection
Operation-based normalisation
Remove the irrelevant variables (Spearman rank correlation)
Remove Redundant variables (Pearson's correlation coefficients)
Tuning of the training structure
LSTM_Layer1
LSTM_Layer2
Tuning ofActivation functions
ReLU
Leaky ReLU
PReLU
ELU
Swish
Tuning of Optimisers
Adagrad
Adadelta
RMSprop
Adam
Prognostics
Training data
Testing data
Data
Label
Training
Two-layer LSTM with tuned activation function and optimiser
Weights
Biases
Trained LSTM
RUL

FIGURE 1 Flow chart of the proposed remaining useful life (RUL) prediction framework.

prediction model, a high underestimation and overestimation of RUL is observed. In contrast, the prediction accuracy improves significantly when the operating condition data is involved in the model.

In this paper, a three-stage feature selection approach, enhanced by an optimised DL-based model, is presented to better predict the RUL of a system under multiple operating conditions. An operation-based normalisation method is proposed to better expose the hidden degradation patterns from the data and enhance the RUL prediction performance. A public dataset is then employed to demonstrate the superior performance of the introduced feature engineering methods and DL optimisation.

2 | METHODOLOGY

As shown in Figure 1, the proposed RUL prediction framework consists of three phases: data exploration, feature selection, and RUL prognostics. In the first phase (the green block), a k-medoids clustering method is applied to address the multiple operating conditions. An operation-based normalisation method is used to reconstruct the raw dataset in the second phase (the yellow block). Then significant and lean sensors are selected using two correlation analysis methods for preparing training and testing datasets. In the third phase (the blue–green block), the transformed training data are then used to train a two-layer LSTM predictive model, where different activation functions and optimisers are optimised to form the best-performed model (the purple block). After that, the transformed testing data are fed into the trained predictive model to produce the RUL estimation for the model deployment. Details of each phase are presented below.

2.1 | Data exploration

The data collected for the RUL prediction of a system is generally time

series data. In many cases, data sets are simply made up of columns of numbers, sometimes even with recording mistakes. An exploration and pre-processing of the data sets are often necessary to ensure reliability before directly feeding the data into the model.

When the system works under multiple operating conditions, the degradation mechanism is often different for different operating conditions. While in many cases, the information on the operation conditions is not recorded in the dataset directly, such as the Commercial Modular Aero-Propulsion System Simulation (C-MAPSS) dataset used in this paper. A classical partitional clustering algorithm is adopted to classify the operational condition (*OC*) from the original datasets, naming k-medoids., which divides the dataset of n objects into k clusters. In the k-medoids algorithm, the most representative objects in the clusters are picked as medoids to represent the clusters. A medoid is defined as the object of a cluster whose average dissimilarity to all the objects in the cluster is minimal.[21] Each remaining object is clustered with the medoid to which it is the most similar. The following equation calculates the total absolute error for the clustering configuration:

$$E\left(M,\ X\right) = \sum_{i=1}^{n} \min_{j=1}^{k} |M_j - X_i| \tag{1}$$

where M is the set of medoids, X the dataset, n the number of objects, k the number of clusters, M_j the jth medoid and X_i is the ith object in the dataset. The function min is used to find the medoid closest to the given object.[22]

2.2 | Three-stage feature selection method

Feature selection or feature reduction is the process of reducing the number of input variables to reduce the computational cost of modelling and improve the model's prediction performance. It primarily focuses on removing non-informative or redundant variables from the data that do not contribute to the predictive model's performance or cause overfitting problems. In this paper, we present a three-stage feature selection method.

In stage 1, we propose an operation-based normalisation approach. This method is used to cope with datasets with multiple operating conditions. By utilising the unsupervised k-medoids clustering based on the selected input variables containing the operating condition information, all raw input variables X_i ($X_1, X_2, \ldots, X_n$) in terms of different *OC*s (OC_1, $OC_2, \ldots, OC_N$) can be written as

$$X_i = \cup_{j=1}^{N} OC_j\left(X_k\right) \tag{2}$$

where n is the number of input variables, N the number of *OC*s, i the index of the raw input and k is the index of the input in the new series OC_j. Then the commonly used normalisation method is applied to each *OC* of each input variable. And then, each input variable $\widehat{X}_i$ is reconstructed by connecting with all normalised segments with the same time index shown in Equation (2), written as

$$\widehat{X}_i = \cup_{j=1}^{N} \mathrm{Norm}|OC_j(X_k)| \tag{3}$$

where Norm$|\cdot|$ denotes the normalisation operation, which can be any of the basic normalisation methods normally used in Machine learning (see Appendix A). Based on the data exploration of the input variable $\widehat{X}_i$ including data description and visualisation, a preliminary decision on feature selection can be made.

In stage 2, we propose to use pair-wise Spearman's rank coefficient between each input and each output to remove the irrelevant input variables. The limitation of Pearson correlation is that it can not be adopted to tell the nonlinear relationship between a dependent variable and an independent variable. Spearman rank correlation does not have any assumptions about the data distribution. Therefore it is the appropriate correlation analysis method when the variables are measured on a scale that is at least ordinal. The formula used to calculate the Spearman rank correlation is written as

$$p = 1 - \frac{6\sum d_i^2}{n\left(n^2 - 1\right)} \tag{4}$$

where p is the Spearman rank correlation, d_i is the difference between the ranks of corresponding variables (x and y), and n is the number of observations. The rank is calculated as the average value when there are ties in input or output. The scores range between −1 and 1, where −1 refers to perfectly negatively correlated, and 1 refers to perfectly positively correlated.

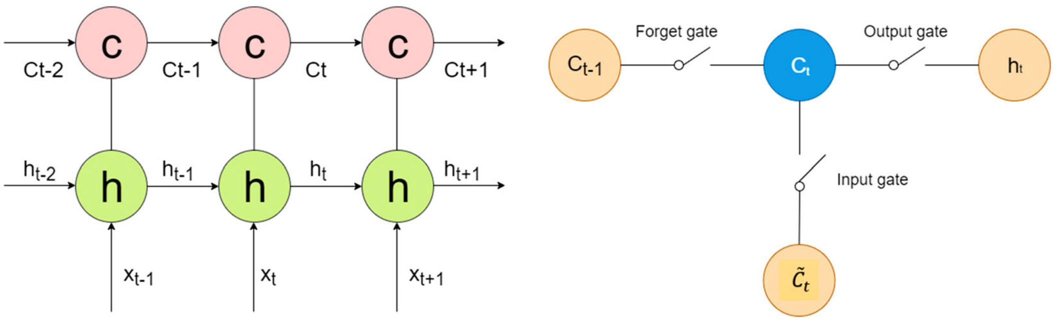


FIGURE 2 Cell state (left) and three gates (right) in long-short term memory (LSTM).

In stage 3, since the RUL prediction of a system is a regression predictive modelling problem with numerical input variables and output variables, we propose to inspect the pair-wise Pearson's correlation coefficients among all input variables to remove the redundant input variables. Pearson's correlation coefficient measures the strength of the linear relationship between two data samples, either time-series data or non-time-series data. It is calculated as the covariance of two variables (x and y) divided by the product of the standard deviation of each variable, written as:

$$r_{xy} = \frac{S_{xy}}{s_x s_y} \tag{5}$$

where r_{xy} is the correlation coefficient, S_{xy} is the covariance, and s_x, s_y are the standard deviations. r_{xy} ranges from -1 to +1. A correlation coefficient of 1 indicates that for every positive increase in one variable, there is a positive increase of fixed proportion in another variable. A correlation coefficient of -1 means that for every positive increase in one variable, there is a negative decrease of fixed proportion in another variable. A correlation coefficient of 0 means that these two variables are unrelated. The absolute value of the correlation coefficient refers to the relationship strength between the two variables.

2.3 | Construction of the predictive model

After the three-stage feature selection, the run-to-failure training data is used to train the DL-based network. Then, the prediction of the RUL of the test engines is achieved by feeding the pre-processed testing data into the trained predictive model. Since RNN is good at dealing with time series data, various RNN-type network structures are tested to validate the practical value of the proposed feature section approach.

2.3.1 | Long short-term memory

LSTM network is a modified structure of the recurrent cell that incorporates the standard recurrent layer along with additional 'memory' control gates. The original LSTM was developed by Hochreiter and Schmidhuber[23] when researchers discovered a vanishing and exploding gradient issue in traditional RNNs. LSTM uses storage elements to transfer information from the past output instead of having the output of the RNN cell be a nonlinear function of the weighted sum of the current input and previous output. In other words, instead of using a hidden state h only, LSTM adopts a cell state C to keep the long-term information, as shown in Figure 2. The central concept of LSTM utilises three gates to control the cell state C (forget gate, input gate and output gate). The forget gate is used to manage the information from the previous cell state C_{t-1} to the current cell state C_t; the input gate decides how many inputs should be kept in the current cell state C_t; and the output gate determines the output h_t from the current cell state C_t.

The output of LSTM at step t is calculated using the following equations:

$$i_t = \sigma\left(U^i x_t + W^i h_{t-1} + b_i\right) \tag{6}$$

$$f_t = \sigma\left(U^f x_t + W^f h_{t-1} + b_f\right) \tag{7}$$

$$o_t = \sigma\left(U^o x_t + W^o h_{t-1} + b_o\right) \tag{8}$$

$$\tilde{c}_t = \tanh\left(U^{\tilde{c}} x_t + W^{\tilde{c}} h_{t-1} + b_{\tilde{c}}\right) \tag{9}$$

$$C_t = C_{t-1} \cdot f_t + \tilde{c}_t \cdot i_t \tag{10}$$

$$h_t = \tanh\left(c_t\right) \cdot o_t \tag{11}$$

where U, W and b are the trainable weights and biases, respectively, and i, f and o represent the input gate, forget gate and output gate, respectively. These three gates have the same shape with different parameters U and W, which need to be learned from the training process. The candidate state $\tilde{c}_t$ cannot be used directly. It must pass through the input gate and then be used to calculate the internal storage C_t. While C_t is not only affected by the hidden state but also by C_{t-1} that is controlled by the forget gate. Based on C_t, a layer of $\tanh$ function is applied to the output information h_t, which is constrained by the output gate. The gates enable LSTM to fulfil the long-term dependencies in the sequence, and by learning the gate parameters, the network can find the appropriate internal storage. Therefore, LSTMs are naturally suited for the RUL prediction task using sensor data with the inherent sequential nature due to their capability of remembering information over long periods.

The introduction of the RNN and its' other popular variants are in Appendix B.

