# A Noise-Boosted Remaining Useful Life Prediction Method for Rotating Machines Under Different Conditions

Lei Xiao, Fabing Duan, Junxuan Tang, and Derek Abbott, *Fellow, IEEE*

***Abstract*— Remaining useful life (RUL) prediction methods for rotating machines have been successfully developed in recent decades. More attention should be paid to predictions with inconsistent data distributions under different conditions. To solve this problem, this article proposes a new RUL prediction method that includes two phases. In the first phase, degradation features are extracted from both the training and testing data sets using probabilistic principal component analysis (PPCA). In the second phase, additive white Gaussian noise (AWGN) is intentionally injected into the degradation features; thereafter, the features that are mixed with manually injected noise are imported into a bidirectional long short-term memory (BLSTM) network. The AWGN can enhance the robustness of the RUL prediction method and achieve prediction for machines under different conditions. In contrast to most deep learning-based RUL prediction methods, the training samples are intentionally "polluted" by manually injected noise. The effectiveness of the proposed method is validated using the C-MAPSS lifetime data set for aeroengines and compared with the effectiveness of state-of-the-art approaches.**

***Index Terms*— Additive white Gaussian noise (AWGN), bidirectional long short-term memory (BLSTM) recurrent neural network, noise-boosted prediction, rotating machines, remaining useful life (RUL).**

## I. Introduction

IN THE era of Industry 4.0, a modern and critical rotating machine is always required to be precise and high speed with a high level of reliability considering safety and costs [1]. By analyzing condition monitoring data, the health status and remaining useful life (RUL) of a machine can be estimated. The existing RUL prediction models for rotating machines can be roughly divided into three categories: physics-based methods [2]–[4], data-driven methods [5]–[10], and hybrid methods [11], [12]. Regardless of which category of methods is used, most methods require the samples for training or estimating parameters to be in the same condition or subject to consistent data distribution. However, a machine usually experiences various operating conditions (OCs) and fault conditions (FCs); therefore, the collected condition monitoring data are always subject to different data distributions. Transfer learning (TL) is an effective method to solve such an RUL prediction problem.

TL has been successfully applied to fault diagnosis [13]–[16]. However, its application to RUL prediction is relatively limited. Mao *et al.* [7] developed a TL-based RUL prediction method for rolling bearings. Fan *et al.* [17] predicted the RULs for turbofan engines using TL and consensus self-organizing models. Zhang *et al.* [18] proposed a TL algorithm based on bidirectional long short-term memory (BLSTM) networks for the RUL estimation of aeroengines. Shen *et al.* [19] combined transfer compact coding for hyperplane classifiers with an exponential semideterministic extended Kalman filter to transfer the RUL prediction models among bearings under multiple working conditions. Sun *et al.* [20] predicted the RUL of tools in manufacturing based on deep TL. Da Costa *et al.* [21] proposed a method based on deep domain adaption to predict RULs for machines under different conditions. In addition to TL, improving the robustness of the RUL prediction method is also an effective method.

The condition monitoring data from different conditions have different distribution characteristics. If the data distribution characteristics from the training data sets and testing data sets are the same or close, more accurate RUL prediction results can be obtained. In other words, the data distribution bias may generate negative impacts and decrease the RUL prediction accuracy. However, most current machine learning algorithms are constructed based on an independent and identical distribution (i.i.d.), which demands that the data distribution for training and testing should be kept the same [7].

The collected condition monitoring data always contain measurement errors that are generated due to vibrations, current fluctuations, or sensor delays. These errors are always regarded as disturbance and background noise. Smoothing is usually adopted to remove such noise in a machine learning-based RUL prediction method [22].

Manuscript received November 15, 2020; revised February 22, 2021; accepted February 25, 2021. Date of publication March 9, 2021; date of current version March 22, 2021. This work was supported in part by the National Natural Science Foundation of China under Grant 52075094 and Grant 51705321; in part by the Fundamental Research Funds for the Central Universities under Grant 2232019D3-29; in part by the China Post-Doctoral Science Foundation under Grant 2017M611576; and in part by the Initial Research Funds for Young Teachers of Donghua University. The Associate Editor coordinating the review process was Loredana Cristaldi. *(Corresponding author: Lei Xiao.)*

Lei Xiao and Junxuan Tang are with the College of Mechanical Engineering, Donghua University, Shanghai 201620, China (e-mail: leixiao211@dhu.edu.cn).

Fabing Duan is with the Institute of Complexity Science, Qingdao University, Qingdao 266071, China.

Derek Abbott is with the Centre for Biomedical Engineering, The University of Adelaide, Adelaide, SA 5005, Australia, and also with the School of Electrical and Electronic Engineering, The University of Adelaide, Adelaide, SA 5005, Australia.

Digital Object Identifier 10.1109/TIM.2021.3064810

In fact, noise is not always undesirable for fault diagnostics. If properly used, noise can be beneficial to extract information on fault characteristics. Some noise-boosted methods for weak fault-signal detection have been developed based on the stochastic resonance (SR) or vibrational resonance (VR) mechanism [23]–[27]. In the SR and VR mechanisms, stochastic noise and high-frequency interference are manually generated and injected into a nonlinear system.

An artificial neural network (ANN) can be regarded as a nonlinear system. Many experiments have demonstrated that injecting uncorrelated noise can enhance the information transmission among neurons [28], [29]. For example, intentionally added noise can boost the performance of CNNs [30] and adversarial learning in bidirectional backpropagation [31]. However, these experiments can be categorized as classification experiments. ANNs usually utilize regressions when predicting RULs for machines. However, there are few precedents for RUL prediction boosted by the injection of noise into an ANN. For example, the RUL prediction methods [32]–[34] that are based on deep learning (DL) do not have manually generated noise intentionally injected. This article is a primary exploration of RUL prediction boosted by noise injection.

Based on the above analysis, this article proposes a new RUL prediction method that includes two phases: degradation feature representation and BLSTM network-based RUL prediction. In the first phase, all data sets, including training samples and testing samples, are jointly analyzed to extract the common characteristics based on probabilistic principal component analysis (PPCA), which is a nonlinear unsupervised feature extraction method. The extracted principal components (PCs) can represent the main degradation characteristics and avoid redundant information.

In the second phase, the BLSTM network is adopted to employ its good ability to handle time-series data. The input of the BLSTM is the extracted degradation features with manually generated noise. Due to the existence of manually generated noise, the degradation features have more noise and bias from their real characteristics. The adoption of manually generated noise aims at improving the robustness of the proposed RUL prediction method even though the training and testing data sets are from different conditions; therefore, this method is referred to as the noise-boosted RUL prediction method.

The main contributions of this article are summarized as follows. A new RUL prediction method that improves the robustness by introducing manually generated additive white Gaussian noise (AWGN) into a BLSTM network is proposed. This is the first primary exploration of noise-injection DL for RUL prediction to solve the prognostics of machines under different conditions.

The remainder of this article is organized as follows. Section II describes the proposed noise-boosted RUL prediction method. In Section III, the proposed method is validated using the C-MAPSS of the aeroengine lifetime data set. Some state-of-the-art approaches are compared, and the results are discussed in Section IV. The entire paper is concluded in Section V.

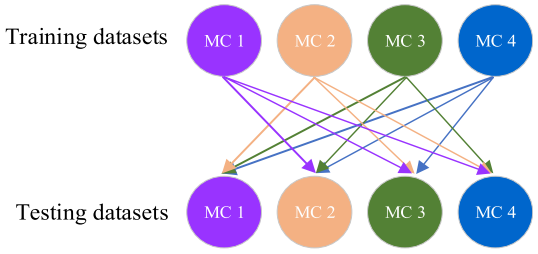


Fig. 1. Illustration of the focal problem of RUL prediction for machines under different conditions.

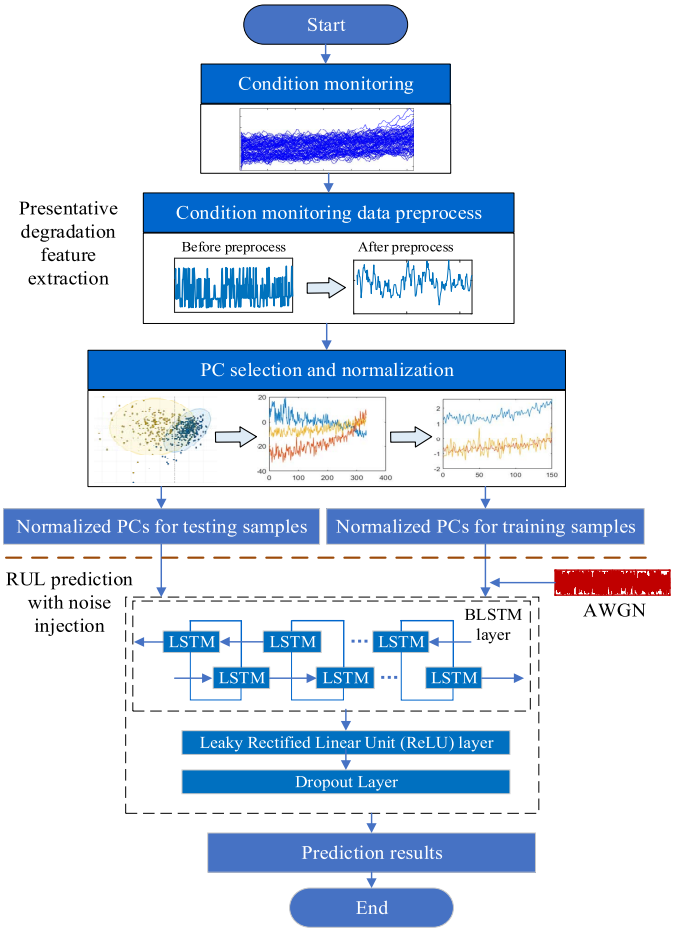


Fig. 2. Framework of the proposed noise-boosted RUL prediction method.

