Simultaneous Bearing Fault Recognition and Remaining Useful Life Prediction Using Joint-Loss Convolutional Neural Network

Ruonan Liu, *Member, IEEE*, Boyuan Yang, and Alexander G. Hauptmann

***Abstract*—Fault diagnosis and remaining useful life (RUL) prediction are always two major issues in modern industrial systems, which are usually regarded as two separated tasks to make the problem easier but ignore the fact that there are certain information of these two tasks that can be shared to improve the performance. Therefore, to capture common features between different relative problems, a joint-loss convolutional neural network (JL-CNN) architecture is proposed in this paper, which can implement bearing fault recognition and RUL prediction in parallel by sharing the parameters and partial networks, meanwhile keeping the output layers of different tasks. The JL-CNN is constructed based on a CNN, which is a widely used deep learning method because of its powerful feature extraction ability. During optimization phase, a JL function is designed to enable the proposed approach to learn the diagnosis–prognosis features and improve generalization while reducing the overfitting risk and computation cost. Moreover, because the information behind the signals of different problems has been shared and exploited deeper, the generalization and the accuracy of results can also be improved. Finally, the effectiveness of the JL-CNN method is validated by run-to-failure dataset. Compared with support vector regression and traditional CNN, the mean-square-error of the proposed method decreases 82.7% and 24.9%, respectively. Therefore, results and comparisons show that the proposed method can be applied for the intercrossed applications between fault diagnosis and RUL prediction.**

***Index Terms*—Bearing, deep learning, fault diagnosis, joint-loss (JL) learning, remaining useful life (RUL) prediction.**

I. Introduction

ROLLER bearings are widely used in industrial fields, whose reliability has always been a concern in many mechanical systems like aerospaces, wind turbines, and vehicles. Due to the tough operation conditions, bearing failures often happen, which can lead to catastrophic consequences. Therefore, the need for improving performances in practical systems, in terms of higher reliability and productivity, has necessitated more and more applications of bearing fault diagnosis and remaining useful life (RUL) prediction systems [1], [2].

Fault diagnosis is a process of determining fault presence in machinery as early as possible and identifying kinds, locations, and degrees of faults, while RUL prediction is a process using prediction methods to forecast the future performance of machinery and obtain the time left before machinery loses its operation ability [3], [4]. In the literature, most works reported focus on fault diagnosis, which mostly includes two stages: feature extraction and fault recognition [5], [6]. Statistical indicators, wavelet transform [7], spectrum kurtosis [8], and sparse representation [9], [10] are all widely applied feature extractor, which are followed by machine learning methods for fault recognition, such as support vector machine [11] and artificial neural network (ANN) [12]. Specifically, many neural networks have been developed for fault diagnosis, including multilayer perceptron neural network [13], generalized regression neural network [14], probabilistic neural network [15], self-organizing map [16], and recurrent neural network [17]. Recently, deep learning also provides an end-to-end approach for fault diagnosis to enable a hierarchical nonlinear learning of high-level features built on top of low-level features to discriminate different conditions [18]. It has been successfully applied in damage localization in plate-like structures [19]–[21], fault diagnosis of motors [22], bearings [23], and other important components in industrial systems. Jiao *et al.* [24] proposed a deep coupled dense convolutional network for intelligent fault diagnosis. A novel deep learning network by multiscale inner product with locally connected feature extraction is proposed by Pan *et al.* for intelligent fault detection [25]. Han [26] proposed adversarial learning framework in deep convolutional network for intelligent diagnosis of mechanical faults.

In practical engineering, fault diagnosis alone is not enough to guarantee the security of industrial systems. RUL prediction is also needed to make the maintenance plan in advance. The principle of predictive maintenance is to predict the RUL based on condition monitoring information and make optimal maintenance decisions before the breakdown of equipment [27]. Approaches to prognostics can be categorized broadly into

Manuscript received January 3, 2019; revised April 18, 2019; accepted May 4, 2019. Date of publication May 8, 2019; date of current version January 4, 2020. Paper no. TII-19-0020. *(Corresponding author: Boyuan Yang.)*

R. Liu and A. G. Hauptmann are with the School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213 USA (e-mail: liuruonan04@163.com; Hauptmann,alex@cs.cmu.edu).

B. Yang is with the School of Electrical and Electronic Engineering, University of Manchester, Manchester M13 9PL, U.K. (e-mail: yangboyuanxjtu@163.com).

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/TII.2019.2915536

model-based, data-driven, and hybrid methods [28], [29]. Model-based methods are set up mathematical or physical models to describe the degradation process of a machine, and update model parameters using measured data. The commonly used models include the exponential model [30], the Winner process model [31], the Gaussian mixture model [32], etc. However, the requirements of prior knowledge to estimate parameters, and the uncertainty due to the assumptions and simplifications may pose significant limitations on this approach. To address this problem, data-driven method is proposed, which utilize monitored operational data related to system health. Machine learning techniques have been widely used in data-driven methods, such as support vector regression (SVR) [33], [34], ANN [35], [36], hidden Markov models [37], [38], and Bayesian networks. Hybrid method is usually the combination of the two aforementioned methods that are applied to prognostic tasks [39], [40]. However, though fault diagnosis techniques have been well developed, there are few research works about RUL prediction. This is not only due to the much more difficult prognosis models and techniques, but also because it is hard to acquire the life test data.

With the development of industrial requirements, demands of both fault diagnosis and RUL prediction are increasing rapidly. On one hand, when fault diagnosis is implemented, because most machines cannot be halted immediately, RUL prediction is also needed to give a basic idea of how long could the machine operate. On the other hand, when predict RUL, if the failure type is known in advance, then the corresponding measurements or maintenance can be prepared earlier to decrease the downtime and increase production.

In the literature, almost all methods deal with these two tasks separately. That is, if both diagnosis and prognosis are needed in an industrial system, then two models are constructed. It is plausible because the problem can be much easier, and the results can also be obtained. However, to some degree, there are some commonalities and inherent correlation between these two tasks that they both implement the diagnosis or prognosis task by extracting faulty information and comparing with baselines. Therefore, it can be inferred that there is some certain information that can be shared in these two research methods, which has been ignored by the previous methods. These two problems are relative and can be connected by one or several share representations. If we deal with these two problems separately as usual, not only will the method be time-consuming and costly, but also may waste a lot of associated information between these two problems that can be used for both tasks, thus reduce the accuracy of results.

Therefore, to improve the performances of both the diagnosis and prognosis methods without increasing the computation and labor needs, and take full advantage of the valuable experimental signals, a joint-loss convolutional neural network (JL-CNN) architecture for simultaneous bearing fault diagnosis and RUL prognosis is proposed. The major novelties and contributions of this paper can be summarized in four aspects.

