Using Deep Learning-Based Approach to Predict Remaining Useful Life of Rotating Components

Jason Deutsch and David He

***Abstract*—In the age of Internet of Things and Industrial 4.0, prognostic and health management (PHM) systems are used to collect massive real-time data from mechanical equipment. PHM big data has the characteristics of large-volume, diversity, and high-velocity. Effectively mining features from such data and accurately predicting the remaining useful life (RUL) of the rotating components with new advanced methods become issues in PHM. Traditional data driven prognostics is based on shallow learning architectures, requires establishing explicit model equations and much prior knowledge about signal processing techniques and prognostic expertise, and therefore is limited in the age of big data. This paper presents a deep learning-based approach for RUL prediction of rotating components with big data. The presented approach is tested and validated using data collected from a gear test rig and bearing run-to-failure tests and compared with existing PHM methods. The test results show the promising RUL prediction performance of the deep learning-based approach.**

***Index Terms*—Bearing, condition monitoring, gearbox, machine learning, prediction methods, prognostics.**

I. Introduction

IN THE age of Internet of Things and Industrial 4.0, prognostic and health management (PHM) systems are used to collect massive real-time data from mechanical equipment. Mechanical big data has the characteristics of large volume, diversity, and high velocity. Remaining useful life (RUL) has been used as an important parameter for condition-based maintenance decision making [1]. Effectively mining features from such data and accurately predicting RUL of the rotating components of the equipment in use with new advanced methods become issues for PHM.

In recent years, many RUL prediction methods have been proposed for PHM. These methods can be classified as model-based, data-driven, or combination of the two. Model-based approaches rely on the knowledge of the inherent system failure mechanism to build a degradation model to describe the physical nature of the failure. Data-driven approaches on the other hand, use massive data to find a degradation law without knowing the physical nature of the failure mechanism. In comparison with model-based techniques, data-driven approaches can design an easily applied system when massive sensor data is available. A recent review of data-driven approaches can be found in [2]. Traditionally, data-driven prognostics is largely dependent on signal processing and feature extraction techniques. Over the past years, many hybrid prognostic methods that require explicit model equations have been developed [3]. For example, recurrent neural networks [4], [5], Kalman filter [6]–[8], dynamic Bayesian networks [9], k-reliable decentralized prognosis [10], particle filter [11]–[15], and combined particle filter and neural networks [16]. Among all the approaches, particle filters [17] have emerged in recent years as a comparatively good RUL prediction method and are becoming more and more widespread, mainly due to their capability of dealing with dynamic systems characterized with nonlinear and non-Gaussian nature. For example, [18] showed the superior RUL prediction performance of a particle filter-based approach using the gear data provided by the NASA Glenn Spiral Bevel Gear Test Facility.

However, all above-mentioned methods require either complicated signal processing techniques to extract features from the sensor data or knowledge of the system dynamics to build the explicit model equations. This type of requirement involves manual processing and analysis of data by human experts and therefore makes these methods not suitable for automatic data processing and feature extraction for big data. There also maybe situations in which these models are unavailable, such as in offshore well drilling and wind turbines or for some bearings in which online measurements of the crack depth may not be available in which the traditional particle filter approach cannot be used [12]. The recent development in deep learning has provided an attractive opportunity to build advanced RUL prediction methods for big data.

Since, the introduction of a deep belief network (DBN) [19], DBNs and other deep learning methods have become a popular approach for big data processing and analysis. Deep learning has the ability to yield useful and important features from data that can ultimately be useful for improving predictive power [20]. It has also the capability of processing big data and mining hidden information due to its multiple layer structure. The recent success of AlphaGo by Google Deepmind [21] has demonstrated the power of deep learning for big data processing and feature learning. There have been great successes in building deep neural

Manuscript received September 15, 2016; revised February 13, 2017; accepted April 18, 2017. Date of publication May 12, 2017; date of current version December 14, 2017. This paper was recommended by Associate Editor J. Wu. *(Corresponding author: David He.)*

J. Deutsch is with the Department of Mechanical and Industrial Engineering, University of Illinois at Chicago, Chicago, IL 60607 USA.

D. He is with the College of Mechanical Engineering and Automation, Northeastern University, Shenyang 110819, China (e-mail: davidhe@uic.edu).

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/TSMC.2017.2697842

2168-2216 © 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

Authorized licensed use limited to: UNIVERSIDADE FEDERAL DE GOIAS. Downloaded on May 10,2023 at 18:32:50 UTC from IEEE Xplore. Restrictions apply.

network architectures in various domains, such as image recognition, automatic speech recognition, natural language processing [22], and many more. It has also recently shown promising results for machine fault diagnostics on extraction of raw vibration signals [23] as well as time-domain features [24]. Although much success in deep learning has been focused on classification problems, deep learning has also proven to be successful in solving prediction problems. These domains include predicting car traffic [25], weather [26], wind speed [27], and Internet traffic [28]. There are many types of deep learning algorithms present including auto-encoders, restricted Boltzman machines (RBNs), DBNs, convolutional neural networks, and more that can also be used for prediction problems. In summary, deep learning represents an attractive option to process mechanical big data for RUL prediction as it has the ability to automatically learn features that otherwise require much skill, time, and experience. Up to date, no research on RUL prediction using deep learning-based approaches for rotating components, such as gears and bearings has been reported in the literature.

In this paper, a deep learning-based approach for RUL prediction using vibration sensors is presented. The developed deep learning-based approach is a DBN-feedforward neural network (FNN) algorithm that takes the advantages of self-taught feature learning capability of the DBN and the predicting power of the FNN. It can either take processed vibration features or extract features from the vibration data to predict the RUL. The presented method overcomes the above mentioned limitations of the traditional data-driven approaches and particle filter-based approaches to allow automatic feature extraction and RUL prediction without human intervention in the age of big data. The presented approach is tested and validated using data collected from a gear test rig and bearing run-to-failure tests and compared with existing PHM methods.

II. Methodology

A DBN, which is a stacked version of an RBM, is used in this paper in order to predict the RUL at L steps ahead in the future. Next, RBM, DBN, and DBN-FNN algorithm for RUL prediction are explained.

A. Restricted Boltzman Machine

An RBM [29] is a type of unsupervised machine learning algorithm. It is a generative stochastic artificial neural network that learns a probability distribution over a set of its inputs. An RBM is a bipartite graph, which contains undirected edges connecting its two layers: 1) a visible and 2) a hidden layer. Each layer contains a collection of neurons/nodes. The visible layer consists of the data's input, where each node/neuron represents a feature of the data. The hidden layer represents the latent variables. As shown in Fig. 1, an RBM is "restricted" because there are no connections between each neuron/node within either the visible or hidden layer. An RBM contains a matrix of weights W_{ij} representing the connection between visible node v_i and hidden node h_j. We will let a_i represent the bias term for the visible layer and b_j for the hidden layer.

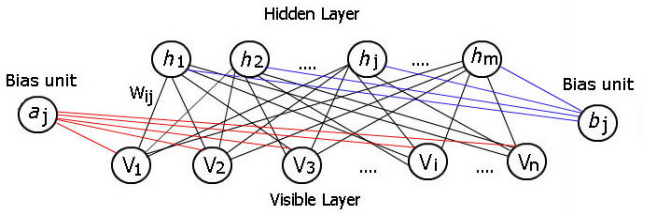

Fig. 1. Restricted Boltzmann machine.