## II. Noise-Boosted RUL Prediction Method

The focal problem in this article is the RUL prediction for machines under different conditions, which can be illustrated in Fig. 1. Here, MC means machine condition. From Fig. 1, separate machines from the training and testing data sets are under different MCs. For example, the condition monitoring data from the machines under MC 1 are used to predict the RULs of machines under MC 2–4. The framework of the proposed noise-boosted RUL prediction method is given in Fig. 2.

### A. Representative Feature Extraction

The representative degradation features are usually extracted from condition monitoring data. However, the originally

collected condition monitoring data may contain measurement errors and redundant information. Therefore, preprocessing including constant value deletion and measurement error reduction should be conducted.

The collected data from a sensor may be constant over time when a machine is under a certain condition. However, the data may vary when the machine is under a different condition. If constant values are generated, the condition monitoring data are biased and should be removed since constant values are meaningless for a time-series regression problem. Smoothing is a commonly adopted way to reduce measurement errors and extract the general tendency of the degradation of a machine.

Degradation information from different sensors (or behind the condition monitoring data) may be correlated, contain redundant information, and have different contributions that reflect a machine's degradation. Based on these concerns, PPCA, which considers the probabilistic distribution of each variable, is adopted to extract the PCs but avoid redundant information.

PPCA is simultaneously conducted on both the training and testing samples in this article, which is different from most of the existing studies. In this way, the similarities and common characteristics among training and testing samples are mined even though machines are under different conditions.

After processing using PPCA, the PCs with high contribution levels to reflecting machine degradation are selected. The selected PCs decrease the dimensions of the inputs of a BLSTM network. Therefore, the training process of a BLTM network may be lightweight. Due to the different orders of magnitudes of the selected PCs, normalization should be conducted. Here, z-score normalization is adopted as follows:

$$p_k^* = \frac{p_k - \mu_k}{\sigma_k} \tag{1}$$

for $k = 1, 2, \ldots, K$, where $p_k$ denotes the $k$th PC and $K$ denotes the total number of selected PCs. The notations $\mu_k$ and $\sigma_k$ denote the mean value and standard deviation of $p_k$, respectively, and $p_k^*$ denotes the normalized PC. It is worth noting that all the training and testing samples are considered when calculating $\mu_k$, $\sigma_k$, and $p_k^*$. This is consistent with the PPCA process.

### *B. Training Process of a BLSTM Network With AWGN*

Before training a BLSTM network, manually generated independent AWGN is injected into the normalized PCs. The AWGN obeys a normal distribution $\mathcal{N}(0, 1)$. Noise can be injected into the input layer, hidden layer, and output layer of an ANN. In this article, noise is only injected into the input data at the input layer of the BLSTM network. Therefore, AWGN should be mixed with the normalized PCs, and the mixed features are written as

$$\mathbf{P}_{\text{noisy}} = \mathbf{P}^* + \mathcal{N} \tag{2}$$

where $\mathbf{P}_{\text{noisy}}$ denotes the matrix of normalized PCs after injecting AWGN. The notations $\mathbf{P}^*$ and $\mathcal{N}$ denote the matrices of normalized PCs and AWGN, respectively.

Due to the impacts caused by the AWGN on the normalized PCs, normalization should be conducted on the "polluted" degradation features, which are $\mathbf{P}_{\text{noisy}}$ in (2). The normalization method is also the z-score. The notation $\mathbf{P}^*_{\text{noisy}}$ denotes the normalized $\mathbf{P}_{\text{noisy}}$.

A BLSTM network is used to predict the RULs of machines under different conditions. A BLSTM is an improvement of LSTM that is governed by gates, including input, forget, and output gates. Compared with LSTM, BLSTM consists of two separate hidden layers to handle the sequence data in the forward and backward directions. By combining the output from the two hidden layers, information related to the past and future is captured and fully used.

Suppose that there is a training set $\{\mathbf{X}_s, \mathbf{Y}_s\}$, for $s = 1, 2, \ldots, S$, where $S$ denotes the total number of training samples. In terms of a training tuple, the input set is $\mathbf{X}_s = [x_s^1, x_s^2, \ldots, x_s^t, \ldots, x_s^T]$, where $\mathbf{X}_s$ is the time-series data of sample $s$ and $x_s^t$ is the corresponding input data at time $t$. The dimension of $x_s^t$ is determined by the feature extraction method. For example, assume that a vibration signal is collected from condition monitoring, and the wavelet packet analysis is used to extract degradation features. The wavelet base is "db4," and the decomposition level is three. Therefore, the degradation features are 8-D. Consequently, the dimension of the input set is 8 at time $t$. In this article, the dimension of $x_s^t$ is determined by the selected PCs. The output set is $\mathbf{Y}_s = [y_s^1, y_s^2, \ldots, y_s^t, \ldots, y_s^T]$, with $y_s^t$ is the output at time $t$. Since an output value represents the RUL of a machine, the dimension of $y_s^t$ is 1 at time $t$. The outputs from the forward and backward calculation of a BLSTM network can be obtained from (3)–(14) according to [35]

$$\overrightarrow{\mathbf{h}_t} = \tanh\left(\overrightarrow{\mathbf{c}_t}\right) \odot \overrightarrow{\mathbf{o}_t} \tag{3}$$

$$\overrightarrow{\mathbf{o}_t} = \sigma\left(\overrightarrow{\mathbf{W}_o}\mathbf{X}\mathbf{N}_t^* + \overrightarrow{\mathbf{R}}_o \overrightarrow{\mathbf{h}}_{t-1} + \overrightarrow{\mathbf{b}}_o\right) \tag{4}$$

$$\overrightarrow{\mathbf{c}_t} = \overrightarrow{\mathbf{z}_t} \odot \overrightarrow{\mathbf{i}_t} + \overrightarrow{\mathbf{c}}_{t-1} \odot \overrightarrow{\mathbf{f}_t} \tag{5}$$

$$\overrightarrow{\mathbf{f}_t} = \sigma\left(\overrightarrow{\mathbf{W}}_f\mathbf{X}\mathbf{N}_t^* + \overrightarrow{\mathbf{R}}_f \overrightarrow{\mathbf{h}}_{t-1} + \overrightarrow{\mathbf{b}_f}\right) \tag{6}$$

$$\overrightarrow{\mathbf{i}_t} = \sigma\left(\overrightarrow{\mathbf{W}}_i\mathbf{X}\mathbf{N}_t^* + \overrightarrow{\mathbf{R}}_i \overrightarrow{\mathbf{h}}_{t-1} + \overrightarrow{\mathbf{b}_i}\right) \tag{7}$$

$$\overrightarrow{\mathbf{z}_t} = \tanh\left(\overrightarrow{\mathbf{W}}_z\mathbf{X}\mathbf{N}_t^* + \overrightarrow{\mathbf{R}}_z \overrightarrow{\mathbf{h}}_{t-1} + \overrightarrow{\mathbf{b}_z}\right) \tag{8}$$

$$\overleftarrow{\mathbf{h}_t} = \tanh\left(\overleftarrow{\mathbf{c}_t}\right) \odot \overleftarrow{\mathbf{o}_t} \tag{9}$$

$$\overleftarrow{\mathbf{o}_t} = \sigma\left(\overleftarrow{\mathbf{W}_o}\mathbf{X}\mathbf{N}_t^* + \overleftarrow{\mathbf{R}_o} \overleftarrow{\mathbf{h}}_{t+1} + \overleftarrow{\mathbf{b}_o}\right) \tag{10}$$

$$\overleftarrow{\mathbf{c}_t} = \overleftarrow{\mathbf{z}_t} \odot \overleftarrow{\mathbf{i}_t} + \overleftarrow{\mathbf{c}}_{t+1} \odot \overleftarrow{\mathbf{f}_t} \tag{11}$$

$$\overleftarrow{\mathbf{f}_t} = \sigma\left(\overleftarrow{\mathbf{W}_f}\mathbf{X}\mathbf{N}_t^* + \overleftarrow{\mathbf{R}_f} \overleftarrow{\mathbf{h}}_{t+1} + \overleftarrow{\mathbf{b}_f}\right) \tag{12}$$

$$\overleftarrow{\mathbf{i}_t} = \sigma\left(\overleftarrow{\mathbf{W}_i}\mathbf{X}\mathbf{N}_t^* + \overleftarrow{\mathbf{R}_i} \overleftarrow{\mathbf{h}}_{t+1} + \overleftarrow{\mathbf{b}_i}\right) \tag{13}$$

$$\overleftarrow{\mathbf{z}_t} = \tanh\left(\overleftarrow{\mathbf{W}_z}\mathbf{X}\mathbf{N}_t^* + \overleftarrow{\mathbf{R}_z} \overleftarrow{\mathbf{h}}_{t+1} + \overleftarrow{\mathbf{b}_z}\right). \tag{14}$$