1) First, an intelligent fault diagnosis and RUL prediction framework is proposed in an end-to-end way, which can learn features adaptively from complex mechanical data and use those features for condition diagnosis and prognosis. Therefore, no predefined features are required, which not only reduces the reliance on prior knowledge, but also liberates the labor spent on developing sensitive and robust features.
2) By sharing the parameters and partial networks, meanwhile keeping the output layers of different tasks, the proposed method can recognize faults and predict RUL simultaneously, at the same time reduce the overfitting risk and computation cost. The information behind the signals of different tasks can also be shared and exploited deeper, which can lead to a better generalization and improve the accuracy of both tasks. In addition, the experiment finds that traditional network is more easily getting local optimum solution, while the JL-CNN can solve this problem by its specific architecture.
3) RUL prediction is always a challenging task. The data-driven model, which is constructed only for RUL prediction can be easily overfitting, or converge to a local minimal. Therefore, the aforementioned problem can be solved by adding the loss of classification task, thus improving the RUL prediction performance.
4) The JL-CNN framework is verified by the practical signals of bearing run-to-failure tests. The analysis shows that the JL-CNN could adapt to fault diagnosis task and RUL prediction task simultaneously. The effectiveness and superiority are also verified by the comparison with state-of-the-art methods.

Thus, the proposed method can be applied for the intercrossed applications between fault diagnosis and RUL prediction with a better performance.

The rest of this paper is organized as follows. Section II illustrates the proposed approach in detail. The proposed JL function and its training algorithm are described in Section III. In Section IV, the proposed method is applied to analyze two bearing run-to-failure datasets. The results and comparisons with classical and state-of-the-art methods verify the effectiveness of the proposed JL-CNN framework. Finally, Section V concludes this paper.

II. Proposed Approach

Recent decades have witnessed the rapid development of deep learning techniques. In essence, deep learning is a multilayer neural network, which can represent input signals with high-level and hierarchical features. As one of the most widely applied deep neural network, CNNs have shown great improvement over hand-crafted features for many problems including object recognition, face detection, activity recognition. In recent years, CNNs also have attracted the attention in industrial field because of their powerful signal represent capability. Based on the CNN model, a JL-CNN is proposed in this paper by introducing a JL function in traditional CNN architecture, which can be applied for simultaneous fault recognition and RUL prediction.

A. CNN

In a CNN, a convolutional layer is used to extract elementary visual features by convolution operation first. There are two architectural ideas in convolutional layer: local receptive fields

(or sparse connectivity) and shared weights. In order to reduce the memory requirements of the model, improve its statistical efficiency, and describe the complicated interactions between units more efficiently, CNN extract features from local receptive fields. Moreover, instead of using each weight matrix only once in traditional neural networks, each kernel (or weight matrix) is used across the entire input, which is the so-called shared weights. Convolution is a mathematical operation on two functions and can be defined as the integral of the product of the two functions after one is reversed and shifted. In CNN, the function x to the convolution is often referred to as the input and the function w is the kernel. The convolution formula can be described as a weighted average of the function $x(a)$ at the time instant t where the weighting is given by $w(a)$ simply shifted by amount t. We often use convolutions over more than one axis. For example, a two-dimensional (2-D) image I is used as input, therefore the kernel K is also 2-D

$$z[i,j] = (I * K)[i,j] = \sum_m \sum_n I[i-m, i-n]K[m,n] \quad (1)$$

where i and j are the width and height of I. m and n are constants.

After convolution layer, the results $z[i,j]$ are obtained and used as the input of hidden units. The hidden units are computational units that take $z[i,j]$ as inputs and $h_{W,b}(z)$ as outputs, given by

$$h_{W,b}(z) = f(z[i,j] + b) \quad (2)$$

where W is the connective weights matrix. b is the bias term. The function f is called the activation function, which is usually sigmoid function or tanh function.

Then, the results of $h_{W,b}(z)$ are modified by the pooling layer. The pooling layer performs a subsampling by replacing the output of the net at a certain location with a summary statistic of the nearby outputs. The most commonly used pooling operation is max pooling, which reports the maximum output within a rectangular neighborhood. The pooling operation can not only reduce the resolution of the feature map and the sensitivity of the output to shifts and distortions, it also shows invariant to small translation or rotation of the input.

B. JL-CNN Architecture

JL learning is an inductive transfer method that uses the domain specific information contained in the training signals of related tasks by learning the multiple tasks in parallel while using a shared representation [41]. It can improve learning for one task by using the information contained in related tasks via learning tasks in parallel, while using a shared representation. The information from one task can help the related tasks to be learned more effectively [42].

Fig. 1 graphically shows two independent neural networks, which is called the single-task learning. Backpropagation algorithm is used to train each network. Because there is no connection between them, features of each network cannot be shared or used to help learn another network. Fig. 2 illustrates a network that has two outputs, each output is corresponding to a task in Fig. 1. What should be noted here is that these outputs

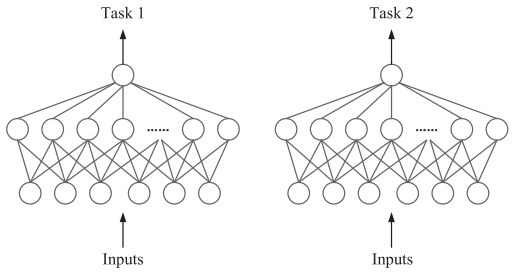

Fig. 1. Illustration of two single-task models.

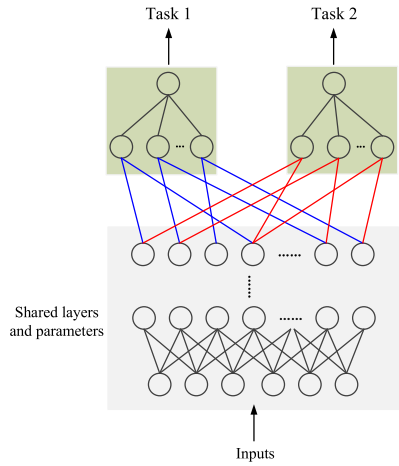

Fig. 2. Illustration of a JL model. The black lines represent the shared parameters. The blue lines represent the exclusive parameters of task 1. The red lines represent the exclusive parameters of task 2.

can connect all neural units in the hidden layer they shared, as shown in Fig. 2. Then, independent subnetworks are constructed after the shared hidden layers, and the corresponding parameters are trained separately. In this way, a JL-CNN architecture is constructed. In a JL-CNN, backpropagation algorithm optimize parameters of the two outputs concurrently. Because the bottom layers are shared with the two tasks, the feature representations of one task can be also used by other tasks, which can lead to co-learning. Biologically, JL learning can be regard as being inspired by human learning. For learning new tasks, we often apply the knowledge we have acquired by learning related tasks. In conclusion, the core of the JL-CNN is to train more than one tasks in parallel and share the feature representations of different tasks.

There are four architectural mechanisms that contribute to the wide application and effectiveness of the JL-CNN: implicit data augmentation, attention focusing, eavesdropping, and representation bias.

1) Implicit Data Augmentation: The JL-CNN model can increase the sample numbers for training, because there is strong noise interference in the mechanical vibration signals, and the noise level varies from different signals. Therefore, learning one task A may lead to the overfitting problem of task A, while learning two tasks simultaneously can obtain more general representations by averaging noise patterns.

2) Attention Focusing: In industrial systems, the signals that can be analyzed are usually very complicated and limited. If the dataset of a task is noisy or limited and high-dimensional, the model may not distinguish the relevant and irrelevant features. The JL-CNN architecture can help the model to focus attention on the important features because other tasks may provide additional evidence that the features are relevant or not.

