Mechanical Systems and Signal Processing 129 (2019) 764–780

Contents lists available at ScienceDirect

Mechanical Systems and Signal Processing

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

Remaining useful life estimation using a bidirectional recurrent neural network based autoencoder scheme

Wennian Yu *, Il Yong Kim, Chris Mechefske

Department of Mechanical and Materials Engineering, Queen's University, Kingston, ON K7L 3N6, Canada

ARTICLE INFO

Article history:
Received 20 August 2018
Received in revised form 5 March 2019
Accepted 2 May 2019
Available online 9 May 2019

Keywords:
Remaining useful life
Bidirectional recurrent neural network
Autoencoder
Health index

ABSTRACT

System remaining useful life (RUL) estimation is one of the major prognostic activities in industrial applications. In this paper, we propose a sensor-based data-driven scheme using a deep learning tool and the similarity-based curve matching technique to estimate the RUL of a system. The whole procedure consists of two steps: in the first step, a bidirectional recurrent neural network based autoencoder is trained in an unsupervised way to convert the multi-sensor (high-dimensional) readings collected from historical run-to-failure instances (i.e. multiple units of the same system) to low-dimensional embeddings, which are used to construct the one-dimensional health index (HI) values to reflect various health degradation patterns of the instances. In the second step, the test HI curve obtained from sensor readings collected from an on-line instance is compared with the degradation patterns built in the offline phase using the similarity-based curve matching technique, from which the RUL of the test unit can be estimated at an early stage. The proposed scheme was tested on two publicly available run-to-failure datasets: the turbofan engine datasets (simulation datasets) and the milling datasets (experimental datasets). The prognostic performance of the proposed procedure was directly compared with the existing state-of-art prognostic models in terms of various prognostic metrics on the two datasets respectively. The comparison results demonstrate the competitiveness of the proposed method used for RUL estimation of systems.

1. Introduction

Prognostic and health management (PHM) has gained significant attention recently as it helps to improve system health management and ensure systems working as designed under their optimal functioning conditions [1]. Remaining useful life (RUL) estimation is the most common task in the research filed of PHM [2]. It aims at providing valuable information of the health state of a system, and estimating the time left before the system reaches the safe operational limit, i.e. the point beyond which the system will no longer perform its desired function with an acceptable level of reliability [3]. However, the development of reliable prognostic systems for the RUL estimation has been a challenging task due to several inherent difficulties [3]. Numerous prognostic algorithms have been reported in the literature. They can be classified into three main categories: physics-based (model-based) methods [4–6], sensor-based data-driven methods [7–9], and hybrid methods [10–12].

* Corresponding author.
E-mail address: wennian.yu@queensu.ca (W. Yu).

https://doi.org/10.1016/j.ymssp.2019.05.005
0888-3270/

Physics-based methods rely on the mathematical descriptions of the system degradation, which are usually expressed as a series of ordinary or partial differential equations, for instance, the Paris crack growth model [5], gear tooth cracks [13] and spall progression models [14]. However, the establishment of physics-based models usually requires accurate and specific physical knowledge about system degradation or damage propagation processes, which are typically complex and difficult to obtain. Besides, most of the physics-based models are unable to be updated with on-line new measurements [15]. Thus, these methods have little applicability in reality. The sensor-based methods estimate the RUL of a machine through routinely collected run-to-failure data from machines of same type by various on-line monitoring sensors via data-driven models such as artificial neural networks [7,8], support vector machines [9], and hidden Markov models [16]. Compared with the physics-based methods, the sensor-based data-driven models collects the input and output data to train model, and thus are popular to handle complex modeling problems. Besides, they can be updated in real time and show good adaptability with machine operating conditions. Thus, sensor-based data-driven models are widely used in industrial applications but rely on the availability of run-to-failure data. Hybrid methods combine the physical knowledge and sensor measurements for the RUL estimation. One common strategy is to construct an analytical or empirical model based on physical knowledge and practical experience of the system operation, and the model parameters are learned and updated according to the on-line measurements using data-driven techniques, such as the particle filter [12,17,18]. Since 2006, deep learning has become a rapidly growing research area, and has been adopted by many researchers as a bridge connecting multi-sensor big data and intelligent machine health monitoring [19]. On the other hand, with the continuous development of instrumentation technology and computation systems, many companies can afford to continuously collect data from operating machines via numerous sensors and store the huge amount of data in devices or clouds with incredible storage capacity. Thus, the sensor-based data-driven models using deep learning tools have become more and more attractive for machine health monitoring.

Most current sensor-based data-driven methods for RUL estimation either use statistical methods [20,21] (e.g., stochastic process techniques), or artificial intelligence tools [22,23] (e.g., neural network). The former rely on the statistical models (Wiener process, Gamma process, etc.) to determine the RUL in a probabilistic way. The latter rely on machine learning tools and do not have a probabilistic orientation. In the literature, data-driven methods based on machine learning tools have been widely used for the RUL estimation, especially with the emergence of deep learning techniques since 2006. They can be further divided into two main groups [1,9]: direct mappings between inputs and RUL, and similarity-based health index (HI) curve matching, as shown in Fig. 1. The first group of methods directly builds the mappings between the inputs and the output target, i.e. RUL, via some popular machine learning models, such as neural networks. Normally, the inputs are carefully selected feature vectors extracted from sensor readings. The mappings are usually learnt in a supervised way by subjectively assigning RUL labels to the corresponding input feature vectors. The second group of methods first transform the run-to-failure multi-sensor readings (high-dimensional data) from a training instance (unit) into a one-dimensional time-history health index (HI) curve, which is used to represent the degradation of the unit from perfect healthy condition (HI = 1) to the failed condition (HI = 0). After HI curves of all available training instances of the same machine are obtained, they are maintained in a library representing various possible degradation trends (trajectories) of the machine. For a given test instance, its corresponding HI curve is compared with each training HI curve in the library, and the most similar training HI curves with the test HI curve are selected using a similarity-based curve matching strategy [2]. The final RUL is usually obtained as the weighted average of individual RUL estimations derived from selected similar training instances.

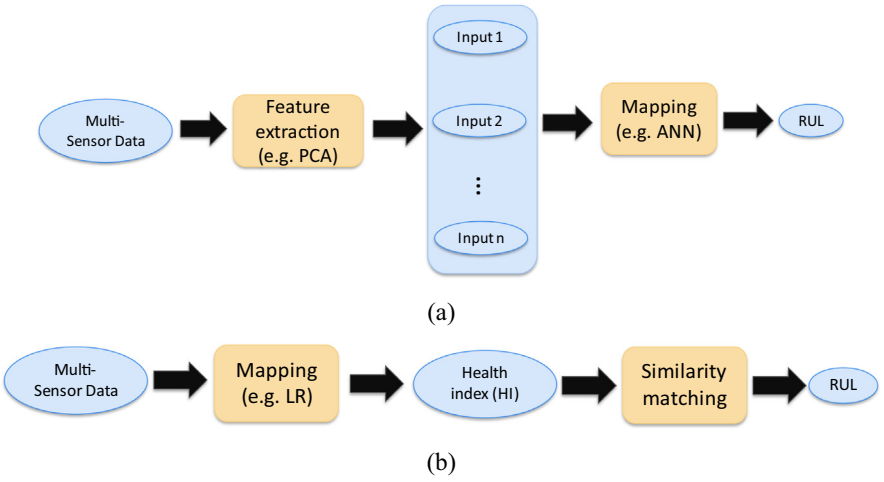


Fig. 1. Two main groups of data-driven methods using machine learning tools: (a) direct mappings between inputs and RUL, and (b) similarity-based health index curve matching.

One inherent challenge of the direct mapping methods is to determine the desired output (RUL) for a given input feature vector [24]. A sensible solution is to simply assign the desired RUL as the time left before reaching functional failure [7,9]. This solution inadvertently assumes that the RUL of a system linearly decreases with its operational time as long as the system starts to operate from its fresh new state. Another strategy is to subjectively assign the desired RULs over operational time based on degradation functions such as exponential functions and the piece-wise linear functions. A commonly used degradation function consists of two pieces of linear functions [8,23,24]. The first function has a constant value up to a time representing the maximum value of RUL. This constant function assumes that a unit starts to degrade only after an initial time of usage. The second function is a linearly decreasing function representing the linear decreasing of the allowable operating time margin of the system with its operational time. Other challenges related with the direct mapping methods include the manual feature selection procedure in order to yield satisfying performance, and the difficulty of incorporating new training instances.

As to the similarity-based HI curve matching methods, a key step is to transform the multivariate time series into univariate HI values. Yan et al. [25] presented a logistic regression model to achieve the transformation. However, it was later pointed out by Wang et al. [2] that the logistic regression will distort the original degradation pattern. They proposed a linear regression (LR) model to preserve the degradation pattern. The LR model was learnt by purposely assigning a target HI value of 1 with the sensor reading in the early stage, whereas a target HI value of 0 with the sensor readings near the end life of the unit. However, this methodology used only the early-life and end-life data to train the LR model. Ramasso [1] introduced local LR models which make use of the entire sensor reading by assigning the target HI values in between the early-life and end-life sensor readings through an exponential degradation function. The above-mentioned models assume the target HI values of a unit based purely on domain knowledge. Malhotra et al. [26] proposed a long short-term memory (LSTM) based autoencoder scheme to obtain the unsupervised HI values based solely on the multi-sensor measurements. The LSTM based autoencoder was trained to reconstruct the sensor readings, and the reconstruction errors were used to calculate the HI values. Recently, Gugulothu et al. [27] proposed a novel gated recurrent unit (GRU) based autoencoder to learn robust embeddings (representations) for multivariate time series subsequences, and used the robust embeddings to construct the HI values in an unsupervised manner. They found that the HI construction based on embeddings was superior to that of reconstruction errors as the reconstruction errors are sensitive to the noise level in the sensor readings whereas the embeddings are much robust against it. Compared with the direct mapping methods, one noticeable advantage of the HI curve matching methods is that new instances can be easily incorporated by just adding their HI curves into the library [28]. In addition, it has been demonstrated that the HI curve matching methods show good generalization ability and generally higher prediction performance on publicly available datasets. However, most reported HI curve matching methods require some domain-specific rules to improve prognostics on given datasets [1,2,26,27].