The weights and biases are computed by maximizing $P(\boldsymbol{v})$, which is the probability that the network assigns to a visible vector $\boldsymbol{v}$

$$P(\boldsymbol{v}) = \frac{1}{Z}\sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})} \tag{1}$$

where Z is the normalization constant which can be found by summing over all the possible pairs of visible and hidden vectors

$$Z = \sum_{\boldsymbol{v}}\sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})} \tag{2}$$

and the energy function of the joint configuration $(\boldsymbol{v}, \boldsymbol{h})$ is given by

$$E(\boldsymbol{v}, \boldsymbol{h}) = -\boldsymbol{a}^T\boldsymbol{v} - \boldsymbol{b}^T\boldsymbol{h} - \boldsymbol{v}^T\boldsymbol{W}\boldsymbol{h}. \tag{3}$$

The maximization of (1) can in theory, be determined by taking its partial log derivative with respect to its parameters $\boldsymbol{W}$, $\boldsymbol{a}$, and $\boldsymbol{b}$

$$\frac{\partial \log(P(\boldsymbol{v}))}{\partial \boldsymbol{W}, \boldsymbol{a}, \boldsymbol{b}} = \sum_{\boldsymbol{v},\boldsymbol{h}} E(v, h) - \sum_{h} E(v, h). \tag{4}$$

Typically (4) is also written as

$$\frac{\partial \log(P(\boldsymbol{v}))}{\partial \boldsymbol{W}, \boldsymbol{a}, \boldsymbol{b}} = <v_i h_j>_{\text{data}} - <v_i h_j>_{\text{model}} \tag{5}$$

where $<*>$ denotes the expectation of $*$. However, the expectation $<v_i h_j>_{\text{model}}$ in the maximum log likelihood function cannot be easily computed and is thus estimated using contrastive divergence which leads to the following parameter update equation [30]:

$$\begin{aligned}
W_{ij}^{\,k} &= W_{ij}^{\,k} + \mathrm{N}\Big(<v_i^{\,k} h_j>_{\text{data}} - \ <v_i^{\,k} h_j>_T\Big) \\
&= P\big(h_j = 1|\boldsymbol{v}\big)v_i - v_i^{\,k} P\Big(h_j = 1|\boldsymbol{v}^k\Big) \\
a_i^{\,k} &= a_i^{\,k} + \mathrm{N}\Big(<v_i^{\,k}>_{\text{data}} - \ <v_i^{\,k}>_T\Big) \\
&= P(v_i = 1|\boldsymbol{h}) - v_i^{\,k} \\
b_j^{\,k} &= b_j^{\,k} + \mathrm{N}\Big(<h_j>_{\text{data}} - <h_j^{\,k}>_T\Big) \\
&= P\big(h_j = 1|\boldsymbol{v}\big) - P\Big(h_j = 1|\boldsymbol{v}^k\Big)
\end{aligned} \tag{6}$$

where T represents a full step of Gibbs sampling, N represents the learning rate, and k represents the $k-step$ of contrastive divergence. The neuron activation probabilities are given by the following equations:

$$P\big(h_j = 1|\boldsymbol{v}\big) = \sigma\left(b_j + \sum_{i=1}^{n} W_{ij}v_i\right) \tag{7}$$

Authorized licensed use limited to: UNIVERSIDADE FEDERAL DE GOIAS. Downloaded on May 10,2023 at 18:32:50 UTC from IEEE Xplore. Restrictions apply.

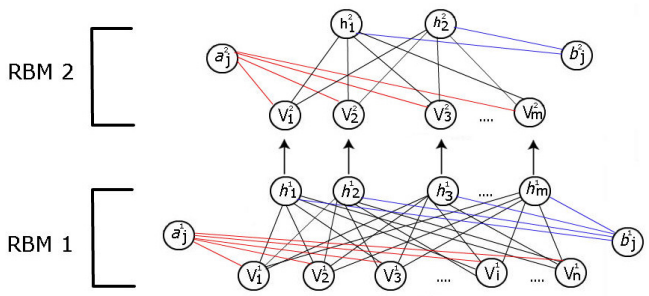

Fig. 2. DBN with two hidden layers.

$$P(v_i = 1|\boldsymbol{h}) = \sigma\left(a_i + \sum_{j=1}^{m} W_{ij} h_j\right) \tag{8}$$

where n represents the number of visible units, m the number of hidden units, and σ is the activation function. The activation function is typically the logistic function used as a threshold defined as

$$\sigma(x) = \frac{1}{1 + e^{-x}}. \tag{9}$$

B. Deep Belief Network

A DBN is formed by stacking multiple RBMs on top of each other in order to create high representations of data that can be used for classification, regression (continuous output) tasks as well as unsupervised learning (see Fig. 2).

The RBM becomes a building block for forming a DBN. The DBN can be trained in a greedy-layer-wise fashion by stacking RBMs on top of each other [31]. The output of one RBM, that is the activation values in the hidden layer, simply becomes the input for the next RBM and the parameters of the previous RBM do not change. This next RBM is trained by the same process as illustrated in the previous section. This process allows for creating multiple hidden layers.

C. DBN-FNN for RUL Prediction

In order to build a deep learning-based prognostic model, we can first learn the weights and biases in the unsupervised stage of learning using DBN. Once the optimal parameters (weights and biases) have been determined, a supervised fine tuning stage is performed. This is done by creating a final output layer on the top of the DBN which outputs the predicted RUL value given a set of vibration features. This is illustrated in Fig. 3 below. The parameters of the entire network are then updated using the backpropagation algorithm in the same way as a FFN is trained. The DBN in this way pretrains the network which serves as an initialization step for the parameters of the FNN, instead of a random initialization of the weights and biases, which has been shown to add robustness to deep architectures and decrease the probability of obtaining poor local minima [32].

The DBN–FNN performs unsupervised learning first by training DBN on a set of fault features in order to learn a latent representation of the data. After the end of the training, the DBN–FNN is formed by adding an output layer on top of the last layer of the DBN, where it is fully connected to the last hidden layer. This output layer contains one neuron which represents the continuous prediction $\widehat{\text{RUL}}_{t+L}$ for some time t at L-steps ahead into the future.

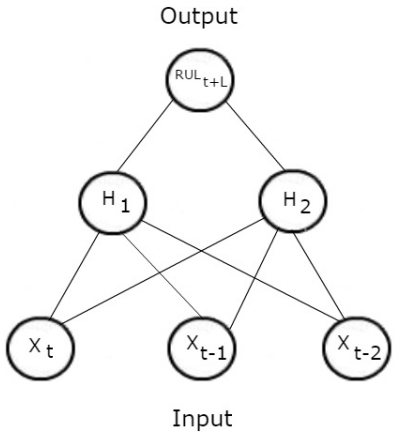

Fig. 3. FFN with $d = 3$ and a single hidden layer.

The time series of the fault features X_t is reconstructed, into a matrix, where each feature (column) represents a lagged order of the time series, and whose output is the L-step ahead into the future RUL value, and each row represents an index in time. Formally, the input can be denoted as

$$\left[X_t, X_{t-1}, \ldots, X_{t-d+1}\right], \epsilon \mathbb{R}^d \tag{10}$$

and the output as

$$\left[\text{RUL}_{t+L}, \text{RUL}_{t+L+1}, \ldots, \text{RUL}_n\right] \tag{11}$$

where d represents the embedding dimension, and determines the size of the first visible layer in the DBN–FNN. Thus, yielding training tuples of $([X_t, X_{t-1}, \ldots, X_{t-d+1}], \text{RUL}_{t+L})$, $t = d-1, \ldots, n-L-1$.

Equations (10) and (11) define a method often called the sliding window technique [33]. This windowing approach is best illustrated below in Fig. 4.