2.3.2 | Parameter selection

The optimiser and activation functions are two integral parts of constructing the predictive model. The choice of the optimiser and the activation function will largely affect the model's prediction performance. A preceding work introduced some popular optimisers (RMSprop, Adam, AdaGrad and AdaDelta) and activation functions (Rectification of Linear Unit [ReLU], Exponential Linear Unit [ELU], Leaky_ReLU, Parametric Rectified Linear Unit [PReLU] and Swish) and systematically analysed their performance in RUL prediction.[24] These variants have been tested in this study to achieve the best performance. A brief introduction to these activation functions and optimisers can be seen in Appendix C.

2.4 | Benchmark dataset

NASA's C-MAPSS dataset aimed at modelling the damage propagation of aircraft gas turbine engines[26] was used in the case study. This dataset consists of four subsets, and each subset has different numbers of engines with varied operational cycles.

In the dataset, engine profiles were simulated with different initial degradation conditions. The maintenance was not considered during the simulation. The dataset includes one training set and one testing set for each engine, and the training set consists of the historical run-to-failure measurement records of the engines from 21 onboard sensors. The objective is to predict the RUL of each engine based on the given sensor measurements. Table 1 lists the details of these

TABLE 1 Commercial Modular Aero-Propulsion System Simulation (C-MAPSS) dataset.

	Dataset			
Parameters	**FD001**	**FD002**	**FD003**	**FD004**
Data for training	100	260	100	249
Data for test	100	259	100	248
Operating conditions	Single	Multiple	Single	Multiple
Fault conditions	Compressor	Compressor	Compressor and fan	Compressor and fan

four datasets. Specifically, FD001 refers to the engine failure arising from the high-pressure compressor under a single operating condition. FD002 refers to the engine failure from the high-pressure compressor under six operation conditions. FD003 refers to the engine failure from a high-pressure compressor and fan under a single operating condition. FD004 refers to the engine failure from both the high-pressure compressor and fan under six operation conditions. FD002 was primarily used in this case study to validate the proposed framework.

2.5 | Performance evaluation

In this case study, the root mean square error (RMSE) was used to evaluate the performance of the trained neural networks. The mathematical expression is:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} e_i^2} \tag{12}$$

where n is the total number of actual RUL targets in the related test set and e_i refers to the difference between the true RUL and the predicted RUL.

3 | RESULTS

3.1 | Data exploration

The employed data set FD002 (training set) is composed of information relating to a total of 260 different turbines with a total number of observations that varies from 128 to 378. The mean and standard deviation of the response and the predictor variables are summarised in Table 2. Since this dataset has six operating conditions ($k = 6$), the three columns of setting data are fed to a k-medoids clustering model with six clusters. To make sure the labels of the training set and testing set are the same for the same operation condition, the setting data in both the training dataset and the test dataset is used. The distribution of the six operating conditions in FD002 is demonstrated in Figure 3A, with the setting values shown in Figure 3B.

3.2 | Parameter optimisation

This section reports the results of the RUL prediction, which inform the selection of the normalisation method, RNN models and model parameters, where the operation-based data scaling method and feature selection are not considered.

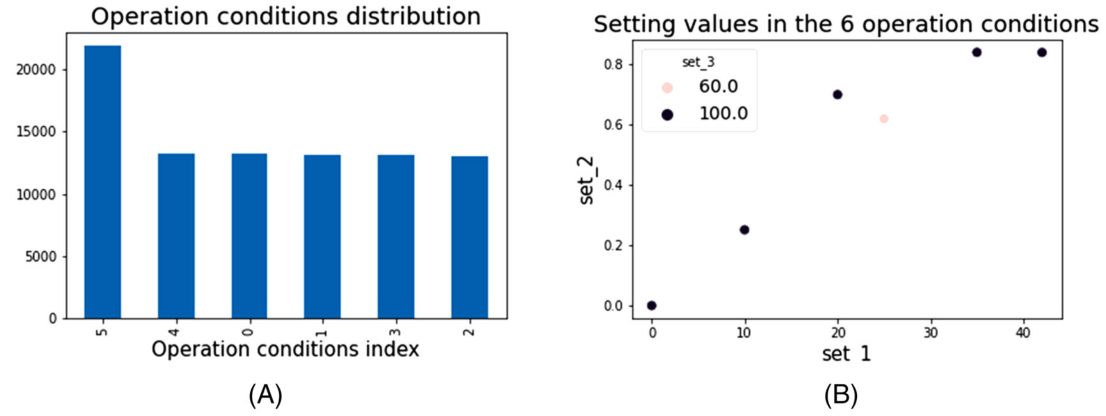


FIGURE 3 Result of k-medoids clustering model on FD002; (A) operation condition distribution; (B) setting values in the six operation conditions.

TABLE 2 Descriptive statistics of all the variables of dataset FD002 (training set).

Index	Symbol	Input variables	Mean	Standard deviation
1	ID	Engine ID		
2	T	Time cycle	109.15	69.18
3	set_1	Operation condition 1	24	14.74
4	set_2	Operation condition 2	0.57	0.31
5	set_3	Operation condition 3	94.05	14.24
6	sensor_1	Total temperature at fan inlet (°R)	472.91	26.39
7	sensor_2	Total temperature at LPC outlet (°R)	579.67	37.29
8	sensor_3	Total temperature at HPC outlet (°R)	1419.97	105.94
9	sensor_4	Total temperature at LPT outlet (°R)	1205.44	119.12
10	sensor_5	Pressure at fan inlet (psia)	8.03	3.61
11	sensor_6	Total pressure in bypass-duct (psia)	11.6	5.43
12	sensor_7	Total pressure at HPC outlet (psia)	282.61	146
13	sensor_8	Physical fan speed (rpm)	2228.88	145.21
14	sensor_9	Physical core speed (rpm)	8525.2	335.81
15	sensor_10	Engine pressure ratio (P50/P2)	1.09	0.13
16	sensor_11	Static pressure at HPC outlet (psia)	42.99	3.23
17	sensor_12	Ratio of fuel flow to Ps30 (pps/psi)	266.07	137.66
18	sensor_13	Corrected fan speed (rpm)	2334.56	128.07
19	sensor_14	Corrected core speed (rpm)	8066.6	84.84
20	sensor_15	Bypass ratio	9.33	0.75
21	sensor_16	Burner fuel–air ratio	0.02	0.005
22	sensor_17	Bleed enthalpy	348.31	27.75
23	sensor_18	Demanded fan speed (rpm)	2228.8	145.33
24	sensor_19	Demanded corrected fan speed (rpm)	97.76	5.36
25	sensor_20	HPT coolant bleed (lbm/s)	20.79	9.87
26	sensor_21	LPT coolant bleed (lbm/s)	12.47	5.92

TABLE 3 Remaining useful life (RUL) prediction performance (root mean square error [RMSE]) without and with normalisation methods.

	Normalisation methods			
Dataset	None	Min–Max normalisation	Robust standardisation	Standardisation
FD002	Fail	37.92 ± 12.61	32.17 ± 10.51	37.06 ± 11.67

It should be noted to evaluate the uncertainty of prediction better, each model was repeated 20 times, and the statistical results are presented below.

Normalisation methods: Before feeding the raw data into the RNN-based model for training or testing, the data need to be normalised. If not, gradient explosion or gradient vanishing problems will occur, which leads to invalid results (Fail). We applied three classic normalisation methods for FD0002, including Min–Max normalisation, Robust standardisation and standardisation. For the RNN model, we used a two-layer LSTM with the optimiser of *Adam*, the activation function of *ReLU*, the sequence length of 50 and the neuron number of 128. The results can be found in Table 3, which suggests the stability of the model prediction is largely improved after the normalisation. Generally, Robust standardisation scaling outperforms both Min–Max normalisation scaling and standardisation scaling. Therefore, Robust standardisation is selected for further analysis.

RNN model selection: Figure 4 displays the RUL prediction performance for FD002 using 12 algorithms constructed by four different model types and three layers. The optimiser of *Adam*, the activation function of *ReLU*, the sequence length of 50 and the neuron number of 128 were used. The data was transformed using Robust standardisation. Inspection of Figure 3 indicates that, for this dataset, LSTM, Bi_LSTM and GRU outperformed the original RNN models in terms of

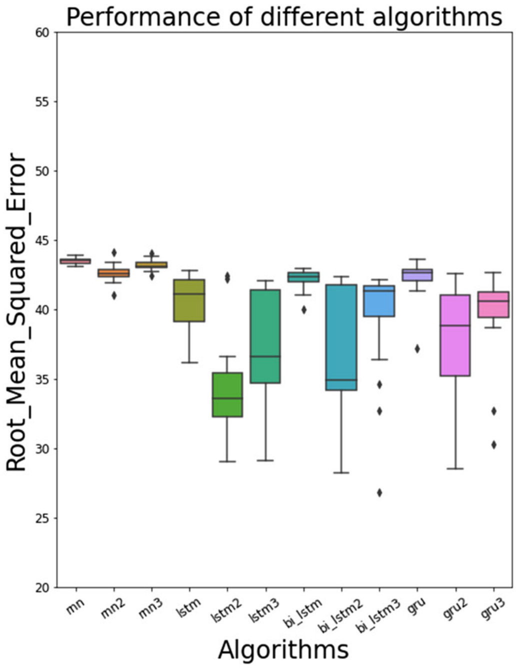


FIGURE 4 RUL prediction performance of FD002 using different RNN-based algorithms. RNN, Recurrent Neural Network; RUL, remaining useful life.

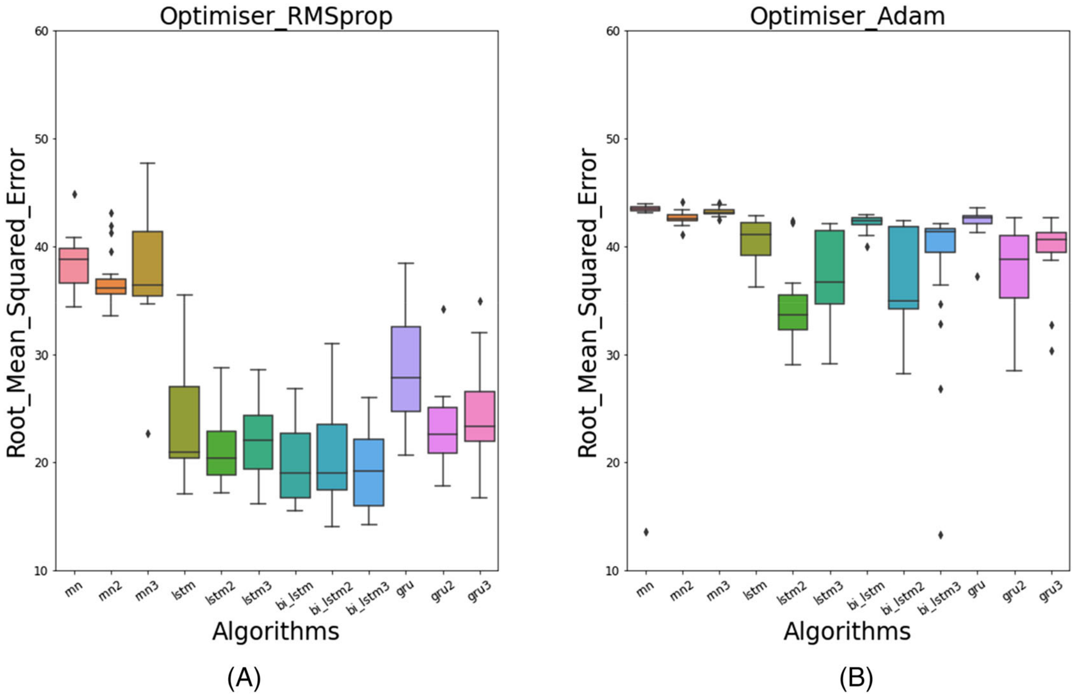


FIGURE 5 RUL prediction performance using different optimisers for FD002; (A) RMSprop; (B) Adam.

precision. Considering the balance of accuracy, precision and model complexity, a two-layer LSTM is selected for further optimisation using feature engineering and operation-based data scaling.

