Early Fault Detection Approach With Deep Architectures

Weining Lu, Yipeng Li, Yu Cheng, Deshan Meng, *Student Member, IEEE*, Bin Liang, and Pan Zhou

***Abstract*— Early fault detection technique is crucial to reduce the machine downtime and has high impact on a wide variety of industrial applications. However, early fault detection is still subject to the following challenges: 1) extracting features from incipient fault signals; 2) detecting anomalies with considering sequential data correlation; and 3) enhancing the reliability of fault alarm. In this paper, we introduce a novel deep-structured framework to solve the early fault detection problem. First, the system variation is measured with the deviation value generated by a current feature extraction model using deep neural network (DNN) and a distribution estimator based on the long short-term memory (LSTM) network. DNN has the ability of representing a complicated and intrinsic distribution for data, which is suitable for handling the early fault data masked by heavy noise, and LSTM is able to discover temporal dependencies in high-dimensional sequential data, which allows distribution estimator making use of previous context information as well as makes the distribution estimator more robust to warp along the time axis. Second, a circular indirect alarm assessment strategy is designed for collecting deviation values and confirming the fault appearance only when a specified confidence level is reached. Experimental results on the typical real-world bearing data sets demonstrate the effectiveness and the reliability of our model.**

***Index Terms*— Deep neural network (DNN), early fault detection, long short-term memory (LSTM).**

I. Introduction

EARLY fault detection techniques allow the prediction of fault evolution, so that maintenance actions could be taken before the fault reaches a severe level that endangers the system operational safety. Therefore, early fault detection methods have been found high impact on a wide variety of industrial applications, such as wind turbine monitoring system [1], bearing, and electrical machine condition monitoring [2], [3].

Manuscript received August 9, 2017; revised December 19, 2017; accepted December 21, 2017. Date of publication February 27, 2018; date of current version June 7, 2018. The Associate Editor coordinating the review process was Dr. Lorenzo Ciani. *(Corresponding author: Yipeng Li.)*

W. Lu is with the School of Aerospace Engineering, Tsinghua University, Beijing 100084, China (e-mail: luwn@mail.tsinghua.edu.cn).

Y. Li and B. Liang are with the Department of Automation, Tsinghua University, Beijing 100084, China (e-mail: liep@tsinghua.edu.cn; luwn517@gmail.com).

Y. Cheng is with the IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598 USA (e-mail: chengyu@us.ibm.com).

D. Meng is with Graduate School at Shenzhen, Tsinghua University, Shenzhen 518055, China (e-mail: meng.deshan@sz.tsinghua.edu.cn).

P. Zhou is with the School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: panzhou@hust.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIM.2018.2800978

The strategies that most of published works adopted could be roughly classified into two categories [4]: model driven and data driven. On the basis of a prior physical and mathematical knowledge of the system, model-driven methods complete fault detection by establishing the system degradation model [5], [6], or analyzing the fault frequencies [3], [7]–[9]. These methods can isolate early faults and describe the fault type effectively; however, it is always not an easy task to obtain the prior knowledge from the monitored system. Data-driven methods indicate early faults by performing various analysis techniques on the collected data, which are more suitable to monitor the modern large-scale industrial process. Statistical features in both time domain and frequency domain are chosen frequently to detect early fault [10]–[12]. Methods based on signal processing techniques have also received long-standing attentions of researchers, for instance, fault detection frameworks built on wavelet analysis have been proposed in [13] and [14]. Besides, techniques from other research areas are also studied to assist in separating the outliers from normal instances [15]. Some of them have been utilized to isolate the early faults, mainly including principle component analysis (PCA) [4], autoregressive model [16], artificial neural network with several hidden layers [17], [18], and support vector machine (SVM) and its extensive forms: one-class SVM, support vector data description (SVDD) [19]–[21], K-Nearest Neighbor (KNN) algorithm [22], and local outlier factor (LOF) algorithm [23], [24].

However, despite many data-driven methods have been designed, early fault detection is still subject to the following challenges: 1) extracting features from incipient fault signals; 2) detecting anomalies with considering sequential data correlation; and 3) enhancing the reliability of fault alarm.

Feature extraction is the critical step in making effective detection. Nowadays, handcrafted features, such as statistical features and wavelet-based features, gain the success in many applications, but such features are demanded for manually designing to characterize the system current signal. Meanwhile, noise presence in the full signal sequence makes normal patterns tend to be similar to the actual early faults and hence creates challenge for extracting the recognizable features. Recently, deep neural network (DNN) has brought opportunities to tackle various kinds of feature extraction challenges [25], [26]. The key ingredient to success is that deep models can automatically learn rich and discriminative features via multiple levels of representation with increasing abstraction.

Sequential data correlation, as mentioned in [27], refers to the temporal or spatial correlation in sequential observations. Normal system behavior strongly depends on the previous context and might evolve over time, in particular, an anomalous pattern is often defined as a series of behaviors that are normal individually but abnormal only collectively. In order to enhance the reliability of early fault detection method, it would be a promising research direction to focus on the full historical trajectories of sequential data. Recurrent neural networks (RNNs) and other derivative models, such as long short-term memory (LSTM) networks, have been demonstrated to be particularly useful for modeling sequences containing patterns [28], [29]. And it also has been found that LSTM is more effective in finding and modeling long-range context along a sequence while handling with time-series data.

False alarms occurrence heavily impacts the performance of detection method; therefore, a desirable fault alarm strategy is expected to declare a fault with a specified confidence level.

These challenges motivate us to propose a novel model to implement early fault detection, named as FDDA: early **F**ault **D**etection with **D**eep **A**rchitecture. Our model successfully integrates the following three key ingredients.

1) DNN with an appropriate structure is used to capture the intrinsic features contained in the original data. A transition feature pool (TFP) is constructed to reveal the recognizable state of signal at each time step, in order that these DNN-based features could be used conveniently.
2) A sequence generator is built with a modified LSTM, which is conditioned on the features drawn from the TFP. The generative sequences are considered as the expected normal behaviors of system along the time axis. Afterward, deviation values between generative sequences and current signal states in the TFP are calculated to measure the system variation.
3) A **c**ircular **i**ndirect **a**larm **a**ssessment strategy (CIAS) with four phases is designed to collect and analyze the deviation values obtained above. The fault would be confirmed while a given confidence level is reached.

Accordingly, the main contributions of this paper are summarized as follows.

1) The proposed LSTM-based sequence generator is conditioned on the DNN-based features of the original signal. Therefore, it could provide a better understanding for sequential data with long- and short-range patterns and be little affected by the background noise.
2) Training the LSTM-based sequence generator is fairly complicated since DNN and LSTM parts should be considered as a whole. Therefore, we propose a layer-wise training (LWT) method and a multiwindow fine-tuning (MFT) strategy to update parameters, which significantly helps achieving efficient and stable training.
3) In addition to enhancing the fault alarm confidence while anomalies appear in succession, CIAS realizes another two core functions, one is excluding the anomalies that are transient and recurrent with a low probability (false alarm); the other is paying attention to the normal states whose previous context is abnormal; once anomalies reappear over a period of time, CIAS could make a rapid reaction and confirm the faults as early as possible.
4) Promising results on benchmark data set for rolling bearing are used to demonstrate the effectiveness of proposed methods.

The rest of this paper is organized as follows. Section II states the preliminaries; Section III introduces the proposed methods in detail; a series of experiments is conducted in Section IV; and in Section V, the conclusion is made.

TABLE I
NOTATIONS AND DESCRIPTIONS

Notation	Description
x	sequence data with T steps or T samples, $x = (x_1, ..., x_t, ..., x_T)$, $x_t \in R^{d_x}$
h	DNN-based features from x, $h = (h_1, ..., h_t, ..., h_T)$, $h_t \in R^{d_h}$
e	features in the transition feature pool projected by h
u	output of LSTM-based sequence generator
W, b	weight matrix and bias for network connections
subscript T	matrix transpose
$*$	element-wise product operator
$\sigma(x)$	sigmoid function, $\sigma(x) = 1/(1 + e^{-x})$

II. PRELIMINARIES

In this section, notations, which are used in this paper frequently, are summarized in Table I. Afterward, we brief structures of DNN and standard LSTM.

