applied sciences

MDPI

Article

Remaining Useful Life Prediction of Hybrid Ceramic Bearings Using an Integrated Deep Learning and Particle Filter Approach

Jason Deutsch [1], Miao He [1] and David He [1,2,*]

[1] Department of Mechanical and Industrial Engineering, University of Illinois at Chicago, Chicago, IL 60607, USA; jdeuts4@uic.edu (J.D.); mhe21@uic.edu (M.H.)

[2] College of Mechanical Engineering and Automation, Northeastern University, Shenyang 110819, China

* Correspondence: davidhe@uic.edu; Tel.: +1-312-996-3410 or +86-024-9369-6132

Academic Editor: César M. A. Vasques

Received: 31 March 2017; Accepted: 20 June 2017; Published: 23 June 2017

Abstract: Bearings are one of the most critical components in many industrial machines. Predicting remaining useful life (RUL) of bearings has been an important task for condition-based maintenance of industrial machines. One critical challenge for performing such tasks in the era of the Internet of Things and Industrial 4.0, is to automatically process massive amounts of data and accurately predict the RUL of bearings. This paper addresses the limitations of traditional data-driven prognostics, and presents a new method that integrates a deep belief network and a particle filter for RUL prediction of hybrid ceramic bearings. Real data collected from hybrid ceramic bearing run-to-failure tests were used to test and validate the integrated method. The performance of the integrated method was also compared with deep belief network and particle filter-based approaches. The validation and comparison results showed that RUL prediction performance using the integrated method was promising.

Keywords: prognostics; deep learning; particle filter; bearing; RUL prediction

1. Introduction

Mechanical big data has the characteristics of large volume, diversity, and high velocity. The prediction of remaining useful life (RUL) has been used as an important parameter for condition-based maintenance decision making [1]. Traditional data-driven prognostic methods have their limitations for processing mechanical big data. Many prognostic methods are developed based on explicit model equations [2]. Therefore, they are largely dependent on human expertise and knowledge in model building and signal processing, and hence are difficult to implement for automatic processing of massive data. These prognostic methods include recurrent neural networks [3,4], Kalman filters [5–7], dynamic Bayesian networks [8], *k*-reliable decentralized prognosis [9], particle filter-based [10–13], and combined particle filter and neural networks [14]. In addition, some fuzzy systems-based approaches for prediction have been developed [15]. However, in comparison with fuzzy systems, particle filters take a probabilistic approach, in that the posterior distribution is modeled by sampling from a set of distributions, whereas in fuzzy systems the model output is based on the input variables of fuzzy set membership and an implication of rules. Among all the approaches, particle filters [16] 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 natures. For example, Yoon and He [17] 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 of the above mentioned methods require either complicated signal processing techniques to extract features from the sensor data, or knowledge of 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 unsuitable for automatic data processing and feature extraction for big data. There also may be 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 damage may not be available, in which the traditional particle filter approach cannot be used [11]. To effectively extract features from massive bearing condition monitoring data and accurately predict bearing RUL, new effective methods are needed. The recent developments in deep learning have provided an attractive opportunity to build advanced RUL prediction methods for big data.

Since the introduction of a deep belief network [18], deep belief networks and other deep learning methods have become widely used for big data processing and analysis. Deep learning is capable of extracting useful and important features from data to improve the power of prediction [19]. Equipped with multiple layers of structure, deep learning is also capable of processing massive data and extracting hidden information. One recent successful story of deep learning is AlphaGo by Google Deepmind [20]. AlphaGo has demonstrated the power of deep learning for massive data processing and feature learning by defeating the best human Go player in the world. Deep neural network architectures have been successfully built in various domains such as image recognition, automatic speech recognition, and natural language processing [21]. Recently [22,23] applied deep learning on raw vibration signals and time-domain features for machine fault diagnostics. In addition to classification problems, deep learning also has promising capability to solve prediction problems. These prediction problems include predicting car traffic [24], weather [25], wind speed [26], and internet traffic [27]. A number of deep learning algorithms have been used for solving prediction problems. These include auto-encoders, restricted Boltzmann machines, deep belief networks, convolutional neural networks, and more. Deep learning represents an attractive option to process massive data for RUL prediction. Deutsch and He [28] presented a deep learning-based bearing RUL prediction method using a deep brief network, and compared RUL prediction performance with a particle filter-based approach. Their comparison results showed that even though the RUL prediction performance of the deep learning based approach was comparable with particle filter-based approach, the RUL prediction accuracy of the deep learning-based approach was slightly lower than that of the particle filter. Therefore, it was of interest to investigate an integrated method that can use the strength of deep learning to overcome the limitations of the particle filter. To date, no research on combining deep learning with particle filter for RUL prediction of hybrid ceramic bearings has been reported in the literature.