RNN model parameters selection: To select the optimiser, the activation function of *ReLU*, the neuron number of 128 and the sequence length of 50 were used to test four different optimisers, *RMSprop*, *Adam*, *AdaGrad* and *AdaDelta* are tested using different neural network structures. It has been observed that gradient explosion occurs while applying *AdaGrad* and *AdaDelta* for FD002, but this problem does not take place when applied to FD002.[27] Therefore, Figure 5 only presents

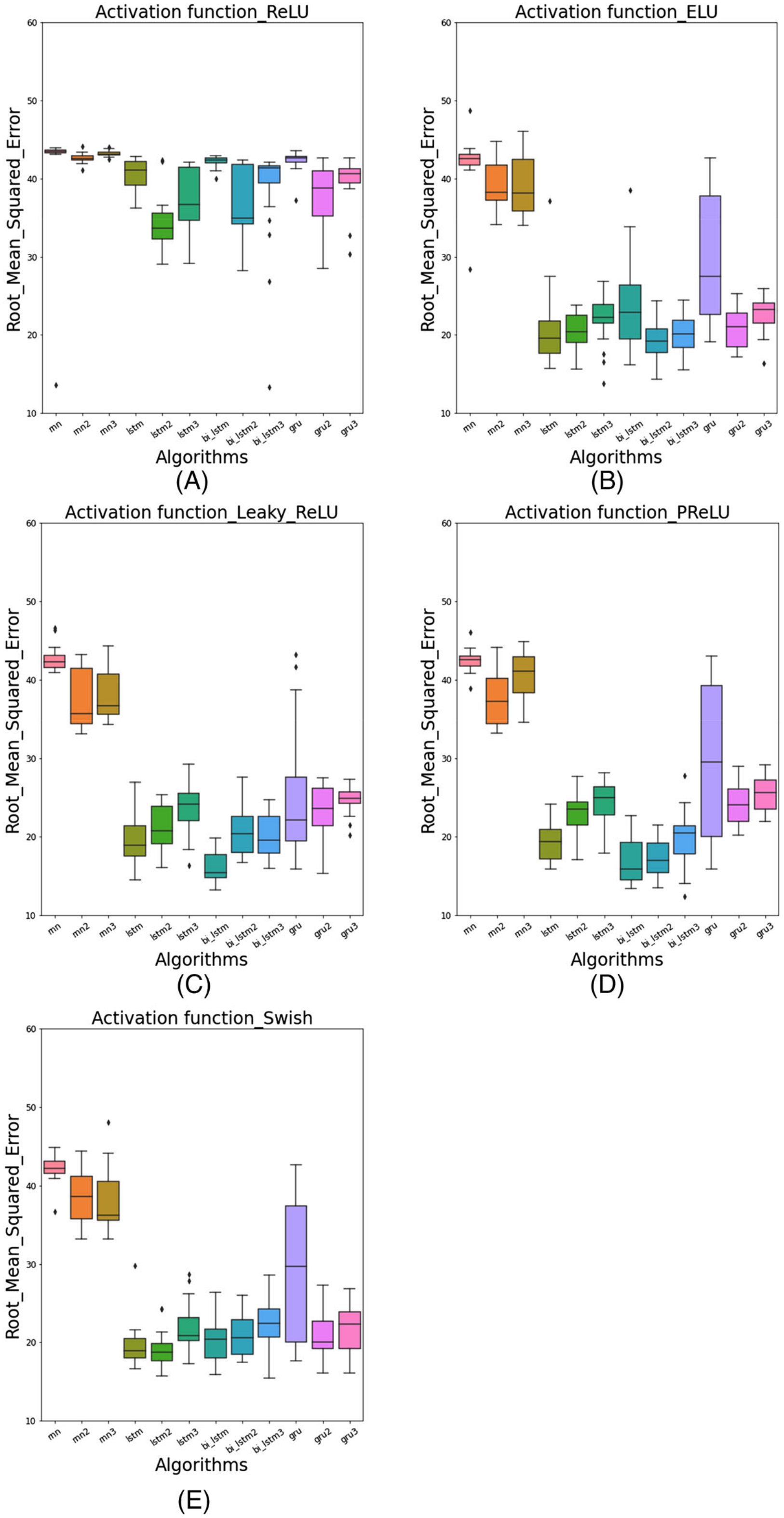


FIGURE 6 RUL prediction performance using different activation functions for FD002; (A) ReLU; (B) ELU; (C) Leaky ReLU; (D) PReLU; (E) Swish. ELU, Exponential Linear Unit; PReLU, Parametric Rectified Linear Unit; ReLU, Rectification of Linear Unit; RUL, remaining useful life.

the prediction performance of *RMSprop* and *Adam*. It can be observed that *RMSprop* outperforms *Adam* for most models in terms of accuracy. Therefore, we select *RMSprop* for further analysis. The same parameters and *Adam* were used to test the prediction performance of five activation functions, including *ReLU*, *ELU*, *Leaky_ReLU*, *PReLU* and *Swish* to choose the optimal activation function. Inspection of the results, shown in Figure 6, suggests that the performance is similar. *ReLU* is selected for further analysis due to its popularity.

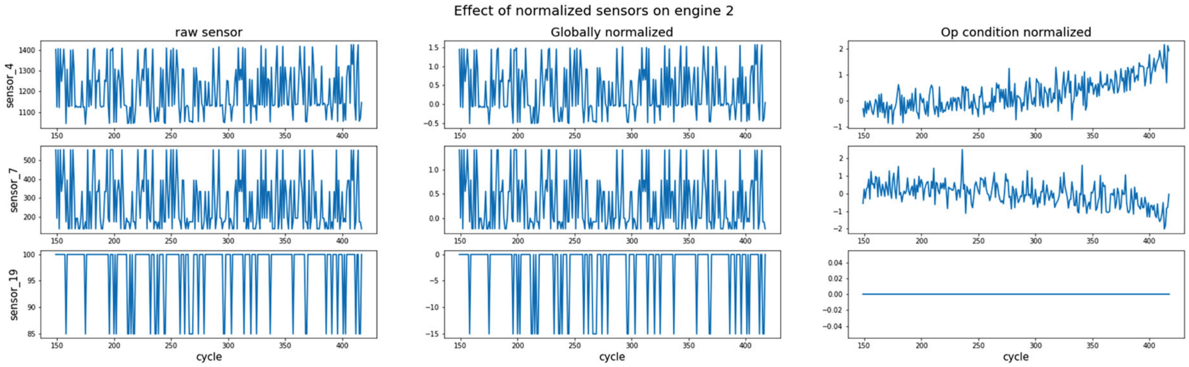


FIGURE 7 Comparison of three selected input variables without normalisation, with the global normalisation, and with the operation-based normalisation approach.

3.3 | Performance of the proposed three-stage feature selection method

Figure 7 gives a detailed comparison of three selected sensors between the data without normalisation, with global normalisation and operation-based normalisation. It can be observed that the signal after the global normalisation remains the same pattern as the raw signal, and the only change is the scale. However, after the operation-based normalisation, the data reveals the trends hidden in the raw signals. For example, there is an apparent increasing trend in sensor_4 and a decreasing trend in sensor_7, particularly after the data point of 200, which is unlikely observed in the raw data or the data after the global normalisation. Such a trend could play an important role in improving the performance of RUL prediction. Furthermore, there is a variation in the raw data and the data after the global normalisation for sensor_19, but this variation disappears after the operation-based normalisation. The reason is that sensor_19 is only influenced by the operating condition. Such a sensor should be removed before proceeding with the RUL prediction.

Figure 8 demonstrates the input signals of engine 2 in FD002 after operation-based normalisation, where it can be observed that signals of set_3, sensor_1, sensor_5, sensor_18 and sensor_19 are constant. Therefore, these five features are dropped first. Then, Spearman's rank coefficient between each input and the RUL was calculated to determine the relevant variables, as shown in Figure 9A. Set 1, set 2, Sensor_8 and Sensor_13 are removed since Spearman's rank coefficients with RUL are much smaller than the other sensors. After that, the pair-wise Pearson correlation coefficients among the input channels are calculated, and the operation-based normalisation results are shown in Figure 10B. Sensor_9 and sensor_14 have a high correlation coefficient of 0.96. Therefore, Sensor_9 was removed according to this observation.

The results of Spearman's rank coefficients between the sensors and the output RUL using two different normalisation methods are displayed in Figure 9. It is shown that the operation-based method (Figure 9B) produces higher correlations than the global normalisation method (Figure 9A). The comparison suggests that the proposed operation-based method produces the input variables, which have a better chance of modelling the output (RUL). Furthermore, revealing irrelevant features in Figure 9B is more straightforward.

It can be observed that different normalisation methods have a significant impact on the correlation coefficients among sensors and lead to a different strategy of feature selection. It has been observed that many features are almost the same after the global normalisation, as shown in Figure 10A. However, when the operation-based normalisation approach is used, the Pearson correlation coefficients among the input channels are smaller in most cases. This is because the global normalisation method can significantly reduce the influence caused by environmental noise and demonstrate the actual degradation trend of the sensors.