3) Eavesdropping: In practical applications, it can be found that a feature G can be easily learned by task B, but harder to be learned by task A. This may be because task A interacts with the features in a more complex way or because other features are impeding the model's ability to learn G. The JL-CNN architecture gives the model the ability of eavesdropping. That is, the architecture can learn G through task B, then G can be shared in task A.

4) Representation Bias: Due to the feature sharing idea, the JL-CNN may bias the model to prefer representations that other tasks also prefer, which will also help the model to generalize new tasks as a hypothesis space that performs well for a sufficiently large number of training tasks and will also perform well for learning novel tasks as long as they are from the same environment.

Therefore, the JL-CNN architecture allows features developed in the hidden layer for one task to be used by other tasks, and also can be used to support several tasks that would not have been developed, which show the potential in the rapidly developed industrial systems. For example, usually the vibration of a roller bearing mainly consists of three components: the vibration caused by the stiffness change of bearings, the vibration caused by the installation, and the vibration caused by the bearing fault. The last component is also the most important component for fault diagnosis. The roller element runs between outer race and inner race during the operation. For a normal bearing, the rolling surface is very smooth, so the vibration caused by fault can be very tiny. For a faulty bearing, when the roller element runs pass the fault point, an impulse vibration will be generated. For different fault location (such as outer race, inner race, roller element), the impulse vibration will be different. The impulse vibration frequency is the passing frequency of roller element, which is the most commonly used fault character frequency for bearing fault diagnosis. Thus, the identification of the fault character frequency may lead to a classification problem. At the same time, the RUL of a bearing is directly related to the gradual expansion of the fault, as well as the impulse vibration. Therefore, a lot of feature information can be shared by the fault diagnosis task and the RUL prediction task. On the other hand, as we all know, RUL prediction is always a challenging task. The data-driven model, which is constructed only for RUL prediction can be easily overfitting, or converge to a local minimal. Therefore, the aforementioned problem can be solved by adding the loss of classification task, thus improving the RUL prediction performance. Some strategies such as sparse connectivity, shared weights, and pooling operation also endow the JL-CNN architecture with the ability to reduce the risk of overfitting.

The JL-CNN architecture applied in this paper is graphically illustrated in Fig. 3. The structure of the JL-CNN is mainly constructed based on the classical VGG network [43]. There are 2048 units in the input layer. The convolutional kernel used in the first layer of the JL-CNN architecture is $3 \times 1@8$ and the convolutional kernels used in other layers are all set to be $3 \times 1@16$. The pooling sizes of the max-pooling layers in this architecture are all set to be 2×1. This is because the feature maps obtained by the convolutional layer with large convolutional kernel can be obtained by the superposition of convolutional layers with small convolutional kernel. The large size of the convolutional kernel can lead to a great increase of algorithm complexity and computation, the same as pooling layers. At the same time, even-numbered kernel size will lead to the difference between input feature maps and output feature maps [44]. Therefore, the kernel sizes in the JL-CNN are all set to be small odd number, that is 3×1. Adam algorithm is used in this paper for parameter optimization.

III. JL Function Training

In order to construct the JL-CNN architecture, a JL function is proposed in this paper. Assume that the input samples are $((x^{(1)}, y_1^{(1)}, y_2^{(1)}), (x^{(2)}, y_1^{(2)}, y_2^{(2)}), \ldots, (x^{(m)}, y_1^{(m)}, y_2^{(m)}))$, where the ith sample $x^{(i)} = (1, x_1^{(i)}, x_2^{(i)}, \ldots, x_p^{(i)})$. $y_1^{(m)}, y_2^{(m)}$ are the labels of the RUL prediction task and classification task, respectively. The model parameters are $W = (W_0, W_1, W_2, \ldots, W_p)^T$.

As for RUL prediction problem, mean-square-error (MSE) loss function is applied for regression as

$$J_1(W) = \frac{1}{m}\sum_{i=1}^{m}(y_1^{(i)} - \hat{y_1}^{(i)})^2 \tag{3}$$

where $\hat{y_1}^{(i)}$ is the output of the first task network.

Then, for binary fault classification problem, the loss function $J_2(W)$ is set to be

$$J_2(W) = -\frac{1}{m}\sum_{i=1}^{m} y_2^{(i)}\log(\hat{y_2}^{(i)}) + (1 - y_2^{(i)})\log(1 - \hat{y_2}^{(i)}) \tag{4}$$

where $\hat{y_2}^{(i)}$ is the output of the fault diagnosis task network. This loss function is called the cross-entropy loss function.

In this paper, the JL function is constructed by the loss functions of both classification and regression problem as follows:

$$J(W) = J_1(W) + \lambda J_2(W) \tag{5}$$

where λ is the penalty factor that control the weight of the two tasks. That is, if $\lambda = 0$, the network is a model only for RUL prediction; if $\lambda = \infty$, the network is a model only for fault diagnosis. The suitable selection of the λ may vary from dataset to dataset. So, λ can be decided by the heuristic method. The selection of λ in this paper has been discussed in detail in the case study.

Let $z_i^{(l)}$ be the total weighted sum of inputs to unit i in layer l and $a_i^{(l)}$ be the activation of the same unit. The computation in this neural network can be given by forward propagation

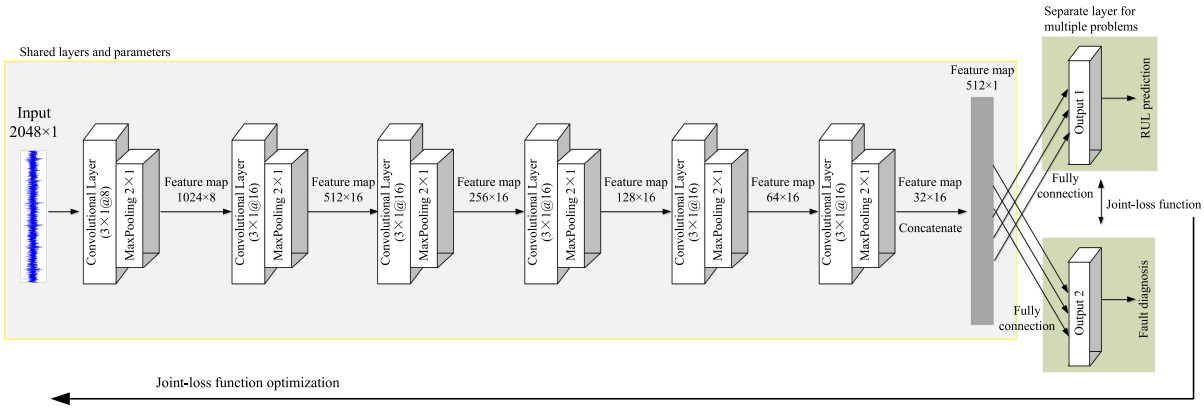

Fig. 3. JL-CNN architecture applied in this paper.

algorithm as

$$\begin{aligned} z^{(2)} &= W^{(1)}x + b^{(1)} \\ a^{(2)} &= f(z^{(2)}) \\ z^{(3)} &= W^{(2)}a^{(2)} + b^{(2)} \\ h_{W,b}(x) &= a^{(3)} = f(z^{(3)}). \end{aligned} \tag{6}$$