A. Deep Neural Network

DNN is a kind of artificial neural network, which is constructed with a desirable complex architecture using deep learning technique [25]. The core idea is to train one layer with an unsupervised representation learning algorithm at a time, and then use the output of this layer as the input for the next layer. This process can be executed iteratively until the desired number of layers is reached. In this paper, the autoencoder (AE) is chosen as the basic single-layer representation model for stacking the deep architecture [30].

The specialty of AE is that its output is equal to the input. From a functional standpoint, an AE consists of two parts: one part is the encoder for extracting features from the input data, denoted as $h = f_e(a_h)$ and $a_h = W_{xh}x + b_{xh}$; the other part is the decoder for reconstructing the input data from the extracted features, denoted as $r = f_d(a_r)$ and $a_r = W_{hr}h + b_{hr}$. And $f_e(a_h) = \sigma(a_h)$ and $f_d(a_r) = a_r$ are activation functions for AE.

The goal of AE is to minimize the reconstruction error; therefore, the loss function for AE is written as

$$\mathcal{L}_{\text{AE}} = \frac{1}{2T}\|r - x\|_F^2 \tag{1}$$

where $\|\|_F$ denotes *Frobenius norm*. The gradient descent algorithm is used to determine the parameter set $\Theta_{\text{AE}} = \{W_{xh}, b_{xh}, W_{hr}, b_{hr}\}$ for building the model. After training, the reconstruction layer of AE is removed, and the feature layer is used as the input to the next AE.

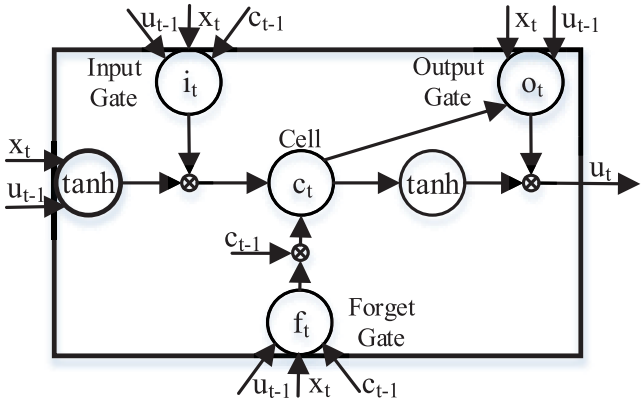


Fig. 1. Structure of LSTM cell.

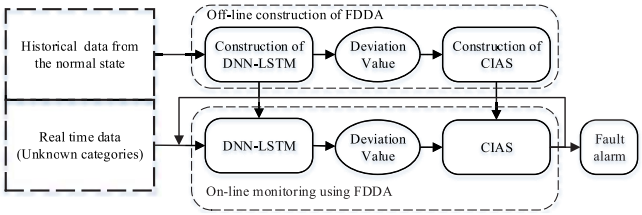


Fig. 2. Usage of the proposed FDDA.

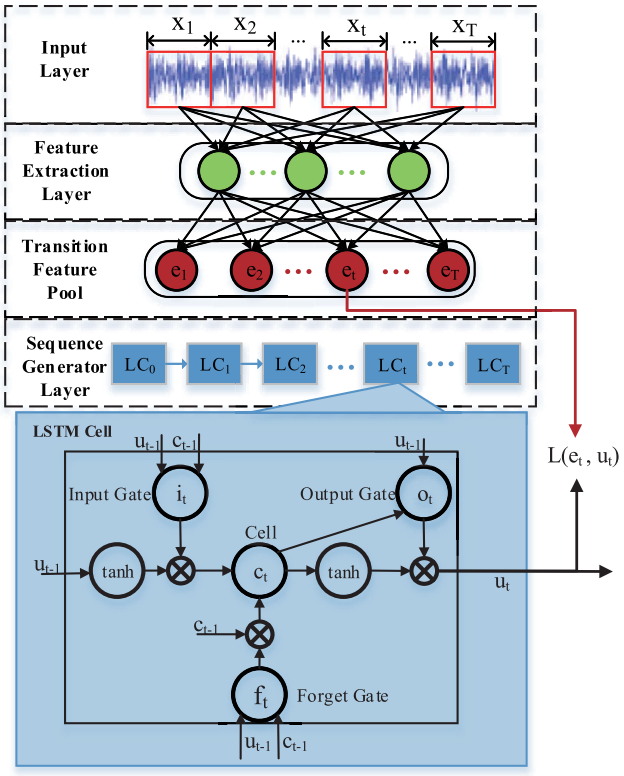


Fig. 3. Framework of DNN–LSTM.

B. Long Short-Term Memory Network

LSTM is a kind of RNNs that computes a nonlinear transformation to convert an input pattern into an output pattern; the structure of LSTM enables the networks to perform sequence reproduction or prediction. Fig. 1 shows a typical structure of a LSTM cell at time t. In order to obtain the output u_t (hidden vector), the input x_t is operated by learning gate functions (input gate, forget gate, and output gate) that determine the input information to be kept or discarded for the output. LSTM cell is defined by the following composite functions:

$$i_t = \sigma(W_{xi}x_t + W_{ui}u_{t-1} + W_{ci}c_{t-1} + b_i) \quad (2)$$

$$f_t = \sigma(W_{xf}x_t + W_{uf}u_{t-1} + W_{cf}c_{t-1} + b_f) \quad (3)$$

$$c_t = f_t \times c_{t-1} + i_t \times \tanh(W_{xc}x_t + W_{uc}u_{t-1} + b_c) \quad (4)$$

$$o_t = \sigma(W_{xo}x_t + W_{uo}u_{t-1} + W_{co}c_t + b_o) \quad (5)$$

$$u_t = o_t \times \tanh(c_t) \quad (6)$$

where i, f, and o are the input gate, forget gate, and output gate, respectively. The values of these functional gates are in the range of 0–1. c denotes the cell activation vectors that store information over different time ranges. W represents the projection matrix between various modules; for instance, W_{xf} denotes the input and forget gate connection. Further details about LSTM could be referred to [28] and [29].

III. Proposed Methods

The usage of our proposed FDDA is shown in Fig. 2, which includes two stages: off-line construction and online monitoring. In the first stage, historical data from normal state are collected to construct the DNN-based feature extractor and the corresponding LSTM-based sequence generator, for simplicity, we denote this section as DNN–LSTM. Afterward, deviation values generated from DNN–LSTM are used to implement the CIAS part. The second stage is online condition monitoring with the real-time input. In this section, FDDA would be described in detail, mainly including the structures annd construction algorithms of DNN–LSTM and CIAS. Considering that the multilayer DNN and LSTM could be constructed easily by stacking single-layer model, hence single-layer structure DNN and LSTM are considered for clarity at first.

A. DNN–LSTM

1) Model Structure: DNN–LSTM can be understood as an LSTM-based sequence generator whose parameters are conditioned on the DNN-based features. Fig. 3 shows the graphical structure of DNN–LSTM, which consists of three main functional modules, feature extraction layer, TFP, and sequence generator layer, respectively. Feature extraction layer is constructed with DNN to obtain the intrinsic features from the input data. TFP is designed to make a bridge between feature extraction layer and sequence generator layer. And sequence generator layer is built with the LSTM network, which is responsible for outputting the expected normal behaviors of system along the time axis; the LSTM cell used in this paper is displayed in the blue box in Fig. 3. Specifically, for each time t, the difference between generative sequence u_t and features stored in TFP e_t is used to represent the system variation. More details about DNN–LSTM will be introduced in the following.

First, DNN with parameter set $\Theta_{\text{DNN}} = \{W_{xh}, b_{xh}\}$ is used to extract features h from original data x.