In this paper, a new integrated method that combines a deep belief network with a particle filter for remaining useful life prediction of hybrid ceramic bearings using vibration signals was presented. Real vibration data collected from hybrid ceramic bearing run-to-failure tests were used to test and validate the integrated method. The performance of the integrated method was also compared with a deep belief network and particle filter-based approaches.

2. Methodology

A deep belief network (DBN), which is a stacked version of a restricted Boltzmann machine (RBM), and a purely data-driven particle filter were combined in this paper in order to predict the RUL at L steps ahead into the future. Next, the relevant background of RBM, DBN and the particle filter was provided. The section concludes with the combined DBN and particle filter-based approach for RUL prediction.

2.1. The Restricted Boltzmann Machine

An RBM [29] is considered as a type of unsupervised machine learning method. It is a stochastic artificial neural network that learns a probability distribution over the set of its inputs. A RBM normally

has two layers: a visible layer and a hidden layer. It can be represented by a bipartite graph that contains undirected edges from its two layers. Each layer contains a collection of neurons/nodes. Each neuron/node of the visible layer represents a feature of the input data, while neurons/nodes of the hidden layer represent the latent variables. A typical RBM structure is shown in Figure 1. The reason that an RBM is "Restricted" is because there are no connections between each neuron/node within either the visible or hidden layers. An RBM contains a matrix of weights W_{ij} representing the connection to visible node v_i and hidden node h_j. In Figure 1, a_i represents the bias term in the visible layer, and b_j in the hidden layer.

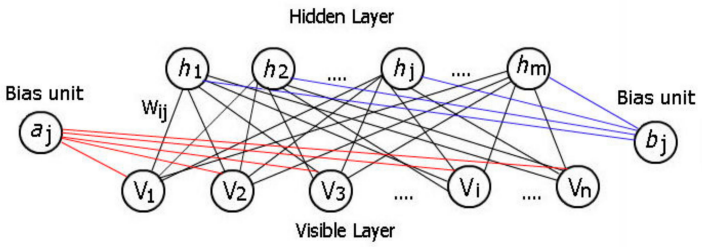

Figure 1. A restricted Boltzmann machine.

The weights and biases are computed by maximizing $P(\boldsymbol{v})$, 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 that can be obtained 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:

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

Theoretically, the problem of maximizing Equation (1) can be solved by taking its partial log derivative with respect to its parameters $\boldsymbol{W}$, $\boldsymbol{a}$, and $\boldsymbol{b}$:

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

Normally Equation (4) can also be written as:

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

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

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

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

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

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

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:

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

2.2. The Deep Belief Network

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

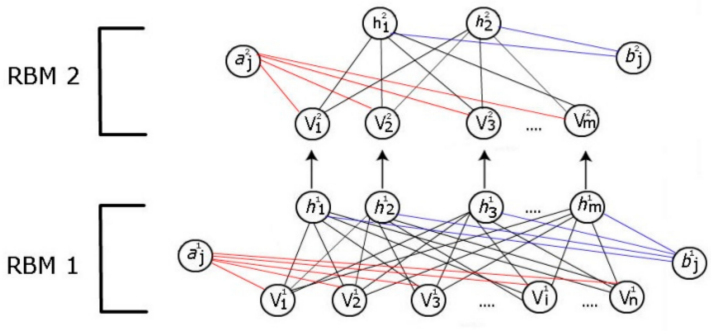

Figure 2. A deep belief network with two hidden layers.