The first plot in Fig. 4 is the complete series of a fault feature. The second plot highlights the first window of size $d = 100$. The last plot simply shows the first window zoomed in. The first window contains points from $t = 0, 1, 2, \ldots, 98, 99$ (first row of features) and the second window would contain points from $t = 1, 2, 3, \ldots, 99, 100$ (second row of features) and so on.

The predicted RUL can then be directly estimated as a function of the fault feature by the following equation:

$$\widehat{\text{RUL}}_{t+L} = \phi(X_t, X_{t-1}, \ldots, X_{t-d+1}) \tag{12}$$

where $\widehat{\text{RUL}}_{t+L}$ is the predicted RUL at some time $t + L$, and $\phi(\cdot)$ is a function of the input space that is to be learned by the DBN–FNN.

Authorized licensed use limited to: UNIVERSIDADE FEDERAL DE GOIAS. Downloaded on May 10,2023 at 18:32:50 UTC from IEEE Xplore. Restrictions apply.

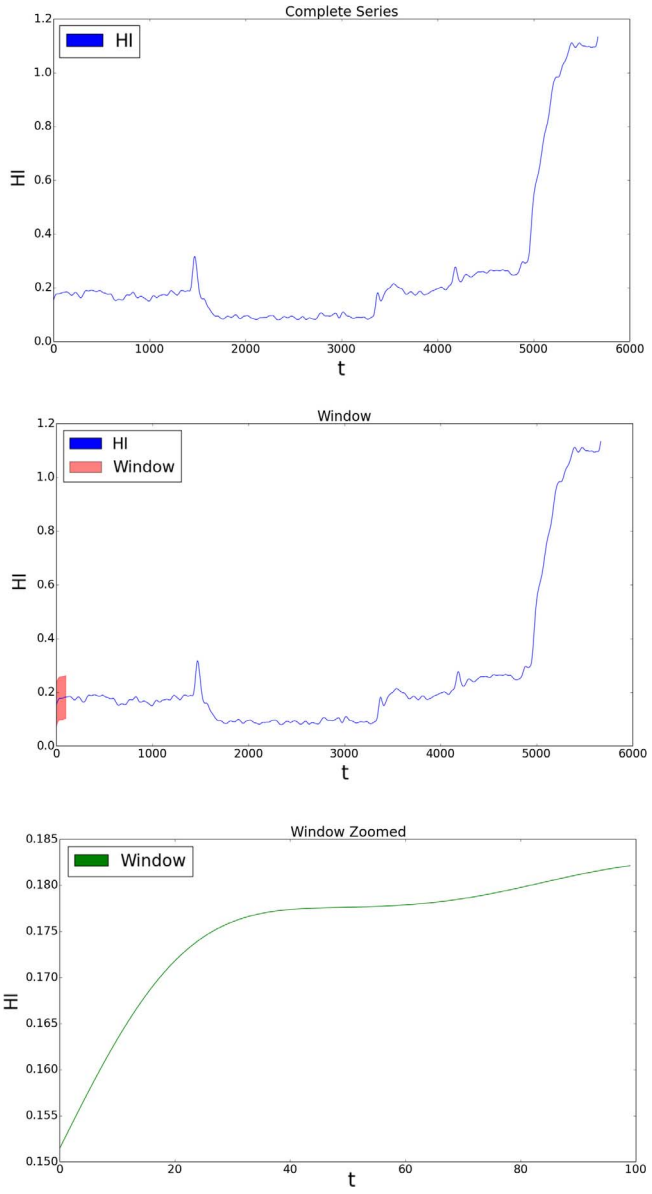

Fig. 4. Windowing approach.

Confidence bounds for the predictions can be obtained by a resampling technique known as a jackknife, which is a linear approximation to the bootstrap method [34]. The method is described next.

Let $\phi(\cdot)_{-i}$ represent the DBN–FNN prediction of the RUL when a single sample i (row) is deleted from the training set. We will simply call this a jackknife sample. Let $\overline{\phi(\cdot)}$ be the average across n jackknife samples

$$\overline{\phi(\cdot)} = \frac{1}{n}\sum_{i=1}^{n}\phi(\cdot)_{-i}. \tag{13}$$

And the estimate of the standard error of the mean is defined as

$$\widehat{\sigma} = \sqrt{\frac{n-1}{n}\sum_{i=1}^{n}\left[\overline{\phi(\cdot)} - \phi(\cdot)_{-i}\right]^2}. \tag{14}$$

The confidence interval with $N-1$ degrees of freedom can then be calculated as follows:

$$\overline{\phi(\cdot)} \pm t_{1-(\alpha/2),N-1}\widehat{\sigma}. \tag{15}$$

The DBN–FNN algorithm for RUL prediction is as follows.

Step 1: Reconstruct the time series of the fault features into a matrix with an embedding dimension of d. This will serve as the input and the output will be the mapped RUL, RUL_{t+L}.

Step 2: Randomly delete one sample (one row) from the data, including the target output. This will serve as one jackknife sample.

Step 3: Initialize the weights and biases of the FNN by training the DBN on the input data using all the data except for the last 100 rows. The last 100 rows will serve as the testing dataset and the rest of the data is the training set.

Step 4: Train the FNN. Fine tune the weights in a supervised fashion by minimizing the loss function on the training set and by using the backpropagation algorithm.

Step 5: Predict the RUL_{t+L}.

Step 6: Let $t = t+1$, and update the training set with input features $(X_t, X_{t-1}, \ldots, X_{t-d+1})$ and output RUL_{t-L}.

Step 7: Repeat steps 4–6, until all 100 points have been predicted.

Step 8: Repeat steps 2–7 for n number of jackknife samples. In step 2, the previous deleted sample is replaced in the data and a new sample is deleted.

In the above DBN–FNN algorithm, step 6 is somewhat analogous to the updating step in the particle filter-based approach [35]. As new information becomes available the weights and biases of the network change (from step 7) to reflect the updated information.

III. Case Studies

To validate the deep learning-based RUL prediction method, vibration data collected at the NASA Glenn Spiral Bevel Gear Test Facility and from hybrid ceramic bearing run-to-failure tests were used.

A. Spiral Bevel Gear RUL Prediction

1) Spiral Bevel Gear Data: Gear tests were performed on a bevel gear test rig at the NASA Glenn Spiral Bevel Gear Test Facility and vibration data were collected during the gear tests. A detailed description of the test rig and test procedure is given in [36]. The rig as shown in Fig. 5 was used to quantify the performance of gear material, gear tooth design, and lubrication additives on the fatigue strength of the gears. During the testing, vibration condition indictors (CIs) and oil debris mass (ODM) data were used to detect pitting damage on spiral bevel gears.

The tests consisted of running the gears under load through a "back to back" configuration, with vibration data collected once per minute using a sampling rate of 100 kHz for a 2 s duration, generating time synchronous averages (TSAs) on the gear shaft (36 teeth). The pinion, on which the damage

Authorized licensed use limited to: UNIVERSIDADE FEDERAL DE GOIAS. Downloaded on May 10,2023 at 18:32:50 UTC from IEEE Xplore. Restrictions apply.