More generally, this step can be written as

$$\begin{aligned} z^{(l+1)} &= W^{(l)}a^{(l)} + b^{(1)} \\ a^{(l+1)} &= f(z^{(l+1)}). \end{aligned} \tag{7}$$

The optimization problem in neural network and CNN is to minimize $J(W, b)$ as a function of parameters W and b

$$\arg\min_{W,b} J(W, b). \tag{8}$$

Based on $a^{(l)}$ and $z^{(l)}$ calculated by forward propagation algorithm in (9), the error term $\delta^{(n_l)}$ of output layer can be given by

$$\delta^{(n_l)} = -(y - a^{(n_l)}) \cdot f'(z^{(n_l)}). \tag{9}$$

Then, for $l = n_l - 1, n_l - 2, \ldots, 2$, the error term of each layer can be given by

$$\delta^{l} = ((W^{(l)})^T \delta^{(l+1)}) \cdot f'(z^{l}). \tag{10}$$

The desired partial derivatives can be computed as

$$\begin{aligned} \nabla_{W^{(l)}} J(W, b; x, y) &= \delta^{(l+1)}(a^{l})^T \\ \nabla_{b^{(l)}} J(W, b; x, y) &= \delta^{(l+1)}. \end{aligned} \tag{11}$$

Let g_t denotes the gradient vector at time t of the cost function $J(W, b; x, y)$; $g_t(W^{(l)})$ and $g_t(b^{(l)})$ denote the partial derivatives of the cost function

$$\begin{aligned} g_t(W^{(l)}) &= \nabla_{W^{(l)}} J(W, b; x, y) \\ g_t(b^{(l)}) &= \nabla_{b^{(l)}} J(W, b; x, y). \end{aligned} \tag{12}$$

Then, m_t is the estimation of first moment (the mean) and v_t is the estimation of second moment (the uncentered variance) of the gradients g_t, respectively, which can be described as

$$\begin{aligned} m_t(W^{(l)}) &= \beta_1 m_{t-1} + (1 - \beta_1) g_t(W^{(l)}) \\ v_t(W^{(l)}) &= \beta_2 v_{t-1} + (1 - \beta_2) g_t(W^{(l)})^2 \end{aligned} \tag{13}$$

and

$$\begin{aligned} m_t(b^{(l)}) &= \beta_1 m_{t-1} + (1 - \beta_1) g_t(b^{(l)}) \\ v_t(b^{(l)}) &= \beta_2 v_{t-1} + (1 - \beta_2) g_t(b^{(l)})^2 \end{aligned} \tag{14}$$

where $\beta_1, \beta_2 \in [0, 1]$. m_0 and v_0 are initialized as zero vectors. To avoid them being biased toward zero especially in the initial iterations, a bias correction is computed on these moment estimations

$$\begin{aligned} \hat{m}_t(W^{(l)}) &= \frac{m_t(W^{(l)})}{1 - \beta_1^t} \\ \hat{v}_t(W^{(l)}) &= \frac{v_t(W^{(l)})}{1 - \beta_2^t} \end{aligned} \tag{15}$$

and

$$\begin{aligned} \hat{m}_t(b^{(l)}) &= \frac{m_t(b^{(l)})}{1 - \beta_1^t} \\ \hat{v}_t(b^{(l)}) &= \frac{v_t(b^{(l)})}{1 - \beta_2^t}. \end{aligned} \tag{16}$$

The vector $\hat{v}_t$ represents an approximation of the diagonal of the Fisher information matrix. The parameters can be updated by the Adam rule as

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t. \tag{17}$$

More specifically

$$\begin{aligned} W_{t+1}^{(l)} &= W_t^{(l)} - \frac{\eta}{\sqrt{\hat{v}_t(W^{(l)})} + \epsilon} \hat{m}_t(W^{(l)}) \\ b_{t+1}^{(l)} &= b_t^{(l)} - \frac{\eta}{\sqrt{\hat{v}_t(b^{(l)})} + \epsilon} \hat{m}_t(b^{(l)}) \end{aligned} \tag{18}$$

where η is the step size and ϵ is a small positive constant used to avoid the division for zero.

To train the JL-CNN, we can now repeatedly take steps of Adam algorithm to reduce the cost function $J(W, b)$ until the

Algorithm 1: Joint-Loss Function Optimization Algorithm.

Input:
Training set $D = \{(x^{(i)}, y^{(i)})\}, i = 1, \ldots, m$;
Learning step η;
Exponential decay rates $\beta_1, \beta_2 \in [0, 1]$;
Joint loss function $J(W, b)$.
Initialization:
Initialize parameters W and b in the range of (0, 1);
Initialize 1^{st} moment vector: $m_0 \leftarrow 0$;
Initialize 2^{nd} moment vector: $v_0 \leftarrow 0$;
Initialize timestep: $t \leftarrow 0$.
While:
W_t and b_t not converged $t \leftarrow t + 1$;
Compute the activations $a_i^{(l)}$ at timestep t by forward propagation algorithm in (10);
Compute the error term $\delta^{(n_l)}$ of output layer by (12);
Compute the error term δ^l of every layer by (13);
Compute the desired partial derivatives $\nabla_{W^{(l)}} J(W, b)$ and $\nabla_{b^{(l)}} J(W, b)$ by (14);
$g_t(W^{(l)}) \leftarrow \nabla_{W^{(l)}} J(W, b; x, y)$;
$g_t(b^{(l)}) \leftarrow \nabla_{b^{(l)}} J(W, b; x, y)$;
$m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$;
$v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$;
$\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$;
$\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$;
$W_{t+1}^{(l)} \leftarrow W_t^{(l)} - \eta \cdot \hat{m}_t(W^{(l)}) / (\sqrt{\hat{v}_t(W^{(l)})} + \epsilon)$;
$b_{t+1}^{(l)} \leftarrow b_t^{(l)} - \eta \cdot \hat{m}_t(b^{(l)}) / (\sqrt{\hat{v}_t(b^{(l)})} + \epsilon)$;
Until the termination condition is satisfied;
Output:
A CNN with optimized parameters.

exit condition is reached. The complete process of the training algorithm is summarized in Algorithm 1.

IV. Case Study

A. Experimental Setup and Dataset Description

The experimental run-to-failure data were generated from Prognostics Center of Excellence through prognostic data repository contributed by Intelligent Maintenance System, University of Cincinnati [45]. This is because this dataset is widely used for RUL prediction, and thus the performance of the proposed method can be easily compared and verified. Bearing test rig consists of four bearings that were installed on one shaft, as presented in Fig. 4. The rotation speed of shaft was kept constantly at 2000 r/min and a radial load of 26 689 N was placed onto the shaft and bearing by a spring mechanism. The bearings used are Rexnord ZA-115 double row bearings. The vibration signals were acquired by eight accelerometers from PCB 353B33 that were installed at vertical and horizontal directions. Four thermocouples were also installed in the outer race of each bearing to record bearing temperature for monitoring lubrication purposes. Vibrations signals were collected every 10 min. Sampling frequency is 20 kHz and the data length is 20 480 points.