As mentioned above, TFP is the bridge between DNN and LSTM. In our proposed framework, outputs of the sequence generator are needed to approach the features from the original data, in order that the sequence generator could convey the information contained in data. However, if we set feature vectors h as the expected output of sequence generator directly, the output dimensionality of sequence generator would be fixed to the hidden layer size of DNN, which constrains the flexibility of the model. The projection way of TFP is defined as follows:

$$e = \tanh(a_e), \quad a_e = W_{he}h + b_{he} \tag{7}$$

where $e = \{e_1, e_2, \ldots, e_t, \ldots, e_T\}$ and e_t is transition feature vector for h_t, $e_t \in R^{d_e}$, d_e is the dimensionality of TFP, which is the same as that of the output layer size of sequence generator. The TFP parameter set is $\Theta_{\text{TFP}} = \{W_{he}, b_{he}\}$. And e is the expected output of sequence generator.

In the sequence generator part, we modify the standard LSTM to realize the function. In particular, at time t, we consider that the LSTM cell (LC_t for short) is only connected to its internal predecessor u_{t-1} and c_{t-1} without receiving external input x_t (as shown in Fig. 3). With this compact form, the sequence generator could be regarded as a pure normal-state predictor, which can display a significant deviation while anomaly appearing. The terms are rewritten as

$$i_t = \sigma(a_{i_t}), \quad a_{i_t} = W_{ui}u_{t-1} + W_{ci}c_{t-1} + b_i \tag{8}$$
$$f_t = \sigma(a_{f_t}), \quad a_{f_t} = W_{uf}u_{t-1} + W_{cf}c_{t-1} + b_f \tag{9}$$
$$c_t = f_t \times c_{t-1} + i_t \times \tanh(a_{c_t}), a_{c_t} = W_{uc}u_{t-1} + b_c \tag{10}$$
$$o_t = \sigma(a_{o_t}), \quad a_{o_t} = W_{uo}u_{t-1} + W_{co}c_t + b_o \tag{11}$$
$$u_t = o_t \times \tanh(c_t) \tag{12}$$

where u_t and c_t have the same size as the transition feature vector e_t. The parameter set of sequence generator part is $\Theta_{\text{LSTM}} = \{W_{uc}, W_{ui}, W_{uf}, W_{uo}, W_{ci}, W_{cf}, W_{co}, b_c, b_i, b_f, b_o\}$. And sequence generator is constructed by minimizing the loss function as follows:

$$\mathcal{L}_{\text{LSTM}} = \frac{1}{2}\sum_{t=1}^{T} \|u_t - e_t\|^2. \tag{13}$$

After completing the construction of DNN–LSTM (the details of training algorithm will be stated in Section III-A2), we proceed to calculate the deviation value (denoted as $\delta_{\text{D–L}}$) as follows:

$$\delta_{\text{D–L}} = \sum_{t=1}^{T} \|u_t - e_t\|^2. \tag{14}$$

2) Training Algorithm: The goal of training DNN–LSTM structure is to determine the parameter set: $\Theta_{\text{DNN-LSTM}} = \{\Theta_{\text{DNN}}, \Theta_{\text{TFP}}, \Theta_{\text{LSTM}}\}$. It is difficult to have a global objective function for the whole DNN–LSTM structure. Instead, we proposed a LWT procedure and an MFT method to update the parameters.

a) Layer-wise training method (LWT): LWT method first trains the DNN part individually and then trains the LSTM by fixing the parameters of DNN. The detailed algorithm is described as follows.

Step 1: Build the DNN part with regular procedures: initialize the AE parameter set $\{W_{xh}, b_{xh}, W_{hr}, b_{hr}\}$ with small random values near 0, and then train it by minimizing the loss function stated in (1) using gradient descent algorithm. The gradients of each parameter are written as

$$\frac{\partial \mathcal{L}_{ae}}{\partial W_{xh}} = \delta_{xh}x^{\text{T}}, \quad \frac{\partial \mathcal{L}_{ae}}{\partial b_{xh}} = \delta_{xh} \tag{15}$$
$$\frac{\partial \mathcal{L}_{ae}}{\partial W_{hr}} = \delta_{hr}h^{\text{T}}, \quad \frac{\partial \mathcal{L}_{ae}}{\partial b_{hr}} = \delta_{hr} \tag{16}$$

where $\delta_{hr} = -(x - r) \times (\partial f_d(z_d)/\partial z_d)$, and $\delta_{xh} = W_{hr}^{\text{T}}\delta_{output} \times (\partial f_e(z_e)/\partial z_e)$. Remove the reconstruction layer of AE to obtain DNN structure and $\Theta_{\text{DNN}} = \{W_{xh}, b_{xh}\}$.

Step 2: Generate the hidden units h using the well-trained DNN.

Step 3: Initialize the parameters of Θ_{TFP} and Θ_{LSTM} with small random values and obtain e with (7). Then, we minimize the loss function in (13) using gradient descent algorithm and backpropagation through time. For each time step t, the gradients with respect to parameters are computed by following the rules:

$$\frac{\partial \mathcal{L}_{\text{LSTM}}}{\partial W_{ui}^{(t)}} = \delta_i^{(t)}u_{t-1}^{\text{T}}, \quad \frac{\partial \mathcal{L}_{\text{LSTM}}}{\partial W_{ci}^{(t)}} = \delta_i^{(t)}c_{t-1}^{\text{T}}, \frac{\partial \mathcal{L}_{\text{LSTM}}}{\partial b_i^t} = \delta_i^{(t)} \tag{17}$$
$$\frac{\partial \mathcal{L}_{\text{LSTM}}}{\partial W_{uf}^{(t)}} = \delta_f^{(t)}u_{t-1}^{\text{T}}, \quad \frac{\partial \mathcal{L}_{\text{LSTM}}}{\partial W_{cf}^{(t)}} = \delta_f^{(t)}c_{t-1}^{\text{T}}, \frac{\partial \mathcal{L}_{\text{LSTM}}}{\partial b_f^t} = \delta_f^{(t)} \tag{18}$$
$$\frac{\partial \mathcal{L}_{\text{LSTM}}}{\partial W_{uo}^{(t)}} = \delta_o^{(t)}u_{t-1}^{\text{T}}, \quad \frac{\partial \mathcal{L}_{\text{LSTM}}}{\partial W_{co}^{(t)}} = \delta_o^{(t)}c_{t-1}^{\text{T}}, \frac{\partial \mathcal{L}_{\text{LSTM}}}{\partial b_o^t} = \delta_o^{(t)} \tag{19}$$
$$\frac{\partial \mathcal{L}_{\text{LSTM}}}{\partial W_{uc}^{(t)}} = \delta_c^{(t)}u_{t-1}^{\text{T}}, \quad \frac{\partial \mathcal{L}_{\text{LSTM}}}{\partial b_c^t} = \delta_c^{(t)} \tag{20}$$
$$\frac{\partial \mathcal{L}_{\text{LSTM}}}{\partial W_{he}^{(t)}} = \delta_{he}^{(t)}h_t^{\text{T}} \quad \frac{\partial \mathcal{L}_{\text{LSTM}}}{\partial b_{he}^{(t)}} = \delta_{he} \tag{21}$$

δ and ε terms are generated in the following formulas:

$$\varepsilon^{(t)} = u_t - e_t \tag{22}$$
$$\delta_i^{(t)} = \varepsilon_c^{(t)} \times \tanh(a_{c_t}) \times \sigma'(a_{i_t}) \tag{23}$$
$$\delta_f^{(t)} = \varepsilon_c^{(t)} \times c_{t-1} \times \sigma'(a_{f_t}) \tag{24}$$
$$\delta_o^{(t)} = \varepsilon^{(t)} \times \tanh(c_t) \times \sigma'(a_{o_t}) \tag{25}$$
$$\delta_c^{(t)} = \varepsilon_c^{(t)} \times i_t \times \tanh'(a_{c_t}) \tag{26}$$
$$\delta_{he} = -\varepsilon^{(t)} \times \tanh'\left(a_e^{(t)}\right) \tag{27}$$

where $\varepsilon_c^{(t)} = o_t \times \tanh'(c_t) \times \varepsilon^{(t)} + f_{t+1}\varepsilon_c^{(t+1)} + W_{ci}^{\text{T}}\delta_i^{(t+1)} + W_{cf}^{\text{T}}\delta_f^{(t+1)} + W_{co}^{\text{T}}\delta_o^{(t)}$ and $\tanh'(z) = 1 - \tanh(z)^2$, $\sigma'(z) = \sigma(z)(1 - \sigma(z))$. In particular, c_0 is set to 0 and u_0 is fixed with e_1.