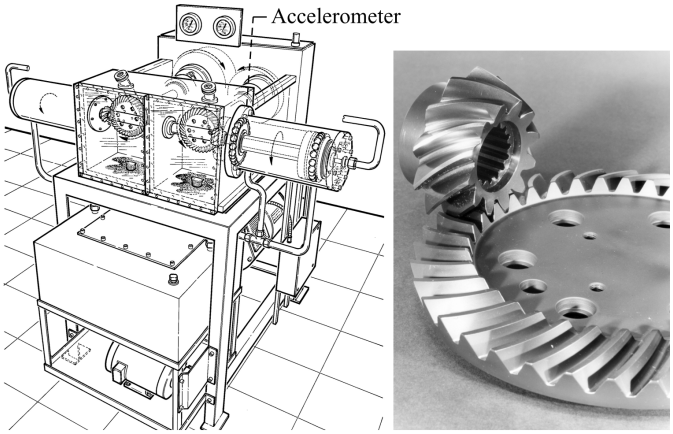

Fig. 5. Bevel gear test rig and bevel gears [36].

Fig. 6. Damaged spiral bevel gear.

occurred, has 12 teeth. The tests were performed for a specific number of hours or until surface fatigue occurs. In this paper, data collected from experiment 5 were used. At test completion of the experiments, destructive pitting could be observed on the teeth of the pinions (see Fig. 6).

CIs obtained from vibration data are commonly used for mechanical fault detection and diagnosis. For example, the health and usage monitoring system currently installed in helicopters utilizes a large number of vibration-based CIs. As pointed out in [37], there is no single CI that is sensitive to every failure mode of a gear. Combining multiple CIs into one health index (HI) is an attractive approach for gear health prognostics. TSA data was processed with gear CI algorithms presented in [38] and [39]. A total of 6 CIs were used for computing a 1-D HI as the fault feature in order to predict the RUL: residual root mean square (RMS), energy operator RMS, FM0, narrowband kurtosis, amplitude modulation kurtosis, and frequency modulation RMS. To compute the 1-D HI, the set of correlated CIs were first de-correlated by applying the Cholesky decomposition. The Cholesky decomposition of a Hermitian, a positive definite matrix results in $\mathbf{A} = \mathbf{L}\mathbf{L}^*$, where $\mathbf{L}$ is a lower triangular, and $\mathbf{L}^*$ is its conjugate transpose. Let $\mathbf{F}$ be a set of correlated CIs. It then follows that

$$\mathbf{L}\mathbf{L}^* = \Sigma^{-1} \tag{16}$$

and

$$\mathbf{Y} = \mathbf{L} \times \mathbf{F}^T \tag{17}$$

where $\mathbf{Y}$ is a vector of n independent CIs with unit variance and $correlation(\mathbf{Y}) = 0$. Equation (11) creates the necessary independent and identical distributions (IIDs) required to define the HI for a function of distributions.

Assuming that the distributions of the CIs follow a Gaussian distribution, then three statistical HI generation models can be used: 1) the Gaussian order statistic; 2) the sum of n Gaussian; and 3) the total energy of n Gaussian. These three models are explained as follows [37].

1) The HI is defined as the Gaussian order statistic, it can be computed as the following:

$$\mathbf{Y} = \mathbf{L} \times \left(\mathbf{F}^T - \mathbf{m}\right) \tag{18}$$

$$\mathrm{HI} = \frac{(\max\{\mathbf{Y}\} + .34) * .5}{(3.41 + 0.34)} \tag{19}$$

where $\mathbf{m}$ is the mean of $\mathbf{F}$. Subtracting the mean and multiplying by $\mathbf{L}$ transforms the features into n, Z distributions (zero mean, IID Gaussian distributions).

2) When the HI is defined as the sum of n Gaussian, it can be computed as the following:

$$\mathbf{Y} = \mathbf{L} \times \mathbf{F}^T$$
$$\mathrm{HI} = 0.5/(8.352 - 0.15)\left(-0.15 + \sum_{i=1}^{n} \mathbf{Y}_i\right). \tag{20}$$

3) When the HI is defined as the total energy of n Gaussian, it can be computed as follows:

$$\mathbf{Y} = \mathbf{L} \times \mathbf{F}^T$$
$$\mathrm{HI} = 0.5/(3.368)\sqrt{\sum_{i=1}^{n} \mathbf{Y}_i^2}. \tag{21}$$

The HI used for RUL prediction was the order statistic defined by (19) as the order statistic gave a more consistent trending of the HI than other statistics [14]. To predict the RUL of the gear, here HI at time t is set as fault feature X_t. The RUL_t for the gear data was calculated by finding the first index of time in which the $X_t \geq 1.0$, which can be assumed is when the gear has failed and is denoted as T'_{end}. Since, the fault feature and the ODM (assumed to be the true state) are correlated, the total life of the gear is the time index of the ODM that correlates with $X_{T'_{\mathrm{end}}}$ and is denoted as T_{end}. This process is shown below in Figs. 7 and 8.

The RUL at each time step can then be calculated by a linear interpolation using the following equation:

$$\mathrm{RUL}_t = T_{\mathrm{end}} - t\left(\frac{T_{\mathrm{end}}}{T'_{\mathrm{end}}}\right). \tag{22}$$

Since, the fault feature and the ODM are not a one to one mapping, (22) allows to map the RUL by the length of the fault feature.

2) Spiral Bevel Gear RUL Prediction Results: A total of 5670 time steps were extracted from the gear data, in which all the data up until $T'_{\mathrm{end}} = 5280$ was used. We set the last

Authorized licensed use limited to: UNIVERSIDADE FEDERAL DE GOIAS. Downloaded on May 10,2023 at 18:32:50 UTC from IEEE Xplore. Restrictions apply.

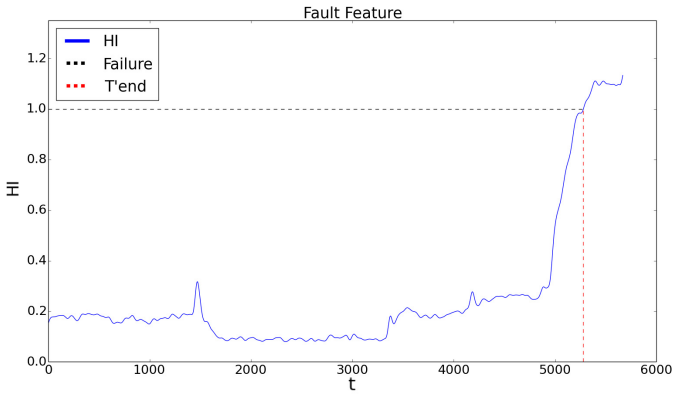

Fig. 7. Determining $T_{\text{end}}^{'}$ from the fault feature.

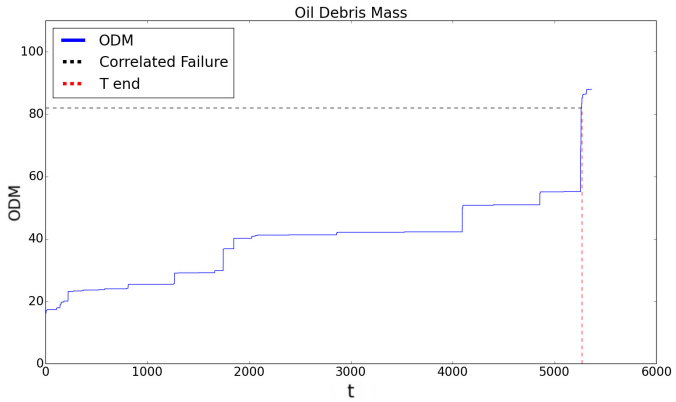

Fig. 8. Determining T_{end} from the ODM.