In (3)–(14), the notations $\overrightarrow{\mathbf{\Theta}}_{\text{BLSTM}}$ and $\overleftarrow{\mathbf{\Theta}}_{\text{BLSTM}}$ denote the parameter sets in the forward and backward processes, respectively. The notations $\overrightarrow{\mathbf{W}_z}$, $\overrightarrow{\mathbf{W}_i}$, $\overrightarrow{\mathbf{W}_f}$, $\overrightarrow{\mathbf{W}_o}$ and $\overleftarrow{\mathbf{W}_z}$, $\overleftarrow{\mathbf{W}_i}$, $\overleftarrow{\mathbf{W}_f}$ $\overleftarrow{\mathbf{W}_o}$ denote the input weights with respect to the current input $\mathbf{X}\mathbf{N}_t^*$, respectively. The notation $\mathbf{X}\mathbf{N}_t^*$ is the normalized input at time $t$. The notations $\overrightarrow{\mathbf{R}_z}$, $\overrightarrow{\mathbf{R}_i}$, $\overrightarrow{\mathbf{R}_f}$, $\overrightarrow{\mathbf{R}_o}$ and $\overleftarrow{\mathbf{R}_z}$, $\overleftarrow{\mathbf{R}_i}$, $\overleftarrow{\mathbf{R}_f}$, $\overleftarrow{\mathbf{R}_o}$ are the recurrent weights with respect to previous recurrent input $\overrightarrow{\mathbf{h}}_{t-1}$ and future recurrent input $\overleftarrow{\mathbf{h}}_{t+1}$ of the

forward and backward processes, respectively. The notations $\overrightarrow{\mathbf{b}_z}$, $\overrightarrow{\mathbf{b}_i}$, $\overrightarrow{\mathbf{b}_f}$, $\overrightarrow{\mathbf{b}_o}$ and $\overleftarrow{\mathbf{b}_z}$, $\overleftarrow{\mathbf{b}_i}$, $\overleftarrow{\mathbf{b}_f}$, $\overleftarrow{\mathbf{b}_o}$ are the bias weights of the forward and backward processes, respectively. The notations $\sigma$ and tanh are the activation functions using the logistic sigmoid and hyperbolic tangent, respectively. The notations $\overrightarrow{\mathbf{i}_t}$ and $\overleftarrow{\mathbf{i}_t}$ are the input gates in the forward and backward processes, respectively. The notations $\overrightarrow{\mathbf{f}_t}$ and $\overleftarrow{\mathbf{f}_t}$ represent the forget gates in the forward and backward processes, respectively. The notations $\overrightarrow{\mathbf{o}_t}$ and $\overleftarrow{\mathbf{o}_t}$ are the output gates in the forward and backward processes, respectively. The notation $\odot$ is the pointwise multiplication operator. The notations $\overrightarrow{\mathbf{c}_t}$ and $\overleftarrow{\mathbf{c}_t}$ are the memory cells in the forward and backward processes, respectively.

The output from the BLSTM can be calculated by combining the outputs from the two directions as

$$\mathbf{y}_t = \mathbf{W}_{\overrightarrow{hy}}\overrightarrow{\mathbf{h}_t} + \mathbf{W}_{\overleftarrow{hy}}\overleftarrow{\mathbf{h}_t} + \mathbf{b}_y \tag{15}$$

where $\mathbf{y}_t$ denotes the calculated output at time $t$. The notations $\mathbf{W}_{\overrightarrow{hy}}$ and $\mathbf{W}_{\overleftarrow{hy}}$ denote the weights connected to the hidden layer with the output layer. The notation $\mathbf{b}_y$ denotes the bias at the output layer.

The input samples of the BLSTM network are $\mathbf{P}^*_{\text{noisy}}$ during the training process. It is acceptable that the more complex the configuration of an ANN is, the more time-consuming the training process is. To decrease the calculation time and complexity, only one BLSTM layer is adopted in this article. The number of hidden neurons in the BLSTM can be determined by an empirical formula [36] as follows:

$$N_h = \frac{N_s}{\alpha \times (N_i + N_o)} \tag{16}$$

where $N_h$, $N_s$, $N_i$, and $N_o$ denote the number of hidden units in the BLSTM network, the number of samples in the training data set, the number of input neurons, and the number of output neurons, respectively. The notation $\alpha$ represents an arbitrary scaling factor in the interval of [2, 10].

The outputs of the BLSTM network are the machine RULs. A negative RUL is meaningless in real applications. Therefore, a leaky rectified linear unit (ReLU) layer is constructed following the BLSTM layers. The leaky ReLU is an improvement of the ReLU. In a ReLU layer, even though a negative output is revised to zero, its derivative is zero accordingly. Consequently, the corresponding neurons are unable to update and die. The leaky ReLU avoids dead neurons and creates a small gradient for the derivative. Therefore, neurons are still able to update. Thus, the training process is more effective.

Dropout is a widely used technique to reduce the training process overhead when training deep ANNs. It randomly discards a subset of neurons and their neural connections [37]. An ANN that applies the dropout method can be regarded as an ensemble learning framework because randomly discarding neurons is equivalent to sampling a subnetwork from the original network [38]. Besides, dropout can increase the base model's generalization ability to avoid overfitting.

The optimization algorithm for the loss function is the adaptive moment estimation (Adam). The network parameter update process of the Adam algorithm in the $l$th iteration can be calculated as follows:

$$\boldsymbol{\theta}_l = \boldsymbol{\theta}_{l-1} - \eta \times \frac{\widehat{\mathbf{m}}_l}{\sqrt{\widehat{\mathbf{n}}_l} + \varepsilon} \tag{17}$$

$$\widehat{\mathbf{m}}_l = \frac{\mathbf{m}_l}{1 - \beta_1^l} \tag{18}$$

$$\widehat{\mathbf{n}}_l = \frac{\mathbf{n}_l}{1 - \beta_2^l} \tag{19}$$

$$\mathbf{m}_l = \beta_1 \mathbf{m}_{l-1} + (1 - \beta_1)\mathbf{g}_l \tag{20}$$

$$\mathbf{n}_l = \beta_2 \mathbf{n}_{l-1} + (1 - \beta_2)\mathbf{g}_l^2 \tag{21}$$

where $\theta$ denotes the parameter set. Here, $\eta$ and $\varepsilon$ denote the step size and the numerical stability constant, respectively. The notations $\widehat{\mathbf{m}}$ and $\widehat{\mathbf{n}}$ denote the deviation corrections for $\mathbf{m}$ and $\mathbf{n}$, which represent the first moment estimate and the second estimate of the gradient, respectively. The notation $g_l$ denotes the gradient of the loss function in the $l$th iteration.

TABLE I
C-MAPSS DATA SET

| Dataset | FD001 | FD002 | FD003 | FD004 |
|---|---|---|---|---|
| Training trajectories | 100 | 260 | 100 | 249 |
| Test trajectories | 100 | 259 | 100 | 248 |
| Maximum lifespan (cycles) | 362 | 378 | 525 | 543 |
| Average lifespan (cycles) | 206 | 206 | 247 | 245 |
| Minimum lifespan (cycles) | 128 | 128 | 145 | 128 |
| OCs | 1 | 6 | 1 | 6 |
| FCs | 1 | 1 | 2 | 2 |

## III. EXPERIMENT AND VALIDATION

### A. Data Set Description

The C-MAPSS data set for turbofan engine degradation simulation[1] is used for validation. The data set contains four subdatasets that are further divided into training and testing trajectories. Each trajectory indicates an engine's cycle records. The details of the data set are listed in Table I. The engines in data set FD001 suffered the failure of a high-pressure compressor with a single OC. The engines in data set FD002 suffered the failure of a high-pressure compressor with six OCs. The engines in data set FD003 suffered the failure of a high-pressure compressor and fan with a single OC. The engines in data set FD004 suffered the failure of a high-pressure compressor and fan with six OCs according to [39].

### B. Evaluation Indicators of Prediction Performance

Regarding the OCs and FCs, the four data sets are different from each other. In this article, the proposed method focuses on RUL prediction under different conditions; namely, the training and testing data sets are drawn from different conditions (OC and/or FCs). Therefore, there are 12 training and testing tuples in total. It is worth noting that only the training trajectories are used to validate the proposed method.

Three criteria are defined to evaluate the prediction accuracy of the proposed RUL prediction method. The first criterion,

[1] https://ti.arc.nasa.gov/tech/dash/groups/pcoe/prognostic-data-repository/

TABLE II
DIMENSION OF CONSTANT VALUES

| Dataset | Dimension of constant values | Dataset | Dimension of constant values |
|---|---|---|---|
| FD001 | 7 | FD002 | 0 |
| FD003 | 6 | FD004 | 0 |
| FD001 & FD002 | 0 | FD001 & FD003 | 6 |
| FD001 & FD004 | 0 | FD002 & FD003 | 0 |
| FD002 & FD004 | 0 | FD003 & FD004 | 0 |

which is provided by the data creators [39], is given in the following equation:

$$E_{\text{score}} = \begin{cases} \sum_{j=1}^{J} \exp\left(-\frac{P_j - R_j}{13}\right) - 1, & \text{if } P_j - R_j < 0 \\ \sum_{j=1}^{J} \exp\left(\frac{P_j - R_j}{10}\right) - 1, & \text{if } P_j - R_j \geq 0. \end{cases} \tag{22}$$

Equation (22) is used to illustrate the penalty to the late prediction more than early prediction. The positive or negative prediction error $(P_j - R_j)$ has different impacts on the final score result.

The second criterion, which is (23), is the root mean square error (RMSE). It is widely used to evaluate the prediction accuracy and extent of the deviation

$$E_{\text{RMSE}} = \sqrt{\frac{1}{J}\sum_{j=1}^{J}(P_j - R_j)^2}. \tag{23}$$

The third criterion is the prediction error percentage $E_{\text{perc}}$, as given in (24). It is a rigorous criterion to evaluate the prediction confidence interval

$$E_{\text{perc}} = \frac{1}{J}\sum_{j=1}^{J}\sqrt{\left(\frac{P_j - R_j}{R_j}\right)^2}. \tag{24}$$

In (22)–(24), $J$ denotes the total number of engines as testing samples. The notations $P_j$ and $R_j$ denote the predicted and real RULs, respectively. In the three criteria, smaller values represent better prediction accuracy in the different views.

### C. Approach Validation

From Section II, constant values should be removed from the original condition monitoring data. The original condition monitoring data have 24 dimensions. The dimension of constant values is listed in Table II, where "FD001 and FD002" mean that data set FD001 or FD002 is the training sample and the other data set is the testing sample. Even though seven sensory measurements are constants in FD001, they remain if FD001 and FD002 are grouped as the training–testing tuples.

A one-step moving mean is used to smooth and reduce the measurement errors. After smoothing, PPCA is used to extract the PCs from both the training and testing samples. The PCs that explain 98% of the first cumulative summation are selected. These scores of the selected PCs are shown in Fig. 3, where the engines are randomly selected.