This study is a direct extension of the work done by Gugulothu et al. [27]. The novelty of our study is that we applied the bidirectional recurrent neural network (RNN) architecture on the RNN based autoencoder (RNN-ED) proposed in [27] to learn more robust embeddings from the multivariate input time series and use the embeddings to construct the one dimensional HI values without relying on any degradation trend assumption based on domain knowledge. Compared with the standard unidirectional RNNs, the bidirectional RNNs (BiRNNs) can capture the complete and sequential information from the time series in the forward and backward manner, which improves the reconstruction precision of the RNN based autoencoder. We will show that using the bidirectional RNN-ED generally improves the prediction performance compared with that of the standard unidirectional RNN-ED. The proposed approach was evaluated on two publicly available datasets for the RUL estimation of systems. One is the simulated turbofan engine datasets. The other is the measured milling datasets. The estimation results are directly compared with those from state-of-art algorithms reported in the literature. This paper is organized as follows: the overall methodology is described in section 2. The application and results of the proposed methodology on two datasets are presented in Section 3. Section 4 concludes this work.

2. Methodology

The complete procedure of the proposed approach for the RUL estimation is described in Fig. 2. It consists of two stages: offline construction of the HI library and the online RUL estimation for an input instance. In the offline stage, a linear regression model is learnt to directly transform the multi-sensor readings to the HI values of the system. Before learning, the target HI values are obtained from the embeddings generated by the proposed bidirectional RNN based autoencoder (or BiRNN encoder-decoder, BiRNN-ED), which is trained based on the available run-to-fail instances in an unsupervised way. For each offline run-to-failure instance, there is a HI curve reflecting the degradation path of the instance. Since the degradation of a system is usually a time-dependent, continuous and monotonically nonhealing phenomenon [3], the HI curve thus obtained usually needs to be smoothed by methods like moving average [26,27] and curve fitting [2,29] to alleviate the effect of measurement noise. A set of HI curves showing various possible degradation paths of the instances are maintained in the HI library. In the online stage, a duration of online sensor readings is first transformed to a section of HI curve, which will be compared with each offline HI curve in the library. The HI curves in the library that are most similar to the online HI curve will be selected by using the similarity-based curve matching method initially introduced by Wang et al. [2]. The RUL estimation of the online instance is inferred based on the selected instances as they show similar degradation trend with the online instance.

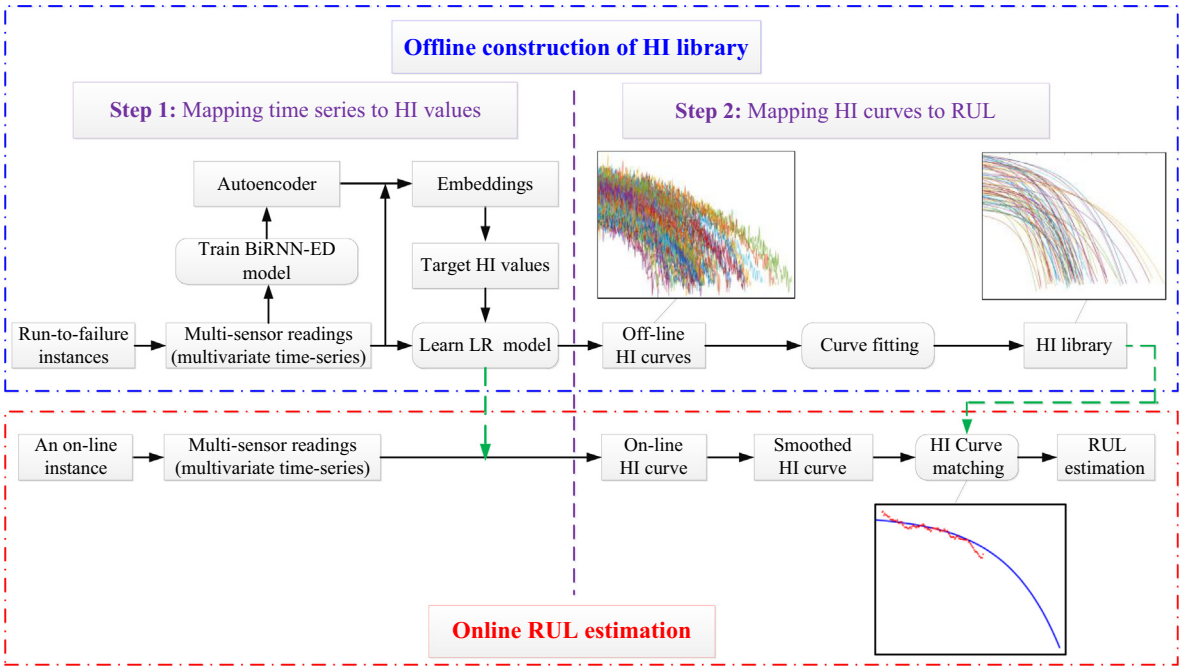
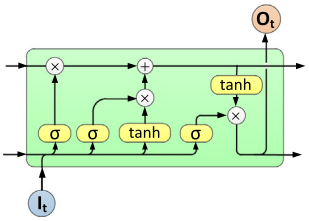
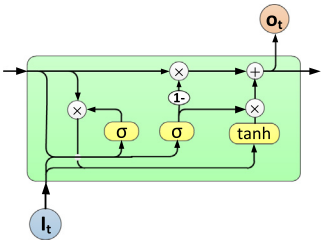


Fig. 2. Flowchart of the methodology.

2.1. Bidirectional RNN based autoencoder

In deep learning, an autoencoder is a type of neural network used to learn efficient code (embedding) in an unsupervised manner [30]. It consists of an encoder and decoder. The encoder learns to compress the input data into a short code, whereas the decoder learns to un-compress the code into a set of data that closely matches the input data. For the RNN autoencoder, it was found that only the RNNs with gating mechanism (e.g. LSTM [31] and GRU [32], as shown in Table 1) can achieve the purpose of recovering input time series [32,33]. Traditional RNN autoencoders are all unidirectional, meaning that the input time series can only be accessed in a positive time direction. Compared with the unidirectional RNNs, the bidirectional RNNs [34] maintain two groups of hidden layers, one for input sequences in the positive time direction (forward states), and the

Table 1
The architectures of the hidden unit of RNNs with gating mechanism.

	Structure	Notations
LSTM [31]		⊗ element-wise product ⊕ sum σ sigmoid function tanh hyperbolic tangent
GRU [32]		⊗ element-wise product ⊕ sum ① subtraction from 1 σ sigmoid function tanh hyperbolic tangent

Note: I_t and O_t are the input and output at the tth step. The details regarding the mathematic operations of the LSTM and GRU can be referred from [31,32].

other for input sequences in the negative time direction (backward states). By this special structure, the bidirectional RNNs are able to capture the time dependencies within a sequence in a forward and a backward manner. Inspired by the special structure of bidirectional RNNs, the bidirectional RNN encoder-decoder (BiRNN-ED) is proposed in this study to learn more robust embeddings from the input time series, as shown in Fig. 3.

Given an input time sequence $\boldsymbol{\Omega} = \{\boldsymbol{I}_1, \boldsymbol{I}_2, \ldots, \boldsymbol{I}_t\}$, where each point $\boldsymbol{I}_i$ ($i = 1, 2, \ldots, t$) is an m-dimensional vector depending on the dimension of input, which is either a univariate (i.e. $m = 1$) or a multivariate time series, the encoder BiRNN iterates through each point $\boldsymbol{I}_i$ in the time sequence $\boldsymbol{\Omega}$ in both forward and backward manners so that the final hidden state is obtained:

$$\boldsymbol{H}_F = \boldsymbol{H}^f{}_t \oplus \boldsymbol{H}^b{}_1 = f_e(\boldsymbol{\Omega}) \tag{1}$$

where $\boldsymbol{H}^f_t$ and $\boldsymbol{H}^b{}_1$ are the final hidden states resulting from the forward and backward processes respectively.

The decoder BiRNN has the same structure as the encoder BiRNN. However, it uses only the final hidden state of encoder $\boldsymbol{H}_F$ as the initial input to reconstruct the input time series in a reverse order $\boldsymbol{\Omega}' = \{\boldsymbol{I}'_t, \boldsymbol{I}'_{t-1}, \ldots, \boldsymbol{I}'_1\}$:

$$\boldsymbol{\Omega}' = f_d(\boldsymbol{H}_F) \tag{2}$$

The reconstruction error at the ith time step is $\boldsymbol{e}_i = \boldsymbol{I}'_i - \boldsymbol{I}_i$. The BiRNN encoder-decoder is trained to minimize the squared reconstruction error E given by:

$$E = \frac{1}{2}\sum_{i=1}^{t}(\| \boldsymbol{e}_i \|_2)^2 \tag{3}$$

where $\| \cdot \|_2$ is the 2-norm operator of a vector. Once the BiRNN encoder-decoder is well trained, the final hidden state of the encoder carries all the relevant information to reconstruct the input time series via the decoder. It is a usual practice to transform a multivariate time series into a one-dimensional vector representation via the autoencoder. Thus, if there are multiple hidden layers in the BiRNN autoencoder, the representation (embedding, or coding) $\boldsymbol{z}_t$ of the input is often obtained by the concatenation of the final hidden states from all the hidden layers in the encoder RNN:

$$\boldsymbol{z}_t = \boldsymbol{H}^1_F \oplus \boldsymbol{H}^2_F \oplus \ldots \boldsymbol{H}^L_F \tag{4}$$

where $\boldsymbol{H}_F{}^j$ is the final hidden state vector at the jth layer of encoder ($j = 1, 2, \ldots L$), and L is the number of hidden layers.