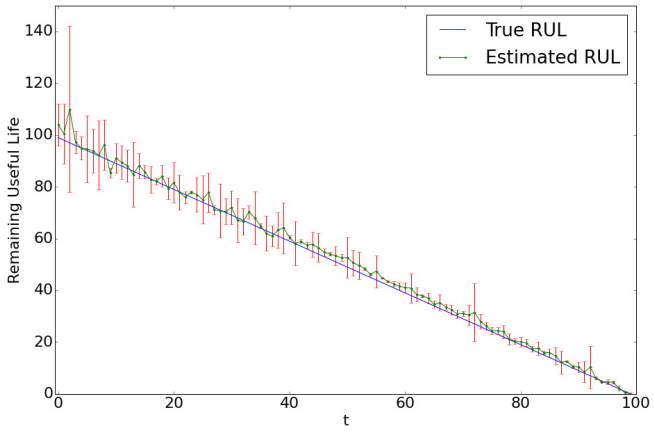

Fig. 9. Plot of gear $\widehat{\text{RUL}}_t$ values with with $L = 1$.

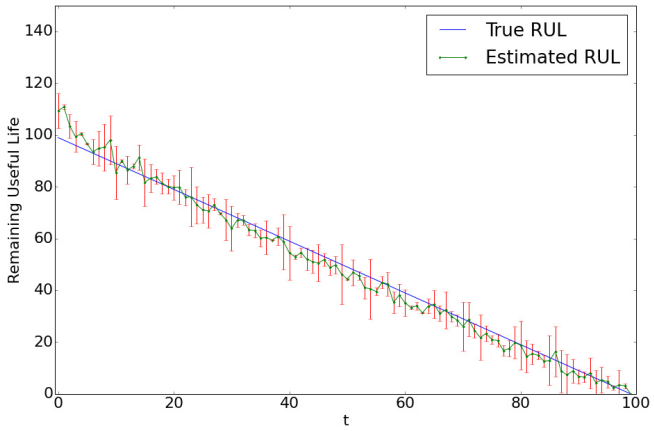

Fig. 10. Plot of gear $\widehat{\text{RUL}}_t$ values with $L = 10$.

100 points of the gear's life as the testing set. For the training set, the last 1280 time steps (elimination of the first 4000 time steps) of data were used for $L = 1$ and 580 points (elimination of the first 4700 points) of data were used for $L = 10$ in the gear data.

The reason for the above setting was twofold: 1) speed of the training/testing is the key to finding appropriate hyperparameters and 2) some removal of noise, since large regions of the fault features are flat and contain a similar input space with different output RUL values.

Since, the DBN assumes a binary input or a real valued input in [0, 1] the input values were scaled to be in [0, 1]. The predictions for $L = 1$ and $L = 10$ are shown below in Figs. 9 and 10, respectively. The error metrics and hyperparamters for predictions with $L = 1$ and $L = 10$ are provided in Tables I and II, respectively. The L step ahead predictions for $L = 1$ predict approximately 1 min into the future and 10 min into the future for $L = 10$.

In Figs. 9 and 10, the green color represents the average predicted RUL values across the jackknife samples. The red error bars represent the 90% confidence bounds.

For both L step ahead predictions it can be seen that they accurately model the true RUL, while the $L = 1$ predictions are less varied than that for $L = 10$. The confidence bounds for $L = 10$ prediction seem to predict the RUL of the gear slightly early when compared to the $L = 1$ predictions.

The common error metrics used in Table II are the root mean squared error (RMSE) and the mean absolute percentage error (MAPE). The MAPE and RMSE are defined by the following equations:

$$\text{MAPE} = \left(\frac{1}{n} \sum_{t=1}^{n} \frac{A_t - F_t}{A_t} \right) * 100\% \tag{23}$$

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{t=1}^{n} (A_t - F_t)^2}. \tag{24}$$

In (23) and (24), A_t = actual value and F_t = predicted value from the DBN–FNN and n = the number of points to be predicted.

The hyperparameters were mostly the same for both L step predictions with 130 hidden neurons in the first layer and 30 neurons in the second layer and were chosen by using a grid search. A large embedding dimension of $d = 100$ was set which was empirically found to yield good results. Since, step 7 of the DBN–FNN algorithm requires 100 different FNNs to be trained, the hyperparameters of the model were found more efficiently by simply training one DBN–FNN and testing the model on the last 100 points and using a grid search that performs exhaustive search throughout a set of hyperparameters to find good candidate solutions. The hyperparameters chosen were simply those that minimized the RMSE on the testing set. The assumption is that if a set of

Authorized licensed use limited to: UNIVERSIDADE FEDERAL DE GOIAS. Downloaded on May 10,2023 at 18:32:50 UTC from IEEE Xplore. Restrictions apply.

TABLE I
RMSE AND MAPE RESULTS FOR GEAR DATA

Deep learning based approach				
L	RMSE		MAPE	
	Average	σ	Average	σ
1	2.54	0.49	6.70%	1.07
10	3.35	1.03	10.04%	2.77
Particle filter based approach				
L	RMSE		MAPE	
	Average	σ	Average	σ
1	2.62	1.12	7.14%	1.54
10	3.48	2.31	10.87%	3.21

TABLE II
DBN–FNN HYPERPARAMETERS FOR GEAR DATA

L	DBN learning rate	DBN epochs	Hidden layer structure	BPNN learning rate	BPNN epochs
1	0.001	200	[130,30]	0.001	125
10	0.001	200	[130,30]	0.001	75

parameters can fit the testing set well without an updating step, then with an updating step it should perform better.

The best results were found by using the MAPE as a loss function instead of the more typical mean squared error loss function used for regression problems. The *Adam* optimizer [40], which is a stochastic optimization algorithm was also used for all of the predictions and yielded good results without having to fine tune the learning rate, which is due in part to the optimizer's ability to adaptively change the learning rate. The exponential decay rates and epsilon values were set to the recommended values as described in this paper. The rectified linear unit (ReLU) activation function for $L = 1$ was used as it empirically provided good results and it solves the vanishing gradient problem that other nonlinear activation functions can cause [41]. The ReLU activation function for the input x of a neuron is defined as

$$\mathrm{ReLU}(x) = \max\{0, x\}. \tag{25}$$

For $L = 10$ on the gear data we found the best results using a Leaky ReLU (LReLU) [42] defined as

$$\mathrm{LReLU}(x) = \begin{cases} x, & x > 0 \\ x\alpha, & x \leq 0 \end{cases} \tag{26}$$

where α is some value which allows for a small nonzero gradient when the neuron is saturated and not active. We found the best results by setting it to a very small value of 0.001.

For a comparison purpose, the RMSE and MAPE of the predicted RUL obtained by the particle filter-based approach are also provided in Table I. The standard deviations (denoted as σ) for both the deep learning-based approach and the particle filter are provided, which are based on the resampled estimates of the RUL, where as the averages are based on the average predicted value. In comparison with the results obtained using the deep learning-based approach, the average RMSE and MAPE values of the particle filter-based approach and their corresponding standard deviations are slightly higher. This comparison result indicates that the RUL prediction accuracy and reliability of the prediction results obtained by the deep learning-based approach are slightly better than the particle filter-based approach for the gear data.

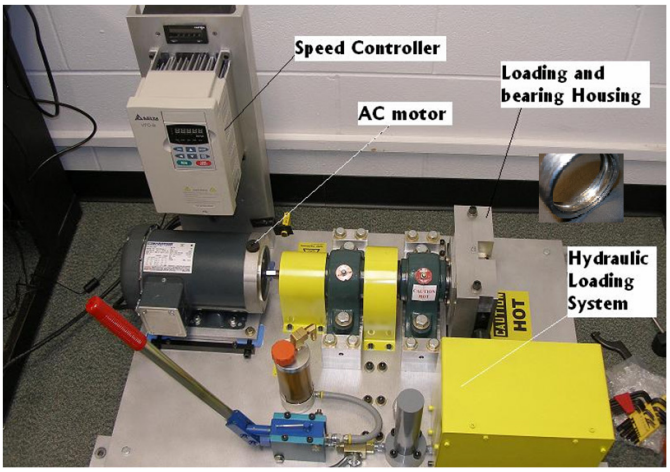

Fig. 11. Bearing run-to-failure test rig.