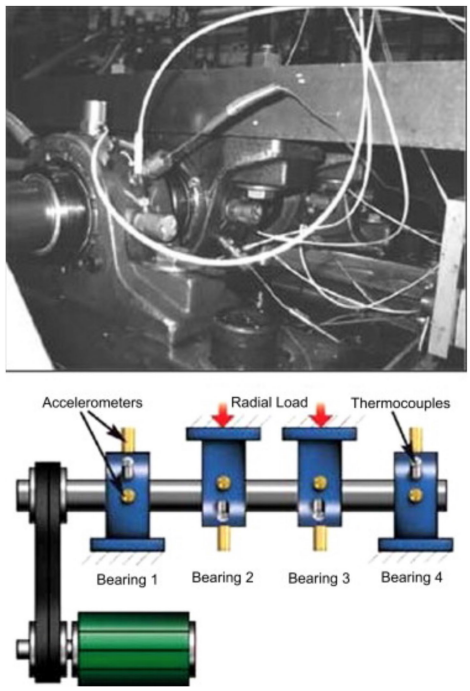

Fig. 4. Bearing test rig and sensor placement illustration.

TABLE I
Training and Testing Samples

	Dataset 1	Dataset 2
Training samples	2135	3136
Testing samples	915	1344
Training accuracy	100%	100%
Testing accuracy	100%	100%

In this experiment, a sample is constructed by 2048 points. There are two bearing run-to-failure datasets used in this experiment: the first one is inner-race fault of bearing 3; the second one is outer-race fault of bearing 1. 70% of the samples are selected randomly for training, and the rest 30% samples are used for testing. The numbers of training and testing samples are given in Table I. For fault diagnosis task, the labels of the inner-race fault samples are set to be 0; and the labels of the outer-race fault samples are set to be 1. For RUL prediction task, the labels are set from 1 linear decrease to 0 after the incipient fault point (IFP). Since

$$\text{RUL}_{\text{relative}} = \frac{\text{RUL}_{\text{current}}}{\text{total degradation time}} \tag{19}$$

where the total degradation time is a constant, and current RUL is a linear function. So, the relative RUL should be a linear function as well. In this paper, the IFP is detected by the method proposed in [46]: the threshold is set to be mean(rms) $\pm$ $(6 \times \sigma)$. Where mean (rms) is the mean value of the rms before fault failure occurrence. For the first dataset, the rms values between 500 and 1000 min are used to calculate the mean (rms). For the second dataset, the rms values between 0 and 400 min are used to calculate the mean (rms). When the rms of current sample

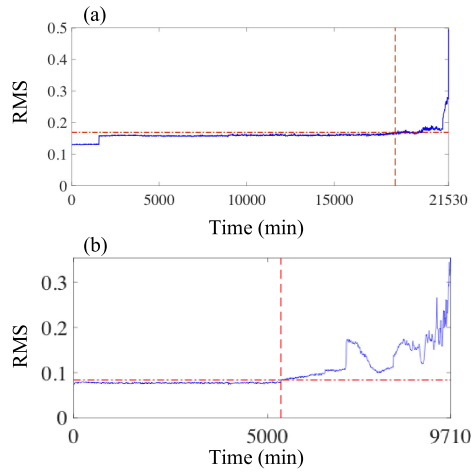

Fig. 5. IFP selection result of (a) dataset 1 and (b) dataset 2. The horizon red line represents the rms threshold. The vertical red line represents the time point of IFP. The blue line represents the rms trend.

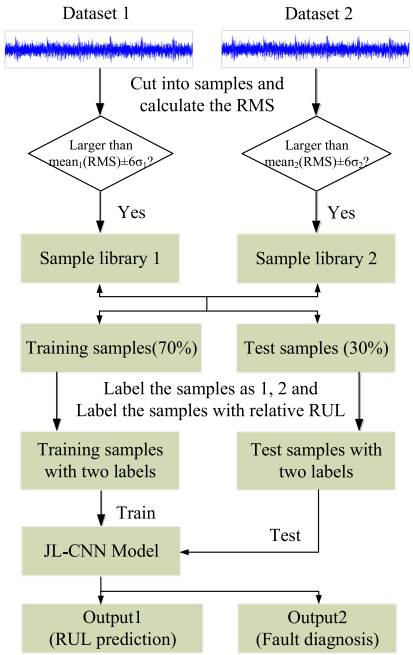

Fig. 6. Flowchart of the proposed method.

is larger than the threshold, the system is detected to begin to degrade, and the current point is set to be the IFP. When the rms of current sample is bigger than 0.35 g, the system is detected as failure. Based on this method, the IFP of the two datasets are set to be 1848 and 534 min, respectively, as shown in Fig. 5.

B. Results and Comparisons

After data collection, the proposed framework is utilized to analyze the collected samples. The flowchart of the proposed method is shown in Fig. 6. The algorithm is implemented by Python based on GPU NVIDIA GTX 1060 6G and CPU Intel I7 6700. The platform is Pytorch1.0. The CNN model is trained 100 epochs, which costs about 400 s. It takes a short time to analyze a single sample, which is less than 1 s. Because fault diagnosis task is relatively simple, both the training accuracy and the test accuracy are all 100%. Therefore, this task will not be discussed later. Since there is an additional parameter λ in the JL function, whose value will influence the performance of the proposed method, we first exploited the loss of the JL-CNN method with different λ. Fig. 7 shows the prediction accuracy trend on the test set with the change of λ. The loss of diagnosis task is small, which is around 1×10^{-5}, so there is no tradeoff problem in this experiment. It clearly shows that when $\lambda \in [0.1, 0.5]$, the loss of the JL-CNN has all been decreased. The optimal range may vary from problem to problem. When $\lambda = 0.25$, the proposed framework gets the best performance. As was inferred before, when $\lambda = 0$, the JL function becomes MSE loss function, and the network is a CNN model only for RUL prediction. It can be seen that the corresponding loss has been increased about 35% compared with the architecture when $\lambda = 0.25$. This phenomenon can be explained from the view of the network architecture: when only the RUL prediction network is used ($\lambda = 0$), the learned features trend to decrease the RUL training error, which may lead to overfitting because there is only one task; when both MSE and cross-entropy loss functions are both contained in the JL function ($\lambda = 0.25$), the learned features trend to satisfy both tasks, therefore decreasing the overfitting risk and obtain a higher test accuracy; when λ further increases towards infinity, only the fault diagnosis network is used and the test loss will increase again.

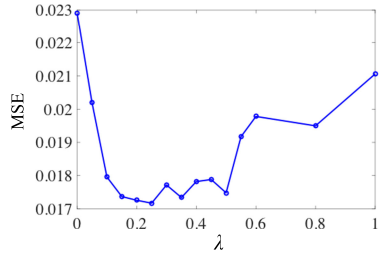

Fig. 7. MSE loss of the JL-CNN method by varying the weighting parameter λ.

The loss trends of training and testing datasets are graphically illustrated in Fig. 8. The red line represents the loss curve when $\lambda = 0.25$; the blue line represents the loss trend when $\lambda = 0$. It can be seen that when $\lambda = 0.25$, both training and testing loss curves converge slower than $\lambda = 0$. This is because when $\lambda = 0.25$, the JL function is constructed by two loss functions that can lead to a slow convergence speed because there are two loss functions that need to be trained. But, the final test accuracy will be higher due to the improved generalization ability, which is also proved in Fig. 8. In addition, the loss trends of training dataset and test dataset are similar with each other, which illustrates that there is no overfitting problem.