3.4 | Performance of RUL prediction

The selected features are then fed to a two-layer LSTM model with the fixed parameters chosen from the above results. It should be noted there are 24 engines whose number of cycles in the test dataset is less than the sequence length of 50 used in the model. Therefore, these engines were removed from the testing dataset due to insufficient sampling data. The degradation of the engines is generally classified into two stages: the normal performance stage showing relative flat RUL values at the initial cycles, and the performance degradation stage showing an exponential drop trend of the RUL values

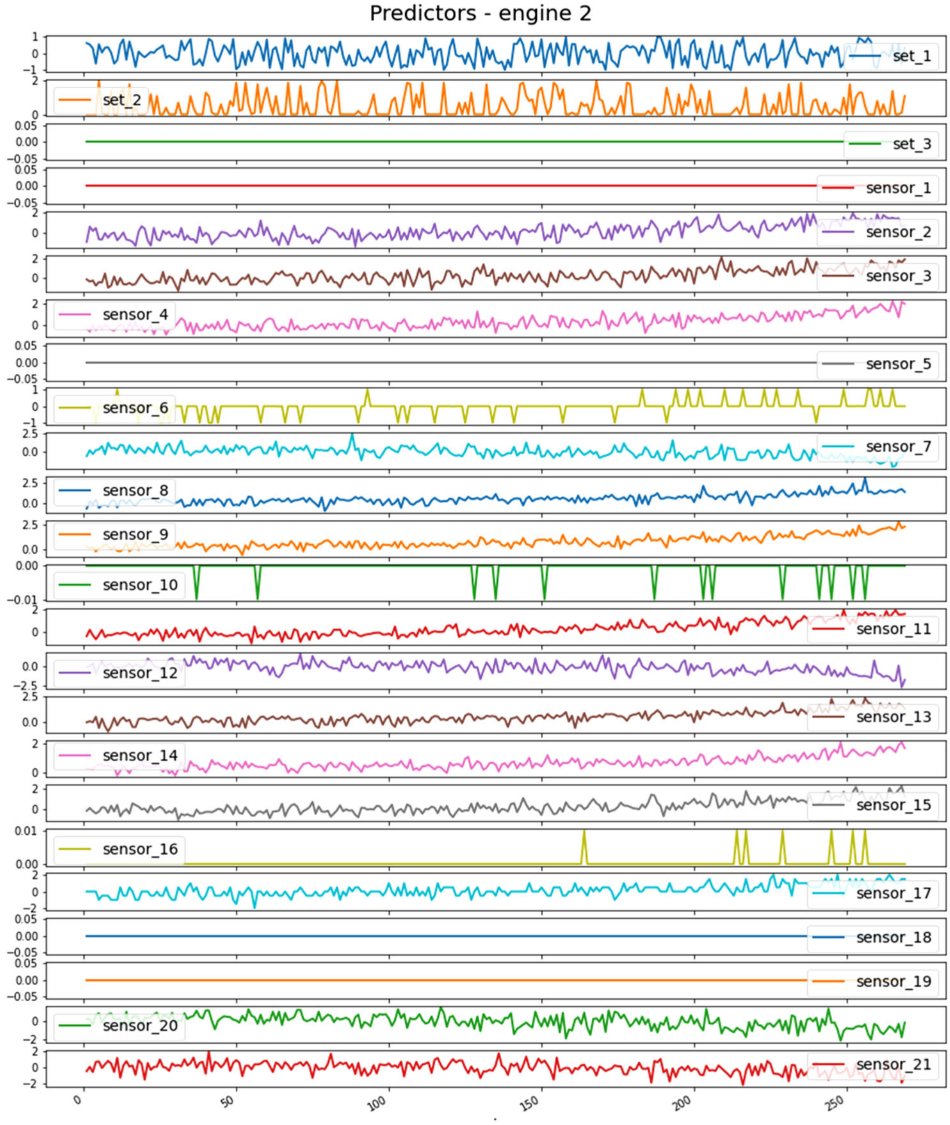


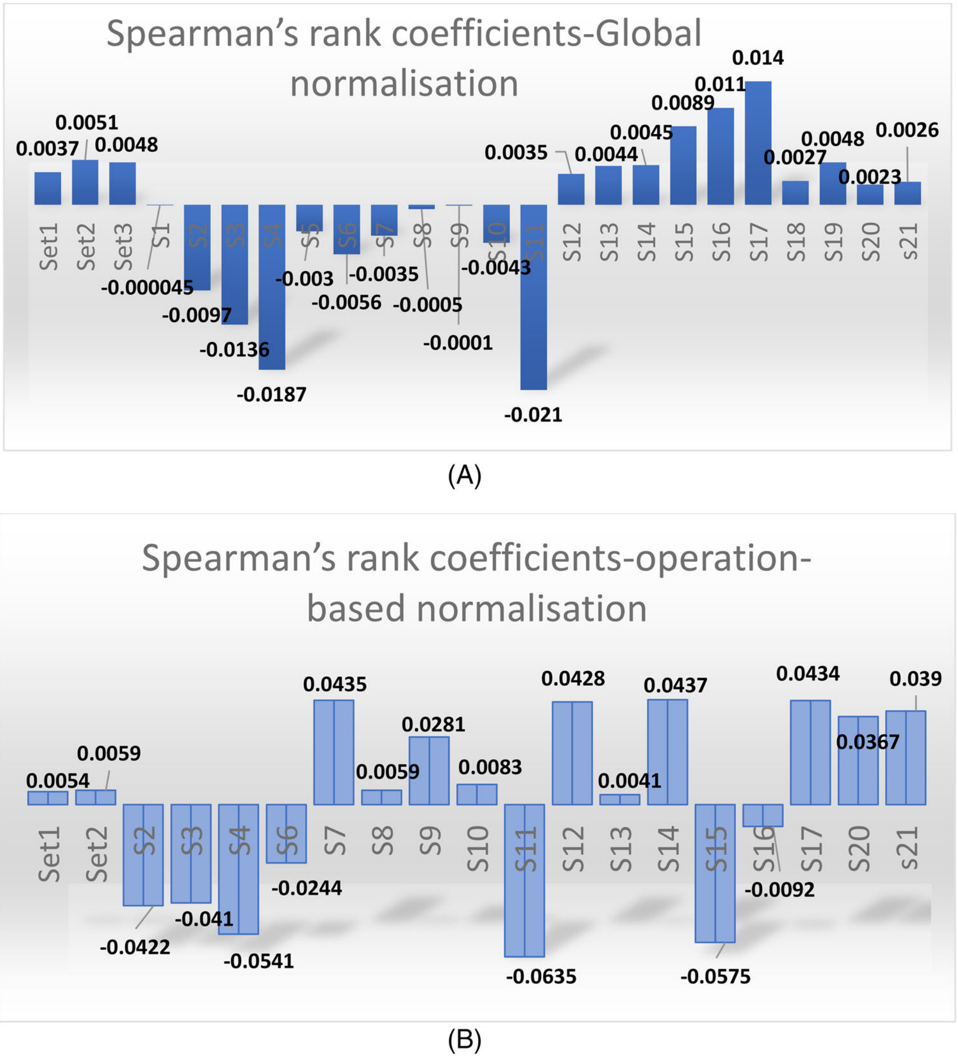
FIGURE 8 The input signals after applying the operation-based normalisation.

at the later cycles. In this study, the RUL is assumed constant in the normal stage until it crosses a specific cut-off value. In the paper, the cut-off value is set to 120. The results shown below are only for the data points whose RUL values are less than 120.

The results of the predicted RUL using the proposed method on the training and testing dataset of FD002 are shown in Figure 11. The left plot illustrates the training result for every cycle and each engine. The right plot illustrates the RUL prediction result of the testing set. It is interesting to observe that the prediction error for different cycles is similar to the training dataset while the error is reduced following the decrease of RUL for the testing dataset.

To detail the prediction performance of each engine, Figure 12 demonstrates the RUL prediction result for engines with lifecycles no less than 100. It can be observed that, in general, the RUL prediction becomes increasingly accurate following the increase in the cycle. This is the reason why most related studies used the RUL value of the last cycle to evaluate the model performance. It has also been observed that the prediction performance compromises if the available testing data number is smaller. This is probably because the system has not wholly entered the performance degradation stage.

To better show the prediction performance, Figure 13 plots the uncertainty quantification of the RUL prediction of 114 engines with cycle length larger than 100. It can be observed that the solid red curve, the prediction average overall, aligns with the ground truth, illustrated by the blue dot line. For most of the cycles, the predicted value is larger than



(A)

(B)

FIGURE 9 Spearman's rank coefficients between each input and the output. (A) Based on global normalisation, (B) based on the operation-based normalisation.

the ground truth, particularly for the initial cycles (<60 cycles). The prediction has a superior performance for the later cycles (>60 cycles). The grey area, indicating the 95% confidence interval, suggests that the uncertainty of the prediction decreases following the increment of the lifecycle.

It should be noted that the proposed framework can accommodate all other DL methods. To demonstrate this, Table 4 shows the RMSE values of the RUL prediction of the final cycle for all engines with the global normalisation and the proposed operation-based normalisation, where 12 popular models are tested. An average improvement of 6.84% has been achieved for all models, demonstrating the proposed solution's contribution.

To further extend the application of the proposed method, Table 5 shows the RMSE values of the RUL prediction using state-of-the-art methods without and with the operation-based normalisation. The superior performance of this approach is demonstrated again with an average improvement of 7.3%

4 | CONCLUSION

In this study, a novel three-stage feature engineering approach enhanced by an optimised DL-based model was presented to improve the RUL prediction performance of a system under multiple operating conditions. The activation function and