The RBM becomes a building block for forming a deep belief network. 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.

In order to use a DBN to predict a continuous output, one can first learn the weights and biases in the unsupervised stage of learning. 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 Figure 3. The parameters of the entire network are then updated using the back-propagation algorithm in the same way as a feedforward neural network (FNN) is trained. In this way, the DBN pre-trains 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 a poor local minima [32].

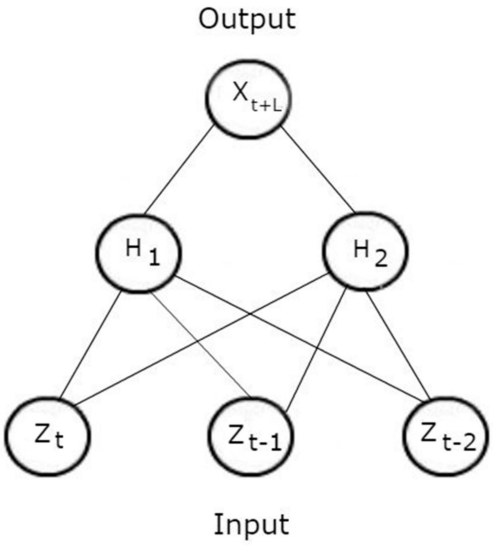

Figure 3. A feedforward neural network with $d = 3$ and a single hidden layer.

The DBN performs unsupervised learning first by training it on a set of signal features in order to learn a latent representation of the data. After the end of training, an output layer is added 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 (without an activation function) which represents the continuous prediction.

2.3. Particle Filter

The particle filter is a Monte Carlo approach that can be used to estimate the state of a system by combining both the state evolution of the system and the observation/measurement parameters obtained from the state.

In discrete time, a system can be described by the following state space model:

$$\boldsymbol{x}_t = f(\boldsymbol{x}_{t-1}, \boldsymbol{\omega}_{t-1}) \tag{10}$$

$$\boldsymbol{z}_t = h(\boldsymbol{x}_t, \boldsymbol{v}_t) \tag{11}$$

From the above equations, $\boldsymbol{x}_t$ represents the state of the system (such as the crack depth) at time t, $\boldsymbol{\omega}_{t-1}$ which is independently and contains identically distributed (*iid*) noise/variance at time $t-1$, $f(\,)$ is a function that maps the transitions between states, $\boldsymbol{z}_t$ represents the measurement/observation (can be thought of as a feature, typically derived from a vibration signal) at time t, $\boldsymbol{v}_t$ is the *iid* noise/variance associated with each measurement, and $h(\,)$ is a function that maps the state with the measurement.

The goal of the particle filter is then to be able to estimate the probability density function (pdf) of $p(\boldsymbol{x}_t|\boldsymbol{z}_{1:t})$, that is, to estimate the state at time t, given the measurements up to time t. It is also assumed that the prior probability of the state $P(\boldsymbol{x}_0)$ is unkown.

In the Bayesian setting, the state estimation is usually computed recursively in two stages; the prediction and the update. For the prediction step, the $P(\boldsymbol{x}_t|\boldsymbol{z}_{1:t-1})$ is given as:

$$\begin{array}{c} \int P(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{z}_{1:t-1})P(\boldsymbol{x}_{t-1}|\boldsymbol{z}_{1:t-1})d\boldsymbol{x}_{t-1} \\ = \int P(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})P(\boldsymbol{x}_{t-1}|\boldsymbol{z}_{1:t-1})d\boldsymbol{x}_{t-1} \end{array} \tag{12}$$