2.2. Linear regression based HI estimation

For a given sensor reading x_t at time t, our goal is to find the HI value h_t and determine the health status at that time. The exact functional mapping between the multi-sensor reading and HI is hard to obtain (or even doesn't exist). However, there are two general rules we can observe by experience: 1) the variation of readings from some sensors (if not all) can reflect the degradation trend of the system health statues (linearly or not), and 2) the degradation trend is normally an exponential pattern with time. A simple way to construct this mapping is through a linear regression model [21]:

$$h_{LR} = f(x) = \theta_0 + \boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{x} \tag{5}$$

where $\boldsymbol{x}^{\mathrm{T}} = [x_1, x_2, \ldots, x_m]$ is a m-dimensional vector representing the simultaneous readings from m sensors, and $\boldsymbol{\theta} = [\theta_1, \theta_2, \ldots, \theta_m]$ is the corresponding parameter vector. The parameters θ_0 and $\boldsymbol{\theta}$ can be learnt by minimizing the objective $\sum(h_{LR} - h_{Tg})^2$, where h_{Tg} is the target HI value. Thus, a sample set $\boldsymbol{\Omega} = \{(\boldsymbol{x}, h_{Tg})\}$ is required to train the linear regression

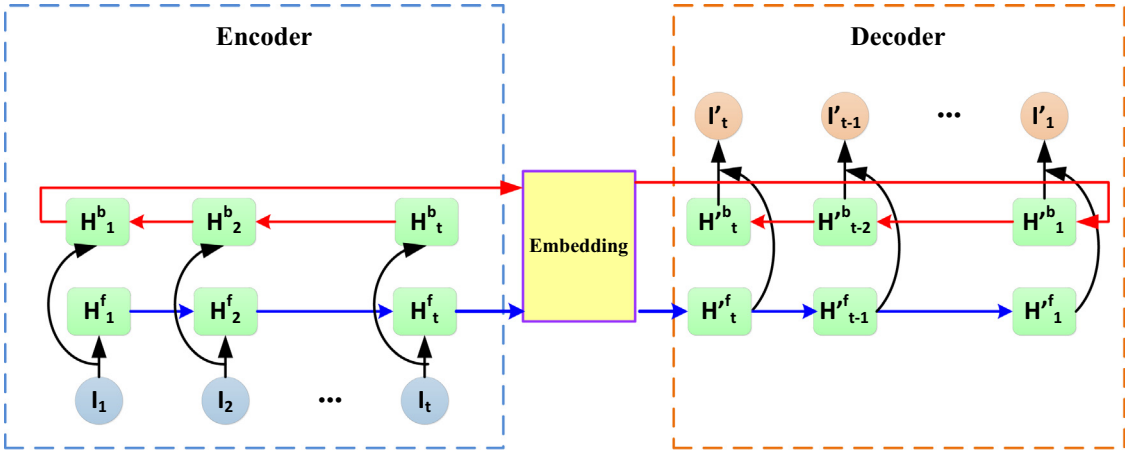


Fig. 3. The bidirectional RNN encoder-decoder.

model. The linear regression models have been widely used in the literature [1,2,18,20,23,26,27,29]. It has been shown that the prediction performance based on LR model works much better than the other regression models [26,27]. Most researchers obtained the targeted HI values based on domain experience or degradation models. In this study, we will use the above-mentioned BiRNN-ED to learn the LR model without relying on the domain knowledge or degradation models.

Supposing that a multivariate time series $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T\}$ represents the sensor signals collected from a machine up to its failure at T. Each point $\boldsymbol{x}_i$ has m dimensions representing the simultaneous readings from m sensors. Considering a window with a fixed size W sliding along $\boldsymbol{X}$ with a step size of 1, there will be $T - W + 1$ windows (sub-sequences) denoted by $\{\boldsymbol{\Omega}_1, \boldsymbol{\Omega}_2, \ldots, \boldsymbol{\Omega}_{T\text{-}W+1}\}$, and for the ith window, $\boldsymbol{\Omega}_i = \{\boldsymbol{x}_i, \boldsymbol{x}_{i+1}, \ldots, \boldsymbol{x}_{i+W\text{-}1}\}$. Using the **BiRNN** based autoencoder presented above, each window $\boldsymbol{\Omega}_i$ can be represented by a one-dimensional embedding $\boldsymbol{z}_{i+W\text{-}1}$. Thus, the multivariate time series $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T\}$ representing the entire operational life of a machine can be transformed into a univariate time series $\boldsymbol{Z} = \{\boldsymbol{z}_W, \boldsymbol{z}_{W+1}, \ldots, \boldsymbol{z}_T\}$ which stores the degradation scenario of the machine. This is just for one run-to-failure instance of the machine. For a population of historical run-to-failure instances $\boldsymbol{X}^{(j)} = \{\boldsymbol{x}_1^{(j)}, \boldsymbol{x}_2^{(j)}, \ldots, \boldsymbol{x}_{Tj}^{(j)}\}$ ($j = 1, 2, \ldots J$) of the same mechanical component, they can be similarly transformed into a set of univariate time series $\boldsymbol{Z}^{(j)} = \{\boldsymbol{z}_W^{(j)}, \boldsymbol{z}_{W+1}^{(j)}, \ldots, \boldsymbol{z}_{Tj}^{(j)}\}$ ($j = 1, 2, \ldots J$) via the BiRNN autoencoder representing different degradation scenarios of the component. It should be mentioned that the first few embeddings (say $\boldsymbol{z}_1$, $\boldsymbol{z}_2$, and $\boldsymbol{z}_3$) in the univariate time series $\boldsymbol{Z}$ for each instance can be considered as in a completely healthy state. These embeddings from all instances constitute a set of normal embeddings, $\boldsymbol{Z}_{norm}$. As a machine operates, its health degrades resulting in the embeddings gradually deviating from the set of normal embeddings [27]. The deviation of an embedding $\boldsymbol{z}_i^{(j)}$ from the normal embeddings $\boldsymbol{Z}_{norm}$ can be estimated by:

$$d_i^{(j)} = \frac{1}{N} \sum_{\boldsymbol{z} \in \boldsymbol{Z}_{norm}} \| \boldsymbol{z}_i^{(j)} - \boldsymbol{z} \|_2 \tag{6}$$

where N is the total number of embeddings within Z_{norm}. For the jth instance, the deviation $d_i^{(j)}$ of each embedding in $\boldsymbol{Z}^{(j)}$ with respect to $\boldsymbol{Z}_{norm}$ represents the gradual degradation of the jth instance. Thus, the health index (HI) of the jth instance can be obtained by normalizing $d_i^{(j)}$ into the range of $0 \sim 1$ by:

$$h_i^{(j)} = \frac{(d_i^{(j)})_{\max} - d_i^{(j)}}{(d_i^{(j)})_{\max} - (d_i^{(j)})_{\min}}, i = W, W+1, \cdots, T_j \tag{7}$$

where $(d_i^{(j)})_{\max}$ and $(d_i^{(j)})_{\min}$ are the maximum and minimum values of the deviation $d_i^{(j)}$ for the jth instance over its operational life, i.e. $i = W, W + 1, \ldots, T_j$, respectively. The HI curve of the jth instance $\boldsymbol{HI}^{(j)} = [h_W^{(j)}, h_{W+1}^{(j)}, \ldots, h_{Tj}^{(j)}]$ represents its degradation path. In this way, we can obtain the HI curves based on all sensor readings for all run-to-failure instances in an unsupervised manner without imposing any domain-specific knowledge. The HI curves thus obtained can be used as the target HI values to estimate θ_0 and $\boldsymbol{\theta}$ of the linear regression model. Once the regression trained, the HI curves for each training instance and any test instance for which the RUL is to be estimated can be obtained according to Eq. (5) by directly inputting corresponding sensor readings.

2.3. Similarity-based HI curve matching

The basic idea of this method is to find the top few run-to-failure training instances which have similar degradation trends (HI curves) with the test instance, and use them to estimate the RUL of the test instance [2]. In order to perform this task, a set of HI curves showing various degradation paths of the same unit are maintained in a library $\boldsymbol{\Sigma} = \{\boldsymbol{HI}^{(j)}\}$ ($j = 1, 2, \ldots J$) based on the available run-to-failure training instances. It should be mentioned that the HI curve obtained by the methodologies described in the last section is usually non-monotonic and noisy due to inherent measurement noise in sensor readings. Some researchers adopted common smoothing techniques such as the moving average [26,27] and curve fitting [2,29] to smooth the HI curve of the training instance. The HI curve of the test unit $\boldsymbol{HI}' = [h_1', h_2', \ldots, h_{T'}]$ is compared with each curve stored in the library $\boldsymbol{\Sigma}$ to find their similarity level. Since the training instance and test instance usually have different degrees of initial health, the test HI curve $\boldsymbol{HI}'$ needs to be shifted with a time lag τ to match with the training HI curve $\boldsymbol{HI}^{(j)}$. The similarity measure between a test HI curve $\boldsymbol{HI}'$ and a training HI curve $\boldsymbol{HI}^{(j)}$ with a time lag τ can be defined as [26,29]:

$$Sim(j, \tau) = \exp\left(\frac{-d(\boldsymbol{HI}', \boldsymbol{HI}^{(j)}, \tau)}{\lambda}\right) \tag{8}$$

where $d(\boldsymbol{HI}', \boldsymbol{HI}^{(j)}, \tau)$ is the distance measure between $\boldsymbol{HI}'$ and $\boldsymbol{HI}^{(j)}$ with a time lag τ. A larger distance means a smaller similarity. λ is a relaxing factor controlling the degree of similarity for a given distance measure. The distance measure can be defined as the average squared Euclidean distance between two vectors given by:

$$d(\boldsymbol{HI}', \boldsymbol{HI}^{(j)}, \tau) = \frac{1}{T'} \sum_{k=1}^{T'} (h_k' - h_{k+\tau}^{(j)})^2 \tag{9}$$

where T' is the time length of the test instance. The estimated RUL of the test instance benchmarked by the jth training instance with a given time tag τ is given by:

$$Rul(j,\tau) = T_j - T' - \tau \tag{10}$$

where T_j is the time length of the jth training instance, and τ is limited in the range $[0, T_j - T']$ as the RUL should be a positive value. Each individual estimation $Rul(j, \tau)$ is assigned a weight of $Sim(j, \tau)$. The final RUL estimation of the test instance is the weighted average of those RUL estimations which have higher similarity measures. The expression is given by [2,26,27,29,35]:

$$RUL = \frac{\sum_{j,\tau_g} Sim(j,\tau) * Rul(j,\tau)}{\sum_{j,\tau} Sim(j,\tau)},\ s.t. Sim(j,\tau) \geq \beta * \left(\max_{\substack{j\in[1,J] \\ \tau\epsilon[0,T_j-T']}} Sim(j,\tau) \right) \tag{11}$$

where $\max(Sim(j, \tau))$ is the maximum similarity value for all possible combinations of j and τ. β is a coefficient within the range [0, 1], which controls the number of individual estimation $Rul(j, \tau)$ to be considered for the final estimation.