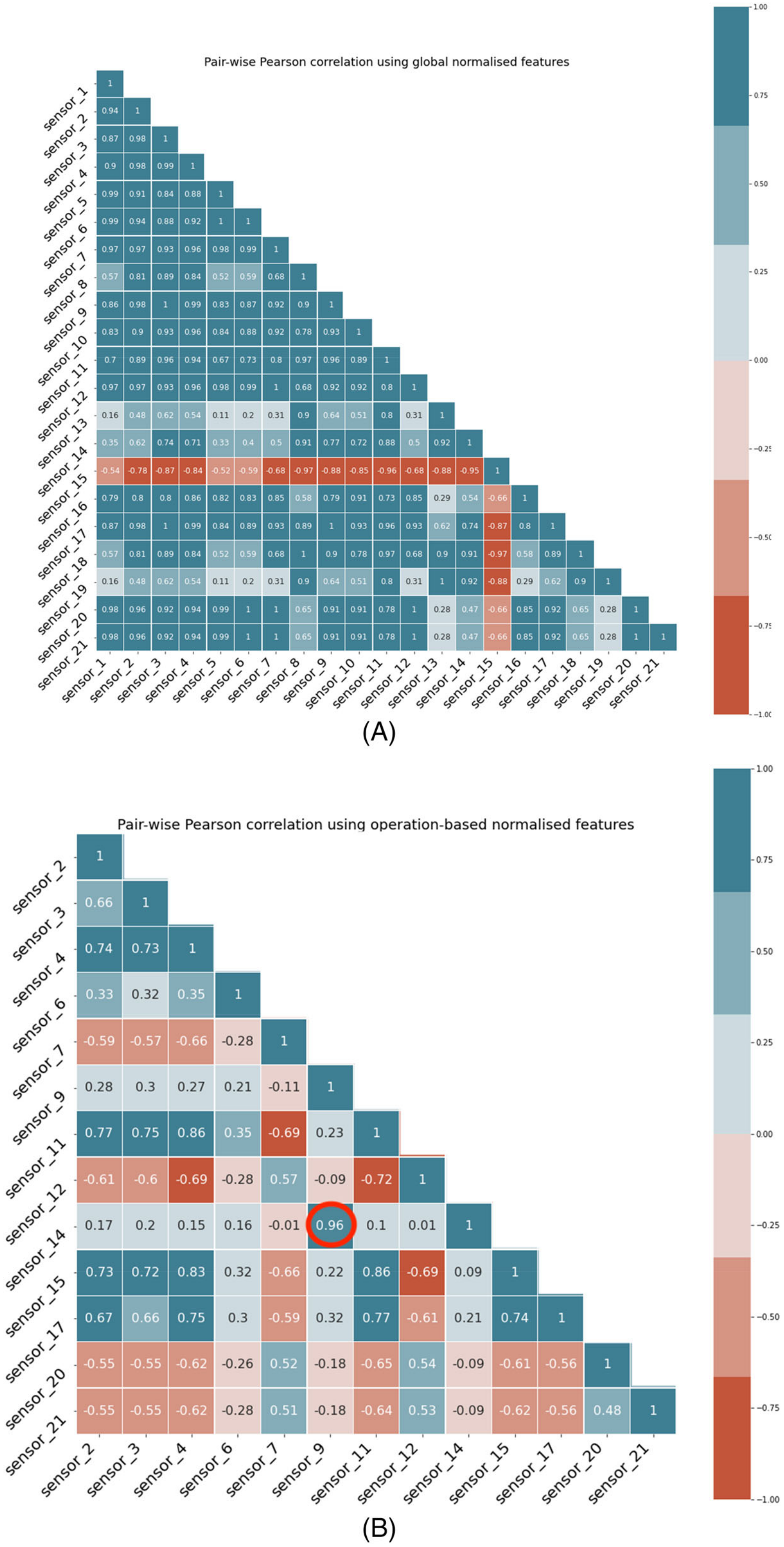


FIGURE 10 Pair-wise Pearson correlation coefficient for the input channels using (A) the global normalisation and (B) the operation-based normalisation.

optimiser optimisation of DL was also introduced to improve the prediction results. A k-medoids clustering model was used to categorise the raw data into six clusters based on the operation conditions. Then, the operation-based normalisation approach is applied to find the actual degradation trend hidden in the raw signal. The preliminary feature selection decision is made by plotting the operation-based normalised signals. After that, Two correlation analysis methods are used to eliminate the irrelevant and redundant input variables to reduce the computational cost further and avoid the over-fitting problem. The result demonstrates that operation-based normalisation improves the correlation between the

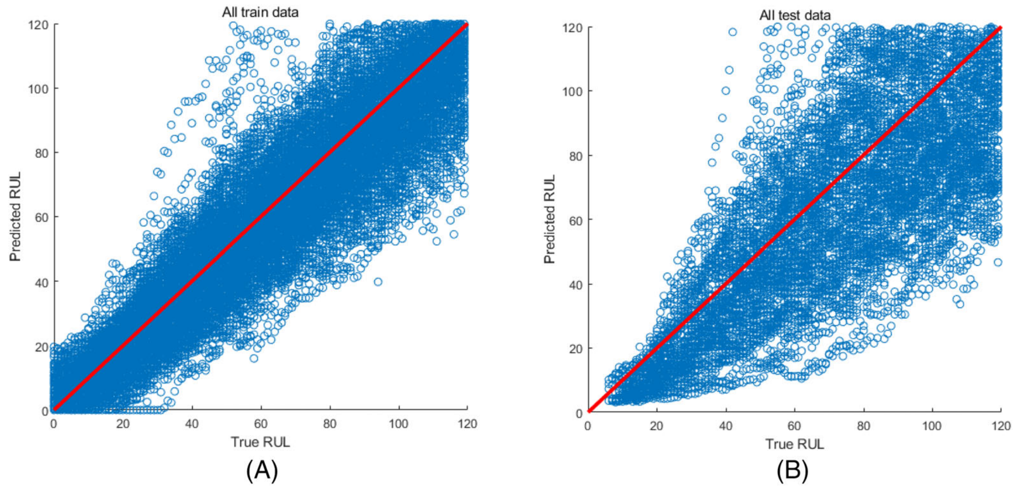


FIGURE 11 The predicted remaining useful life (RUL) versus the true RUL of each cycle and each engine for the (A) training and (B) testing dataset.

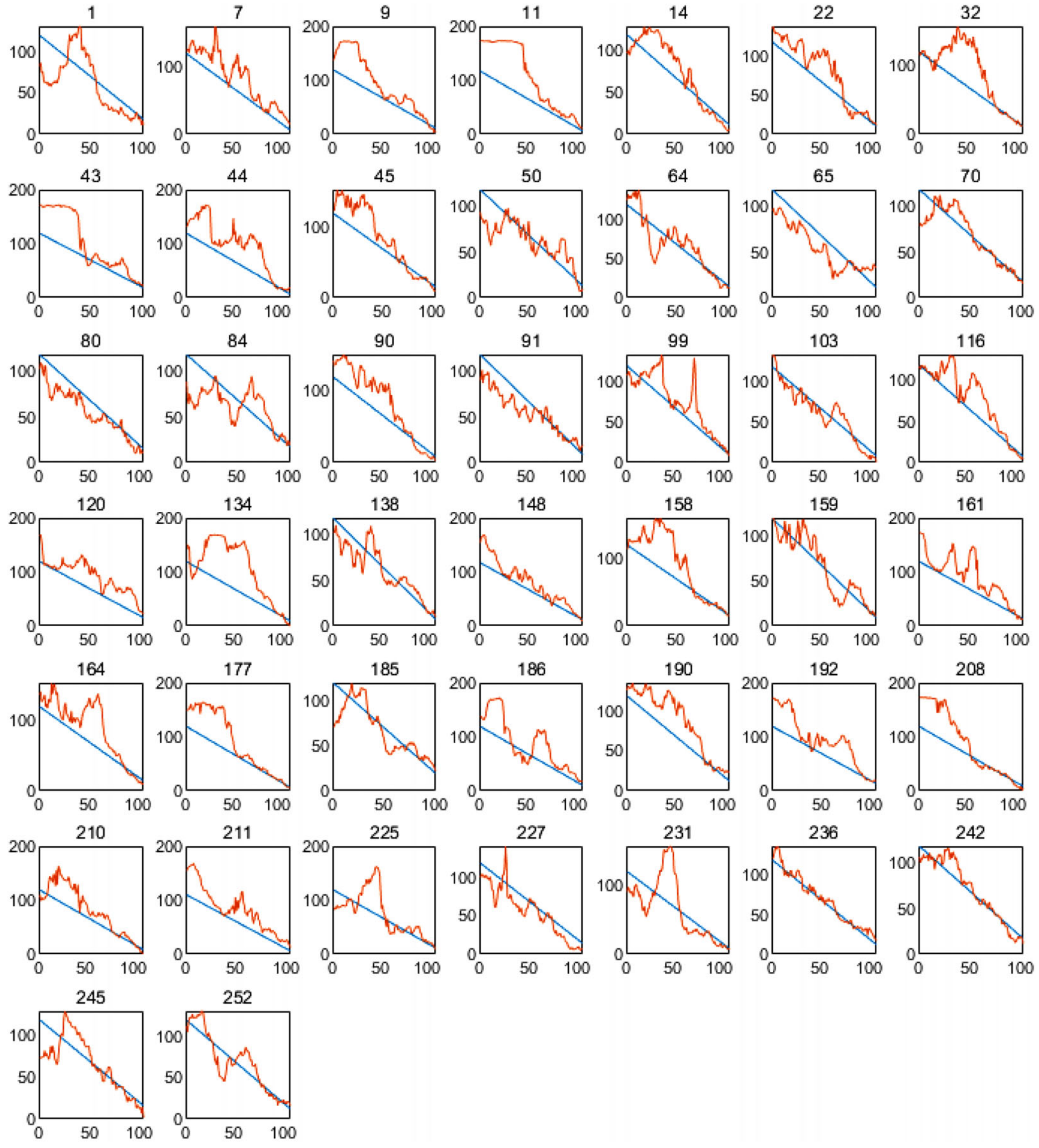


FIGURE 12 Prediction results for all engines with lifecycles no less than 100.

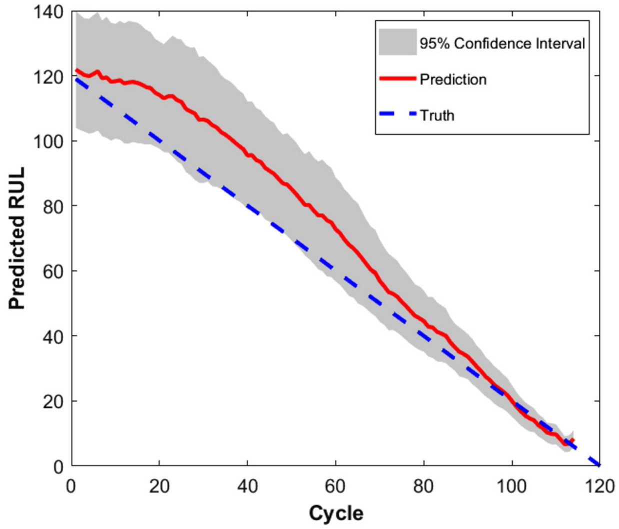


FIGURE 13 Uncertainty quantification of the remaining useful life (RUL) prediction using the proposed method by combining all 1114 engines.

TABLE 4 Root mean square error (RMSE) values of the remaining useful life (RUL) prediction of the final cycle for all engines with the global normalisation and the proposed operation-based normalisation.