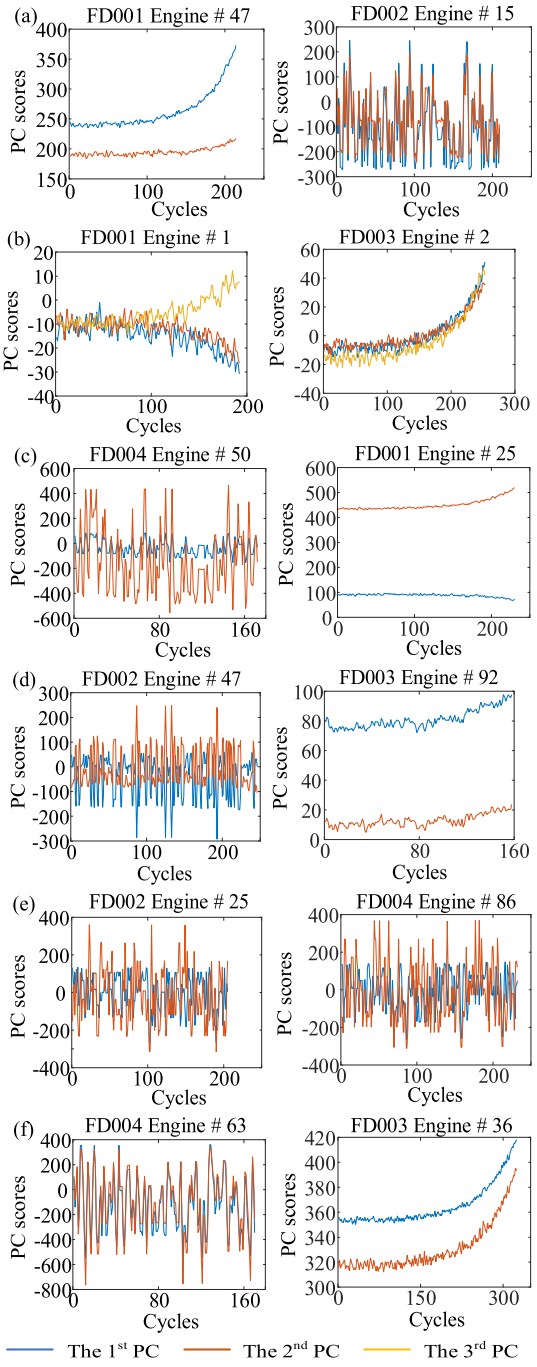


Fig. 3. Scores of the selected PCs of the training and testing samples. (a) FD001 and FD002. (b) FD001 and FD003. (c) FD001 and FD004. (d) FD002 and FD003. (e) FD002 and FD004. (f) FD003 and FD004.

From Fig. 3, only two or three PCs are extracted from the six training–testing tuples. The 2-D or 3-D PCs can reduce the training time for a BLSTM network compared with using all the 24-D raw features. The selected PCs show different tendencies: increasing, decreasing, and oscillating.

The selected PCs have different orders of magnitude and vary greatly. Normalization was conducted before the manually generated AWGN injection. Otherwise, the AWGN injection has little influence. After normalization and AWGN injection, the PCs that contain mixed noise should be renormalized.

TABLE III
THEORETICAL NUMBER OF HIDDEN UNITS ACCORDING TO (16)

| Datasets | Number of selected PCs | $\alpha$=5 | $\alpha$=6 | $\alpha$=7 | $\alpha$=8 | $\alpha$=9 | $\alpha$=10 |
|---|---|---|---|---|---|---|---|
| FD001 | 2 | 69 | 57 | 49 | 43 | 38 | 34 |
| FD002 | 2 | 69 | 57 | 49 | 43 | 38 | 34 |
| FD003 | 3 | 62 | 52 | 44 | 39 | 34 | 31 |
| FD004 | 2 | 82 | 68 | 58 | 51 | 45 | 41 |

It is worth noting that noise is only injected into the input samples during the training process rather than the output samples of the BLSTM network. In addition, noise is not injected into the input data when predicting the RULs of the testing engines.

The output of the BLSTM network is the RULs of the engines. In fact, the label value of the RUL has a significant impact on the prediction performance. Some researchers have proven that piecewise linear labeling of the C-MAPSS engine data set is effective and beneficial [5], [33], which means that the RUL of an engine is labeled with a constant value at the early stage of its lifetime and then decreases linearly. The constant value of the RUL label is suggested to be 125 cycles [5], [33].

The minimum batch size is set to 5; therefore, there are 20, 52, 20, and 50 batches for data sets FD001, FD002, FD003, and FD004, respectively. The average numbers of training samples in a batch are 1031.55, 1033.83, 1236, and 1224.98 for the four subdatasets, respectively. The number of hidden units is determined by (16) and the enumeration method. The scaling factor $\alpha$ in (16) is set in the interval of [5, 10]. To determine the proper number of hidden units, one-fold cross validation is performed on the training samples. The maximum and minimum numbers of hidden units are 82 and 31, respectively, as shown in Table III. To reduce the calculation cost, the number of hidden units is varied from 30 to 80 by increments of 10 in each experiment, each of which is run five times to avoid randomness. Then, the proper number of hidden units is determined according to the minimum average $E_{\text{score}}$, $E_{\text{RMSE}}$, and $E_{\text{perc}}$. Here, the number of hidden units should be set to 70, 70, 40, and 60 when the data sets are FD001, FD002, FD003, and FD004, respectively. The corresponding $\alpha$ should be 5 (for FD001 and FD002) or 7 (for FD003 and FD004). Here, the dropout rate is set to 0.5, which is recommended in [40].

There is a fully connected layer with the same number of hidden units as in the BLSTM layer. The initial learning rate is set to 0.01, and the learning rate is reduced by a factor of 0.3 every 30 epochs. All the algorithms are programed in MATLAB 2018b and run on a desktop computer with an Intel i7 1.87-GHz CPU with 120 GB of RAM running Windows 10.

The prediction errors from the proposed method are listed in Table IV, where "MC changes" indicate the changes in engine conditions (including OCs and FCs) with respect to the training and testing data sets. In Table IV, $\rightarrow$ denotes the training and testing tuples. Although all the tuples are from different OCs and/or FCs, the proposed method achieves high performance in terms of the evaluation criteria. In particular,

TABLE IV
PREDICTION ERRORS FROM THE PROPOSED METHOD

| Datasets | MC changes | $E_{\text{score}}$ | $E_{\text{RMSE}}$ (cycles) | $E_{\text{perc}}$ |
|---|---|---|---|---|
| FD001→FD002 | (1, 1)→(6, 1) | 20.89 | 1.06 | 0.78 |
| FD001→FD003 | (1, 1)→(1, 2) | 5.66 | 0.58 | 0.55 |
| FD001→FD004 | (1, 1)→(6, 2) | 60.38 | 2.35 | 2.12 |
| FD002→FD001 | (6, 1)→(1, 1) | 11.97 | 1.13 | 1.13 |
| FD002→FD003 | (6, 1)→(1, 2) | 6.78 | 0.73 | 0.95 |
| FD002→FD004 | (6, 1)→(6, 2) | 18.34 | 0.82 | 0.70 |
| FD003→FD001 | (1, 2)→(1, 1) | 15.69 | 1.70 | 1.43 |
| FD003→FD002 | (1, 2)→(6, 1) | 48.72 | 1.88 | 1.68 |
| FD003→FD004 | (1, 2)→(6, 2) | 40.07 | 1.80 | 1.43 |
| FD004→FD001 | (6, 2)→(1, 1) | 4.53 | 0.45 | 0.44 |
| FD004→FD002 | (6, 2)→(6, 1) | 25.12 | 1.12 | 0.90 |
| FD004→FD003 | (6, 2)→(1, 2) | 32.23 | 2.79 | 2.79 |

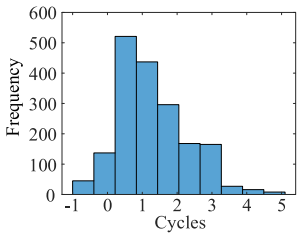


Fig. 4. Histogram of prediction errors from the proposed method.

$E_{\text{score}}$ is less than 70 with respect to the total scores from all the engines in the testing data sets. The histogram of prediction errors for the 12 training–testing tuples is shown in Fig. 4.

## IV. RESULTS AND DISCUSSION

To illustrate the effectiveness and performance of the proposed method, the following comparison examples are conducted and compared.

*Case 1:* The proposed method.

*Case 2:* The same prediction procedures but without AWGN.

*Case 3:* The same prediction procedures but without PPCA.

*Case 4:* Training using all the data sets but without AWGN.

*Case 5:* Training using all the data sets but without PPCA or AWGN.

In case 2, the prediction procedures are the same as the proposed method, but AWGN is not injected. Therefore, the normalized selected PCs without noise injection are used to train the BLSTM network. Then, the normalized PCs of the testing samples are imported into the well-trained BLSTM network for RUL prediction. Case 2 is used to illustrate the effectiveness of AWGN injection.

In case 3, the prediction procedures are the same as the proposed method, but PPCA is not performed to reduce the dimension of the input samples of the network. Case 3 is used to illustrate the reduction of the calculation time by using 2-D or 3-D PCs in the proposed method.