For the update step, new measurements are collected, which are then used to update the prior distribution. The posterior distribution can then be written as:

$$P(\boldsymbol{x}_t|\boldsymbol{z}_{1:t}) = \frac{P(\boldsymbol{x}_t|\boldsymbol{z}_{1:t-1})P(\boldsymbol{z}_t|\boldsymbol{x}_t)}{\int P(\boldsymbol{x}_t|\boldsymbol{z}_{1:t-1})P(\boldsymbol{z}_t|\boldsymbol{x}_t)d\boldsymbol{x}_{t-1}} \tag{13}$$

Typically, the calculations for Equations (12) and (13) are intractable and the particle filter sampling approach is used to approximate them. This can be accomplished by using a set of samples/particles $\{\boldsymbol{x}_t{}^i, w_t{}^i\}_{i=1}^N$, where $w_t{}^i$ is the ith weight at time t and N is the user-specified number of particles generated. The weights can be updated by using the sequential importance resampling (SIR) algorithm [33], and samples are drawn from the state transition distribution:

$$\boldsymbol{x}_t^i \sim p(\boldsymbol{x}_t \Big| \boldsymbol{x}_{t-1}^i) \tag{14}$$

$$w_t^i = \frac{p(\boldsymbol{z}_t|\boldsymbol{x}_t^i)}{\sum_{i=1}^N p(\boldsymbol{z}_t \Big| \boldsymbol{x}_t^i)} \tag{15}$$

Finally, the posterior distribution $P(\boldsymbol{x}_t|\boldsymbol{z}_{1:t-1})$ is obtained by resampling from $\{\boldsymbol{x}_t^1, \boldsymbol{x}_t^2, \ldots \boldsymbol{x}_t^N\}$ where $\hat{\boldsymbol{x}_t^i}$ (predicted estimate) is drawn with probability w_t^i.

2.4. Combined DBN and Particle Filter-Based Approaches for RUL Prediction

The ultimate objective of the combined DBN and particle filter approach is to estimate the pdf of $P(\boldsymbol{x}_{t+L}|\boldsymbol{z}_{1:t})$, where $L \, \epsilon \, \{1, 2, \ldots N-t\}$ and N represents the length of the signal. It should be noted that no new (future) information about the system is provided at the generic time step t, and only information provided from time steps $\{0, 1, \ldots t\}$ may be used for the prediction.

There are two functions that are of interest in modeling; the state transition distribution Equation (14) (sometimes referred to as the proposal distribution) and the measurement distribution Equation (15). Both of these functions can be modeled by the DBN, since neural networks in general are universal function approximators [34].

In the prediction step, the state transition model can simply be approximated by first reconstructing the time series of the measurements ($\boldsymbol{z}_t$ for $t = 1, 2, \ldots N$) into a matrix, where each feature (column) represents a lagged order of the time series, and whose output is an L-step ahead into the future state value, and each row represents an index in time. Formally, the input can be denoted as:

$$[\boldsymbol{z}_t, \boldsymbol{z}_{t-1}, \ldots., \boldsymbol{z}_{t-d+1}], \, \epsilon \, \mathbb{R}^d \tag{16}$$

and treat the state $\boldsymbol{x}_t$ as the actual RUL of the bearing at time t as the output:

$$[x_t, x_{t+1}, \ldots, x_N] \tag{17}$$

where d represents the embedding dimension, and determines the size of the first visible layer in the DBN. Thus, the network requires training tuples of the form $([\boldsymbol{z}_t, \, \boldsymbol{z}_{t-1}, \, \ldots, \, \boldsymbol{z}_{t-d+1}], \, x_t)$, $t = d-1, \ldots, N$. Using this training set, the state transition model is instead actually modeling the pdf of $p(\boldsymbol{x}_{t+L}^i \big| \boldsymbol{z}_t, \boldsymbol{z}_{t-1}, \ldots., \boldsymbol{z}_{t-d+1})$. The L step ahead prediction, x_{t+L}^i, is simply made by subtracting the network's output x_t by L. The advantages of treating the state x_t (instead of treating it as the crack