3. Application and results

3.1. Turbofan engine datasets

We first applied the methodology on the publicly available turbofan engine datasets provided by the NASA prognostic data repository [36,37]. Six datasets were created using the turbofan engine simulation model called C-MAPSS (Commercial Modular Aero-Propulsion System Simulation) [38] to simulate the degradation scenarios of the turbofan engines under different operating conditions and fault modes as shown in Table 2. Datasets #1 through #4 represent an increasing level of complexity due to the effects of fault modes and operating conditions on the sensor readings. Each dataset consists of a training set and a test set. The training set includes a number of instances (units or engines) with complete run-to-failure data, which allows users to train their prediction models. The test set includes a number of instances with incomplete data that end prior to failure, whose RULs are to be determined by the users' prediction models once well trained. The last two datasets (Challenge Datasets) share the same training set. Each instance either from training set or test set is composed of a multivariate time series with 26 variables. The 1st variable represents the engine ID (j). The 2nd variable represents the operating time in cycles (t). The 3rd to 5th variables represent the three operating parameters (altitudes, Mach number and sea-level temperature) which have significant effects on the engine performance. The remaining 21 variables represent readings from 21 different sensors, which are contaminated with noise. Details of these sensors can be found in [38].

The ground true RUL values of the test engines are revealed for datasets #1 to #4 so that users can evaluate the performance of their models on their own. However, the ground truth RUL values for datasets #5 and #6 are not known to the public. For dataset #5, users have to upload their results (once per day) to receive a performance score based on an asymmetrical scoring function defined by the data creators. Dataset #6 is the final validation set that was used to rank the challenge participants. Users need to send their results to the challenge organizer and a score will be mailed back. Therefore, this dataset is seldom used in the literature due to the difficult access to the ground truth information for quick feedback during model development [28].

Three common metrics used to evaluate the performance of prediction models on the C-MAPSS datasets are the Score (S), the Accuracy (A), and the Root Mean Square Error (RMSE).

(1) Score S

This metric is defined by the data creators [38], which was used in the PHM 2008 Data Challenge to rank the prediction models proposed by different participants. The score of a prediction model on a test dataset with N units is given by the following scoring function:

$$S = \begin{cases} \sum_{i=1}^{N} (\exp(-\frac{e_i}{13}) - 1), & \text{if } e_i \leq 0 \\ \sum_{i=1}^{N} (\exp(\frac{e_i}{10}) - 1), & \text{if } e_i > 0 \end{cases} \tag{12}$$

Table 2
Details of the six datasets simulated by C-MAPSS.

Dataset	Turbofan Engine Degradation Simulation Dataset [36]				PHM08 Challenge Dataset [37]	
	#1	#2	#3	#4	#5	#6
Conditions	1	6	1	6	6	6
Fault Modes	1	1	2	2	1	1
Train Units	100	260	100	249	218	
Test Units	100	259	100	248	218	435

where e_i is the difference between the estimated RUL and the actual RUL for the ith test unit (e_i = estimated RUL – actual RUL). The smaller the value of the score S, the better the performance of the prediction model. As can be seen from Eq. (12), the scoring function is asymmetric such that late predictions ($e_i > 0$) are more heavily penalized than early predictions ($e_i < 0$). In either case, the score increases exponentially with the prediction error, meaning that RUL estimates far from the actual value are penalized exponentially.

(2) Accuracy A

This metric evaluates the performance of a prediction model in terms of the percentage of correct predictions. A correct prediction is confirmed when the prediction error e_i falls within the range of -13 and 10, i.e. $e_i \in [-13, 10]$. Thus,

$$A = \frac{100}{N}\sum_{i=1}^{N}\mathrm{Cor}(e_i) \tag{13}$$

where $\mathrm{Cor}(e_i) = 1$ if $e_i \in [-13, 10]$, otherwise $\mathrm{Cor}(e_i) = 0$ (incorrect prediction). A prediction is considered as false positive (FP) if $e_i < -13$, and false negative (FN) if $e_i > 10$.

(3) RMSE

In addition to the above two metrics, the root mean square error (RMSE) of the estimated RUL with respect to the actual RUL is also a commonly used performance measure. It is defined as:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N} e_i^2} \tag{14}$$

Compared with the Score function defined by the data creator, the RMSE assigns equal weight to both early and late predictions.

3.1.1. Data preprocessing

Although there are 21 sensors available in the C-MAPSS datasets, not all of them are informative [1,2,9,22,39,40]. Some sensors provide constant or discrete values in the entire lifetime of engines, where some others do not show consistent trend with increasing engine life cycles. Table 3 lists several commonly used sensor subsets in the literature. In this study, we used the 14 sensors suggested by Zhang et al. [39] and Li et al. [22] for analysis, which provide continuous-value readings and consistent trend with the degradation of the engines.

For the selected sensor sets, we use the z-score normalization to transform the sensor readings within acceptable range before inputting them into the RNN autoencoder. In order to remove the effect of the operating conditions on the normalization, we first use the K-mean clustering method to partition the sensor readings of each dataset into the corresponding number of clusters (operating conditions). The K-mean clustering method is especially advantageous in this case as the number of operating conditions for each C-MAPSS dataset is known (as shown in Table 2). Thus, the original time series $\boldsymbol{x}'_{m,k}$ corresponding to the mth sensor and kth operating condition are transformed by:

$$\boldsymbol{x}_{m,k} = \frac{\boldsymbol{x}'_{m,k} - \mu_{m,k}}{\sigma_{m,k}} \tag{15}$$

where $\mu_{m,k}$ and $\sigma_{m,k}$ are the mean value and the standard deviation for the mth sensor readings and kth operating condition over all instances in each dataset. It should be noted that the z-score normalization holds for large populations due to the central limit theorem. Thus, if the number of instances at one operating condition is few, the z-score normalization is not applicable. Fig. 4 show the raw data and its normalized data for the readings of sensor #3 during the entire operational life of engine #1 in the training set of the C-MAPSS dataset #2, which has 6 operating conditions. It can be seen that the raw data shows large variations and no clear trend with the life cycle of the engine due to the effect of the operating conditions. However, the normalized data shows a clear rising trend within acceptable range as the effect of operating conditions is removed. This demonstrates the necessity of using the z-score normalization method on the raw data before inputting them into the prediction models.

Table 3
Sensor subsets used in the literature.

Researchers	Selected subsets of sensors
Wang et al. (2008) [2]	{2, 3, 4, 7, 11, 12, 15, 20, 21} and {2, 3, 4, 7, 11, 12, 15}
Coble and Hines (2008) [40]	{2, 3, 4, 9, 11, 14, 15, 17, 20, 21}
Ramasso et al. (2012) [41], Khelif et al. (2014) [29]	{2, 3, 4, 8, 11}
Khelif et al. (2017) [9]	{3, 4, 5} and {1, 3, 5}
Zhang et al. (2017) [39], Li et al. (2018) [22]	{2, 3, 4, 7, 8, 9, 11, 12, 13, 14, 15, 17, 20, 21}

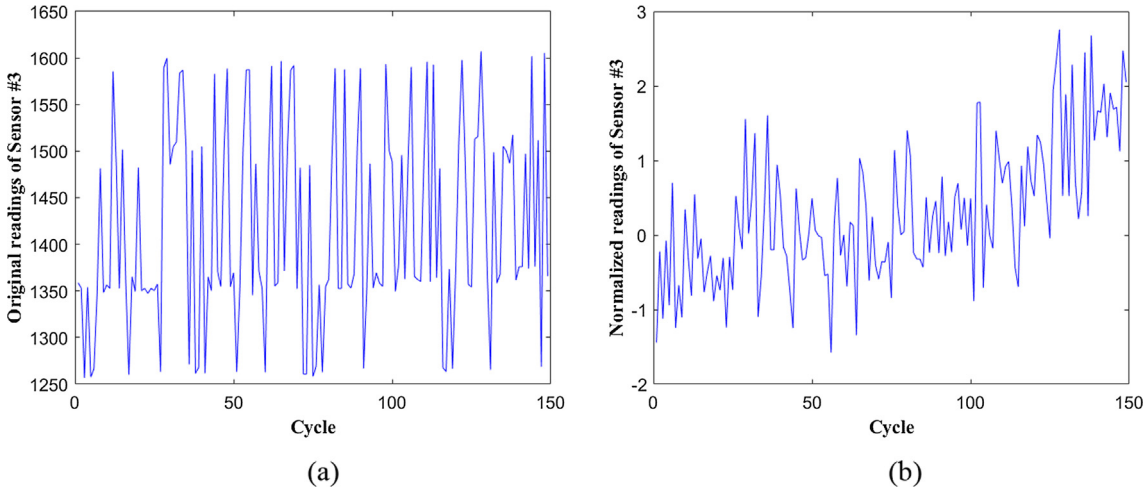


Fig. 4. Readings of the sensor #3 during the entire operational life of engine # 1 in the training set of C-MAPSS dataset #2: (a) raw data, (b) normalized data.