Model	Global normalisation (GN)	Operation-based normalisation (ON)	Improvement ($\frac{GN-ON}{GN}$)
RNN-1LAYER	35.88	31.40	12.4%
RNN-2LAYERS	32.12	30.80	4.0%
RNN-3LAYERS	33.47	30.65	8.4%
LSTM-1LAYER	30.78	30.48	1.0%
LSTM-2LAYERS	33.01	30.32	8.1%
LSTM-3LAYERS	34.26	31.23	8.8%
Bi_LSTM-1LAYER	31.66	30.66	3%
Bi_LSTM-2LAYERS	32.59	31.01	4.8%
Bi_LSTM-3LAYERS	35.62	32.86	7.5%
GRU1-LAYER	32.41	31.48	2.9%
GRU2-LAYERS	34.74	30.76	11.4%
GRU3-LAYERS	35.46	32.00	9.8%

input variable and the output (RUL), which leads to higher prediction accuracy. Furthermore, it greatly enhances the stability of the model by reducing the gradient explosion or gradient vanishing problems. The results demonstrate that the proposed framework can consistently improve the performance of popular RNN variants and state-of-the-art for RUL prediction. This DL-based prognostics framework provides a reliable RUL prediction strategy for a complex system that works under multiple operating conditions. However, it should be noted that the optimisation is based on testing every feature engineering method and prognostic parameter. It could be very time consuming and can not necessarily present the most accurate RUL prediction performance. Therefore, future work will focus on designing new implementation algorithms to speed up the training progress and achieve better prediction results.

TABLE 5 Improvement of the proposed method for state-of-the-art methods.

Model	Global normalisation (GN)	Operation-based normalisation (ON)	Improvement $(\frac{GN-ON}{GN})$
Baseline LR	34.10	32.30	5.3%
Baseline ridge	34.10	32.10	5.9%
XGBoost	29.70	28.50	4%
Attention-based LSTM[28]	24.50	22.40	8.6%
Attention-based GRU[28]	28.50	24.90	12.6%

ACKNOWLEDGEMENT

The authors received no financial support for the research and publication of this article.

DATA AVAILABILITY STATEMENT

Data sharing is not applicable to this article as no new data were created or analysed in this study.

ORCID

Yifan Zhao https://orcid.org/0000-0003-2383-5724

REFERENCES

1. Vogl GW, Weiss BA, Helu M. A review of diagnostic and prognostic capabilities and best practices for manufacturing. *J Intell Manuf*. 2019;323:148-156. https://doi.org/10.1007/s10845-016-1228-8
2. Li Q, Gao Z, Tang D, Li B. Remaining useful life estimation for deteriorating systems with time-varying operational conditions and condition-specific failure zones. *Chin J Aeronaut*. 2016;133:223-236. https://doi.org/10.1016/j.cja.2016.04.007
3. Salunkhe T, Jamadar NI, Kivade SB. *Prediction of remaining useful life of mechanical components – a review*. 2014;3:125–135.
4. Cubillo A, Perinpanayagam S, Esperon-Miguez M. A review of physics-based models in prognostics: application to gears and bearings of rotating machinery. *Adv Mech Eng*. 2016;185:372-382. https://doi.org/10.1177/1687814016664660
5. Javed K, Gouriveau R, Zerhouni N. State of the art and taxonomy of prognostics approaches, trends of prognostics applications and open issues towards maturity at different technology readiness levels. *Mech Syst Signal Process*. 2017;94:214–236. https://doi.org/10.1016/j.ymssp.2017.01.050
6. An D, Kim NH, Choi JH. Practical options for selecting data-driven or physics-based prognostics algorithms with reviews. *Reliab Eng Syst Saf*. 2015;133:223–236. https://doi.org/10.1016/j.ress.2014.09.014
7. Lei Y, Li N, Guo L, Li N, Yan T, Lin J. Machinery health prognostics: a systematic review from data acquisition to RUL prediction. *Mech Syst Signal Process*. 2018;8:799–834. https://doi.org/10.1016/j.ymssp.2017.11.016
8. Ma J, Su H, Zhao W, Liu B. Predicting the remaining useful life of an aircraft engine using a stacked sparse autoencoder with multilayer self-learning. *Complexity*. 2018;323:148-156. https://doi.org/10.1155/2018/3813029
9. Zhao R, Yan R, Chen Z, Mao K, Wang P, Gao RX. Deep learning and its applications to machine health monitoring. *Mech Syst Signal Process*. 2019;72-73:92-104. https://doi.org/10.1016/j.ymssp.2018.05.050
10. Zhao R, Yan R, Chen Z, Mao K, Wang P, Gao RX. Deep learning and its applications to machine health monitoring. *Mechanical Systems and Signal Processing*. 2019;115:213-237. https://doi.org/10.1016/j.ymssp.2018.05.050
11. Lu C, Wang ZY, Qin WL, Ma J. Fault diagnosis of rotary machinery components using a stacked denoising autoencoder-based health state identification. *Signal Process*. 2017;94:214-236. https://doi.org/10.1016/j.sigpro.2016.07.028
12. Gan M, Wang C, Zhu C. Construction of hierarchical diagnosis network based on deep learning and its application in the fault pattern recognition of rolling element bearings. *Mech Syst Signal Process*. 2016;72–73:92–104. https://doi.org/10.1016/j.ymssp.2015.11.014
13. Wang J, Ma Y, Zhang L, Gao RX, Wu D. Deep learning for smart manufacturing: methods and applications. *J Manuf Syst*. 2018;134:144–156. https://doi.org/10.1016/j.jmsy.2018.01.003
14. Zheng S, Ristovski K, Farahat A, Gupta C. Long short-term memory network for remaining useful life estimation. In: *2017 IEEE International Conference on Prognostics and Health Management, ICPHM*. Vol. 104. IEEE; 2017:799-834. https://doi.org/10.1109/ICPHM.2017.7998311
15. Chen J, Jing H, Chang Y, Liu Q. Gated recurrent unit based recurrent neural network for remaining useful life prediction of nonlinear deterioration process. *Reliab Eng Syst Saf*. 2019;29:662-674. https://doi.org/10.1016/j.ress.2019.01.006
16. Elsheikh A, Yacout S, Ouali M-S, Elsheikh A, Yacout S, Ouali MS. Bidirectional handshaking LSTM for remaining useful life prediction. *Neurocomputing*. 2018;182:208-218. https://doi.org/10.1016/j.neucom.2018.09.076
17. Li X, Zhang W, Ding Q. Deep learning-based remaining useful life estimation of bearings using multi-scale feature extraction. *Reliab Eng Syst Saf*. 2019;130:377-388. https://doi.org/10.1016/j.ress.2018.11.011

18. Tran VT, Thom Pham H, Yang BS, Tien Nguyen T. Machine performance degradation assessment and remaining useful life prediction using proportional hazard model and support vector machine. *Mech Syst Signal Process*. 2012;2018:1-13. https://doi.org/10.1016/j.ymssp.2012.02.015
19. Zhu J, Chen N, Shen C. A new data-driven transferable remaining useful life prediction approach for bearing under different working conditions. *Mech Syst Signal Process*. 2020;139:106602. https://doi.org/10.1016/j.ymssp.2019.106602
20. Kundu P, Darpe AK, Kulkarni MS. Weibull accelerated failure time regression model for remaining useful life prediction of bearing working under multiple operating conditions. *Mech Syst Signal Process*. 2019;134:106302. https://doi.org/10.1016/j.ymssp.2019.106302
21. Madhulatha TS. *Comparison between K-Means and K-Medoids Clustering Algorithms, CCIS*. Springer; 2011.
22. Chitrakar R, Chuanhe H. Anomaly detection using support vector machine classification with k-medoids clustering. In: Asian Himalayas International Conference on Internet. 2012. https://doi.org/10.1109/AHICI.2012.6408446
23. Hochreiter S, Schmidhuber J. Long short-term memory. *Neural Comput*. 1997;32:320-330. https://doi.org/10.1162/neco.1997.9.8.1735
24. Wang Y, Zhao Y, Addepalli S. Practical options for adopting recurrent neural network and its variants on remaining useful life prediction. *Chin. J. Mech. Eng. (Engl. Ed.)*. 2021;30:79-95. https://doi.org/10.1186/s10033-021-00588-x
25. Zhang J, Wang P, Yan R, Gao RX. Long short-term memory for machine remaining life prediction. *J Manuf Syst*. 2018;48:78–86. https://doi.org/10.1016/j.jmsy.2018.05.011
26. Saxena A, Goebel K, Simon D, Eklund N. Damage propagation modeling for aircraft engine run-to-failure simulation. In: 2008 International Conference on Prognostics and Health Management, PHM 2008. 2008. https://doi.org/10.1109/PHM.2008.4711414
27. Ruder S. An overview of gradient descent optimization algorithms. 2016:1–14. https://arxiv.org/abs/1609.04747
28. Ayodeji A, Wang W, Su J, Yuan J, Liu X. An empirical evaluation of attention-based multi-head models for improved turbofan engine remaining useful life prediction. 2021. https://arxiv.org/abs/2109.01761
29. Cho K, Van Merriënboer B, Gulcehre C, et al. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). 2014. 1724-1734, Doha, Qatar. Association for Computational Linguistics. https://doi.org/10.3115/v1/d14-1179
30. Elsheikh A, Yacout S, Ouali MS. Bidirectional handshaking LSTM for remaining useful life prediction. *Neurocomputing*. 2019;115:213-237. https://doi.org/10.1016/j.neucom.2018.09.076
31. Kingma DP, Ba JL. Adam: a method for stochastic optimization. In: 3rd International Conference on Learning Representations, ICLR 2015 – Conference Track Proceedings. 2015:1–15.
32. Zhao R, Yan R, Wang J, Mao K. Learning to monitor machine health with convolutional Bi-directional LSTM networks. *Sensors (Switzerland)*. 2017;139:1–18. https://doi.org/10.3390/s17020273
33. Zoph B, Le QV. Searching for activation functions. In: 6th International Conference on Learning Representations, ICLR 2018 – Workshop Track Proceedings. 2018:1–13. https://arxiv.org/abs/1710.05941

How to cite this article: Wang Y, Zhao Y. Three-stage feature selection approach for deep learning-based RUL prediction methods. *Qual Reliab Eng Int*. 2023;1-25. https://doi.org/10.1002/qre.3288

APPENDIX A: NORMALISATION APPROACHES

Standardisation

Standardisation is a transform for data with a Gaussian distribution. It standardises features by subtracting the mean and dividing the result by the standard deviation of the data sample. It can be regarded as subtracting the mean value or centring the data. The standard score z of the variable x is calculated as:

$$\hat{x} = \ (x - u)\,/s \tag{A.1}$$

where u and s are the mean and standard deviation of the variable, respectively. Standardising a dataset means rescaling the distribution of values so that the mean of observed values is 0 and the standard deviation is 1. Centring and scaling are applied to every feature by calculating the relevant statistics on the samples in the training dataset. The stored mean u and standard deviation s are then used on the test dataset.

Min–Max normalisation

Min–Max normalisation is a transform that rescales the data from the original range to a new fixed range of 0 and 1. It is based on the assumption that the data's minimum and maximum observable values are provided or can be accurately estimated. The transformation equation is given by:

$$\hat{x} = (x - \min) / (\max - \min) \tag{A.2}$$

where the minimum and maximum values pertain to the value x being normalised. Normalisation is often used as an alternative to standardisation. However, normalisation may not be the best method to estimate these expected values if the time series trends up or down.