depth for instance) as the actual RUL allow for a simpler model to be developed. One does not need to utilize another neural network to determine the RUL, given the crack depth, or to recursively compute the state transition model multiple times until it exceeds such a threshold, which needs to be defined. By training the network directly on the RUL, the network is minimizing the error of predicting the RUL, rather than an intermediate value. This results in a network that is more easily trainable and reduces the potential problem of having the network's eventual forecast of the state die off (converging to a single value). The disadvantage of this approach however, is that the parameters of the state transition model will be updated/relearned less frequently. The measurement z_t (often a multi dimensional vector) can be constructed into a mono-dimensional feature vector by setting the input of the DBN as z_t and then setting the size of the last hidden layer to one. The output of the DBN in its unsupervised stage of learning is then the reconstructed mono-dimensional feature vector of z_t.

Equations (16) and (17) define a method often called the sliding window technique [35]. This windowing approach is illustrated in Figure 4.

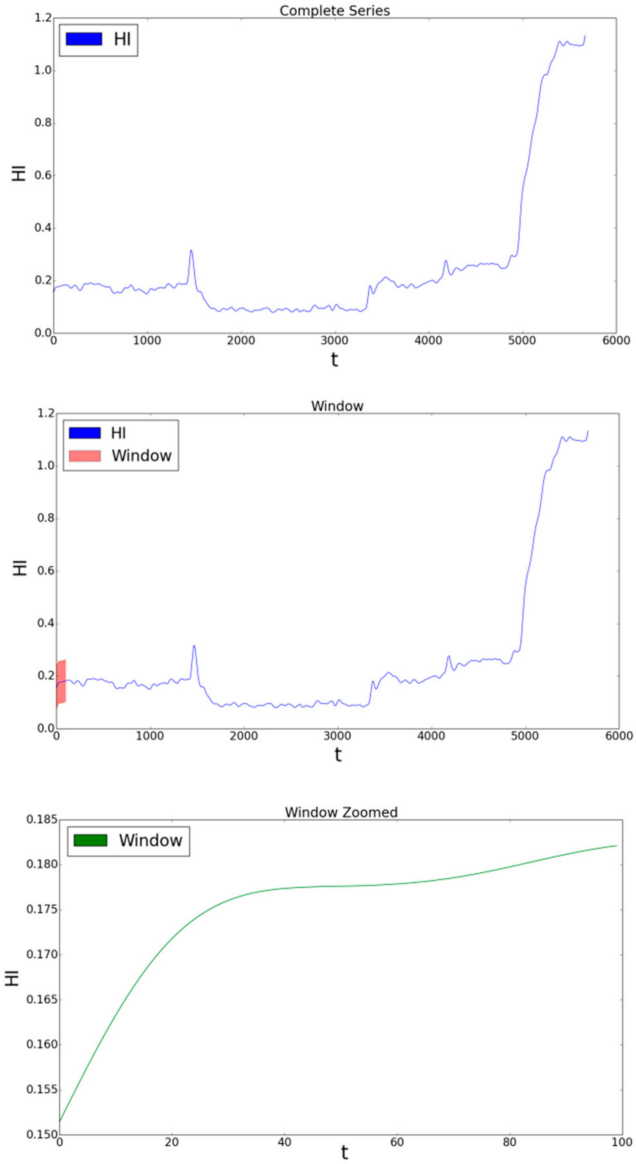

Figure 4. The windowing approach.

The first plot in Figure 4 is the complete series of a fault feature. The second plot highlights the first window of size $d = 100$. The last figure simply shows a zoom-in of the first window. 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 state transition model and its respective variance in Equation (10) $\boldsymbol{\omega}_{t-1}$ can be modeled by taking B bootstraps, which randomly samples (with replacement) each row of the training data TD, $|TD|$ times. The DBN is then trained B times for each bootstrap and the predicted values will be defined as:

$$ST_b(\boldsymbol{z}) = F(\boldsymbol{z}; ST_b^*) \tag{18}$$

where, $\boldsymbol{z}$ represents the generic testing input vector of the past d measurements [Equation (16)] and $F(\boldsymbol{z}; ST_b^*)$ represents the DBN-FNN output [Equation (17)] with respect to the bth bootstrap. The mean and variance are simply then calculated as:

$$ST_{avg}(\boldsymbol{z}) = \frac{1}{B}\sum_{b=1}^{B} ST_b(\boldsymbol{z}) \tag{19}$$

$$\hat{\sigma}_{ST}^2(\boldsymbol{z}) = var(ST_b(\boldsymbol{z})) \tag{20}$$

and then samples x_t^i can be generated from the following Gaussian distribution:

$$\mathrm{N}(ST_{avg}(\boldsymbol{z}), \hat{\sigma}_{ST}^2(\boldsymbol{z})) \tag{21}$$

The measurement distribution can be modeled in a somewhat similar approach to the state transition distribution. The technique [13] requires a dataset of tuples of $(x_t, \boldsymbol{z}_t)_{t=1}^{N_{training}}$ where B bootstraps are sampled from this training data set, and an interpolator $\varphi(x)$ is created. $\boldsymbol{z}_t$ is typically a multi-dimensional vector, which can be reduced to a single mono-dimensional vector for simplicity and ease of calculation. This can be performed by the DBN in its unsupervised stage of learning, by setting the last layer of the DBN's hidden layer to a size of one. A simple FFN can also be used as an interpolator for computing the following:

$$\varphi_{avg}(x) = \frac{1}{B}\sum_{b=1}^{B} \varphi(x; T_b^*) \tag{22}$$

where, T_b^* is the bth bootstrap sampled from the dataset. The measurement model in Equation (11) can be hypothesized as:

$$z(x) = f(x) + v(x) \tag{23}$$

From Equation (23), we can subtract $\varphi_{avg}(x)$ from both sides such that:

$$z(x) - \varphi_{avg}(x) = [f(x) - z(x)] + v(x) \tag{24}$$

The error term $z(x) - \varphi_{avg}(x)$ is then a function of $[f(x) - z(x)]$ and $v(x)$, which we will call the model error and intrinsic noise respectively. Their variances are then denoted as $\sigma_m^2(x)$ and $\alpha^2(x)$ respectively. The model error variance can be modeled by taking M groups of interpolators from $\varphi(x; T_b^*)$ of length K networks and computing the average as:

$$\varphi_{com}^m(x) = \frac{1}{K}\sum_{i=1}^{K} \varphi_i(x) \tag{25}$$

The set, $\Psi = \{\varphi_{com}^{m}(x)\}_{m=1}^{M}$ is then resampled with replacement using P bootstraps, denoted as $\{\Psi_p\}_{p=1}^{P}$, where Ψ_p is the pth bootstrap of Ψ. The estimate of the model error variance can then be computed as:

$$\hat{\sigma}_m^2(x) = \frac{1}{P}\sum_{p=1}^{P} var(\Psi_p) \tag{26}$$

The noise variance can be modeled by building up a training set $(x_t, \hat{\alpha}_t^2)_{t=1}^{N_{training}}$, where $\hat{\alpha}_t^2$ is defined as:

$$\hat{\alpha}_t^2 = max\Big\{\big(z_t - \varphi_{avg}(x_t)\big)^2 - \hat{\sigma}_m^2(x_t), 0\Big\} \tag{27}$$

A single FNN can then be trained on the aforementioned training set in order to estimate $\hat{\alpha}_t^2(x)$ for an arbitrary x vector. Finally $p(z_t|x_t)$ can be approximated by:

$$p(z_t|x_t) \approx \mathrm{N}\Big(\varphi_{avg}(x), \hat{\sigma}_m^2(x) + \hat{\alpha}_t^2(x)\Big) \tag{28}$$

The confidence intervals for a generic prediction $\hat{x}$ can then be obtained with 1-α% confidence by the following formula [36]:

$$\hat{x} \pm t_{1-(\alpha/2),B-1}\sqrt{\frac{\hat{\sigma}_{ST}^2(x)}{B-1}} \tag{29}$$

The training/testing procedure for the integrated DBN and particle filter is then as follows:

Step 1: Let the last 100 rows be equal to the testing set, and set the rest of the data to the training set.