After normalization, a window with a fixed size W is moved along the sensor readings of each instance of the training set, as shown in Fig. 5. The resulting sub-sequences denoted by $\{\boldsymbol{\Omega}_1^{\{j\}}, \boldsymbol{\Omega}_2^{\{j\}}, \ldots, \boldsymbol{\Omega}_{Tj\text{-}W+1}^{\{j\}}\}$ for all instances available ($j = 1, 2, \ldots, J$, and J is the number of instances) are used to train the BiRNN based autoencoder. Once the BiRNN based autoencoder is trained, each point in the original sensor readings is predicted as many times as the number of the windows

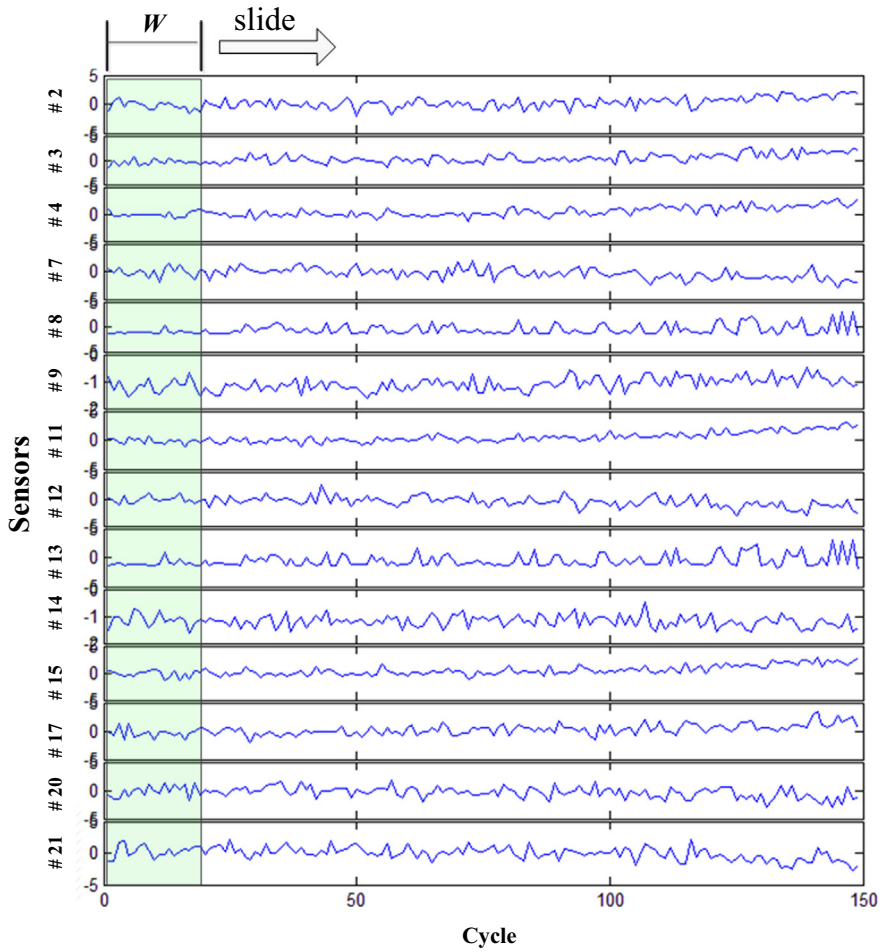


Fig. 5. Sliding window on the normalized sensor readings of an instance.

(sub-sequence) which include the point to be predicted [26]. Hence, the final prediction of a point is the average of its original predictions.

3.1.2. Parameter study

Without loss of generality, we used only C-MAPSS dataset #1 for study. The training set of dataset #1 was used to train the BiRNN based autoencoders. The RUL estimations were made on the test set. The effects of several key parameters on the prognostic performance of the proposed method are studied in this section. The default values of some parameters are given in Table 4. The number of hidden layers was selected to be one, because it was found that multiple hidden layers did not show obvious prediction improvement and the computational cost was high. The parameters of neural networks were randomly initialized, which were drawn from the standard uniform distribution in the interval between 0 and 1. No specific optimization strategy (e.g. dropout) was adopted during the training as the training instances are sufficiently abundant. In this study, we used the moving average technique to smooth the HI curves of training instances, and the window size is set as 10. Other parameter values like β and λ were set as 0.95 and 0.001 according to the suggestions in the literature [26,27,29]. For each case, the simulations were run for 10 times to cope with the stochastic process caused by the random initialization of the neural networks. The prognostic results in terms of three metrics (score, accuracy and RMSE) are expressed by boxplots [42] which can give us a holistic impression on the prognostic performance of the proposed method with different parameter sets. The parameter study was carried out on a personal computer equipped with Inter Core i7 processor, 2.6 GHz clock speed and 16 GB RAM.

(1) Architecture

Fig. 6 shows the prognostic performance of the proposed method using different RNN architectures while the other parameters are kept at the default values shown in Table 4. The notch in each box provides the 95% confidence interval of the median value (the horizontal red line) for each case. If notches of given two boxes do not overlap, it can be regarded

Table 4
Default values of some key parameters.

Parameter	Value
Architecture	GRU/LSTM/**BiGRU**/BiLSTM
Number of hidden layers L	1
Number of hidden nodes N_h	20/50/100/**200**
Window length W	**5**/10/20/30
Learning rate	0.02
Training epochs	2

Note: the bold values are the default values while conducting parameter study. For instance, the default architecture is BiGRU while varying the window length or the number of hidden nodes.

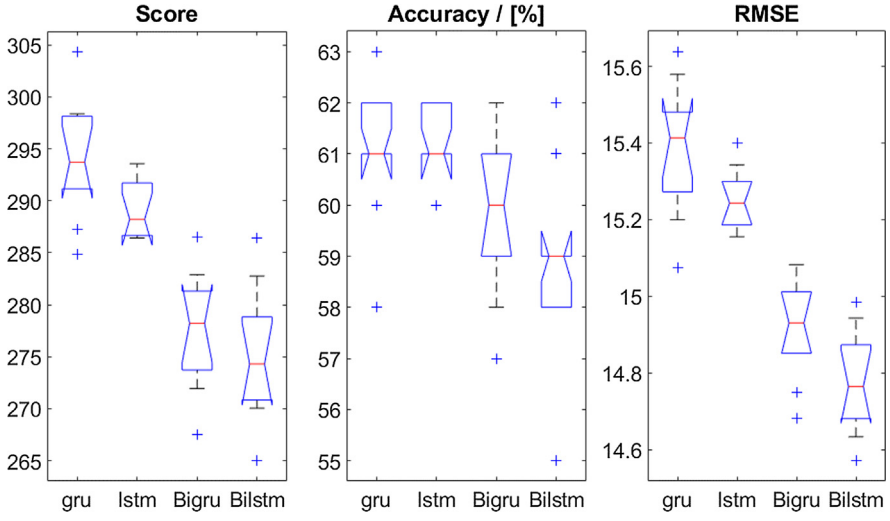


Fig. 6. Boxplots reflecting the prognostic performance of the proposed scheme on the test units of dataset #1 using different architectures (GRU, LSTM, BiGRU and BiLSTM from left to right labels respectively) in terms of three metrics (score, accuracy and RMSE from left to right subfigures respectively) (Note: the default number of hidden node is 200, and the default window length is 5.)

that their true median values are different with 95% confidence. From Fig. 6, there is strong evidence that the bidirectional RNNs show improved prognostic performance in terms of the score and RMSE compared with their unidirectional counterparts. This is due to the special structure of the bidirectional RNNs which was designed to increase the amount of input information to the neural networks. However, it seems that bidirectional RNNs do not yield improved prediction accuracy (or even poorer), which needs further investigation in the future. It can also be observed that the LSTM architecture works slightly better than the GRU architecture for the case studied, but with little confidence as notches of these two cases overlap with each other.

(2) Window length

Fig. 7 shows the prognostic performance of the proposed method using different sliding window length while the other parameters are kept in the default values shown in Table 4. The notches of most boxes are far apart from each other, especially for the score and RMSE metrics. Thus, it can be confidently concluded that a smaller window length generally yields better prognostic performance, at least for the case studied.

(3) Number of hidden nodes

Fig. 8 shows the prognostic performance of the proposed method using different numbers of hidden nodes while the other parameters are kept at the default values shown in Table 4. There is very little evidence that the median values

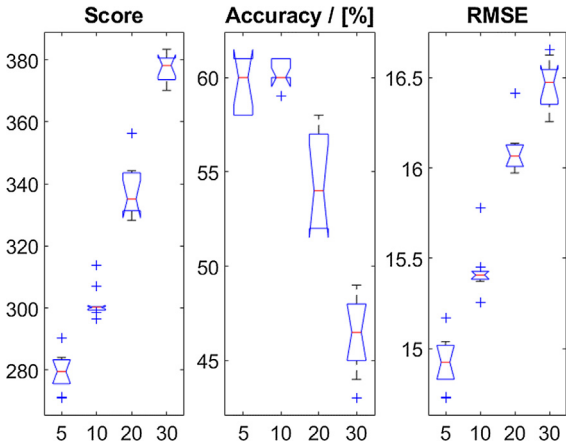


Fig. 7. Boxplots reflecting the prognostic performance of the proposed scheme on the test units of dataset #1 using different sliding window length (5, 10, 20 and 30 from left to right labels respectively) in terms of three metrics (score, accuracy and RMSE from left to right subfigures respectively) (Note: the default architecture is BiGRU, and the default number of hidden nodes is 200).

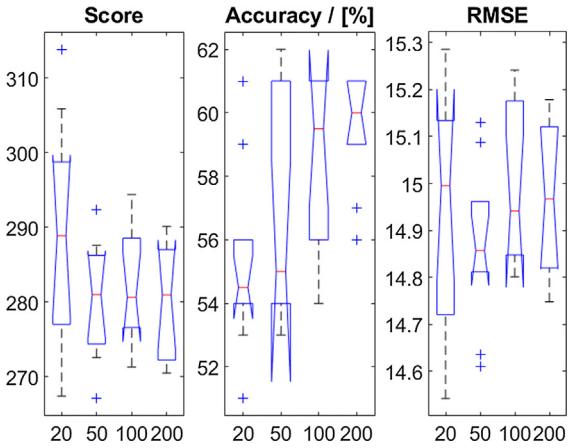


Fig. 8. Boxplots reflecting the prognostic performance of the proposed scheme on the test units of dataset #1 using different numbers of hidden nodes (20, 50, 100 and 200 from left to right labels respectively) in terms of three metrics (score, accuracy and RMSE from left to right subfigures respectively) (Note: the default architecture is BiGRU, and the default window length is 5).

obtained by using different numbers of hidden nodes are different from each other. This indicates that the number of hidden nodes has comparatively little effect on the final prognostic performance. Increasing the number of hidden nodes does not significantly improve the prognostic performance, compared with strategies like reducing the sliding window length and using the bidirectional RNN architectures.

3.1.3. Prognostic results on the C-MAPSS datasets

In this section, we used the bidirectional LSTM based autoencoder (BiLSTM-ED) to construct the offline HI libraries for each C-MAPSS dataset using the corresponding training sets. The RUL estimations were made on the test sets. The window length and the number of hidden nodes are 5 and 200 respectively. Fig. 9(a), (b), (c) and (d) show the actual and estimated RUL estimations on the test engines for dataset #1, dataset #2, dataset #3 and dataset #4 respectively. The challenge datasets are not included as the actual RUL values of the test engines are not exposed to public. It can be found that the estimated RULs are generally consistent with the actual RULs for each dataset. From dataset #1 to dataset #4, the prediction accuracy becomes poorer due to the increasing level of complexity. It can also be found that as the actual RUL increases, the prediction error in the estimated RUL generally increases, which is expected as it is more difficult to estimate the RUL of a machine in the early stage than in the late stage, especially when the machine is in the fresh healthy state. The performance of the proposed BiLSTM-ED scheme for the RUL estimation of turbofan engines is compared with several recent prognostic algorithms in terms of three metrics, i.e. score (S), accuracy (A) and RMSE, as shown in Table 5. It should be noted that data set #1 was the most used one, followed by the data set #5, where the rest data sets are relatively under-utilized, especially data set #6, which is due to the difficult access to the true RUL values [28]. Different researchers used slightly different metric values to assess the performance of their algorithms. Table 5 listed the available values in terms of the three metrics reported by them.