B. Hybrid Ceramic Bearing RUL Prediction

1) Hybrid Ceramic Bearing Test Setup: Run-to-failure tests were performed using hybrid ceramic bearings on a bearing test rig in the laboratory and vibration data were collected during the bearing run-to-failure tests. Fig. 11 shows the customized bearing run-to-failure test rig in the laboratory. The key features of the test rig include: 1) it is driven by a 3-HP ac motor with a maximum speed up to 3600 rpm and variable speed controller; 2) it is equipped with a hydraulic dynamic loading system with a maximum radial load up to 4400 lbs or 19.64 kN; and 3) an integrated loading and bearing housing that can be used for testing both ball and tapered roller bearings.

An automatic data acquisition system based on National Instrument CI 4462 board and NI LabVIEW software was constructed for data collection purposes. The automatic data acquisition system has the following features: 1) maximum sampling rate up to 102.4 kHz; 2) four input simultaneous anti-aliasing filters; 3) software-configurable ac/dc coupling and IEPE conditioning; and 4) vibration analysis functions, such as envelope analysis, cepstrum analysis, and so on for computing necessary condition indicators.

The hybrid ceramic bearings used in the test were ball bearings with stainless steel inner and outer races and ceramic balls. Two accelerometers were stunt mounted on the bearing housing in the direction perpendicular to the shaft. The test bearing was mounted on the test rig and the rig was run at a speed of 1800 rpm (30 Hz) and was subjected to a radial load of 600 psi. A sampling rate of 102.4 kHz was used for 2 s of data collection at each sampling point. The data was collected every 5 min during the test. At the end of the test, the test bearing was disassembled, checked, and photographed. The bearing contained a total of 849 data files with a length of approximately 71 h. Table III provides the run-to-failure test setting and Table IV the specification of the tested bearing.

The RMS of the vibration signals was computed to represent the degradation of the bearing over time during the run-to-failure tests. The vibration signals were preprocessed using the fast Fourier transform (FFT) and the FFT values were used as the fault feature as the input into the DBN–FNN to predict the RUL of the bearing. The RMS at each time

Authorized licensed use limited to: UNIVERSIDADE FEDERAL DE GOIAS. Downloaded on May 10,2023 at 18:32:50 UTC from IEEE Xplore. Restrictions apply.

TABLE III
RUN-TO-FAILURE TEST SETTING

Name	Type	Load (psi)	Input shaft speed (Hz)
B2	Hybrid ceramic bearing	600	30

TABLE IV
HYBRID CERAMIC BEARING SPECIFICATIONS

Parameter	Specification
Bearing material	Stainless steel 440c
Ball material	Ceramic SI3N4
Inner diameter (d)	25 m
Outer diameter (D)	52 m
Width	15 m
Enclosure	Two shields
Enclosure material	Stainless steel
Enclosure type	Removable (S)
Retainer material	Stainless steel
ABEC/ISO rating	ABEC #3 / ISOP6
Radial play	C3
Lube	Klubber L55 grease
RPM grease (x 1000 rpm)	19
RPM oil (x 1000):	22
Dynamic load (kgf)	1429
Basic Load (kgf)	804
Working temperature (° C)	121
Weight (g)	110.32

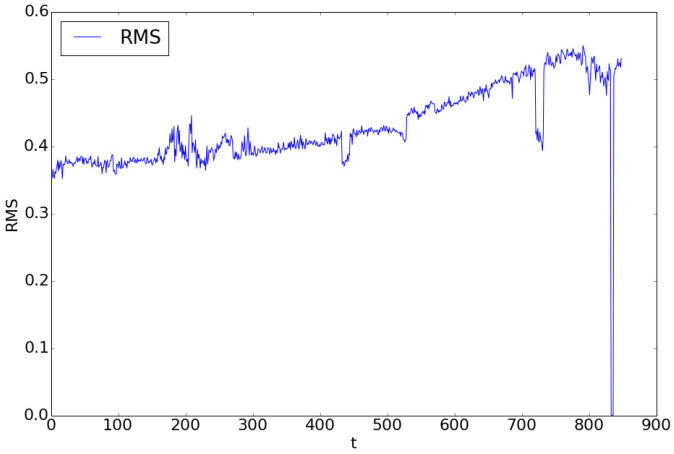

Fig. 12. Bearing RMS values.

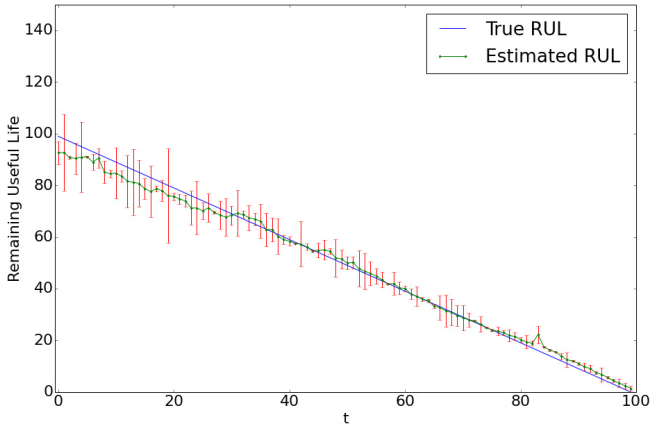

Fig. 13. Plot of bearing $\widehat{\text{RUL}}_t$ values with $L = 1$.

interval (denoted as RMS_t) can be calculated as follows:

$$\text{RMS}_t = \sqrt{\frac{1}{n}\sum_{i=1}^{n} f_{ti}^2} \tag{27}$$

where f_{ti} represents the ith raw vibration data point at time interval t and n is the length of the signal at time interval t. The RMS plot for the bearing data can be seen in Fig. 12.

The FFT, which is an efficient algorithm for computing the discrete Fourier transform at some time interval t can be calculated as follows:

$$\text{DFT}_{tn} = \sum_{k=0}^{N-1} f_{tk} e^{\frac{-2\pi ikn}{N}} \tag{28}$$

where f_{tk} is the kth raw vibration signal at time interval t, N is the length of the signal at time interval t, $i = \sqrt{-1}$ and $n = 0, 1, \ldots, N-1$.

Equation (28) transforms the vibration signals from a time domain to a frequency domain in which we extracted eight equal bands ranging from 0 to 20 kHz.

For the bearing data, the fault features were a one to one mapping and the RUL_t was calculated simply by taking the time index of the maximum recorded RMS value as the point of failure denoted as $\text{RMS}_{T_{\text{end}}}$ and subtracting it from each time step

$$\text{RUL}_t = \text{RMS}_{T_{\text{end}}} - t. \tag{29}$$

2) Hybrid Ceramic Bearing RUL Prediction Results: A total of 849 time steps were extracted from the bearing data in which all the data up until time step 792 were used in the DBN–FNN. As seen above from Fig. 12, a rather large dip in the RMS values occurs from time steps 720 through 735. Features collected from those points were simply removed from the data and treated as outliers.

$L = 1$ and $L = 10$ were used to predict 5 min and 50 min into the future for the bearing data, respectively.