b) Multiwindow fine-tuning (MFT) method: The goal of fine-tuning method is to make the features extracted by

DNN more appropriate for constructing the recurrent relation according to the sequential information. We denote M window sequential data as $[x^{(m)}]_{m=1}^{M}$, $x^{(m)} = \{x_1^{(m)}, \ldots, x_t^{(m)}, \ldots, x_T^{(m)}\}$. During the fine-tuning stage, M window data are treated as a whole to update the parameter set $\Theta_{\text{DNN-LSTM}}$. In order to find more details about the discrepancy between feature extraction part and sequence generator part, we want to integrate the loss from each time window in a product way. Therefore, instead of (13), we define the cost function for fine-tuning (ft) stage over the M window sequence

$$\mathcal{L}_{ft} = \exp\left(\sum_{m=1}^{M} \|u^{(m)} - e^{(m)}\|^2/2\right) \tag{28}$$

where $u^{(m)}$ and $e^{(m)}$ are outputs of sequence generator and transition feature vector at m time window, respectively. The usage of exponential function makes gradient calculation convenient to realize. For each M windows set, parameters gradients at time step t are described as follows:

DNN Part:

$$\frac{\partial \mathcal{L}_{ft}}{\partial W_{xh}^{(t)}} = \left[\delta_{ft,xh}^{(m,t)}\left(x_t^{(m)}\right)^{\mathrm{T}}\right]_{m=1}^{M}, \quad \frac{\partial \mathcal{L}_{ft}}{\partial b_{he}^{(t)}} = \left[\delta_{ft,xh}^{(m,t)}\right]_{m=1}^{M}. \tag{29}$$

TFP Part:

$$\frac{\partial \mathcal{L}_{ft}}{\partial W_{he}^{(t)}} = \left[\delta_{ft,he}^{(m,t)}\left(h_t^{(m)}\right)^{\mathrm{T}}\right]_{m=1}^{M}, \quad \frac{\partial L_{ft}}{\partial b_{he}^{(t)}} = \left[\delta_{ft,he}^{(m,t)}\right]_{m=1}^{M} \tag{30}$$

where

$$\varepsilon_{ft}^{(t)} = \mathcal{L}_{ft}\sum_{m=1}^{M}\left(u^{(m)} - e^{(m)}\right) \tag{31}$$

$$\delta_{ft,he}^{(m,t)} = -\varepsilon_{ft}^{(t)} \times \tanh'\left(a_e^{(m,t)}\right) \tag{32}$$

$$\delta_{ft,xh}^{(m,t)} = \left(W_{he}\right)^{\mathrm{T}}\delta_{ft,he}^{(m,t)} \times \sigma'\left(a_r^{(m,t)}\right). \tag{33}$$

LSTM Part: The gradients of LSTM part for fine-tuning stage are similar to those used in the LWT method, (17)–(20). We just need to replace $\varepsilon^{(t)}$ in (22) with $\varepsilon_{ft}^{(t)}$ in (31). And then concatenate the gradient term window by window, which is same as the way listed in (29) and (30). Due to the page limit, we would not list these calculations in this section.

c) Stop criterion: The training process of DNN–LSTM construction or DNN–LSTM fine tuning will not terminate until the loss function value goes lower than the preset threshold or training epochs exceed the preset times.

d) Whole training strategy: In the training stage, LWT and MFT methods would be used interchangeably to build the DNN–LSTM model. The procedures are shown in Algorithm 1.

Algorithm 1 Training Strategy for DNN-LSTM

Input: N sequence data $\{x^{(n)}\}_{n=1}^{N}$; window count: w_count = 1; window number for fine-tuning in one epoch: M.
Ouput: $\Theta_{\text{DNN-LSTM}}$=$\{\Theta_{\text{DNN}}, \Theta_{\text{TFP}}, \Theta_{\text{LSTM}}\}$
Execute Step 1 of LWT to build DNN with $\{x^{(n)}\}_{n=1}^{N}$
Execute Step 2 of LWT to generate feature vectors $\{h^{(n)}\}_{n=1}^{N}$ using $\{x^{(n)}\}_{n=1}^{N}$
while (w_count $\leq$ N)
  set input as $x^{(\text{w\_count})}$
  if (w_count can be divided exactly by M)
    while (not meet stop criterion of fine-tuning stage)
      Execute MFT method to update $\Theta_{\text{DNN-LSTM}}$
  else
    Execute Step 3 of LWT to update Θ_{TFP} and Θ_{LSTM}
    w_count = w_count + 1

B. CIAS

CIAS is a novel general empirical strategy for declaring a timely fault alarm with as less false alarms as possible. In this paper, CIAS is designed by obeying the following requirements.

1) The fault alarm confidence level could be enhanced while anomalies appear in succession.
2) The anomalies, which are transient and does not reproduce within a certain time interval, are refused to consider as the fault indication.
3) Making fault alarm judgment should be taken into account of previous state information; for instance, a normal input with the abnormal previous state should be thought as a more likely fault indication than that with the normal previous state.

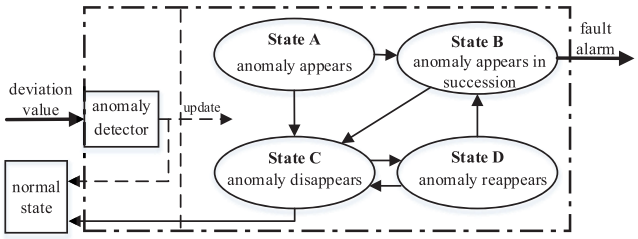


Fig. 4. Framework of CIAS.

1) Structure of CIAS: On the basis of the above factors, CIAS consists of one anomaly detector and four state cells, which is displayed in Fig. 4.

Anomaly detector module is used to receive and determine whether $\delta_{\text{D-L}}$ [calculated with (14)] is an anomaly or not. And state cells represent the various stages while evaluating the fault alarm confidence level. In particular, **State A** indicates the beginning of anomaly appearance; **State B** represents the situation that anomalies appears in succession, in this stage, the fault alarm confidence level keeps enhancing; **State C** is the stage in which anomalies disappear while their previous contexts are abnormal, and the fault alarm confidence level starts to reduce; and **State D** represents the stage in which anomalies reappear and the fault alarm confidence level restarts to raise on the basis of residual confidence level of **State C**. **State A**, **State B**, and **State C** make CIAS meet the demands of the first two requirements, and **State C** and **State D** realize the function for the last requirement term.

The connections of various modules in CIAS also could be observed in Fig. 4. The specific switching conditions are listed as follows.

1) *Anomaly Detector to Normal State:* $\delta_{\mathrm{D-L}}$ is in the normal range, meanwhile its previous state is normal.
2) *State A/State D to State B:* Anomalies appear in succession.
3) *State A/State B to State C:* $\delta_{\mathrm{D-L}}$ performs normal whose previous state is abnormal.
4) *State C to State D:* $\delta_{\mathrm{D-L}}$ is out of normal range, meanwhile its previous context in a certain time interval contains both normal and abnormal information.
5) *State C to Normal State:* The score in **State C** is lower than a predefined threshold.

2) Formulation of CIAS: In this section, we introduce the construction methods for modules in CIAS. In particular, we calculate scores to evaluate fault alarm confidence level in each state cell, denoted as Score_A, Score_B, Score_C, and Score_D; high score means high fault alarm confidence level.

a) Anomaly detector: Detecting the outlier from the normal sequence data is typically one-class problem; therefore, we choose SVDD method [21] to construct a classifier for detecting anomalies. The training set contains the samples from the normal state only.