The RUL prediction results of the two datasets with different λ are shown in Fig. 9. The red lines in Fig. 9 represent the real RUL. It can be seen that the degradation trends of both the inner-race fault in Fig. 9(a) and the outer-race fault in Fig. 9(b)

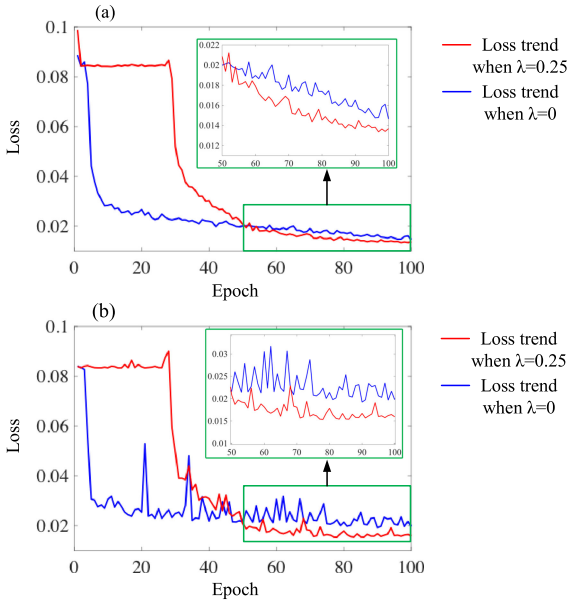

Fig. 8. Loss trends of (a) training and (b) testing datasets. The red line represents the loss trend when $\lambda = 0.25$. The blue line represents the loss trend when $\lambda = 0$.

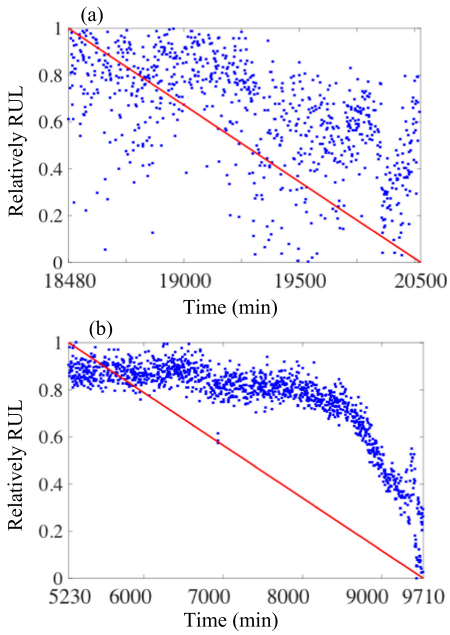

Fig. 10. SVR RUL prediction results of (a) dataset 1 and (b) dataset 2. The red line represents the real RUL. The blue points represent the predicted RUL.

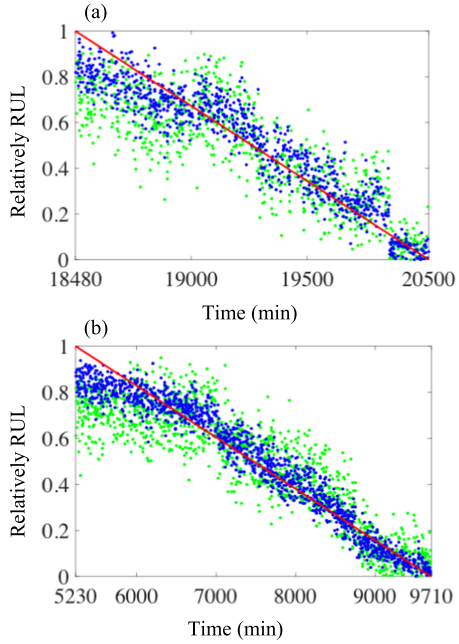

Fig. 9. RUL prediction results of (a) dataset 1 and (b) dataset 2. The red line represents the real RUL. The blue points represent the predicted RUL when $\lambda = 0.25$. The green points represent the predicted RUL when $\lambda = 0$ that the network becomes a traditional CNN architecture.

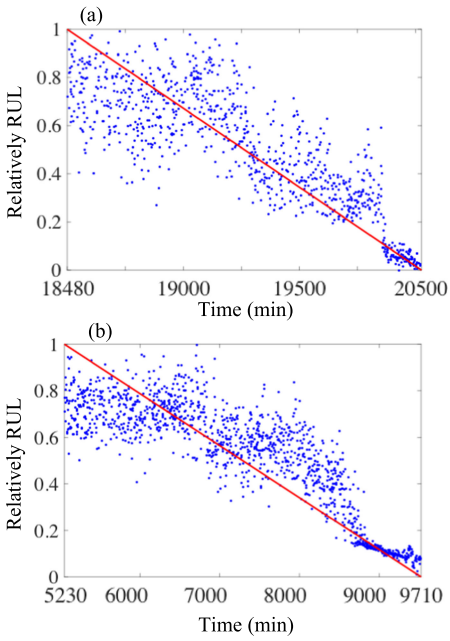

Fig. 11. HI-based RUL prediction results of (a) dataset 1 and (b) dataset 2. The red line represents the real RUL. The blue points represent the predicted RUL.

have been extracted successfully. When $\lambda = 0$, the network is a traditional CNN, which is represented as green points in Fig. 9. When $\lambda = 0.25$, the network is a JL-CNN, which is represented as blue points. It can be seen that the blue points are closer to the real RUL. Compared with green points, blue points are closer to and more concentrated on real RUL line.

To verify the effectiveness of the proposed method, SVR, which is a widely used RUL prediction method, is also applied to analyze the same data, and the results are shown in Fig. 10. 11 statistic features such as rms and kurtosis are used as the input of SVR. In addition, a state-of-the-art health index (HI) based prognostics method [1] is also applied in this paper for comparison. The results are shown in Fig. 11. It can be seen that

TABLE II
MSE LOSS OF SVR, HI-BASED METHOD, CNN, AND THE PROPOSED METHOD

Methods	SVR	HI-based method	CNN(λ=0)	JL-CNN(λ=0.25)
Test MSE	0.0994	0.0236	0.0229	0.0172
Train MSE	0.0285	0.0232	0.0134	0.0136

SVR performs worst. The prediction result of dataset 1 is too decentralized to be applied in practical industrial systems and the result of dataset 2 is not as decentralized as the first one, but still far from real RUL. The HI-based prediction method performs better than SVR, but the prediction results of the two datasets still show high standard deviation. It can be seen from Fig. 9 that the proposed method can provide both a high prediction accuracy and a low dispersity. This is because the feature extracted by the deep network can be more representative than statistic features, meanwhile the shared network can enlarge the generalization of the proposed method, and therefore increase the test accuracy.