The predicted RUL values for the last 100 steps can be seen in Figs. 13 and 14 for $L = 1$ and $L = 10$, respectively. The error metrics and hyperparameters for prediction with $L = 1$ and $L = 10$ are provided in Tables V and VI, respectively. Input data was also scaled to be in [0, 1], the same $d = 100$ embedding dimension, loss function, and optimizer that were used in the gear data were used for the bearing data.

In Figs. 13 and 14, the green color represents the average predicted RUL values across the jackknife samples. The red error bars represent the 90% bounds. The predicted results show that for both $L = 1$ and $L = 10$ that it can accurately predict the true RUL and as the bearing approaches the point

Authorized licensed use limited to: UNIVERSIDADE FEDERAL DE GOIAS. Downloaded on May 10,2023 at 18:32:50 UTC from IEEE Xplore. Restrictions apply.

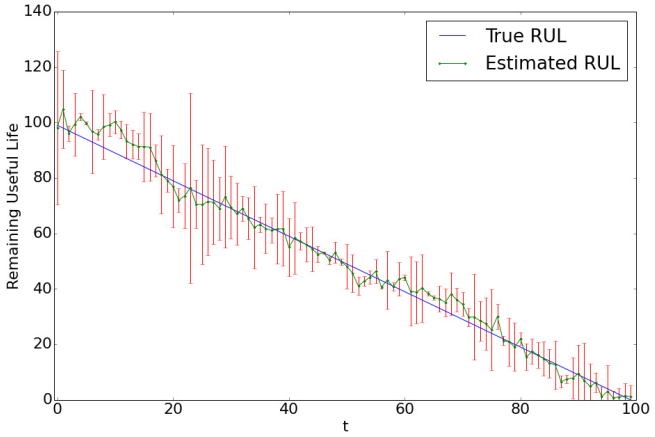

Fig. 14. Plot of bearing $\widehat{\text{RUL}}_t$ values with $L = 10$.

TABLE V
RMSE AND MAPE RESULTS FOR BEARING DATA

Deep learning based approach				
L	RMSE		MAPE	
	Average	σ	Average	σ
1	2.64	0.78	8.40%	1.28
10	3.71	1.62	9.31%	3.37
Particle filter based approach				
L	RMSE		MAPE	
	Average	σ	Average	σ
1	2.53	1.14	7.47%	2.11
10	3.65	2.08	8.73%	3.57

TABLE VI
DBN–FNN HYPERPARAMETERS FOR BEARING DATA

L	DBN learning rate	DBN epochs	Hidden layer structure	BPNN learning rate	BPNN epochs
1	0.004	50	[130,50]	0.001	79
10	0.02	100	[130,30]	0.001	79

of failure, the accuracy of the predictions tends to increase and converge.

Similar to the $L = 10$ prediction for the gear data, an LReLU was used for both the hidden layers in the bearing data, with $\alpha = 0.0085$ and $\alpha = 0.0015$, for the first and second hidden layers, respectively. Again, similar to the $L = 10$ prediction for the gear data, the confidence bounds for the $L = 10$ predicts the RUL of the bearing slightly early when compared to the $L = 1$ predictions and exhibits a greater variance. The increase of variance in comparison to the ReLU is most likely due to a slightly wider band of information that passes through each neuron when an input $x < 0$, rather than the neuron being simply being turned off.

Again, for a comparison purpose, the RMSE and MAPE of the predicted RUL obtained by the particle filter-based approach are also provided in Table V. The standard deviations for both the deep learning-based approach and the particle filter are provided, which are based on the resampled estimates of the RUL, whereas the averages are based on the average predicted value.

In comparison with the particle filter-based approach, the average RMSE and MAPE values of the deep learning-based approach are slightly higher while the corresponding standard deviation are slightly lower. This comparison result indicates that for the bearing data, the RUL prediction accuracy obtained by the deep learning-based approach is slightly worse than the particle filter-based approach but the reliability of the results is slightly better than the particle filter-based approach. Note that the RUL prediction results obtained by the particle filter-based approach were based on the features extracted using Hilbert-Huang transform which is a complex signal processing technique while the RUL prediction results obtained by the deep learning-based approach were based on the features extracted by DBN–FNN directly from the preprocessed vibration data without assuming any explicit state transition equations.

In summary, in both the gear and bearing RUL prediction validation case studies, the deep learning-based DBN–FNN has achieved the prediction accuracy that is compatible to that of the most popular RUL prediction method based on particle filters. Given that the deep learning-based approach does not require complicated signal processing and explicit model equations like the particle filter-based approach and is scalable for big data applications, the RUL prediction performance achieved by the deep learning-based DBN–FNN has shown its potential for RUL prediction for rotating components with big data.

IV. CONCLUSION

In this paper, a deep learning-based approach, a DBN–FNN algorithm for RUL prediction of rotating equipment using vibration data was presented. The developed DBN–FNN algorithm takes the advantages of self-taught feature learning capability of the DBN and the predicting power of the FNN. It can take either processed vibration features or extract features from the vibration data to predict the RUL. The presented method overcomes the above mentioned limitations of the traditional data-driven approaches and particle filter-based approaches to allow automatic feature extraction and RUL prediction without human intervention in the age of big data.

To validate the presented DBN–FNN-based RUL prediction method, vibration data collected at the NASA Glenn Spiral Bevel Gear Test Facility and from hybrid ceramic bearing run-to-failure tests in the laboratory were used to predict the RUL of the gears and bearings. In the first case study, the vibration features for gear RUL prediction were extracted using specialized signal processing techniques. However, in the second case study, the vibration features for bearing RUL prediction were extracted automatically by the DBB–FNN algorithm. The optimal hyperparameters of DBN–FNN structure were determined by running a grid search to minimize the RMSE on the testing set in all the case studies. The RUL prediction performance of the DBN–FNN algorithm was compared with that of a particle filter-based approach. The results have shown that the deep learning-based approach achieved a slightly better accuracy for gear but slightly worse accuracy than the particle filter-based approach. However, given that the deep learning-based DBN–FNN does not require manual feature extraction using specialized signal processing

Authorized licensed use limited to: UNIVERSIDADE FEDERAL DE GOIAS. Downloaded on May 10,2023 at 18:32:50 UTC from IEEE Xplore. Restrictions apply.

techniques and explicit model equations like the particle filter-based approaches, the RUL prediction performance achieved by the deep learning-based DBN–FNN has shown its promising capability for RUL prediction of rotating components with big data.

REFERENCES

[1] K. T. Huynh, I. T. Castro, A. Barros, and C. Bérenguer, "On the use of mean residual life as a condition index for condition-based maintenance decision-making," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 44, no. 7, pp. 877–893, Jul. 2014.

[2] X.-S. Si, W. Wang, C.-H. Hu, and D. H. Zhou, "Remaining useful life estimation—A review on the statistical data driven approaches," *Eur. J. Oper. Res.*, vol. 213, no. 1, pp. 1–14, 2011.

[3] G. Vachtsevanos, F. L. Lewis, M. Roemer, A. Hess, and B. Wu, *Intelligent Fault Diagnosis and Prognosis for Engineering System*. Hoboken, NJ, USA: Wiley, 2006.

[4] A. Malhi, R. Yan, and R. X. Gao, "Prognosis of defect propagation based on recurrent neural networks," *IEEE Trans. Instrum. Meas.*, vol. 60, no. 3, pp. 703–711, Mar. 2011.

[5] F. O. Heimes, "Recurrent neural networks for remaining useful life estimation," in *Proc. IEEE Int. Conf. Prognostics Health Manag.*, Denver, CO, USA, 2008, pp. 1–6.