State A: Factors that mainly correlate with Score_A include: 1) $\delta_{\mathrm{D-L}}$ and baseline deviation value δ_{base} obtained from the normal samples and 2) how many times have **s**uccessive **a**nomalies (N_{sa}) occurred ($N_{sa} = 1$ in **State A**). Therefore, the formulation of Score_A is designed as follows:

$$\mathrm{Score}_A = \sigma(\delta_{\mathrm{D-L}}/\delta_{\mathrm{base}}) \exp((N_{sa} - N_{\mathrm{thre}})/\Delta_1) \quad (34)$$

where $\delta_{\mathrm{D-L}}/\delta_{\mathrm{base}}$ evaluates the variation between two deviation values, meanwhile it is a dimensionless term which is reasonable to compute the score; δ_{base} is the average value of δ_{D-L} of normal data. The value of $\sigma()$ is limited in the range of 0–1, therefore it could prevent the situation that single anomaly with large value reaches the fault alarm level directly; in the second term, the characteristics of exponential function $\exp(z)$ are suitable for meeting our demands, specifically, if $z < 0$, the value of $\exp(z)$ raises with a relative slow pace, and if $z > 0$, the $\exp(z)$ value would increase more rapidly, it means that anomalies that appear in succession and exceed certain times are more likely to be a fault indication, which is an appropriate assumption. N_{thre} is the predefined turning point and Δ_1 controls the variation rate.

State B: The calculation method for Score_B is similar to that used for Score_A, the specific form is written as follows:

$$\mathrm{Score}_B = \sum_{n_{\mathrm{sa}}=2}^{N_{\mathrm{sa}}} \sigma\left(\frac{\delta_{\mathrm{D-L}}^{(n_{\mathrm{sa}})}}{\delta_{\mathrm{base}}^{(n_{\mathrm{sa}})}}\right) \exp\left(\frac{n_{\mathrm{sa}} - N_{\mathrm{thre}}}{\Delta_1}\right) + \mathrm{Score}_{\mathrm{input}} \quad (35)$$

where the subscript n_{sa} denotes the state of n_{sa}th successive anomaly and $\mathrm{Score}_{\mathrm{input}}$ is the residual score coming from State_A or State_D.

State C: The fault alarm confidence level keeps decreasing since the anomaly disappears

$$\mathrm{Score}_C = [1 - \sigma(n_{\mathrm{ms}}/\Delta_2)]\mathrm{Score}_{\mathrm{input}} \quad (36)$$

where n_{ms} is the number of **m**onitored **s**amples since the last anomaly occurred, and Δ_2 controls the decreasing rate. And the specific form of $\mathrm{Socre}_{\mathrm{input}}$ would be Score_A, Score_B, or Score_D.

State D: Score_D consists of two sources: the score brought by emerging anomaly and the residual score from **State C**. The calculation form is designed as

$$\mathrm{Score}_D = \sigma\left(\frac{\delta_{\mathrm{D-L}}}{\delta_{\mathrm{base}}}\right) \exp\left(\frac{N_{\mathrm{sa}} - N_{\mathrm{thre}}}{\Delta_1}\right) + \mathrm{Score}_C \quad (37)$$

where $N_{\mathrm{sa}} = 1$.

3) Fault Alarm Condition of CIAS: The fault alarm condition of CIAS is as follows:

$$\mathrm{Score}_B > \mathrm{Score}_{\mathrm{normal}} \quad (38)$$

where

$$\mathrm{Score}_{\mathrm{normal}} = N_{\mathrm{thre}} > \sum_{n_{\mathrm{sa}}=1}^{N_{\mathrm{thre}}} \exp\left(\frac{n_{\mathrm{sa}} - N_{\mathrm{thre}}}{\Delta_1}\right) > \sum_{n_{\mathrm{sa}}=1}^{N_{\mathrm{thre}}} \sigma\left(\frac{\delta_{\mathrm{D-L}}^{(n_{\mathrm{sa}})}}{\delta_{\mathrm{base}}^{(n_{\mathrm{sa}})}}\right) \exp\left(\frac{n_{\mathrm{sa}} - N_{\mathrm{thre}}}{\Delta_1}\right). \quad (39)$$

IV. Experiment Results and Analysis

A. Data Preparation

To investigate the effectiveness of our proposed FDDA approach for early fault detection, a bearing faulty data set and a run-to-failure bearing data set are considered in this paper.

1) Rolling Bearing Fault Data Set: The data set was acquired from the bearing data center of the Case Western Reserve University (CWRU) [31]. The test rig consists of two SKF rolling bearings and a motor shaft. Data points were collected using accelerometers mounted at the drive end of an induction (Reliance Electric IQPreAlert motor). Bearings were seeded with faults using electrodischarge machining. The faulty data set consists of four categories: normal state (N), and fault in the inner race (FI), the outer race (FO), and the ball (FB), respectively. Moreover, for the same fault type, there are four fault sizes corresponding to the various fault degrees, 0.007, 0.014, 0.021, and 0.028 in, respectively.

2) Run-to-Failure Bearing Data set: The experimental data set was acquired from intelligent maintenance system (IMS) of University of Cincinnati [32]. The bearing test rig hosts four bearings on one shaft, which is driven by an ac motor (shown in Fig. 5). The rotation speed kept 2000 r/min. And a radial load of 6000 lbs is added to the shaft and bearing by a spring mechanism. An oil circulation system measures the flow and the temperature of the lubricant. Besides, a magnetic plug installed in the oil feedback pipe collects debris from the oil as evidence of bearing system degradation. The system will stop while the accumulated debris adhered to the magnetic plug exceeds a certain level. A high sensitivity accelerometer was installed on each bearing housing. Vibration data were recorded at 10-min intervals. The data sampling rate is 20 kHz and each vibration signal snapshot length contains 20 480 points. At the end of the test-to-failure experiment, outer race failure occurred in bearing 1 and 984 snapshots was collected. And first 500 snapshots represent system normal state.

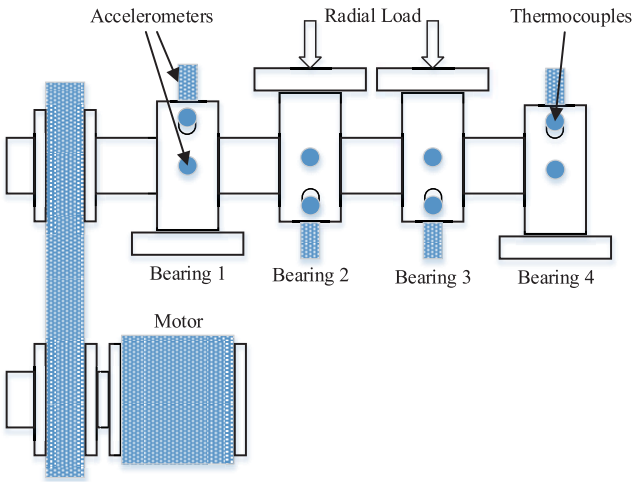


Fig. 5. Testbed for IMS bearing data set.

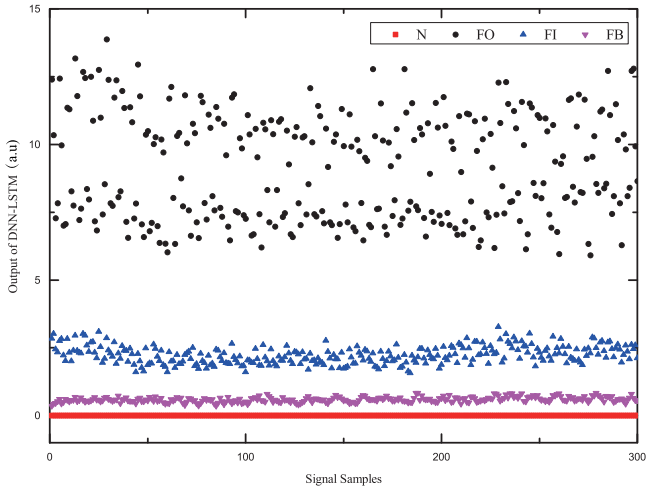


Fig. 6. DNN–LSTM output for various categories rolling bearing samples.

3) Preprocessing Procedures: At the beginning of experiment, several preprocessing procedures need to be completed on the data sets.

1) Samples of 500 points without overlap are selected from both CWRU and IMS data sets. According to data acquisition methods, ten successive sequential samples from CWRU data set are regarded as the one-window input. And 40 sequential samples from the same signal snapshot of IMS data set are considered as the one-window input.
2) *Fast Fourier Transform* is applied to every signal sample to obtain the corresponding frequency amplitude.
3) Take the frequency amplitude calculated in the last step as the final input for the model. The dimension of input is 500.