In case 4, all the data sets are used to train a BLSTM network. For example, the engines in data set FD001 are the testing engines, and the engines in data sets FD002, FD003, and FD004 are used to train the BLSTM network. Similar to case 2, AWGN is not injected into the training samples

TABLE V
PREDICTION ERRORS FROM CASES 2 AND 3

| Datasets | Case 2 | | | Case 3 | | |
|---|---|---|---|---|---|---|
| | $E_{score}$ | $E_{RMSE}$ (cycles) | $E_{perc}$ | $E_{score}$ | $E_{RMSE}$ (cycles) | $E_{perc}$ |
| FD001→FD002 | 48.61 | 2.57 | 1.87 | 491.06 | 10.39 | 9.15 |
| FD001→FD003 | 8.37 | 1.18 | 0.97 | 211.84 | 11.30 | 8.58 |
| FD001→FD004 | 135.91 | 5.97 | 5.31 | 299.33 | 8.10 | 6.95 |
| FD002→FD001 | 13.78 | 1.33 | 1.28 | 104.96 | 7.18 | 7.16 |
| FD002→FD003 | 60.66 | 4.74 | 4.74 | 27.01 | 2.39 | 2.39 |
| FD002→FD004 | 40.80 | 1.86 | 1.45 | 230.69 | 2.68 | 6.36 |
| FD003→FD001 | 35.75 | 3.65 | 2.72 | 95.57 | 6.78 | 6.59 |
| FD003→FD002 | 69.43 | 3.23 | 2.86 | 178.28 | 5.86 | 4.33 |
| FD003→FD004 | 22.54 | 1.27 | 1.06 | 264.26 | 7.52 | 6.61 |
| FD004→FD001 | 9.42 | 0.91 | 0.90 | 88.38 | 6.33 | 6.33 |
| FD004→FD002 | 51.97 | 1.88 | 1.81 | 190.80 | 5.54 | 5.46 |
| FD004→FD003 | 17.07 | 1.58 | 1.57 | 75.07 | 5.60 | 5.60 |

because most of the existing RUL prediction methods based on DL do not inject AWGN. Case 4 is used to illustrate the effectiveness of the proposed method with respect to the few training samples.

In case 5, all the engines in the data sets are used to train the BLSTM network, which is similar to case 4, but PPCA is not conducted. All the training samples are smoothed during preprocessing, and then, the preprocessed data are normalized and imported into the BLSTM network for training. In case 5, neither PPCA nor AWGN injection was performed. Case 5 is used to illustrate the effectiveness of adopting PPCA.

### *A. Validation of Performing AWGN and PPCA*

The prediction errors from cases 2 and 3 are distinctly listed in Table V. The overall prediction evaluation indicators from cases 2 and 3 given in Table V are greater than the corresponding values from case 1 listed in Table IV. This means that the proposed method achieves better performance than the two comparative cases. An engine is randomly selected from the testing data set, and its predicted RULs from different cases are shown in Fig. 5.

From Fig. 5, the prediction results from case 2 show good performance excluding the training–testing tuples FD001→FD002, FD001→FD004, FD003→FD002, FD003→FD004, and FD004→FD003, which means that the prediction method is not steady. Here, "steady" indicates that the results fluctuate within a small interval. It is obvious that the prediction results from case 2 fluctuate greatly. Thus, the robustness of the proposed method due to manually injecting AWGN is illustrated by comparing the results from cases 1 and 2.

In Fig. 5, the prediction results from case 3 have large variations. In case 3, the minibatch size is set to 5, which is the same as case 1. The dimensions of the input samples are 24 or 18. The number of hidden units in case 3 is 10 according to (16), which is much less than the ones in case 1. With the small minibatch size and number of hidden units, a BLSTM cannot be trained well; therefore, the prediction results show poor performance, as shown in Table V and Fig. 5. By comparing cases 1 and 3, the effectiveness of PPCA is illustrated.

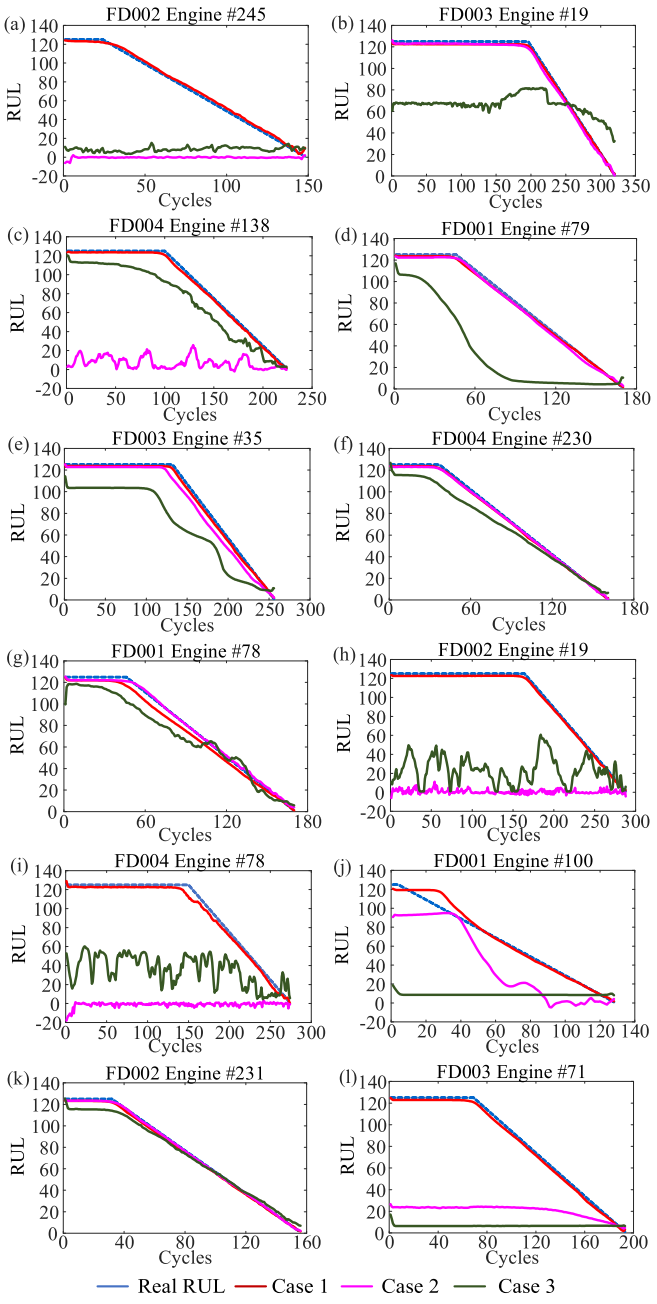


Fig. 5. Real RULs and predicted RULs obtained from different cases. (a) FD001 → FD002. (b) FD001 → FD003. (c) FD001 → FD004. (d) FD002 → FD001. (e) FD002 → FD003. (f) FD002 → FD004. (g) FD003 → FD001. (h) FD003 → FD002. (i) FD003 → FD004. (j) FD004 → FD001. (k) FD004 → FD002. (l) FD004 → FD003.

The prediction errors from cases 4 and 5 are listed in Table VI. To avoid the randomness caused by the parameters in the BLSTM network, the two methods are run five times each. The results listed in Table VI are the average values from the five runs. In cases 4 and 5, the determination of the parameters for a BLSTM network is the same as that in case 1.

In Table VI, the prediction method achieves good performance using all the training samples without AWGN injection (namely, case 4). The evaluation indicators are close to the results from the proposed method. However, case 4 needs more

TABLE VI
PREDICTION ERRORS FROM CASES 4 AND 5

| Cases | Datasets | $E_{score}$ | $E_{RMSE}$ (cycles) | $E_{perc}$ |
|---|---|---|---|---|
| Case 4 | FD002+FD003+FD004→FD001 | 13.36 | 1.28 | 1.23 |
| | FD001+FD003+FD004→FD002 | 38.88 | 1.57 | 1.32 |
| | FD001+FD002+FD004→FD003 | 13.39 | 1.26 | 1.24 |
| | FD001+FD002+FD003→FD004 | 36.67 | 1.60 | 1.31 |
| Case 5 | FD002+FD003+FD004→FD001 | 145.66 | 8.91 | 8.88 |
| | FD001+FD003+FD004→FD002 | 290.86 | 7.41 | 7.14 |
| | FD001+FD002+FD004→FD003 | 104.17 | 6.96 | 6.94 |
| | FD001+FD002+FD003→FD004 | 550.86 | 10.07 | 9.54 |

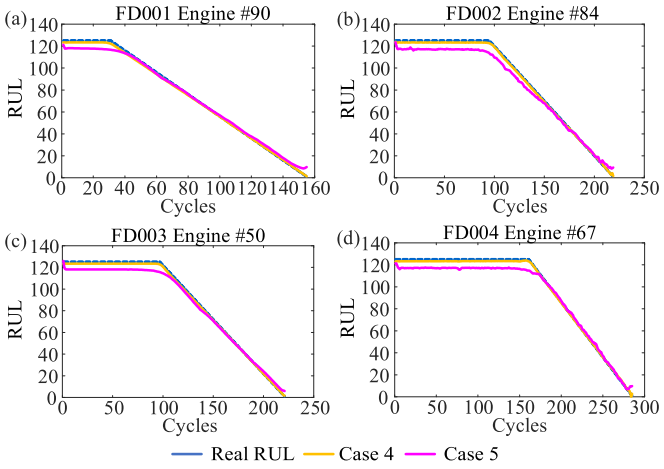


Fig. 6. Predicted and real RULs obtained from cases 4 and 5. (a) FD002 + FD003 + FD004 → FD001. (b) FD001 + FD003 + FD004 → FD002. (c) FD001 + FD002 + FD004 → FD003. (d) FD001 + FD002 + FD003 → FD004.

training samples. In case 5, PPCA and AWGN injection are not conducted. Even though more training samples are used, the obtained evaluation indicators are still greater than those from the proposed method. The comparison of the real RULs and predicted RULs from cases 4 and 5 are shown in Fig. 6, where the predicted RULs are very close to the real RULs. Comparing the results from cases 4 and 5 shows that adopting PPCA is beneficial to RUL prediction.