[6] P. Lim, C. K. Goh, K. C. Tan, and P. Dutta, "Estimation of remaining useful life based on switching Kalman filter neural network ensemble," in *Proc. Annu. Conf. Prognostics Health Manag. Soc.*, Fort Worth, TX, USA, 2014, pp. 2–9.

[7] P. Baraldi, F. Mangili, and E. Zio, "A Kalman filter-based ensemble approach with application to turbine creep prognostics," *IEEE Trans. Rel.*, vol. 61, no. 4, pp. 966–977, Dec. 2012.

[8] E. Bechhoefer, S. Clark, and D. He, "A state space model for vibration based prognostics," in *Proc. Annu. Conf. Prognostics Health Manag. Soc.*, Portland, OR, USA, Oct. 2010, pp. 1–8.

[9] D. Codetta-Raiteri and L. Portinale, "Dynamic Bayesian networks for fault detection, identification, and recovery in autonomous spacecraft," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 45, no. 1, pp. 13–24, Jan. 2015.

[10] X. Yin and Z. Li, "Reliable decentralized fault prognosis of discrete-event systems," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 46, no. 11, pp. 1598–1603, Nov. 2016, doi: 10.1109/TSMC.2015.2499178.

[11] M. J. Daigle and K. Goebel, "Model-based prognostics with concurrent damage progression processes," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 43, no. 3, pp. 535–546, May 2013.

[12] P. Baraldi, M. Compare, S. Sauco, and E. Zio, "Ensemble neural network-based particle filtering for prognostics," *Mech. Syst. Signal Process.*, vol. 41, nos. 1–2, pp. 288–300, 2013.

[13] C. Chen, B. Zhang, G. Vachtsevanos, and M. Orchard, "Machine condition prediction based on adaptive neuro–fuzzy and high-order particle filtering," *IEEE Trans. Ind. Electron.*, vol. 58, no. 9, pp. 4353–4364, Sep. 2011.

[14] D. He, E. Bechhoefer, J. Ma, and R. Li, "Particle filtering based gear prognostics using one-dimensional health index," in *Proc. Annu. Conf. Prognostics Health Manag. Soc.*, Montreal, QC, Canada, Sep. 2011, pp. 1–9.

[15] R. Li, J. Ma, A. Panyala, and D. He, "Hybrid ceramic bearing prognostics using particle filtering," in *Proc. Conf. Soc. Mach. Failure Prevent. Technol.*, Huntsville, AL, USA, Apr. 2010, pp. 57–69.

[16] N. Daroogheh, A. Baniamerian, N. Meskin, and K. Khorasani, "Prognosis and health monitoring of nonlinear systems using a hybrid scheme through integration of PFs and neural networks," *IEEE Trans. Syst., Man, Cybern., Syst.*, to be published, doi: 10.1109/TSMC.2016.2597272.

[17] M. S. Arulampalam, S. Makell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking," *IEEE Trans. Signal Process.*, vol. 50, no. 2, pp. 174–188, Feb. 2002.

[18] J. Yoon and D. He, "Development of an efficient prognostic estimator," *J. Failure Anal. Prevent.*, vol. 15, no. 1, pp. 129–138, 2015.

[19] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, 2006.

[20] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, Aug. 2013.

[21] D. Silver *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[22] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[23] Z. Chen, X. Zeng, W. Li, and G. Liao, "Machine fault classification using deep belief network," in *Proc. IEEE Int. Instrum. Meas. Technol. Conf.*, Taipei, Taiwan, 2016, pp. 1–6.

[24] H. Shao, H. Jiang, X. Zhang, and M. Niu, "Rolling bearing fault diagnosis using an optimization deep belief network," *Meas. Sci. Technol.*, vol. 26, no. 11, pp. 115002–115018, 2015.

[25] Y. Lv, Y. Duan, W. Kang, Z. Li, and F.-Y. Wang, "Traffic flow prediction with big data: A deep learning approach," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 2, pp. 865–873, Apr. 2015.

[26] M. Hossain, B. Rekabdar, S. J. Louis, and S. Dascalu, "Forecasting the weather of Nevada: A deep learning approach," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Killarney, Ireland, 2015, pp. 1–6.

[27] Y. Tao, H. Chen, and C. Qiu, "Wind power prediction and pattern feature based on deep learning method," in *Proc. IEEE PES Asia–Pac. Power Energy Eng. Conf. (APPEEC)*, Hong Kong, 2014, pp. 1–4.

[28] T. P. Oliveira, J. S. Barbar, and A. S. Soares, "Multilayer perceptron and stacked autoencoder for Internet traffic prediction," *Network and Parallel Computing* (LNCS 8707). Heidelberg, Germany: Springer, 2014, pp. 61–71.

[29] P. Smolensky, "Chapter 6: Information processing in dynamical systems: Foundations of harmony theory," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, vol. 1. Cambridge, MA, USA: MIT Press, 1986, pp. 194–281.

[30] G. E. Hinton, "Training products of experts by minimizing contrastive divergence," *Neural Comput.*, vol. 14, no. 8, pp. 1711–1800, 2002.

[31] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Advances in Neural Information Processing Systems*, vol. 19. Cambridge, MA, USA: MIT Press, 2007, p. 153.

[32] D. Erhan *et al.*, "Why does unsupervised pre-training help deep learning?" *J. Mach. Learn. Res.*, vol. 11, pp. 625–660, Mar. 2010.

[33] R. J. Frank, N. Davey, and S. P. Hunt, "Time series prediction and neural networks," *J. Intell. Robot. Syst.*, vol. 31, nos. 1–3, pp. 91–103, 2001.

[34] B. Efron and G. Gong, "A leisurely look at the bootstrap, the jackknife, and cross-validation," *Amer. Stat.*, vol. 37, no. 1, pp. 36–48, 1983.

[35] E. Zio and G. Peloni, "Particle filtering prognostic estimation of the remaining useful life of nonlinear components," *Rel. Eng. Syst. Safety*, vol. 96, no. 3, pp. 403–409, 2011.

[36] P. Dempsey, R. Handschuh, and A. Afjeh, "Spiral bevel gear damage detection using decision fusion analysis," Nat. Aeronaut. Space Admin., Washington, DC, USA, Tech. Rep. NASA/TM-2002-211814, 2002.

[37] E. Bechhoefer, D. He, and P. Dempsey, "Gear health threshold setting based on a probability of false alarm," in *Proc. Annu. Conf. Prognostics Health Manag. Soc.*, Montreal, QC, Canada, Sep. 2011, pp. 275–281.

[38] J. J. Zakrajsek and D. P. Townsend, "An analysis of gear fault detection method as applied to pitting fatigue failure damage," NASA Lewis Res. Center, Cleveland, OH, USA, Tech. Rep. 92-C-035, 1993.

[39] E. Wemhoff, H. Chin, and M. Begin, "Gearbox diagnostics development using dynamic modeling," in *Proc. AHS 63rd Annu. Forum*, Virginia Beach, VA, USA, 2007, pp. 1–11.

[40] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Represent.*, San Diego, CA, USA, 2014, pp. 1–15.

[41] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier networks," in *Proc. AISTATS*, 2011, pp. 315–323.

[42] A. L. Mass, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. Int. Conf. Mach. Learn.*, Atlanta, GA, USA, 2013, pp. 1–6.

Jason Deutsch, photograph and biography not available at the time of publication.

David He, photograph and biography not available at the time of publication.

Authorized licensed use limited to: UNIVERSIDADE FEDERAL DE GOIAS. Downloaded on May 10,2023 at 18:32:50 UTC from IEEE Xplore. Restrictions apply.