B. Results on Rolling Bearing Fault Data Set

In this section, we choose samples of CWRU data set to evaluate performance of the proposed algorithm while describing the system variation. As mentioned in Section IV-A, there are four bearing states contained in the CWRU data set. Naturally, data of normal state represent normal behaviors of the system. And other three states (FI, FO, and FB) are abnormal conditions of the system. We utilize part of normal samples as training data to construct DNN–LSTM structure, which owns the ability of measuring the system variation. Afterward, data from all conditions (N, FI, FO, and FB) are input to the well-trained DNN–LSTM, so that we could observe the system variation by checking the output $\delta_{\mathrm{D-L}}$ of DNN–LSTM. Specifically, 300 samples are selected in each bearing conditions. $\delta_{\mathrm{D-L}}$ for these samples are shown in Fig. 6.

The following two facts are concluded from Fig. 6.

1) We make a clear separation of normal samples (red points) and abnormal samples (black, blue, and magenta points) by using $\delta_{\mathrm{D-L}}$ as signals feature. It proves that our proposed method can reflect the system variation effectively.
2) Three kinds of abnormal conditions (FO, FI, and FB) show obvious difference in Fig. 6 as well. It means that the proposed method is sensitive to the system variation, which could be used to tell the unknown category fault in an unsupervised way in the future.

C. Results on Run-to-Failure Bearing Data Set

The content of this section is designed to verify the effectiveness of FDDA model. In order to construct the FDDA model, we split IMS bearing data set into two parts, one part consists of samples drawn from the first 400 snapshots, and the other part contains samples drawn from the left snapshots. The first part is used as the training data, in order that only normal-state signals are utilized for training. Naturally, the second part consisting of normal and abnormal states used for online monitoring test.

1) Comparison Methods: In order to verify the effectiveness of FDDA model, the proposed method is compared with several early fault detection and anomaly detection methods as follows:

1) KNN;
2) LOF [23];
3) one-class SVM [20];
4) root mean square (RMS) + SVDD;
5) Kurtosis factor + SVDD;
6) PCA + SVDD;
7) DNN + SVDD;
8) FDDA removing the TFP, LSTM, and CIAS parts (FDDA-S1) + SVDD;
9) FDDA without the DNN part (FDDA-S2) + SVDD;
10) FDDA without fine-tuning stage (FDDA-S3);
11) one-class CRF (OCCRF) [33];
12) DSEBM-e [34];
13) UODA [35].

Methods 1–3 are classic anomaly detection methods, and methods 4–10 are detection methods which contain feature extraction stage and anomaly detection stage. We choose SVDD to realize anomaly detection task in 4–9 for staying

consistent with FDDA. In particular, method 7 simply utilizes DNN to extract features and SVDD to detect outliers, the feature extraction process in method 7 is unsupervised. Methods 8–10 could be considered as the simplified versions of FDDA; method 8 is constructed by only keeping DNN part from the well-trained FDDA, this DNN is fine tuned for extracting task-related features which is different from method 7. Method 9 remove the DNN part, and uses the original data as the direct input to compare with the expected system behavior. Method 10 builds the FDDA model without using fine-tuning method.

Methods 11–13 are three effective anomaly detection methods used for sequential data. OCCRF learns from a one-class data set and captures the temporal dependence structure using CRFs; DSEBM-e is the deep generative model with energy function for anomaly detection. The idea of UODA is the closest to ours, it consists of two parts, one is for feature extraction and the other is for system behavior estimator; however, its estimator is built with RNN, whose learning ability for long-term pattern is weaker than LSTM; meanwhile, its model structure is simpler than ours which is lack of extendibility. The effectiveness of all these three methods has been verified in various nonsequential and sequential data application scenarios, such as audiovisual anomaly recognition, body-and-hand gestures recognition, and video event detection. Several results generated by them still keep the state-of-the-art performance.

2) Implementation Details: While using the KNN technique, we sum the distances between each normal data and its KNNs, and then choose the maximum one as the threshold for judging the anomaly. For the LOF technique, we calculate the LOF value for each normal data and choose the maximum one as the threshold for excluding the anomaly as well. For methods SVDD and one-class SVM, we could directly utilize the normal-state data or features to construct a classification hyperplane.

For making a relative fair comparison, the fault declaration condition for these comparison methods are set as: the number of successively anomalies exceeds the N_{thre}, which is weaker than that for FDDA [shown in (38)]. Besides, comparison methods are evaluated by empirically searching the parameter space for the optimal parameter settings and choosing the best result.

We use the LIBSVM package introduced in [36] to implement one-class SVM and SVDD. Besides, the structures of DNN and LSTM used in methods 7–10 are the same as those in the DNN–LSTM part of FDDA. The optimal subspace dimension for PCA is set by searching {4, 8, 16, 32, 64, 128}. For KNN, the optimal number of nearest neighbor is selected by searching {1, 2, 4, 8, 16, 32, 64}.

We implemented OCCRF following the procedures listed in [33]. The version of DSEBM-e model used in this section is designed for sequential data, which is offered by the authors. Besides, the hidden layer size of DNN and RNN part of UODA keep the same with those of DNN–LSTM.

For FDDA, we propose single-layer structure DNN and LSTM, and the specific values for each parameter are listed in Table II.

TABLE II
PARAMETERS FOR FDDA

Parameter	Value	Parameter	Value
DNN hidden layer size	1000	LSTM hidden layer size	10
N_{thre}	10	M	5
Δ_1	10	Δ_2	100

TABLE III
EARLY FAULT DETECTION RESULTS

Method	Result	Method	Result
KNN	563	LOF	626
One-class SVM	609	RMS+SVDD	578
Kurtosis+SVDD	708	PCA+SVDD	574
DNN+SVDD	566	FDDA-S1	549
FDDA-S2	545	FDDA-S3	546
OCCRF	557	DSEBM-e	530
UODA	532	FDDA	**527**

3) Evaluation Metric: We evaluate the effectiveness of the different methods by comparing the location of signal snapshot where the fault alarm appears.

4) Early Fault Detection Results on the Data Set: The fault alarm locations calculated by various methods are detailed in Table III. According to the results, we observe that FDDA could declare the fault alarm at 527 signal snapshot (about 88 h since the test began), this result outperforms other listed methods. For instance, the appearance of early fault with FDDA is 36 snapshots earlier (about 6 h) than that of the best method (KNN) in 1–6.

Besides, the performance of the DNN-related model and three simplified versions of FDDA are worthy of attentions. The DNN + SVDD model gives alarm about 6.5 h later than FDDA, this situation could be explained as follows: the feature extraction process of DNN is unsupervised, thus features might contain much irrelevant information to the early fault detection task while no task-related objective function is added. These redundant features enhance the difficulty of recognizing the early fault. By contrast, FDDA-S1 (DNN with fine tuned) obtains a better result, but it still makes a later alarm decision than FDDA does. It is because the comparison between features and system expected behavior could zoom the anomaly characteristics, which is helpful for detecting the early tiny anomaly. The FDDA-S2 model identifies the fault almost 3 h later than that of FDDA, this result shows the importance of DNN part in FDDA, which proves that extracted features are more effective for recognizing the early fault than original data. Performance comparison between FDDA-S3 and FDDA indicates that the fine-tuning procedures could truly boost the performance by strengthening the temporal dependence into the model.

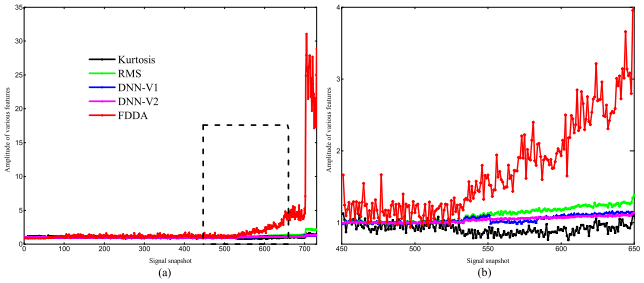


Fig. 7. Comparison of various features. (a) Development process between signal snapshot 1 to 730. (b) Local enlargement of various features between signal snapshot 450 and 650.