Robust standardisation

Robust standardisation is used to scale input variables in the presence of outliers. The probability distribution will be skewed if some outlier values exist in the input variable. Therefore, data standardisation will be very difficult because these outliers skew the calculated mean and standard deviation. Robust standardisation is presented by removing these outliers from the mean and standard deviation calculation. This is fulfilled by calculating the median and the 25% and 75% as shown below:

$$\hat{x} = (x - \text{median}) / (p75 - p25) \tag{A.3}$$

The scaled data has a zero mean and median, and a standard deviation of 1. Although outliers do not skew the scaled data, these outliers still have the same relative relationships to other values.

APPENDIX B: RNN AND ITS' VARIANTS

Recurrent neural network

In a traditional neural network, inputs are independent, while in RNN, the front neurons pass the information to the following neurons. As illustrated in Figure B.1, unlike a traditional feed-forward neural network, an RNN can be regarded as numerous copies of the same neural network cell, in which each cell passes the message to the next through the hidden state. In other words, the output from a recurrent neuron is connected to the next one to characterise the current system state as a function of current sensing data and the initial system state.

In an unrolled RNN, the sensing data $(\ldots x_{t-1},\ x_t,\ x_{t+1}\ldots)$ are fed simultaneously into the corresponding neurons, which generate the corresponding neuron time series $(\ldots\ h_{t-1},\ h_t,\ h_{t+1}\ \ldots)$. The output of a single recurrent neuron can be expressed as:

$$h_t = \sigma(W_x x_t + W_h h_{t-1} + b) \tag{B.1}$$

$$y_t = \text{softmax}\left(W_y h_t + c\right) \tag{B.2}$$

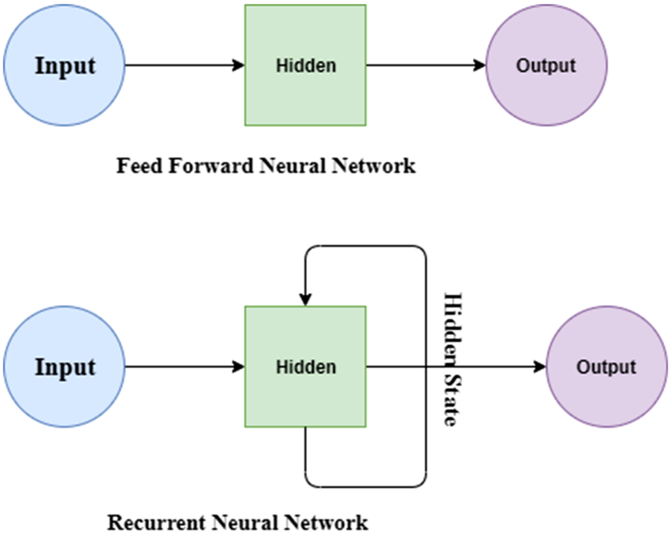


FIGURE B.1 Basic feed-forward neural network structure and recurrent neural network structure.

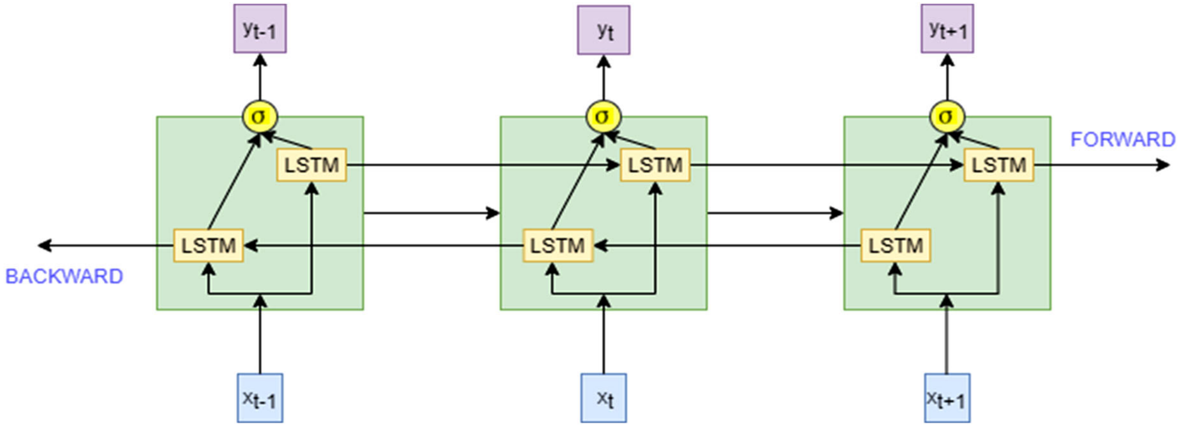


FIGURE B.2 A bi-directional long-short term memory (LSTM) structure.

where W_x, W_h and W_y represent the weight vectors, respectively. The symbol b and c denote the bias term, and σ is the activation function, with the hyperbolic tangent or ReLU being commonly used in RNN. y_t is the recurrent neuron's output based on the hidden state's output h_t, which can be referred to as a memory space containing the information of the current input and the former hidden state h_{t-1}. All the weight vectors are shared at every step, meaning that the same task is repeated at every step with different inputs, and the memory is renewed accordingly.

Gated recurrent unit (GRU)

The shortcoming of LSTMs is that it is usually time-consuming due to the forget gate, input gate and output gate added to the structure of the memory blocks. To address this problem, an improved structure, named GRU, was proposed.[29]

GRU is the latest generation of RNN, which looks very similar to LSTM. Instead of using the cell state, GRU uses the hidden state to transfer information. Moreover, it only has two gates (a reset gate and an update gate) instead of three. Like the forget and input gate of LSTM, the function of the update gate is to decide what information to keep and what to throw away. The function of the reset gate is to decide what to keep from past information.

The output of GRU at step t is calculated using the following equations:

$$z_t = \sigma\left(U^Z x_t + W^Z h_{t-1} + b_Z\right) \tag{B.3}$$

$$r_t = \sigma\left(U^r x_t + W^r h_{t-1} + b_r\right) \tag{B.4}$$

$$\tilde{h}_t = \tanh\left(U^{\tilde{h}} x_t + W^{\tilde{h}} h_{t-1} \cdot r_t + b_{\tilde{h}}\right) \tag{B.5}$$

$$h_t = (1 - z_t) \cdot h_{t-1} + z_t \cdot \tilde{h}_t \tag{B.6}$$

Since GRU has fewer tensor operations, it runs relatively faster when training the structure than LSTM. However, the accuracy is behind LSTM due to fewer gates. Thus, GRU could be a good option when the computational resource is limited or short training is required. For instance, Chen *et al.*[15] adopted a GRU network to predict the RUL for a complex system with multiple components, states, and parameters.

Bi-directional LSTM

Another variant of RNN called bi-directional LSTM can be seen frequently in literature in recent years. The bi-directional LSTM is proposed with the information flowing back to the former LSTM cells. The forward flow of information can discover the system variation, which flows back to smooth the predictions, as illustrated in Figure B.2. The outputs of the forward path and the backward path are then concatenated. The governing equations of Bi-directional LSTM can be presented as follows:

$$h_i^1 = f\left(U^1 \cdot x_i + W^1 \cdot h_{i-1}\right) \tag{B.7}$$

$$h_i^2 = f\left(U^2 \cdot x_i + W^2 \cdot h_{i-1}\right) \tag{B.8}$$

$$y_i = \text{sof tmax}\left(V \cdot \left[h_i^1; h_i^2\right]\right) \tag{B.9}$$

where Equation (B.7) refers to the forward path and Equation (B.8) refers to the backward path, y_i is the output of the bi-directional LSTM obtained by fusing the results from both directional paths.

As for the application, Zhao *et al.*[32] presented an integrated approach of CNN and bi-directional LSTM for machining tool wear prediction named convolutional bi-directional LSTM (CBLSTM) networks. CNN was first used to extract robust local features from the sequential input. Then, bi-directional LSTM was utilised for encoding temporal information. The proposed CBLSTM's capability of predicting the RUL of actual tool wear based on raw sensory data was verified with a real-life tool wear test. Zhang *et al.*[25] presented a bi-directional LSTM network to discover the underlying patterns embedded in time series to track system degradation. The bi-directional LSTM network was implemented to track the variation of the health index, and the RUL was predicted by the recursive one-step ahead method. Elsheikh *et al.*[30] built a bidirectional handshaking LSTM (BHLSTM) network for RUL prediction, where short sequences of monitored observations were given with random initial wear. This method was able to predict the RUL with a random start, which makes it more suitable for real-world application as the initial condition of physical systems is usually unknown, especially in terms of its manufacturing deficiencies.

APPENDIX C: OPTIMISERS AND ACTIVATION FUNCTIONS

Optimisers

Gradient descent is the most used way to optimise neural networks.[27] It is an iterative optimisation algorithm used to find the values of parameters or coefficients of a function that minimises a cost function. Although various algorithms have been developed to optimise gradient descent, they are usually used as black-box optimisers because it is hard to figure out practical explanations of their strengths and weaknesses. Ruder stated that Adagrad, Adadelta, RMSprop and Adam could all significantly improve the robustness of stochastic gradient descent (SGD) and do not need much manual tuning of the learning rate. These four optimisers are therefore selected and discussed in more detail in this paper.

Adagrad

Adagrad is a gradient-based optimiser that adapts the learning rate to the parameters, performing more significant updates for infrequent and more minor updates for frequent updates. Thus, it is very suitable for sparse data. It uses a different learning rate for every parameter θ_i at every time step t, the gradient of the objective function $g_{t,i}$ regarding the parameter θ_i at time step t is written as:

$$g_{t,i} = \nabla_{\theta_t} J\left(\theta_{t,i}\right) \tag{C.1}$$

The SGD updates for every parameter θ_i at each time step t following

$$\theta_{t+1,\,i} = \theta_{t,i} - \eta \cdot g_{t,i} \tag{C.2}$$

Adagrad modifies the general learning rate η at each time step t for every parameter θ_i based on the past gradients:

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,ii} + \in}} \cdot g_{t,i} \tag{C.3}$$

where $G_t \in R^{d\times d}$ is a diagonal matrix where each diagonal element i is the sum of the squares of the gradients regarding the parameter θ_i at time step t, $\in$ is a smoothing term used to avoid division by zero.

One of the main advantages of Adagrad is that it is not required to tune the learning rate manually. The default value is set as 0.01. The main drawback of this optimiser is that its accumulation of the squared gradients in the denominator would result in the learning rate shrinking and becoming infinitesimally small, which means that at a certain point, the algorithm can no longer acquire additional knowledge.