The score of the proposed scheme on challenge dataset #5 was evaluated as 1098, which is lower than most score values reported by previous algorithms. To our best knowledge, the lowest score reported in the published work after 2008 is 752,

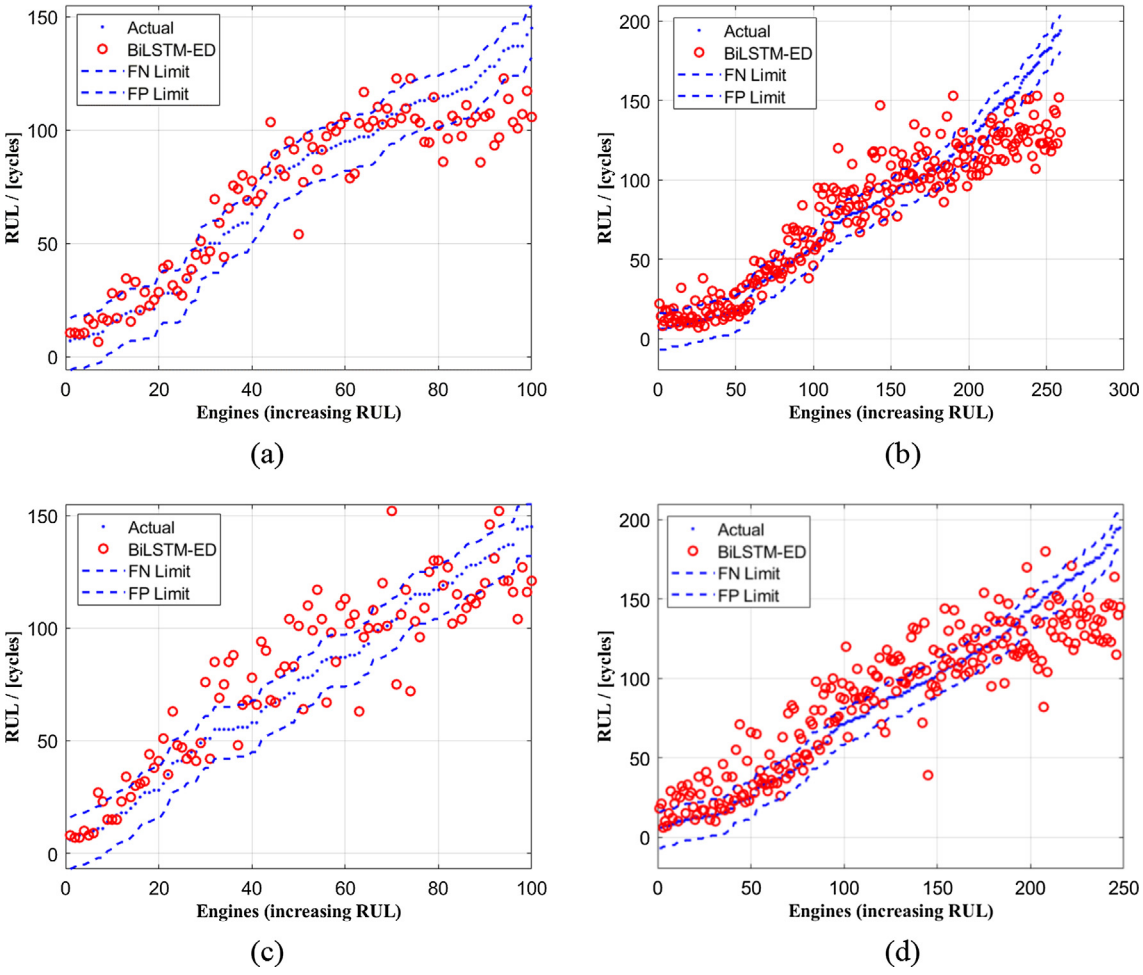


Fig. 9. RUL estimations using the proposed BiLSTM based autoencoder scheme on the test set of the C-MAPSS datasets: (a) dataset #1, (b) dataset #2, (c) dataset #3 and (c) dataset #4.

Table 5
Performance comparisons among several state-of-art algorithms reported in the literature.

	Algorithms	Metrics	#1	#2	#3	#4	#5	#6
Similarity-based curve matching	BiLSTM-ED	S	273	3099	574	3202	1098	
		A (%)	57	49	42	40		
		RMSE	14.74	22.07	17.48	23.49		
	RULCLIPPER (Ramasso 2014) [1]	S	216	2796	317	3132	752	11,672
		A (%)	67	46	59	45		
		RMSE	13.27	22.89	16.00	24.33		
	GRU-ED (Gugulothu et al. 2017) [27]	S	219					
		A (%)	59					
		RMSE	12.45					
	LSTM-Recon (Malhotra et al. 2016) [26]	S	256					
		A (%)	67					
		RMSE	12.81					
Direct mapping	SV regression (Khelif et al. 2017) [9]	S	449					
		A (%)	70					
		RMSE						
	ADNN (Zhao et al. 2017) [23]	S	236				793	
		A (%)						
		RMSE						
	Deep LSTM (Zheng et al. 2017) [43]	S	338	4450	852	5550	1862	
		A (%)						
		RMSE	16.14	24.49	16.18	28.17		
	MODBNE (Zhang et al. 2017) [39]	S	334	5585	421	6557		
		A (%)						
		RMSE	15.04	25.05	12.51	28.66		
Statistical models	Wiener process (Son et al. 2013) [20]	S					5520	
		A (%)						
		RMSE					819	
	Gamma process (Son et al. 2012) [42]	S					4170	
		A (%)						
		RMSE					434	

Notes: It is often that a prediction algorithm with a parameter set yielding the best value for one metric does not perform best in the other two metrics. In this table, values of all metrics from different algorithms are presented when the best score was achieved for each model.

from the algorithm RULCLIPPER proposed by Ramasso [1]. It should be noted that the prediction performance of RULCLIPPER was tested by extensive trial and error (4 empirical rules and 511 combinations of sensor subset for each dataset), from which the best model with an optimal set of hyperparameters that yields the lowest score on each dataset was found. The two recent algorithms: LSTM-Recon by Malhotra et al. [26] and GRU-ED by Gugulothu et al. [27] were also tested in a similar way by grid-searching the optimal values of more than 7 hyperparameters, from which the best model with the lowest score on dataset #1 was found. Compared with these algorithms, the performance of our model was first assessed through a parameter study only on the dataset #1, to find suitable choice of three key parameters (architecture, window length and number of hidden nodes), and then directly applied on the other datasets without further tuning for each dataset. From Table 5, it can be noted that the proposed scheme shows obvious competitiveness compared with most state-of-art algorithms reported in the literature for the RUL estimations of C-MAPSS turbofan datasets. More importantly, the proposed scheme shows good generalization capability for the other datasets when tuned only on dataset #1.

It is worth noticing that, for datasets #1 to #4 where the true RUL values are known, the prognostic performance of our model can be further improved by fine-turning some other hyper parameters (e.g. sensor subsets in Table 3, learning rate, number of hidden layers, β and λ in Eqs. (8) and (11), etc.) using the similar grid search method as did by Malhotra et al. [26] and Ramasso [1]. However, purely grid searching the optimal hyper parameters over high-dimensional configuration space is quite time-consuming and unnecessary. We are planning to adopt the random search strategy as suggested by Bergstra and Bengio [44] for hyper-parameter optimization. The work is undergoing, and the results will be presented in a separate paper.

3.2. Milling datasets

The publicly available milling datasets provided by UC Berkeley represent experiments conducted on a milling machine under various operating conditions [45]. The cutting speed was set constant at 200 m/min (826 rev/min). Two different depths of cut (1.5 mm and 0.75 mm), two feed rates (0.5 mm/rev and 0.25 mm/rev), and two different workpiece material types (cast iron and steel) were investigated. The experimental matrix is $2 \times 2 \times 2$. Hence there are eight different cutting conditions, as shown in Table 6. The experiment under each condition was done a second time with a second set of inserts. Therefore, there are 16 cases with a variable number of runs (cuts) per tool life representing the run-to-fail instances of cutting tools. The number of runs was dependent on the degree of flank wear *VB* that was measured between runs at irregular

Table 6
Milling data set provided by UC Berkeley [45].

Material	Cutting condition	Depth of cut (D, mm)	Feed (F, mm/rev)	Number of runs	
				1st insert	2nd insert
Cast iron	#1	1.5	0.5	17	9
	#2	1.5	0.25	7	10
	#3	0.75	0.5	14	14
	#4	0.75	0.25	14	23
Steel	#5	1.5	0.5	6	6
	#6	1.5	0.25	1	7
	#7	0.75	0.5	6	9
	#8	0.75	0.25	8	15

intervals up to a wear limit. There is a total of 167 runs across cases with 109 runs and 58 runs for cast iron and steel materials, respectively. However, there is only one run for the first insert at cutting condition #6. Hence, this case will not be considered in the following analysis. The dataset contains readings from 6 different sensors: AC spindle motor current (smc AC), DC spindle motor current (smc DC), table vibration (table VB), spindle vibration (spindle VB), table acoustic emission (table AE) and spindle acoustic emission (spindle AE). Fig. 10 shows the raw readings of the six sensors for cutting condition #1 using the 1st insert which has 17 runs.

The average life cycle using steel material is much shorter than that when using cast iron material. Thus, it is necessary to separate these instances according to the material type and develop two separate material specific prediction models. This strategy has proved to significantly improve prediction performance [46]. For cast iron, there are eight instances whereas there are only seven for steel. Since the number of instances is small, we use the leave-one-out cross validation method for model learning and validation [26]. That is, each time we select one instance from the eight instances (seven instances) of cast iron (steel) material, and the remaining seven instances (six instances) are used to develop the prediction model.