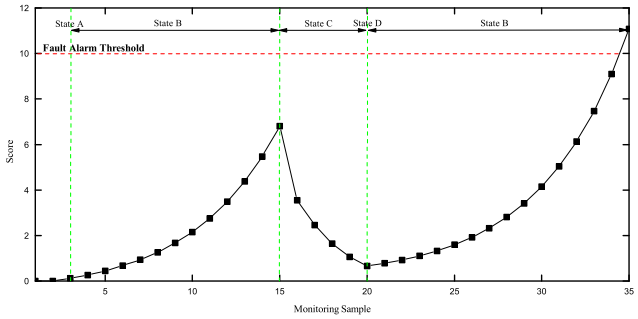


Fig. 8. Response Curve of CIAS for the test situation.

OCCRF makes an alarm declaration at 557 signal snapshot, which is 30 snapshots later than that of FDDA. OCCRF aims to learn the latent model contained in the data; however, it cannot hold the sensitivity to the incipient change of system which might be regarded as the unexpected disturbance caused by noise. DSEBM-e obtains a satisfactory result, which is only three snapshots later than that of FDDA. UODA also performs well, which locates fault appearance at 532 signal snapshot. The result of comparison between UODA and FDDA indicates that our proposed method is more sensitive to the system variation.

These results are obtained from the benchmark data set of early fault detection research under a relatively fair experiment condition, hence it can verify that FDDA is effective for solving the early fault detection problems.

D. Empirical Analysis

In this section, we would further illustrate the effectiveness of FDDA.

1) Features Extracted by FDDA: $\delta_{\mathrm{D-L}}$ calculated by FDDA is the output feature of monitoring signals, which can be used to visualize the damage development process. Furthermore, another four features are adopted to reveal this process for the comparison purpose, namely, RMS, Kurtosis factor, and DNN features in method 7 (DNN-V1) and DNN features in method 8 (DNN-V2).

First, signal snapshots from 1 to 730 are chosen, which cover the process from normal operation to fault appearance. For each signal snapshot, we utilize five feature extraction methods above (FDDA, RMS, Kurtosis, DNN-V1, and DNN-V2) to obtain the corresponding feature values. Afterward, to aid visualization, feature values are divided by respective average values of features in the normal state. By doing this, we get five curves for system life cycle, which are generated by various feature extraction methods. We draw these five curves in Fig. 7(a). What is more, in order to make a more clear illustration, we draw the local enlargement of features curves between snapshots 450 and 650 in Fig. 7(b).

The main conclusions are drawn as follows.

1) From a long-term perspective [shown in Fig. 7(a)], FDDA-based feature (red line) represents the degradation assessment in a more recognizable way than other features do, which is suitable to be a health indicator.
2) As denoted in Fig. 7(b), FDDA-based feature is more sensitive to incipient fault, and its performance degradation curve increases faster while anomalies appear successively.
3) In Fig. 7(b), the oscillations of degradation assessment are observed by using DNN-V1 feature (blue line), it might be caused by the irrelevant information retained during the unsupervised learning stage. By contrast, DNN-V2 feature (magenta line) has a more stable performance, but its sensitivity to state change should be strengthen.

2) Score Response Curve of CIAS: For evaluating the performance of CIAS, suppose the situation that a false alarm comes first, and then several normal input follows, at last a true early fault appears and triggers the fault alarm.

In order to simulate the situation, we design an input sequence like: $[\underbrace{-1\ldots-1}_{15},\underbrace{1\ldots1}_{5},\underbrace{-1\ldots-1}_{15}]$, where -1 represents the anomaly alarm and 1 is the symbol of normal state. Afterward, we input this sequence into CIAS system to get the score response curve, which is shown in Fig. 8.

The characteristics of CIAS could be observed clearly: 1) the growth rate of fault alarm confidence level become higher while several anomalies have been detected and 2) the anomaly memory phase is triggered as the anomaly disappears, in order that a rapid reaction could be made while the anomaly reappears within a period of time (State C to State D to State B).

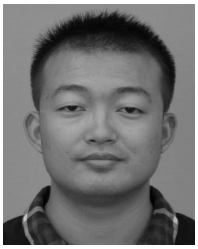
V. Conclusion

This paper designs a model, FDDA, to address early fault detection problem. FDDA aims to learn the recognizable features and predict the system behavior, which can be used to measure changes of system development process. Furthermore, FDDA also designs a useful fault alarm assessment strategy to ensure the fault declaration with a certain confidence level. The experimental results on real-world data sets confirm the superiority of proposed model. Besides, existing DNN architectures, e.g., DBN and CNN, can be conveniently incorporated into the FDDA model.

References

[1] T. Gerber, N. Martin, and C. Mailhes, "Time-frequency tracking of spectral structures estimated by a data-driven method," *IEEE Trans. Ind. Electron.*, vol. 62, no. 10, pp. 6616–6626, Oct. 2015.