### *B. Validation of the Calculation Time and Data Volume*

In addition to the prediction accuracy, the required number of training samples and the calculation time of an RUL prediction method are critical to real applications. Even though the RUL prediction methods in cases 4 and 5 achieve high prediction accuracy, the BLSTM network needs a larger volume of data and more training time than the proposed method. The training times of the BLSTM networks in the five comparative cases are shown in Fig. 7, where the four columns indicate that the testing data sets are FD001, FD002, FD003, and FD004, respectively. In cases 1–3, the average training time of a BLSTM network is adopted. For example, 281 s in case 1 means it is the average training time of a BLSTM network when FD002→FD001, FD003→FD001, and FD004→FD001.

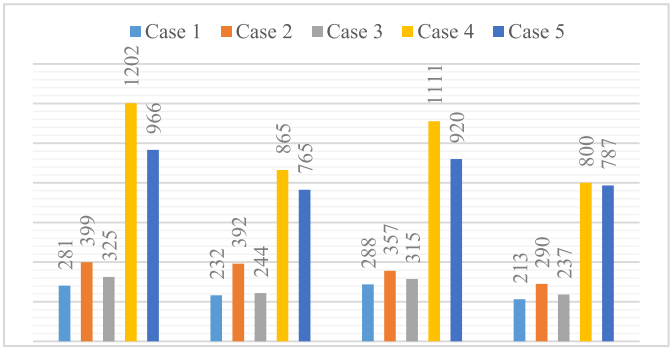


Fig. 7. Training time (seconds) of a BLSTM network.

Fig. 7 shows that the prediction methods of cases 4 and 5 need more training time for a BLSTM network than case 1 even though these three methods can obtain good prediction results and are steady. However, cases 4 and 5 need more training samples to guarantee the prediction performance. In addition, the fact that prediction can be enhanced by noise injection is illustrated by comparing cases 1 and 2. The training time of a BLSTM network declines due to noise injection. Compared with case 1, more dimensions are input into a BLSTM network in case 3; thereafter, case 3 needs more training time than case 1. Therefore, the proposed method is more effective and needs fewer training samples and less training time with high prediction accuracy.

Regarding the data volume for training a neural network, the minibatch size is set to 5 in the proposed method. That is, there are five engines gathered in a batch. To further validate the effectiveness using a small number of training samples, a series of comparisons are conducted, and the results are shown in Fig. 8. In the comparisons, the minibatch is still set to 5, but the number of total engines for training a BLSTM network is increased by 20 engines during each prediction. The number of engines in the testing samples is unchanged.

A simplified explanation is given from Fig. 8 as follows: the initial number of total engines in the training samples is 20, and the minibatch is set to 5. The number of engines in the testing samples is 260 when FD001→FD002. Then, 40 engines from FD001 are used as the training samples, 260 engines from FD002 are regarded as the testing samples in the next prediction, and the minibatch is still set to 5.

The prediction results shown in Fig. 8 are from one prediction; therefore, the results have the tendency to fluctuate. Despite all this, the prediction results have small error indicators when a small number of samples are used to train a BLSTM network. Using a small number of engines as the training samples can still obtain good prediction results as measured by the evaluation indicators. From Fig. 8, regardless of how many engines are used as the training samples, the maximum score (namely, $E_{score}$) is still less than 200, and the error deviation (namely, $E_{RMSE}$) is less than ten cycles.

Using a small number of training samples to test another whole data set means that the numbers of engines in the training and testing data sets are unbalanced. For example, the first point shown in Fig. 8 means that 20 engines from

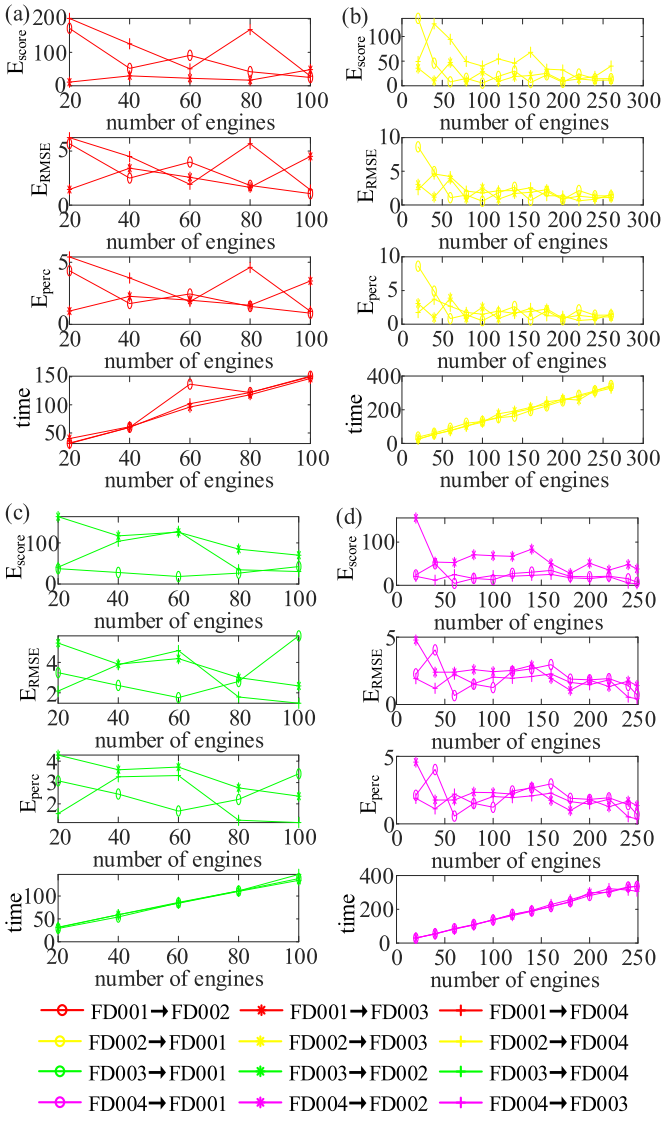


Fig. 8. Prediction evaluation indicators and training time of BLSTM networks when using different numbers of engines. (a) FD001 is the training data set. (b) FD002 is the training data set. (c) FD003 is the training data set. (d) FD004 is the training data set.

FD001 are used to predict the RULs of 260 engines in FD002. The engines from FD001 and FD002 have different OCs and FCs. In addition, there are ten times more test engines than training engines. Even in such situations, the proposed method still achieves good prediction performance according to the evaluation indicators and needs less training time than the traditional method.

### *C. Discussion of the State-of-the-Art Approaches*

The C-MAPSS data set is popular and has been widely used to validate the performance of different RUL prediction models. In [18] and [21], the same prediction accuracy criteria are used, which makes the methods comparable. In [18], the model score and RMSE of the prediction error were used as the evaluation indicators, and their calculations are the same as (22) and (23), respectively, in this article. In [18], $10, 20, \ldots, 90$, and 100 engines from the testing data sets were selected to calculate the mean $E_{\text{score}}$, which was used for evaluation. That is different from the proposed method. To make the results comparable, the mean $E_{\text{score}}$ values are recalculated and listed in Table VII, where "-" means that no values are given in [18].

TABLE VII

COMPARISON OF THE MEAN PREDICTION ERRORS FROM THE PROPOSED METHOD AND [18]

| Datasets | The proposed method | | Method in [18] | |
|---|---|---|---|---|
| | Mean $E_{\text{score}}$ | $E_{\text{RMSE}}$ | Mean $E_{\text{score}}$ | $E_{\text{RMSE}}$ |
| FD001→FD002 | 0.08 | 1.06 | 7.04 | 20.77 |
| FD001→FD003 | 0.06 | 0.58 | 4.00 | 14.34 |
| FD001→FD004 | 0.24 | 2.35 | - | - |
| FD002→FD001 | 0.12 | 1.13 | 4.30 | 18.30 |
| FD002→FD003 | 0.07 | 0.73 | - | - |
| FD002→FD004 | 0.07 | 0.82 | 16.16 | 25.44 |
| FD003→FD001 | 0.16 | 1.70 | 2.24 | 13.65 |
| FD003→FD002 | 0.19 | 1.88 | - | - |
| FD003→FD004 | 0.16 | 1.80 | 11.75 | 23.4 |
| FD004→FD001 | 0.05 | 0.45 | - | - |
| FD004→FD002 | 0.10 | 1.12 | 8.45 | 20.83 |
| FD004→FD003 | 0.32 | 2.79 | 9.60 | 24.10 |

In Table VII, the mean $E_{\text{score}}$ from the proposed method is in the interval of (0, 1) and much smaller than values from the method in [18]. In addition, the RMSE, which is the same as (23) in the proposed method, is also smaller than the corresponding values in [18]. Furthermore, the training trajectories of FD001, FD002, FD003, and FD004 are used as the testing samples in the proposed method. This is different from the testing samples in [18].

Da Costa *et al.* [21] proposed an RUL prediction method via deep domain adaptation. In their method, the RULs of engines were piecewise separated by 125 cycles and then rescaled into the interval of (0, 1). The RUL prediction results are shown in Fig. 9 by the blue dotted lines. The magenta lines and red lines in Fig. 9 represent the results from the target- and source-only models. The black piecewise lines represent the RULs of engines after rescaling, which are in the interval of (0, 1). The engines listed in Fig. 9 are randomly selected from the target data sets. In Fig. 9, the results from source-only models and target-only models represent the lower bound and the upper bound of the RUL prediction results, respectively.

In Fig. 9, the predicted RULs show better fitting ability with the real RULs for FD004→FD002, FD003→FD001, FD002→FD004, and FD001→FD003. One commonality behind the four training–testing tuples is that the OCs of FD004→FD002 versus FD002→FD004 and FD003→FD001 versus FD001→FD003 are the same. This is different from the proposed method. Even though the OCs are different between the training and testing tuples, the predicted RULs from the proposed method still approximate the real RULs. This can also be demonstrated in Fig. 5. To make the comparison clearer, $E_{\text{RMSE}}$ and $E_{\text{score}}$ are compared in Table VIII, where $E_{\text{RMSE}}$ and $E_{\text{score}}$ of the proposed method are obviously much smaller than those from [21]. This is consistent with the comparison between Figs. 5 and 9, which also demonstrates the outperformance of the proposed method.