The length of each sensor reading in a run is 9000. However, it was observed that there are obvious unstable regions in the initial and late stages due to the cut in and cut out processes, as shown in Fig. 10. Thus, only the readings in the middle stable region 3000–6000 (3000 values) were used to train the BiLSTM-ED model, which was supposed to be directly related with the cutting condition. The window length W is 100 after down-sampling the original sensor readings by 30. The first run of each instance is supposed to be normal. The target HI value for a run is determined by the deviation of its embeddings from the normal embeddings, which are obtained from the trained BiLSTM-ED model. Compared with the simulation data of turbofan engines, one of the biggest challenges of the milling machine experimental data is that the degradation data is not strictly time-continuous. For convenience, we used the similar linear interpolation strategy [26] to reduce the gap between two consecutive runs of an instance into one cycle. The tool wear and target HI values of an instance are also interpolated in

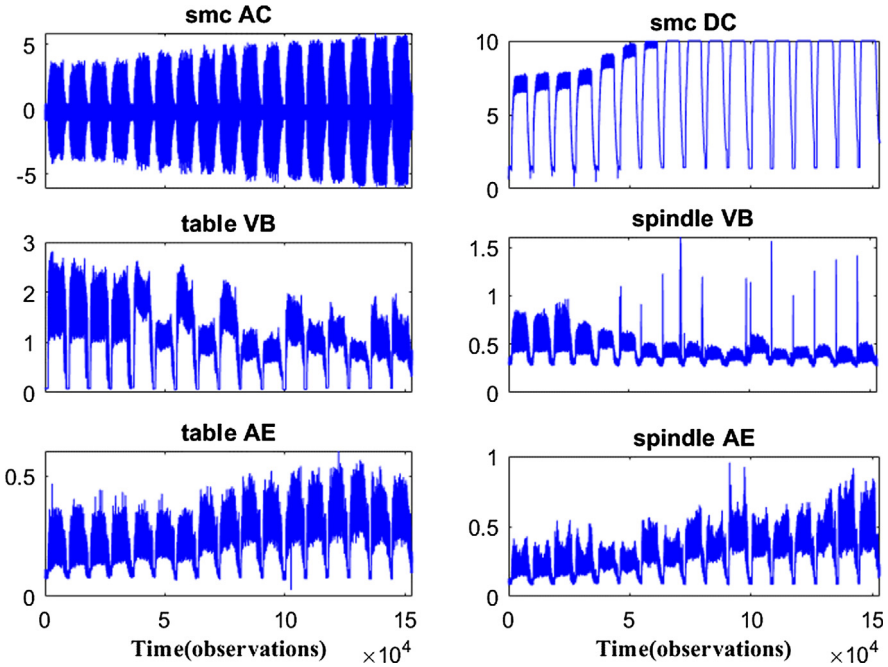


Fig. 10. Sensor readings of the cutting condition #1 with 1st insert (17 runs).

the same manner, and the tool was assumed to be failed when the measured tool wear reached 0.45 mm for the first time [26,45]. When learning the LR model, initially we considered the mean values and standard deviations of the six sensors as the derived sensors (i.e. 12 features) to represent the sensory readings for each run as suggested by Malhotra et al. [26] and Coble [46]. However, we found that considering all 12 derived sensors is not necessary, and it can yield poor results as some features may deteriorate the prediction performance. For instance, the smc DC signal became saturated in most runs, and the vibration signals are not correlated well with the degradation of the cutting tool, as shown in Fig. 10. Thus, we tried different feature sets for learning LR model, and found that for the cast iron material, the optimal feature set includes standard deviation and mean value of smc AC, whereas for the steel material, the optimal feature set includes the standard deviations of the smc AC signal, table AE and spindle AE.

Fig. 11(a) and (b) show the actual and estimated RULs at the specified runs of experiments for the cast iron and steel materials. The number of hidden nodes is 100. Other tunable parameters like β, τ and moving average window were set as the same values as for the turbofan engine datasets. The red circles (BiLSTM-ED$_1$) represent estimation using the optimal feature set whereas the green squares (BiLSTM-ED$_2$) represent estimation using the full 12 features as suggested in the literature. It can be found that the proposed scheme using the optimal feature set yields much better RUL predictions than that using the full features.

Coble [46] defined a metric to determine the prognostic performance of algorithms on the milling dataset, which is the mean absolute percent error (MAPE). It is defined as:

$$MAPE = \frac{100}{N}\sum_{i=1}^{N}\frac{e_i}{\text{actual RUL}_i} \tag{16}$$

Table 7 shows the performance comparisons of the proposed algorithm using the optimal feature set and full features with respect to other two state-of-art algorithms on the milling dataset reported in the literature in terms of the RMSE and MAPE. It can be noted that the proposed scheme using the optimum feature set has good performance on RUL estimation

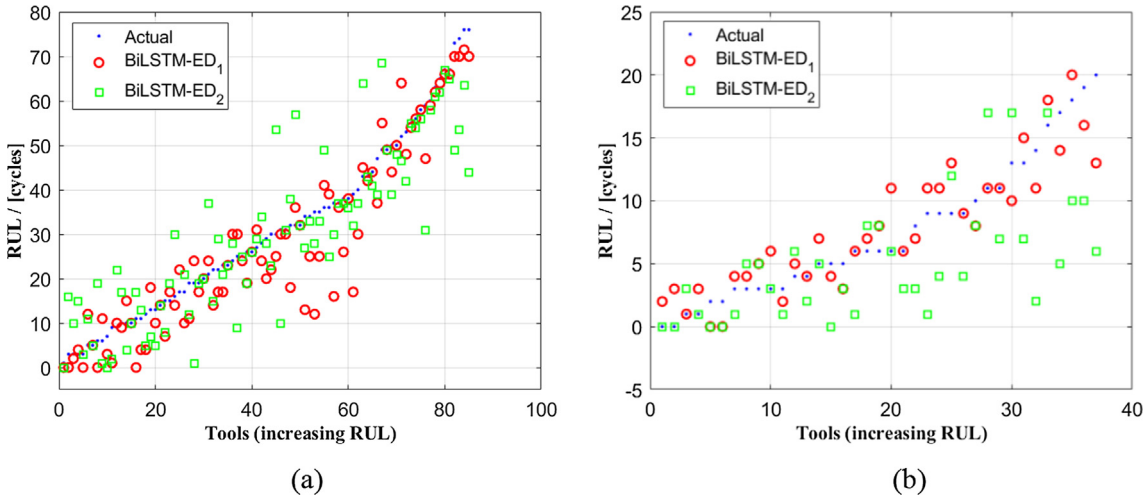


Fig. 11. RUL estimations using the proposed BiLSTM based autoencoder scheme on the milling datasets with material types of: (a) cast iron, (b) steel (note that BiLSTM-ED$_1$ and BiLSTM-ED$_2$ uses the optimal feature set and full features respectively).

Table 7
Performance comparisons among different algorithms on the milling datasets.

Prediction model	Metrics	Cast iron	Steel
BiLSTM-ED$_1$	RMSE	7.14	2.36
	MAPE	24%	38%
BiLSTM-ED$_2$	RMSE	11.27	5.18
	MAPE	40%	41%
LSTM-Recon (Malhotra et al. 2016) [26]	RMSE	8.45	2.66
	MAPE	26%	31.7%
GPM (Coble 2010) [46]	RMSE		
	MAPE	12.3%	

Note: The value provided by Coble [46] is the average MAPE value of the predictions on the two material types. No RMSE value was reported by Coble [46] on the milling dataset.

of the cutting tools of the milling machine compared with other algorithms that are available in the literature. However, due to the limited number of run-to-failure instances, there are comparatively large deviations of the RUL estimations with respect to true values. The prediction performance can be significantly improved as more and more run-to-failure instances are available.

4. Conclusions

In this study, we proposed a sensor-based data-driven scheme for the RUL estimation of systems. It first utilizes the bidirectional recurrent neural network based autoencoder to map the original run-to-failure multi-sensor readings into one-dimensional HI values, which represent the degradation patterns of the units of the system. Compared with the unidirectional RNN based autoencoders, the bidirectional ones can learn more robust embeddings by increasing the amount of input information to the neural networks. The similarity-based HI curve matching technique was adopted to match the test HI curve with each HI curve in the library built in the offline phase, from which the RUL of the on-line unit was estimated in the early stage based on the matched offline instances with similar degradation patterns.

The proposed scheme was tested on two publicly available datasets: the turbofan engine datasets and the milling datasets. Through parameter studies on one of the turbofan engine datasets, it was demonstrated that the bidirectional RNN based autoencoders yield improved RUL estimations compared with their unidirectional counterparts. Suitable choice of several key tunable parameters was also found through parameter studies. The proposed scheme was then applied on other turbofan datasets and the milling datasets, which shows obvious competitiveness compared with most of state-of-art algorithms reported in the literature. Moreover, the proposed scheme shows good generalization capability on unseen datasets. Its application to the RUL estimation of mechanical systems is promising. However, like most sensor-based data-driven algorithms, the prediction accuracy of the proposed scheme is dependent on the availability of run-to-failure instances.

Acknowledgement

This project was financially supported by the Natural Sciences and Engineering Research Council of Canada (Grant number: RGPIN/05922-2014).