For a more intuitive comparison, MSE is used as the prediction index. The MSE loss of SVR, HI-based method, CNN when $\lambda = 0$, and JL-CNN when $\lambda = 0.25$ are given in Table II. The MSE of the JL-CNN decreases 82.7% compared with SVR and decreases 24.9% compared with traditional CNN, which also verify the effectiveness and superiority of the proposed framework. In addition, it can be seen that there is huge variability between these two datasets, not only in the failure time, but also in the failure type and degradation pattern. That is, at the end of each experiment, one bearing is identified as outer-race fault and another one as inner-ring fault. One of the bearing degrades quickly and the other degrades slowly as shown in Fig. 5. Since the proposed method still performs well even though there is huge variability between these two experiments, the robustness and generalization of the proposed method can be verified. The main limitation of the proposed method is the requirement of more than one training datasets. If there is one training dataset, there will be only one label of diagnosis task, which makes the model unable to be trained.

V. CONCLUSION

In this paper, a novel deep learning-based approach was proposed for simultaneous bearing fault diagnosis and RUL prediction in this paper, which is named the JL-CNN. This approach constructed a deep network by sharing partial network and constructing a JL function, which could not only learn the multiple tasks in parallel, but also could improve the performance of both tasks by sharing features. To verify its practicality in industrial systems, the proposed method was applied to analyze two run-to-failure bearing datasets. Compared with traditional fault diagnosis or RUL prediction methods, it did not require predetermined feature selection and was less likely to overfitting. Because of the feature sharing strategy, the proposed method could make full use of the information and improve the accuracy of both tasks. In addition, the experimental results showed that the JL-CNN method could increase the accuracy of both tasks compared with the deep network only for single task, which illustrates the effectiveness of the JL architecture.

There is still some work needed to improve the proposed architecture. First, the selection of parameter λ was an important issue when applying the proposed approach; second, the construction of the JL-CNN was also a difficult problem in practical applications; third, evolutionary computation methods could be used to optimize the network parameters, and subsequently improve the performance. Since there were only two specimens in this paper, more training dataset could be helpful to improve the performance and train a more generalized model. The proposed method could not only be trained by one dataset because the output labels were corresponding to different failure and the network could not be trained with only one output label, which was another limitation. At the same time, cross validation could be helpful to improve the generalization and reduce the risk of overfitting. Since the performance of the proposed method was stable, cross validation did not apply in this paper. But, it could be a helpful tool in real applications. To enlarge the possible applications of the proposed method, our further work would focus on some techniques that can solve these problems.

REFERENCES

[1] 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.
[2] T. H. Loutas, D. Roulias, and G. Georgoulas, "Remaining useful life estimation in rolling bearings utilizing data-driven probabilistic e-support vectors regression," *IEEE Trans. Rel.*, vol. 62, no. 4, pp. 821–832, Dec. 2013.
[3] Y. Lei, N. Li, S. Gontarz, J. Lin, S. Radkowski, and J. Dybala, "A model-based method for remaining useful life prediction of machinery," *IEEE Trans. Rel.*, vol. 65, no. 3, pp. 1314–1326, Sep. 2016.
[4] J. Lee, F. Wu, W. Zhao, M. Ghaffari, L. Liao, and D. Siegel, "Prognostics and health management design for rotary machinery systems—Reviews, methodology and applications," *Mech. Syst. Signal Process.*, vol. 42, no. 1, pp. 314–334, 2014.
[5] R. Liu, B. Yang, E. Zio, and X. Chen, "Artificial intelligence for fault diagnosis of rotating machinery: A review," *Mech. Syst. Signal Process.*, vol. 108, pp. 33–47, 2018.
[6] H. Shao, H. Jiang, H. Zhang, and T. Liang, "Electric locomotive bearing fault diagnosis using novel convolutional deep belief network," *IEEE Trans. Ind. Electron*, vol. 65, no. 3, pp. 2727–2736, Mar. 2018.
[7] R. Yan, R. X. Gao, and X. Chen, "Wavelets for fault diagnosis of rotary machines: A review with applications," *Signal Process.*, vol. 96, pp. 1–15, 2014.
[8] Y. Wang, J. Xiang, R. Markert, and M. Liang, "Spectral kurtosis for fault detection, diagnosis and prognostics of rotating machines: A review with applications," *Mech. Syst. Signal Process.*, vol. 66, pp. 679–698, 2016.
[9] B. Yang, R. Liu, and X. Chen, "Fault diagnosis for a wind turbine generator bearing via sparse representation and shift-invariant K-SVD," *IEEE Trans. Ind. Inform.*, vol. 13, no. 3, pp. 1321–1331, Jun. 2017.
[10] B. Yang, R. Liu, and X. Chen, "Sparse time-frequency representation for incipient fault diagnosis of wind turbine drive train," *IEEE Trans. Instrum. Meas.*, vol. 67, no. 11, pp. 2616–2627, Nov. 2018.
[11] R. Liu, B. Yang, X. Zhang, S. Wang, and X. Chen, "Time-frequency atoms-driven support vector machine method for bearings incipient fault diagnosis," *Mech. Syst. Signal Process.*, vol. 75, pp. 345–370, 2016.
[12] W. Chine, A. Mellit, V. Lughi, A. Malek, G. Sulligoi, and A. M. Pavan, "A novel fault diagnosis technique for photovoltaic systems based on artificial neural networks," *Renewable Energy*, vol. 90, pp. 501–512, 2016.
[13] N. H. Chandra and A. Sekhar, "Fault detection in rotor bearing systems using time frequency techniques," *Mech. Syst. Signal Process.*, vol. 72, pp. 105–133, 2016.