[2] M. Riera-Guasp, J. A. Antonino-Daviu, and G.-A. Capolino, "Advances in electrical machine, power electronic, and drive condition monitoring and fault detection: State of the art," *IEEE Trans. Ind. Electron.*, vol. 62, no. 3, pp. 1746–1759, Mar. 2015.
[3] Y. Li, M. Xu, Y. Wei, and W. Huang, "Health condition monitoring and early fault diagnosis of bearings using SDF and intrinsic characteristic-scale decomposition," *IEEE Trans. Instrum. Meas.*, vol. 65, no. 9, pp. 2174–2189, Sep. 2016.
[4] S. Yin, S. X. Ding, X. Xie, and H. Luo, "A review on basic data-driven approaches for industrial process monitoring," *IEEE Trans. Ind. Electron.*, vol. 61, no. 11, pp. 6418–6428, Nov. 2014.
[5] D. U. Campos-Delgado and D. R. Espinoza-Trejo, "An observer-based diagnosis scheme for single and simultaneous open-switch faults in induction motor drives," *IEEE Trans. Ind. Electron.*, vol. 58, no. 2, pp. 671–679, Feb. 2011.
[6] I. Hwang, S. Kim, Y. Kim, and C. E. Seah, "A survey of fault detection, isolation, and reconfiguration methods," *IEEE Trans. Control Syst. Technol.*, vol. 18, no. 3, pp. 636–653, May 2010.
[7] H. Li, X. Lian, C. Guo, and P. Zhao, "Investigation on early fault classification for rolling element bearing based on the optimal frequency band determination," *J. Intell. Manuf.*, vol. 26, no. 1, pp. 189–198, 2015.
[8] A. Brkovic, D. Gajic, J. Gligorijevic, I. Savic-Gajic, O. Georgieva, and S. Di Gennaro, "Early fault detection and diagnosis in bearings for more efficient operation of rotating machinery," *Energy*, vol. 136, pp. 63–71, Oct. 2017. http://www.sciencedirect.com/science/article/pii/S0360544216311525
[9] T. W. Rauber, F. de A. Boldt, and F. M. Varejão, "Heterogeneous feature models and feature selection applied to bearing fault diagnosis," *IEEE Trans. Ind. Electron.*, vol. 62, no. 1, pp. 637–646, Jan. 2015, doi: 10.1109/TIE.2014.2327589.
[10] M. D. Prieto, G. Cirrincione, A. G. Espinosa, J. A. Ortega, and H. Henao, "Bearing fault detection by a novel condition-monitoring scheme based on statistical-time features and neural networks," *IEEE Trans. Ind. Electron.*, vol. 60, no. 8, pp. 3398–3407, Aug. 2013.
[11] A. Purarjomandlangrudi, A. H. Ghapanchi, and M. Esmalifalak, "A data mining approach for fault diagnosis: An application of anomaly detection algorithm," *Measurement*, vol. 55, pp. 343–352, Sep. 2014.
[12] A. Soualhi, H. Razik, G. Clerc, and D. D. Doan, "Prognosis of bearing failures using hidden Markov models and the adaptive neuro-fuzzy inference system," *IEEE Trans. Ind. Electron.*, vol. 61, no. 6, pp. 2864–2874, Jun. 2014.
[13] D. S. Singh and Q. Zhao, "Pseudo-fault signal assisted EMD for fault detection and isolation in rotating machines," *Mech. Syst. Signal Process.*, vol. 81, pp. 202–218, Dec. 2016.
[14] B. Merainani, D. Benazzouz, B. O. Bouamama, and C. Rahmoune, "Early fault diagnosis of gearbox using empirical wavelet transform and Hilbert transform," in *Proc. 24th Medit. Conf. Control Autom. (MED)*, Jun. 2016, pp. 1367–1372
[15] Y. Cheng, N. Ratha, and S. Pankanti, "Outlier faces detector via efficient cohesive subgraph identification," in *Proc. IEEE Conf. Image Process. (ICIP)*, Sep. 2016, pp. 1659–1663, doi: 10.1109/ICIP.2016.7532640.
[16] X. Jin, Y. Sun, Z. Que, Y. Wang, and T. W. S. Chow, "Anomaly detection and fault prognosis for bearings," *IEEE Trans. Instrum. Meas.*, vol. 65, no. 9, pp. 2046–2054, Sep. 2016.
[17] J. Zarei, M. A. Tajeddini, and H. R. Karimi, "Vibration analysis for bearing fault detection and classification using an intelligent filter," *Mechatronics*, vol. 24, no. 2, pp. 151–157, 2014.
[18] T. Ince, S. Kiranyaz, L. Eren, M. Askar, and M. Gabbouj, "Real-time motor fault detection by 1-D convolutional neural networks," *IEEE Trans. Ind. Electron.*, vol. 63, no. 11, pp. 7067–7075, Nov. 2016.
[19] P. Bangalore and L. B. Tjernberg, "An artificial neural network approach for early fault detection of gearbox bearings," *IEEE Trans. Smart Grid*, vol. 6, no. 2, pp. 980–987, Mar. 2015.
[20] D. FernáNdez-Francos, D. Martínez-Rego, O. Fontenla-Romero, and A. Alonso-Betanzos, "Automatic bearing fault diagnosis based on one-class υ-SVM," *Comput. Ind. Eng.*, vol. 64, no. 1, pp. 357–365, Jan. 2013.
[21] J. Huang and X. Yan, "Related and independent variable fault detection based on KPCA and SVDD," *J. Process Control*, vol. 39, pp. 88–99, Mar. 2016.
[22] Q. P. He and J. Wang, "Fault detection using the k-nearest neighbor rule for semiconductor manufacturing processes," *IEEE Trans. Semicond. Manuf.*, vol. 20, no. 4, pp. 345–354, Nov. 2007.
[23] H. Ma, Y. Hu, and H. Shi, "Fault detection and identification based on the neighborhood standardized local outlier factor method," *Ind. Eng. Chem. Res.*, vol. 52, no. 6, pp. 2389–2402, 2013.
[24] Z. Zhao, J. Yang, W. Lu, and X. Wang, "Application of local outlier factor method and back-propagation neural network for steel plates fault diagnosis," in *Proc. 27th Chin. Control Decision Conf. (CCDC)*, May 2015, pp. 2416–2421.
[25] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2015.
[26] W. Lu, B. Liang, Y. Cheng, D. Meng, J. Yang, and T. Zhang, "Deep model based domain adaptation for fault diagnosis," *IEEE Trans. Ind. Electron.*, vol. 64, no. 3, pp. 2296–2305, Mar. 2017, doi: 10.1109/TIE.2016.2627020.
[27] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, p. 15, 2009.
[28] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[29] Z. C. Lipton, "A critical review of recurrent neural networks for sequence learning," *CoRR*, vol. abs/1506.00019, 2015. [Online]. Available: http://arxiv.org/abs/1506.00019
[30] D. Erhan, P. Manzagol, Y. Bengio, S. Bengio, and P. Vincent, "The difficulty of training deep architectures and the effect of unsupervised pre-training," in *Proc. IEEE Int. Conf. Artif., Intell., Statist.*, May 2009, pp. 153–160.
[31] K. Loparo. *Case Western Reserve University Bearing Data Center*. http://csegroups.case.edu/bearingdatacenter/pages/12k-drive-end-bearing-fault-data
[32] H. Qiu, J. Lee, J. Lin, and G. Yu, "Wavelet filter-based weak signature detection method and its application on rolling element bearing prognostics," *J. Sound Vibrat.*, vol. 289, nos. 4–5, pp. 1066–1090, 2006.
[33] Y. Song, Z. Wen, C.-Y. Lin, and R. Davis, "One-class conditional random fields for sequential anomaly detection," in *Proc. 23rd Int. Joint Conf. Artif. Intell.*, 2013, pp. 1685–1691. [Online]. Available: http://dl.acm.org/citation.cfm?id=2540128.2540370
[34] S. Zhai, Y. Cheng, W. Lu, and Z. Zhang, "Deep structured energy based models for anomaly detection," in *Proc. 33nd Int. Conf. Mach. Learn. (ICML)*, New York City, NY, USA, Jun. 2016, pp. 1100–1109. [Online]. Available: http://jmlr.org/proceedings/papers/v48/zhai16.html
[35] W. Lu *et al.*, "Unsupervised sequential outlier detection with deep architectures," *IEEE Trans. Image Process.*, vol. 26, no. 9, pp. 4321–4330, Sep. 2017, doi: 10.1109/TIP.2017.2713048.
[36] C.-W. Hsu and C.-J. Lin, "A comparison of methods for multiclass support vector machines," *IEEE Trans. Neural Netw.*, vol. 13, no. 2, pp. 415–425, Mar. 2002, doi: 10.1109/72.991427.

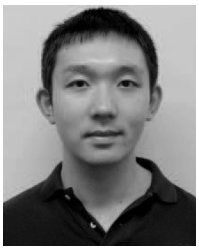


Weining Lu received the B.S. degree from the Department of Physics, Fudan University, Shanghai, China, in 2011, and the Ph.D. degree from the Department of Automation, School of Information Science and Technology, Tsinghua University, Beijing, China, in 2017.

His current research interests include solving anomaly detection problems by using deep architecture networks, computer vision, and data mining.

Yipeng Li received the B.S. and M.S. degrees in electronic engineering from the Harbin Institute of Technology, Harbin, China, in 2003 and 2005, respectively, and the Ph.D. degree in electronic engineering from Tsinghua University, Beijing, China, in 2011.

Since 2011, he has been a Research Associate with the Department of Automation, Tsinghua University. His current research interests include computer vision and data mining by using deep architecture networks.



Yu Cheng received the bachelor's degree from Tsinghua University, Beijing, China, in 2010, and the Ph.D. degree from the Computer Science Department, Northwestern University, Evanston, IL, USA, in 2015.

He is a Research Staff Member with the IBM T.J. Watson Research Center, Yorktown Heights, NY, USA. His work was published in many on top conferences, including ICML, NIPS, CVPR, ICCV, and KDD. His current research interests include deep learning in general, with specific interests in unsupervised learning and compressing deep networks, and its applications in computer vision, data mining, and health informatics.

Deshan Meng (S'15) received the B.E. degree in mechatronics engineering from the Liaoning University of Science and Technology, Anshan, China, in 2011, and the M.E. degree in mechatronics engineering from the Harbin Institute of Technology Shenzhen Graduate School, Shenzhen, China, in 2013, where he is currently pursuing the Ph.D. degree with the Department of Mechatronics and Automation.

His current research interests include fault diagnosis and vibration control.

Bin Liang received the B.S. and M.S. degrees in control engineering from Northwestern Polytechnical University, Xi'an, China, in 1991 and 1994, respectively, and the Ph.D. degree in precision instrument and mechanology from Tsinghua University, Beijing, China, in 1994.

He is currently a Professor with the Department of Automation, Tsinghua University. His current research interests include artificial intelligence, anomaly detection, space robotics, and fault-tolerant control.

Pan Zhou received the B.S. degree in advanced class and the M.S. degree from the Department of Electronics and Information Engineering, Huazhong University of Science and Technology, Wuhan, China, in 2006 and 2008, respectively, and the Ph.D. degree from the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, USA, in 2011.

He is an Associate Professor with the School of Electronic Information and Communications, Wuhan. His current research interests include big data analytics and machine learning, security and privacy, and information networks.