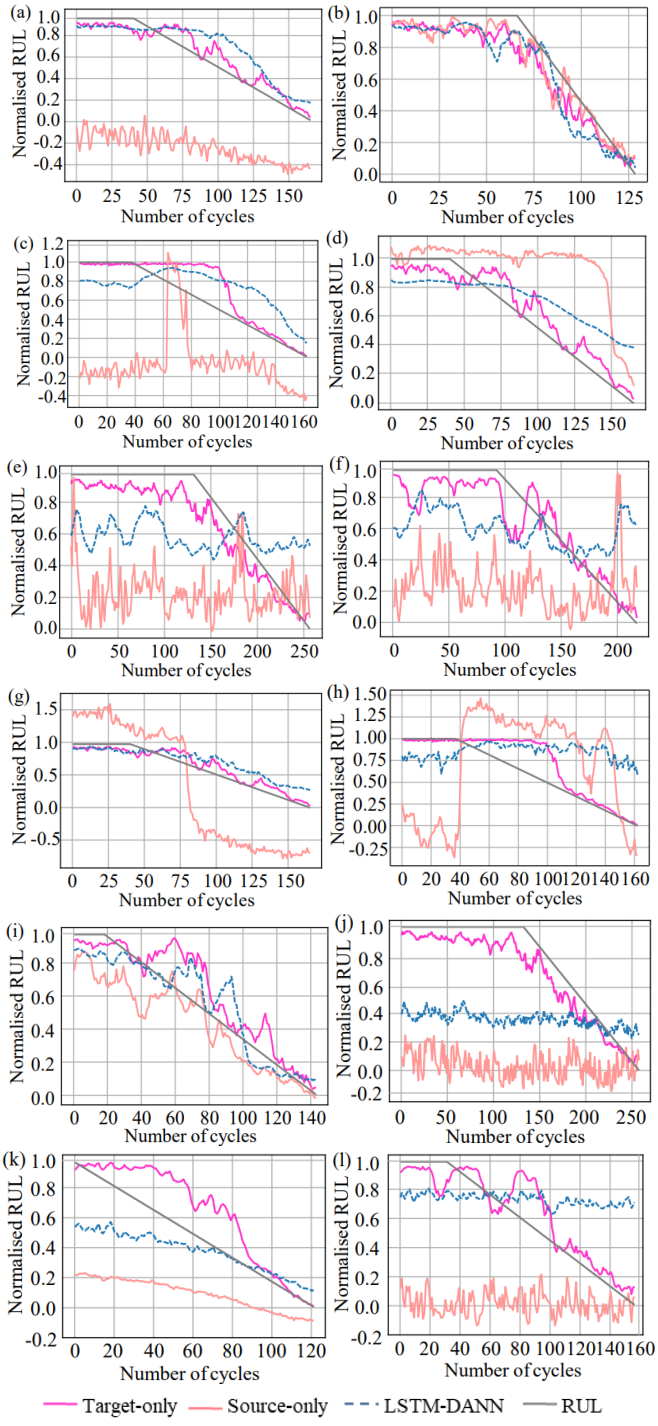


Fig. 9. RUL predictions of the target-only, source-only and LSTM-DANN models for one engine coming from target domain cross-validation data sets [21] where (a) FD004 → FD001, (b) FD004 → FD002, (c) FD004 → FD003, (d) FD003 → FD001, (e) FD003 → FD002, (f) FD003 → FD004, (g) FD002 → FD001, (h) FD002 → FD003, (i) FD002 → FD004, (j) FD001 → FD002, (k) FD001 → FD003, and (l) FD001 → FD004.

By comparing the state-of-the-art methods to the proposed method, the performance of the proposed method is illustrated. The predicted RULs from the proposed method are closer to the real RULs, and the prediction evaluation indicators are smaller than those from the proposed method. Regarding the comparisons in Sections IV-A and IV-B, the proposed method needs fewer training samples and a lower time cost, while the prediction accuracy is high.

TABLE VIII
COMPARISON OF THE PREDICTION ERRORS FROM THE PROPOSED METHOD AND [21]

| Datasets | The proposed method | | Method in [21] | |
|---|---|---|---|---|
| | $E_{score}$ | $E_{RMSE}$ | $E_{score}$ | $E_{RMSE}$ |
| FD001→FD002 | 20.89 | 1.06 | 93,841 | 48.6 |
| FD001→FD003 | 5.66 | 0.58 | 27,005 | 45.9 |
| FD001→FD004 | 60.38 | 2.35 | 57,044 | 43.8 |
| FD002→FD001 | 11.97 | 1.13 | 8,411 | 28.1 |
| FD002→FD003 | 6.78 | 0.73 | 17,406 | 37.5 |
| FD002→FD004 | 18.34 | 0.82 | 66,305 | 31.8 |
| FD003→FD001 | 15.69 | 1.70 | 5,113 | 31.7 |
| FD003→FD002 | 48.72 | 1.88 | 37,297 | 44.6 |
| FD003→FD004 | 40.07 | 1.80 | 141,117 | 47.9 |
| FD004→FD001 | 4.53 | 0.45 | 7,586 | 31.5 |
| FD004→FD002 | 25.12 | 1.12 | 17,001 | 24.9 |
| FD004→FD003 | 32.23 | 2.79 | 5,941 | 27.8 |

## V. CONCLUSION

In this article, a new method to solve the problem of RUL prediction for rotating machines under different conditions is investigated. This method exploits manually generated and intentionally added AWGN to improve the robustness of the RUL prediction method. Besides making predictions under different conditions and achieving prediction robustness, the proposed method also achieves RUL prediction when the data sets are unbalanced, which means that there is a large difference in the numbers of training and testing samples (in different conditions).

Since the good performance of the proposed method mainly depends on noise injection, this method is named the noise-boosted RUL prediction method, which is the opposite of most existing DL-based RUL prediction methods. This is also a primary attempt at injecting noise into a DL-based RUL prediction method. In addition, the proposed RUL prediction method can achieve accurate prediction results even though the training and testing data sets have small numbers of samples. In other words, the method needs fewer training samples and less training time; therefore, the training process has less overhead. The performance of this method is also demonstrated by comparing some methods commonly adopted in practice and state-of-the-art methods.

Even though the preliminary work has been conducted in this article, there is still space to improve the proposed method. In the future, more research will be conducted on how to adaptively set parameters for the network and how to construct more reasonable added noise.

## REFERENCES

[1] L. Xiao, T. Xia, E. Pan, and X. Zhang, "Long-term predictive opportunistic replacement optimisation for a small multi-component system using partial condition monitoring data to date," *Int. J. Prod. Res.*, vol. 58, no. 13, pp. 4015–4032, Jul. 2020.

[2] E. Bechhoefer and R. Schlanbusch, "Calculating remaining useful life in an embedded system," in *Proc. Annu. Conf. PHM Soc.*, Sep. 2018, vol. 10, no. 1, p. 586.

[3] A. Cubillo, S. Perinpanayagam, and M. Esperon-Miguez, "A review of physics-based models in prognostics: Application to gears and bearings of rotating machinery," *Adv. Mech. Eng.*, vol. 8, no. 8, Aug. 2016, Art. no. 168781401666466.

[4] Y. Lei, N. Li, and J. Lin, "A new method based on stochastic process models for machine remaining useful life prediction," *IEEE Trans. Instrum. Meas.*, vol. 65, no. 12, pp. 2671–2684, Dec. 2016.

[5] J. Wu, K. Hu, Y. Cheng, H. Zhu, X. Shao, and Y. Wang, "Data-driven remaining useful life prediction via multiple sensor signals and deep long short-term memory neural network," *ISA Trans.*, vol. 97, pp. 241–250, Feb. 2020.

[6] H. Miao, B. Li, C. Sun, and J. Liu, "Joint learning of degradation assessment and RUL prediction for aeroengines via dual-task deep LSTM networks," *IEEE Trans. Ind. Informat.*, vol. 15, no. 9, pp. 5023–5032, Sep. 2019.

[7] W. Mao, J. He, and M. J. Zuo, "Predicting remaining useful life of rolling bearings based on deep feature representation and transfer learning," *IEEE Trans. Instrum. Meas.*, vol. 69, no. 4, pp. 1594–1608, Apr. 2020.

[8] Y. Cheng, J. Wu, H. Zhu, S. W. Or, and X. Shao, "Remaining useful life prognosis based on ensemble long short-term memory neural network," *IEEE Trans. Instrum. Meas.*, vol. 70, pp. 1–12, 2021.

[9] B. Yang, R. Liu, and E. Zio, "Remaining useful life prediction based on a double-convolutional neural network architecture," *IEEE Trans. Ind. Electron.*, vol. 66, no. 12, pp. 9521–9530, Dec. 2019.

[10] T. Wang, Z. Liu, and N. Mrad, "A probabilistic framework for remaining useful life prediction of bearings," *IEEE Trans. Instrum. Meas.*, vol. 70, pp. 1–12, 2021.

[11] B. Wang, Y. Lei, N. Li, and N. Li, "A hybrid prognostics approach for estimating remaining useful life of rolling element bearings," *IEEE Trans. Rel.*, vol. 69, no. 1, pp. 401–412, Mar. 2020.

[12] H. Sun, D. Cao, Z. Zhao, and X. Kang, "A hybrid approach to cutting tool remaining useful life prediction based on the Wiener process," *IEEE Trans. Rel.*, vol. 67, no. 3, pp. 1294–1303, Sep. 2018.

[13] Z. Wu, H. Jiang, K. Zhao, and X. Li, "An adaptive deep transfer learning method for bearing fault diagnosis," *Measurement*, vol. 151, Feb. 2020, Art. no. 107227.