[14] P. Pławiak, "An estimation of the state of consumption of a positive displacement pump based on dynamic pressure or vibrations using neural networks," *Neurocomputing*, vol. 144, pp. 471–483, 2014.
[15] F. Wang, S. Chen, J. Sun, D. Yan, L. Wang, and L. Zhang, "Time-frequency fault feature extraction for rolling bearing based on the tensor manifold method," *Math. Problems Eng.*, vol. 2014, pp. 1–15, 2014.
[16] S. Hong, Z. Zhou, E. Zio, and W. Wang, "An adaptive method for health trend prediction of rotating bearings," *Digit. Signal Process.*, vol. 35, pp. 117–123, 2014.
[17] D. Pham and P. Pham, "Artificial intelligence in engineering," *Int. J. Mach. Tools Manufacture*, vol. 39, no. 6, pp. 937–949, 1999.
[18] M. Zhao, M. Kang, B. Tang, and M. Pecht, "Deep residual networks with dynamically weighted wavelet coefficients for fault diagnosis of planetary gearboxes," *IEEE Trans. Ind. Electron.*, vol. 65, no. 5, pp. 4290–4300, May 2018.
[19] A. Ebrahimkhanlou and S. Salamone, "Single-sensor acoustic emission source localization in plate-like structures using deep learning," *Aerospace*, vol. 5(2), no. 50, pp. 1–22, 2018.
[20] A. Ebrahimkhanlou, B. Dubuc, and S. Salamone, "Damage localization in plate-like structures using guided ultrasonic waves edge reflections," *Structural Health Monit.*, vol. 2015, no. 2, pp. 2521–2528, 2015.
[21] A. Ebrahimkhanlou and S. Salamone, "Single-sensor acoustic emission source localization in plate-like structures: A deep learning approach," *Proc. SPIE*, vol. 10600, 2018, Art. no. 106001O.
[22] R. Liu, G. Meng, B. Yang, C. Sun, and X. Chen, "Dislocated time series convolutional neural architecture: An intelligent fault diagnosis approach for electric machine," *IEEE Trans. Ind. Inf.*, vol. 13, no. 3, pp. 1310–1320, Jun. 2017.
[23] W. Zhang, C. Li, G. Peng, Y. Chen, and Z. Zhang, "A deep convolutional neural network with new training methods for bearing fault diagnosis under noisy environment and different working load," *Mech. Syst. Signal Process.*, vol. 100, pp. 439–453, 2018.
[24] J. Jiao, M. Zhao, J. Lin, and C. Ding, "Deep coupled dense convolutional network with complementary data for intelligent fault diagnosis," *IEEE Trans. Ind. Electron.*, to be published, doi: 10.1109/TIE.2019.2902817.
[25] T. Pan, J. Chen, Z. Zhou, C. Wang, and S. He, "A novel deep learning network via multi-scale inner product with locally connected feature extraction for intelligent fault detection," *IEEE Trans. Ind. Inform.*, to be published, doi: 10.1109/TII.2019.2896665.
[26] T. Han, C. Liu, W. Yang, and D. Jiang, "A novel adversarial learning framework in deep convolutional neural network for intelligent diagnosis of mechanical faults," *Knowl.-Based Syst.*, vol. 165, pp. 474–487, 2019.
[27] N. Li, Y. Lei, L. Guo, T. Yan, and J. Lin, "Remaining useful life prediction based on a general expression of stochastic process models," *IEEE Trans. Ind. Electron.*, vol. 64, no. 7, pp. 5709–5718, Jul. 2017.
[28] E. Zio and F. Di Maio, "A data-driven fuzzy approach for predicting the remaining useful life in dynamic failure scenarios of a nuclear system," *Rel. Eng. Syst. Saf.*, vol. 95, no. 1, pp. 49–57, 2010.
[29] C. Zhang, P. Lim, A. Qin, and K. C. Tan, "Multiobjective deep belief networks ensemble for remaining useful life estimation in prognostics," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 10, pp. 2306–2318, Oct. 2017.
[30] N. Li, Y. Lei, J. Lin, and S. X. Ding, "An improved exponential model for predicting remaining useful life of rolling element bearings," *IEEE Trans. Ind. Electron.*, vol. 62, no. 12, pp. 7762–7773, Dec. 2015.
[31] X.-S. Si, W. Wang, M.-Y. Chen, C.-H. Hu, and D.-H. Zhou, "A degradation path-dependent approach for remaining useful life estimation with an exact and closed-form solution," *Eur. J. Oper. Res.*, vol. 226, no. 1, pp. 53–66, 2013.
[32] J. Yu, "Health degradation detection and monitoring of lithium-ion battery based on adaptive learning method," *IEEE Trans. Instrum. Meas.*, vol. 63, no. 7, pp. 1709–1721, Jul. 2014.
[33] R. Khelif, B. Chebel-Morello, S. Malinowski, E. Laajili, F. Fnaiech, and N. Zerhouni, "Direct remaining useful life estimation based on support vector regression," *IEEE Trans. Ind. Electron.*, vol. 64, no. 3, pp. 2276–2285, Mar. 2017.
[34] J. Wei, G. Dong, and Z. Chen, "Remaining useful life prediction and state of health diagnosis for lithium-ion batteries using particle filter and support vector regression," *IEEE Trans. Ind. Electron.*, vol. 65, no. 7, pp. 5634–5643, Jul. 2018.
[35] J. B. Ali, B. Chebel-Morello, L. Saidi, S. Malinowski, and F. Fnaiech, "Accurate bearing remaining useful life prediction based on Weibull distribution and artificial neural network," *Mech. Syst. Signal Process.*, vol. 56, pp. 150–172, 2015.
[36] L. Guo, N. Li, F. Jia, Y. Lei, and J. Lin, "A recurrent neural network based health indicator for remaining useful life prediction of bearings," *Neurocomputing*, vol. 240, pp. 98–109, 2017.
[37] D. A. Tobon-Mejia, K. Medjaher, N. Zerhouni, and G. Tripot, "A data-driven failure prognostics method based on mixture of Gaussians hidden Markov models," *IEEE Trans. Rel.*, vol. 61, no. 2, pp. 491–503, Jun. 2012.
[38] P. Baruah and R. B. Chinnam, "HMMs for diagnostics and prognostics in machining processes," *Int. J. Prod. Res.*, vol. 43, no. 6, pp. 1275–1293, 2005.
[39] Z. Zhao, B. Liang, X. Wang, and W. Lu, "Remaining useful life prediction of aircraft engine based on degradation pattern learning," *Rel. Eng. Syst. Saf.*, vol. 164, pp. 74–83, 2017.
[40] M. Djeziri, S. Benmoussa, and R. Sanchez, "Hybrid method for remaining useful life prediction in wind turbine systems," *Renewable Energy*, vol. 116, pp. 173–187, 2018.
[41] R. Caruana, "Multitask learning," *Mach. Learn.*, vol. 28, no. 1, pp. 41–75, 1997.
[42] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, "Joint face detection and alignment using multitask cascaded convolutional networks," *IEEE Signal Process. Lett.*, vol. 23, no. 10, pp. 1499–1503, Oct. 2016.
[43] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.
[44] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 2818–2826.
[45] 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 Vibration*, vol. 289, no. 4/5, pp. 1066–1090, 2006.
[46] A. Ginart, I. Barlas, J. Goldin, and J. L. Dorrity, "Automated feature selection for embeddable prognostic and health monitoring (PHM) architectures," in *Proc. IEEE Autotestcon*, 2006, pp. 195–201.

Ruonan Liu (M'19) received the B.A., M.A., and the Ph.D. degrees in mechanical manufacture and automation from the School of Mechanical Engineering, Xi'an Jiaotong University, Xi'an, China.

She is currently a Postdoctoral Researcher with Language Technologies Institute, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA. Her research interests include intelligent manufacturing, natural language processing, computer vision, and machine learning.

Boyuan Yang received the B.A., M.A., and the Ph.D. degrees in mechanical manufacture and automation from the School of Mechanical Engineering, Xi'an Jiaotong University, Xi,an, China.

He is currently a Postdoctoral Researcher with the School of Electrical and Electronic Engineering, University of Manchester, Manchester, U.K. His research interests include intelligent manufacturing, machine learning, condition monitoring, and wind energy.

Alexander G. Hauptmann received the B.A. and M.A. degrees in psychology from Johns Hopkins University, Baltimore, MD, USA, in 1982, the M.S. degree in computer science from the Technische Universitat Berlin, Berlin, Germany, in 1984, and the Ph.D. degree in computer science from Carnegie Mellon University (CMU), Pittsburgh, PA, USA, in 1991.

He is currently with the Faculty of the Department of Computer Science and the Language Technologies Institute, CMU. From 1984 to 1994, he was working on speech and machine translation, when he joined the Informedia project for digital video analysis and retrieval, and led the development and evaluation of news-on-demand applications. His research interests include several different areas: man–machine communication, natural language processing, speech understanding and synthesis, video analysis, and machine learning.