Adadelta

To reduce the monotonically decreasing learning rate, an extension optimiser of Adagrad has been promoted, named Adadelta. It uses a fixed-size window of accumulated past gradients instead of accumulating all past squared gradients. The sum of the gradient is recursively defined as a decaying average of all history-squared gradients. Thus, the running average of the squared gradients of the objective function at time step t depends on the previous average and the current gradient:

$$E\left[g^2\right]_t = \gamma E\left[g^2\right]_{t-1} + (1-\gamma)\, g_t^2 \tag{C.4}$$

where γ is the fraction of the update vector of the past time step to the current update vector, which is normally set to 0.9.[27]

The SGD update for parameter $\Delta\theta_t$ at each time step t therefore becomes:

$$\Delta\theta_t = -\eta \cdot g_{t,i} \tag{C.5}$$

$$\theta_{t+1} = \theta_t + \Delta\theta_t \tag{C.6}$$

And according to the update rule, through simply replacing the diagonal matrix G_t with the decaying average of past squared gradients $\mathrm{E}[g^2]_t$, the parameter update vector of Adadelta can be derived as:

$$\Delta\theta_t = -\frac{\eta}{\sqrt{E[g^2]_t + \in}} \cdot g_{t,i} \tag{C.7}$$

As $\sqrt{E[g^2]_t + \in}$ is the root mean squared (RMS) error criterion of the gradient, it can then be written as:

$$\Delta\theta_t = -\frac{\eta}{\mathrm{RMS}[g]_t} \cdot g_{t,i} \tag{C.8}$$

Since the update should have the same hypothetical units as the parameter, the exponentially decaying average of the squared parameter should be used:

$$E\left[\Delta\theta^2\right]_t = \gamma E\left[\Delta\theta^2\right]_{t-1} + (1-\gamma)\,\Delta\theta_t^2 \tag{C.9}$$

$$\Delta\theta_t = -\frac{\mathrm{RMS}[\Delta\theta]_{t-1}}{\mathrm{RMS}[g]_t} \cdot g_t \tag{C.10}$$

$$\theta_{t+1} = \theta_t + \Delta\theta_t \tag{C.11}$$

Based on the update rule of Adadelta, there is no need to set a default learning rate.

RMSprop

RMSprop is an adaptive learning rate method designed for neural networks, which has grown in popularity in recent years. Like Adadelta, the central idea of RMSprop is to keep the moving average of the squared gradients for each weight and then divide the gradient by square rooting the mean square. However, a good default value of decay parameter γ and learning rate is set to 0.9 and 0.001:

$$E\left[g^2\right]_t = 0.9E\left[g^2\right]_{t-1} + 0.1g_t^2 \tag{C.12}$$

$$\theta_{t+1} = \theta_t - \frac{0.001}{\sqrt{E[g^2]_t + \in}} \cdot g_{t,i} \tag{C.13}$$

Adam

Another method that computes adaptive learning rates for each parameter is Adaptive Moment Estimation (Adam).[31] Adam not only stores an exponentially decaying average of past squared gradients v_t, but also keeps an exponentially decaying average of past gradients m_t, as shown in Equations (C.14) and (C.15).

$$m_t = \beta_1 \, m_{t-1} + (1 - \beta_1) \, g_t \tag{C.14}$$

$$v_t = \beta_2 \, v_{t-1} + (1 - \beta_2) \, g_t^2 \tag{C.15}$$

where m_t refers to the estimate of the first moment (the mean) of the gradients and v_t refers to the second moment (the uncentered variance) of the gradients. As the initial value of m_t and v_t are vectors of zeros, it is observed that when the decay rates are low during the initial time, they are biased towards zero. The biases are counteracted by computing bias-corrected first and second moment estimates:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{C.16}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{C.17}$$

Therefore, the update rule of Adam can be derived as:

$$\theta_{t+1} = \theta_t \ - \frac{\eta}{\sqrt{\hat{v}_t + \in}} \cdot \hat{m}_t \tag{C.18}$$

The proposed values for β_1, β_2 and $\in$ are 0.9, 0.999 and e10-8, respectively.

Activation function

The activation function is a function working on a neuron in an ANN and mapping the input of the neuron to the output. More specifically, each neuron node in the neural network adapts the output of the neuron in the upper layer as the input and passes it to the next layer (hidden layer or output layer). Thus, the activation function refers to the functional relationship between the output of the upper node and the input of the lower node in the multi-layer neural network. Three types of activation functions are typically used in the DL area: tan*h* and sigmoid, ReLU and swish. This section reviews the primary mathematical expression of these three types of activation functions with their advantages and drawbacks.

Sigmoid and tan*h*

The sigmoid function, expressed in Equation (C.19), also known as the logistic function, is normally used for the output of the hidden layer neurons.

$$\Sigma \ (x) = \frac{1}{1 + e^{-x}} \quad x \in (0, 1) \tag{C.19}$$

The advantage of the Sigmoid function is that the output of the activation function is limited between 0 and 1, which results in a stable optimisation and is thus suitable to be used as the output layer. The drawback is that the function could be very insensitive to small changes in input when a variable takes a substantial positive or negative value. The weight will hardly be updated during the backpropagation when the gradient gets close to zero. Therefore, the gradient will disappear, and the network can complete its training. In addition, the output of the sigmoid function is not zero mean, which leads to the input of neurons in the back layer being non-zero mean, and then affects the gradient. Besides, the computational complexity is very high due to the exponential form in the sigmoid function.

Tanh function, expressed in Equation (C.20), is also called the hyperbolic tangent function:

$$f \ (x) = \ \tan h \ x \ = \frac{\sin hx}{\cos hx} \ = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad x \in (-1, 1) \tag{C.20}$$

Tanh function is the translation and contraction of the sigmoid function: $\tan h(x) \ = \ 2 \cdot \sigma(2x) - 1$. tan*h* function often outperforms sigmoid in practice because its output is zero mean. Nevertheless, it still suffers from gradient saturation and computational complexity.

ReLU

Rectification of Linear Unit (ReLU) is the most used DL neural network activation function. It is the default activation function for most of the feed-forward neural networks. The ReLU function is written as:

$$f\ (x) = \max (0, x) \tag{C.21}$$

The advantage of the ReLU function is that the SGD algorithm converges faster than sigmoid or tan$h$. When the weight is larger than zero, there are no problems like gradient saturation and gradient disappearance. Since there is no need to carry out the exponential operation, the computational complexity is relatively low. A threshold is needed to achieve the activation value. The limitation of the ReLU function is that the output is not zero mean either. Besides, the Dead ReLU Problem will occur when the weight is in the negative field. During the training, when x is less than zero, the gradient of the current neuron and the neurons after it is always zero. In other words, it will no longer respond to any data, and the corresponding parameters will never be updated. To solve this problem, Leaky ReLU, Parametric Rectified Linear Unit (PReLU) and Exponential Linear Unit (ELU) were introduced.

Leaky ReLU function:

$$f(x) = \max (0.01x, x) \tag{C.22}$$

PReLU function:

$$f(x) = \max (\alpha x, x) \tag{C.23}$$

ELU:

$$f(x) = \begin{cases} \alpha \left(e^{x} - 1\right), & x\ \leq 0 \\ x, & x\ > 0 \end{cases} \tag{C.24}$$

The Leaky ReLU uses a small value of 0.01 to initialise the neuron so that the ReLU function can be activated in the negative region. The difference between Leaky ReLU and PReLU is that the α of the PReLU function is learned through backpropagation. ELU has all the advantages of ReLU and no Dead ReLU Problem. It can make the average activation mean value of neurons close to zero at the same time, which suggests that it is more robust to noise. However, the calculation complexity is relatively higher because of the exponential form.

Swish

Swish is a self-gated activation function that uses an automated search technique to find novel activation functions to replace the ReLU function without changing the network architecture. By combining exhaustive and reinforcement learning-based search, they found a few novel promising activation functions and named the best one as swish.

The swish function can be written as:

$$f\ (x) = x \cdot \mathrm{sigmoid}\,(\beta x) \tag{C.25}$$

where β is a constant or trainable parameter.

In stage 2, we propose to use pair-wise Spearman's rank coefficient between each input and each output to remove the irrelevant input variables. The limitation of Pearson correlation is that it cannot be adopted to tell the nonlinear relationship between a dependent variable and an independent variable. Spearman rank correlation does not have any assumptions about the data distribution. Therefore, it is the appropriate correlation analysis method when the variables are measured on a scale that is at least ordinal. The formula used to calculate the Spearman rank correlation is written as

$$p\ =\ 1 - \frac{6 \sum d_i^2}{n\,(n^2 - 1)} \tag{C.26}$$

where p is the Spearman rank correlation, d_i the difference between the ranks of corresponding variables (x and y) and n is the number of observations. The rank is calculated as the average value when there are ties in input or output. The scores range between −1 and 1, where −1 refers to perfectly negatively correlated, and 1 refers to perfectly positively correlated.

In stage 3, since the RUL prediction of a system is a regression predictive modelling problem with numerical input variables and output variables, we propose to inspect the pair-wise Pearson's correlation coefficients among all input variables to remove the redundant input variables. Pearson's correlation coefficient measures the strength of the linear relationship between two data samples, either time-series data or non-time-series data. It is calculated as the covariance of two variables (x and y) divided by the product of the standard deviation of each variable, written as:

$$r_{xy} = \frac{S_{xy}}{s_x s_y} \tag{C.27}$$

where r_{xy} is the correlation coefficient, S_{xy} the covariance and s_x, s_y are the standard deviations. r_{xy} ranges from −1 to +1. A correlation coefficient of 1 indicates that for every positive increase in one variable, there is a positive increase of fixed proportion in another variable. A correlation coefficient of −1 means that for every positive increase in one variable, there is a negative decrease of fixed proportion in another variable. A correlation coefficient of 0 means that these two variables are unrelated. The absolute value of the correlation coefficient refers to the relationship strength between the two variables.

AUTHOR BIOGRAPHIES

Youdao Wang received a BEng degree in Civil Engineering from Xi'an Jiaotong-Liverpool University, China, and the University of Liverpool, United Kingdom, in 2016. He received an MSc degree in Structural Engineering from the University of Manchester, United Kingdom, in 2017. He is pursuing a PhD degree in Manufacture from Cranfield University, United Kingdom. His research interests include data processing for asset management the prediction of the remaining useful life for complex industrial systems.

Yifan Zhao was born in Zhejiang, China. He received a PhD degree in Automatic Control and System Engineering from the University of Sheffield, Sheffield, UK, in 2007. He is currently a Reader in Data Science at Cranfield University, Cranfield, UK. His research interests include computer vision, signal processing, non-destructive testing, active thermography and nonlinear system identification.