[14] L. Guo, Y. Lei, S. Xing, T. Yan, and N. Li, "Deep convolutional transfer learning network: A new method for intelligent fault diagnosis of machines with unlabeled data," *IEEE Trans. Ind. Electron.*, vol. 66, no. 9, pp. 7316–7325, Sep. 2019.

[15] C. Che, H. Wang, Q. Fu, and X. Ni, "Deep transfer learning for rolling bearing fault diagnosis under variable operating conditions," *Adv. Mech. Eng.*, vol. 11, no. 12, Dec. 2019, Art. no. 168781401989721.

[16] D. Xiao, Y. Huang, C. Qin, Z. Liu, Y. Li, and C. Liu, "Transfer learning with convolutional neural networks for small sample size problem in machinery fault diagnosis," *Proc. Inst. Mech. Eng. C, J. Mech. Eng. Sci.*, vol. 233, no. 14, pp. 5131–5143, Jul. 2019.

[17] Y. Fan, S. Nowaczyk, and T. Rögnvaldsson, "Transfer learning for remaining useful life prediction based on consensus self-organizing models," *Rel. Eng. Syst. Saf.*, vol. 203, Nov. 2020, Art. no. 107098.

[18] A. Zhang *et al.*, "Transfer learning with deep recurrent neural networks for remaining useful life estimation," *Appl. Sci.*, vol. 8, no. 12, p. 2416, Nov. 2018.

[19] F. Shen, J. Xu, C. Sun, X. Chen, and R. Yan, "Transfer between multiple working conditions: A new TCCHC-based exponential semi-deterministic extended Kalman filter for bearing remaining useful life prediction," *Measurement*, vol. 142, pp. 148–162, Aug. 2019.

[20] C. Sun, M. Ma, Z. Zhao, S. Tian, R. Yan, and X. Chen, "Deep transfer learning based on sparse autoencoder for remaining useful life prediction of tool in manufacturing," *IEEE Trans. Ind. Informat.*, vol. 15, no. 4, pp. 2416–2425, Apr. 2019.

[21] P. R. D. O. da Costa, A. Akçay, Y. Zhang, and U. Kaymak, "Remaining useful lifetime prediction via deep domain adaptation," *Rel. Eng. Syst. Saf.*, vol. 195, Mar. 2020, Art. no. 106682.

[22] F. Yang, M. S. Habibullah, T. Zhang, Z. Xu, P. Lim, and S. Nadarajan, "Health index-based prognostics for remaining useful life predictions in electrical machines," *IEEE Trans. Ind. Electron.*, vol. 63, no. 4, pp. 2633–2644, Apr. 2016.

[23] S. Kumar, A. Kumar, and R. K. Jha, "A novel noise-enhanced back-propagation technique for weak signal detection in Neyman–Pearson framework," *Neural Process. Lett.*, vol. 50, no. 3, pp. 2389–2406, Dec. 2019.

[24] Z. Qiao, Y. Lei, and N. Li, "Applications of stochastic resonance to machinery fault detection: A review and tutorial," *Mech. Syst. Signal Process.*, vol. 122, pp. 502–536, May 2019.

[25] Z. Qiao, Y. Lei, J. Lin, and F. Jia, "An adaptive unsaturated bistable stochastic resonance method and its application in mechanical fault diagnosis," *Mech. Syst. Signal Process.*, vol. 84, pp. 731–746, Feb. 2017.

[26] L. Xiao, J. Tang, X. Zhang, and T. Xia, "Weak fault detection in rotating machineries by using vibrational resonance and coupled varying-stable nonlinear systems," *J. Sound Vib.*, vol. 478, Jul. 2020, Art. no. 115355.

[27] L. Xiao, R. Bajric, J. Zhao, J. Tang, and X. Zhang, "An adaptive vibrational resonance method based on cascaded varying stable-state nonlinear systems and its application in rotating machine fault detection," *Nonlinear Dyn.*, vol. 103, no. 1, pp. 715–739, Jan. 2021.

[28] V. Pavlovic, D. Schonfeld, and G. Friedman, "Stochastic noise process enhancement of hopfield neural networks," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 52, no. 4, pp. 213–217, Apr. 2005.

[29] M. Kawaguchi, H. Mino, and D. M. Durand, "Stochastic resonance can enhance information transmission in neural networks," *IEEE Trans. Biomed. Eng.*, vol. 58, no. 7, pp. 1950–1958, Jul. 2011.

[30] K. Audhkhasi, O. Osoba, and B. Kosko, "Noise-enhanced convolutional neural networks," *Neural Netw.*, vol. 78, pp. 15–23, Jun. 2016.

[31] O. Adigun and B. Kosko, "Noise-boosted bidirectional backpropagation and adversarial learning," *Neural Netw.*, vol. 120, pp. 9–31, Dec. 2019.

[32] T. Xia, Y. Song, Y. Zheng, E. Pan, and L. Xi, "An ensemble framework based on convolutional bi-directional LSTM with multiple time windows for remaining useful life estimation," *Comput. Ind.*, vol. 115, Feb. 2020, Art. no. 103182.

[33] X. Li, Q. Ding, and J.-Q. Sun, "Remaining useful life estimation in prognostics using deep convolution neural networks," *Rel. Eng. Syst. Saf.*, vol. 172, pp. 1–11, Apr. 2018.

[34] W. Yu, I. Y. Kim, and C. Mechefske, "An improved similarity-based prognostic algorithm for RUL estimation using an RNN autoencoder scheme," *Rel. Eng. Syst. Saf.*, vol. 199, Jul. 2020, Art. no. 106926.

[35] C.-G. Huang, X. Yin, H.-Z. Huang, and Y.-F. Li, "An enhanced deep learning-based fusion prognostic method for RUL prediction," *IEEE Trans. Rel.*, vol. 69, no. 3, pp. 1097–1109, Sep. 2020.

[36] M. T. Hagan, H. B. Demuth, M. H. Beale, and O. De Jesús, *Neural Network Design*. Oklahoma City, OK, USA: Martin Hagan, 2014.

[37] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[38] K. Hara, D. Saitoh, and H. Shouno, "Analysis of dropout learning regarded as ensemble learning," in *Proc. Int. Conf. Artif. Neural Netw.*, 2016, pp. 72–79.

[39] A. Saxena, K. Goebel, D. Simon, and N. Eklund, "Damage propagation modeling for aircraft engine run-to-failure simulation," in *Proc. Int. Conf. Prognostics Health Manage.*, 2008, pp. 1–9.

[40] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," 2012, *arXiv:1207.0580*. [Online]. Available: https://arxiv.org/abs/1207.0580

**Lei Xiao** was born in China, in 1988. She received the Ph.D. degree in management science and engineering from Chongqing University, Chongqing, China, in 2016.

In 2014 and 2015, she studied as a joint Ph.D. student at Rutgers University, New Brunswick, NJ, USA. From 2016 to 2018, she was a Post-Doctoral Fellow at Shanghai Jiao Tong University, Shanghai, China. Since 2018, she has been with Donghua University, Shanghai, where she is currently an Assistant Professor of mechanical engineering. Her research interests are weak-fault signal detection, remaining useful life prediction, and maintenance optimization.

**Fabing Duan** was born in China, in 1974. He received the master's degree in engineering mechanics from the China University of Mining and Technology, Beijing, China, in 1999, and the Ph.D. degree in solid mechanics from Zhejiang University, Hangzhou, China, in 2002.

From 2002 to 2003, he was a Post-Doctoral Fellow at the University of Angers, Angers, France. Since 2004, he has been with Qingdao University, Qingdao, China, where he is currently a Professor of system science. His research interests are in nonlinear systems and signal processing.

**Junxuan Tang** was born in China, in 1997. He received the bachelor's degree in mechanical engineering from Donghua University, Shanghai, China, in 2019, where he is currently pursuing the master's degree in mechanical engineering.

His research interests are in remaining useful life prediction and fault detection.

**Derek Abbott** (Fellow, IEEE) was born in South Kensington, London, U.K., in 1960. He received the B.Sc. (Hons.) degree in physics from Loughborough University, Leicestershire, U.K., in 1982, and the Ph.D. degree in electrical and electronic engineering from The University of Adelaide, Adelaide, SA, Australia, in 1997, under the supervision of K. Eshraghian and B. R. Davis.

From 1978 to 1986, he was a Research Engineer with the GEC Hirst Research Centre, London, U.K. From 1986 to 1987, he was a VLSI Design Engineer with Austek Microsystems, Australia. Since 1987, he has been with The University of Adelaide, where he is currently a full Professor with the School of Electrical and Electronic Engineering. His research interests include multidisciplinary physics and electronic engineering applied to complex systems, networks, game theory, energy policy, stochastics, and biophotonics. He coedited *Quantum Aspects of Life* (Imperial College Press, 2008) and coauthored *Stochastic Resonance* (Cambridge Univ. Press, 2008) and *Terahertz Imaging for Biomedical Applications* (Springer-Verlag, 2012).

Dr. Abbott is a Fellow of the Institute of Physics (IoP), U.K., and an Honorary Fellow of Engineers Australia. He has received a number of awards, including the Tall Poppy Award for Science in 2004, an Australian Research Council Future Fellowship in 2012, the David Dewhurst Medal in 2015, the Barry Inglis Medal in 2018, and the M. A. Sargent Medal in 2019 for eminence in engineering. He has been an Editor and/or Guest Editor for a number of journals, including the IEEE JOURNAL OF SOLID STATE CIRCUITS, *Journal of Optics B*, *Microelectronics Journal*, *PLOS ONE*, PROCEEDINGS OF THE IEEE, and the IEEE PHOTONICS JOURNAL. He is currently on the Editorial Board of IEEE ACCESS, *Nature's Scientific Reports*, *Royal Society Open Science*, and *Frontiers in Physics*. He has served on the Editorial Board of the PROCEEDINGS OF THE IEEE from 2009 to 2014 and the IEEE ACCESS since 2015. He has been serving on the IEEE Publication Services and Products Board (PSPB) since 2019.