References

[1] E. Ramasso, Investigating computational geometry for failure prognostics, Int. J. Progn. Heal. Manag. 5 (2014) 1–18. https://hal.archives-ouvertes.fr/hal-01303405.
[2] T. Wang, J. Yu, D. Siegel, J. Lee, A Similarity-Based Prognostics Approach for remaining useful life estimation of Engineered Systems, Progn. Heal. Manag. 2008. PHM 2008. Int. Conf. (2008) 4–9.
[3] J. Coble, J. Hines, Applying the general path model to estimation of remaining useful life, Int. J. Progn. Heal. Manag. 2 (2011) 1–13. https://www.phmsociety.org/node/154.
[4] E. Usui, T. Shirakashi, T. Kitagawa, Analytical prediction of cutting tool wear, Wear 100 (1984) 129–151, https://doi.org/10.1016/0043-1648(84)90010-3.
[5] Y. Li, S. Billington, C. Zhang, T. Kurfess, S. Danyluk, S. Liang, Adaptive prognostics for rolling element bearing condition, Mech. Syst. Signal Process. 13 (1999) 103–113.
[6] Z. Pálmai, Proposal for a new theoretical model of the cutting tool's flank wear, Wear 303 (2013) 437–445, https://doi.org/10.1016/j.wear.2013.03.025.
[7] L. Peel, Data driven prognostics using a Kalman filter ensemble of neural network models, Int. Conf. Progn. Heal. Manag. (2008) 1–6, https://doi.org/10.1109/PHM.2008.4711423.
[8] F.O. Heimes, Recurrent neural networks for remaining useful life estimation, Progn. Heal. Manag. 2008. PHM 2008. Int. Conf. (2008) 1–6. doi:10.1109/PHM.2008.4711422.
[9] R. Khelif, B. Chebel-Morello, S. Malinowski, E. Laajili, F. Fnaiech, N. Zerhouni, Direct remaining useful life estimation based on support vector regression, IEEE Trans. Ind. Electron. 64 (2017) 2276–2285, https://doi.org/10.1109/TIE.2016.2623260.
[10] D. An, J. Choi, N. Ho, Prognostics 101: a tutorial for particle filter-based prognostics algorithm using Matlab, Reliab. Eng. Syst. Saf. 115 (2013) 161–169, https://doi.org/10.1016/j.ress.2013.02.019.
[11] L. Liao, F. Köttig, Review of hybrid prognostics approaches for remaining useful life prediction of engineered systems, and an application to battery life prediction, IEEE Trans. Reliab. 63 (2014) 191–207, https://doi.org/10.1109/TR.2014.2299152.
[12] H. Hanachi, W. Yu, I.Y. Kim, J. Liu, C.K. Mechefske, Hybrid data-driven physics-based model fusion framework for tool wear prediction, Int. J. Adv. Manuf. Technol. (2018), https://doi.org/10.1007/s00170-018-3157-5.
[13] W. Yu, Y. Shao, C.K. Mechefske, The effects of spur gear tooth spatial crack propagation on gear mesh stiffness, Eng. Fail. Anal. 54 (2015) 103–119, https://doi.org/10.1016/j.engfailanal.2015.04.013.
[14] W. Yu, C.K. Mechefske, M. Timusk, A new dynamic model of a cylindrical gear pair with localized spalling defects, Nonlinear Dyn. 91 (2017) 2077–2095, https://doi.org/10.1007/s11071-017-4003-2.
[15] R. Zhao, R. Yan, J. Wang, K. Mao, Learning to monitor machine health with convolutional Bi-directional LSTM networks, Sensors (Switzerland). 17 (2017) 1–18, https://doi.org/10.3390/s17020273.
[16] X. Li, J. Qian, G.G. Wang, Fault prognostic based on hybrid method of state judgment and regression, Adv. Mech. Eng. 149562 (2013) 10, https://doi.org/10.1155/2013/149562.
[17] H. Hanachi, J. Liu, A. Banerjee, Y. Chen, Sequential state estimation of nonlinear/non-Gaussian systems with stochastic input for turbine degradation estimation, Mech. Syst. Signal Process. 72–73 (2016) 32–45, https://doi.org/10.1016/j.ymssp.2015.10.022.
[18] J. Sun, H. Zuo, W. Wang, M.G. Pecht, Application of a state space modeling technique to system prognostics based on a health index for condition-based maintenance, Mech. Syst. Signal Process. 28 (2012) 585–596, https://doi.org/10.1016/j.ymssp.2011.09.029.
[19] R. Zhao, R. Yan, Z. Chen, K. Mao, P. Wang, R.X. Gao, Deep learning and its applications to machine health monitoring, Mech. Syst. SignalProcess. 115 (2019) 213–237, https://doi.org/10.1016/j.ymssp.2018.05.050.
[20] K. Le Son, M. Fouladirad, A. Barros, E. Levrat, Remaining useful life estimation based on stochastic deterioration models: a comparative study, Reliab. Eng. Syst. Saf. 112 (2013) 165–175, https://doi.org/10.1016/j.ress.2012.11.022.

[21] X.S. Si, W. Wang, C.H. Hu, D.H. Zhou, Remaining useful life estimation - A review on the statistical data driven approaches, Eur. J. Oper. Res. 213 (2011) 1–14, https://doi.org/10.1016/j.ejor.2010.11.018.
[22] X. Li, Q. Ding, J.Q. Sun, Remaining useful life estimation in prognostics using deep convolution neural networks, Reliab. Eng. Syst. Saf. 172 (2018) 1–11, https://doi.org/10.1016/j.ress.2017.11.021.
[23] Z. Zhao, B. Liang, X. Wang, W. Lu, Remaining useful life prediction of aircraft engine based on degradation pattern learning, Reliab. Eng. Syst. Saf. 164 (2017) 74–83, https://doi.org/10.1016/j.ress.2017.02.007.
[24] G. Sateesh Babu, P. Zhao, X. Li, Deep convolutional neural network based regression approach for estimation of remaining useful life, Lect. Notes Comput. Sci. (Including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics) 9642 (2016) 214–228, https://doi.org/10.1007/978-3-319-32025-0_14.
[25] J. Yan, M. Koç, J. Lee, A prognostic algorithm for machine performance assessment and its application, Prod. Plan. Control. 15 (2004) 796–801, https://doi.org/10.1080/09537280412331309208.
[26] P. Malhotra, T.V. Vishnu, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, G. Shroff, Multi-sensor prognostics using an unsupervised health index based on LSTM encoder-decoder, 1st ACM SIGKDD Work. Mach. Learn. Progn. Heal. Manag. San Fransisco, CA, USA, 2016, http://arxiv.org/abs/1608.06154.
[27] N. Gugulothu, T.V. Vishnu, P. Malhotra, L. Vig, P. Agarwal, G. Shroff, Predicting remaining useful life using time series embeddings based on recurrent neural networks, 2nd ML PHM Work. SIGKDD 2017, Halifax, Canada, 2017, https://arxiv.org/abs/1709.01073.
[28] E. Ramasso, A. Saxena, Review and analysis of algorithmic approaches developed for prognostics on CMAPSS dataset, in: Annu. Conf. Progn. Heal. Manag. Soc. 2014., Fort Worth, TX, USA., n.d.
[29] R. Khelif, S. Malinowski, B. Chebel-Morello, N. Zerhouni, RUL prediction based on a new similarity-instance based approach, IEEE Int. Symp. Ind. Electron. (2014) 2463–2468, https://doi.org/10.1109/ISIE.2014.6865006.
[30] G.E. Hinton, R.R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science (80-) 313 (2006) 504–508, https://doi.org/10.1126/science.1127647.
[31] S. Hochreiter, J. Urgen Schmidhuber, Long short-term memory, Neural Comput. 9 (1997) 1735–1780, https://doi.org/10.1162/neco.1997.9.8.1735.
[32] K. Cho, B. van Merrienboer, C. Gulcehre, Machine Translation (2014), ArXiv:1406.1078, http://arxiv.org/abs/1406.1078.
[33] P. Malhotra, T.V. Vishnu, L. Vig, P. Agarwal, G. Shroff, TimeNet: Pre-trained deep recurrent neural network for time series classification, 25th Eur Symp. Artif. Neural Networks, Comput. Intell. Mach. Learn., 2017.
[34] M. Schuster, K.K. Paliwal, Bidirectional recurrent neural networks, IEEE Trans. Signal Process. 45 (1997) 2673–2681, https://doi.org/10.1109/78.650093.
[35] P. Wang, B.D. Youn, C. Hu, A generic probabilistic framework for structural health prognostics and uncertainty management, Mech. Syst. Signal Process. 28 (2012) 622–637, https://doi.org/10.1016/j.ymssp.2011.10.019.
[36] A. Saxena, D. Simon, Turbofan Engine Degradation Simulation Data Set, NASA Ames Progn. Data Repos. (2008). http://ti.arc.nasa.gov/project/prognostic-data-repository.
[37] A. Saxena, K. Goebel, PHM08 Challenge Data Set, NASA Ames Progn. Data Repos. (2008). http://ti.arc.nasa.gov/project/prognostic-data-repository.
[38] A. Saxena, K. Goebel, D. Simon, N. Eklund, Damage propagation modeling for aircraft engine run-to-failure simulation, 2008 Int. Conf. Progn. Heal. Manag. PHM 2008. (2008). doi:10.1109/PHM.2008.4711414.
[39] C. Zhang, P. Lim, A.K. Qin, K.C. Tan, Multiobjective deep belief networks ensemble for remaining useful life estimation in prognostics, IEEE Trans. Neural Networks Learn. Syst. 28 (2017) 2306–2318, https://doi.org/10.1109/TNNLS.2016.2582798.
[40] J.B. Coble, J.W. Hines, Prognostic algorithm categorization with PHM challenge application, 2008 Int. Conf. Progn. Heal. Manag. PHM 2008. (2008). doi:10.1109/PHM.2008.4711456.
[41] E. Ramasso, M. Rombaut, N. Zerhouni, Joint prediction of observations and states in time-series : a partially supervised prognostics approach based on belief functions and KNN, IEEE Trans. Syst. Man, Cybern. Part B Cybern. 43 (2013) 37–50.
[42] K. Le Son, M. Fouladirad, A. Barros, Remaining useful life estimation on the non-homogenous gamma with noise deterioration based on Gibbs filtering: a case study, 2012 IEEE Conf. Progn. Heal. Manag. (2012) 1–6, https://doi.org/10.1109/ICPHM.2012.6299520.
[43] S. Zheng, K. Ristovski, A. Farahat, C. Gupta, Long short-term memory network for remaining useful life estimation, in: 2017 IEEE Int. Conf. Progn. Heal. Manag., 2017, pp. 88–95.
[44] James Bergstra, Y. Bengio, Random search for hyper-parameter optimization, J. Mach. Learn. Res. 13 (2012) 281–305, https://doi.org/10.1162/153244303322533223.
[45] A. Agogino, K. Goebel, Milling Data Set, BEST Lab, UC Berkeley, NASA Ames Progn. Data Repos. (2007). https://ti.arc.nasa.gov/tech/dash/groups/pcoe/prognostic-data-repository/.
[46] J.B. Coble, Merging data sources to predict remaining useful life-an automated method to identify prognostic parameters, (2010) 1–223. http://trace.tennessee.edu/utk_graddiss/683.