Step 2: Approximate the state transition model.

- a: Reconstruct the time series of the signal features into a matrix with an embedding dimension of d. This will serve as the input, and the output will be the mapped state x_t, where $t = d-1, \ldots, N_{training} - L - 1$.
- b: Create B bootstrapped data sets using the data created in Step 1.
- c: Initialize the weights and biases of the FNN by training the DBN on the input data with all the data except for the last 100 rows. The rest of the data is the testing set.
- d: Train the FNN. Fine tune the weights in a supervised fashion by minimizing the loss function on the training set and by using the back-propagation algorithm.
- e: Predict the x_{t+L}. This is accomplished by subtracting the network's output x_t by L, i.e., $x_{t+L} = x_t - L$.
- f: Let $t = t+1$, and update the training vector with input features $(z_t, z_{t-1}, \ldots, z_{t-d+1})$.
- g: Repeat Steps 2e–2f, until all 100 points have been predicted.

Step 3: Estimate the measurement distribution. Create a new dataset of the form $(x_t, z_t)_{t=1}^{N_{training}}$ Generate B bootstraps from this training dataset and apply Equations (22)–(28) to obtain the measurement distribution. The DBN can be used for estimates in Equations (22) and (27).

Step 4: Estimate the state. The predicted state $\hat{x}_{t+L}$ can be obtained by sampling $\{x_{t+L}^1, x_{t+L}^2, \ldots\ x_{t+L}^B\}$ with probability w_t^i [from Equation (15)].

This method allows a complete data-driven approach to be developed in conjunction with a particle filter for RUL estimation. Equation (10), the proposal distribution is built up using training tuples of the form $([z_t, z_{t-1}, \ldots, z_{t-d+1}], x_t)$. The bootstrapping approach is then used to collect the statistics for the mean and variance. Equation (11), the measurement model is then modeled by Equations (22)–(28). Here the DBN can serve two purposes. It can be used to reduce the dimensionality of the measurement z_t and it can also to be used in order to initialize the weights of the neural network.

3. Validation

In this section, the validation of the integrated bearing RUL prediction method using vibration data collected from a single hybrid ceramic bearing run-to-failure test is discussed.

3.1. Hybrid Ceramic Bearing Run-to-Failure Test Setup

The hybrid ceramic bearing run-to-failure tests were performed using a bearing test rig in the laboratory as shown in Figure 5. The bearing test rig consisted of the following main components: (1) 3-HP AC motor with a maximum speed up to 3600 rpm and variable speed controller; (2) Hydraulic dynamic loading system with a maximum radial load up to 4400 lbs or 19.64 kN; (3) Integrated loading and bearing housing. The rig can be used for testing both ball and tapered roller bearings.

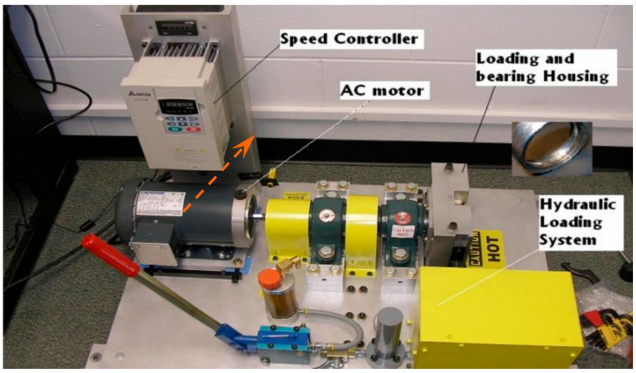

Figure 5. The bearing run-to-failure test rig.

An automatic data acquisition system was constructed using a National Instrument CI 4462 board (NI, Austin, TX, USA) and NI LabVIEW software (LabView 2012, NI, Austin, TX, USA). The automatic data acquisition system is characterized with the following key features: (1) Maximum sampling rate up to 102.4 kHz; (2) 4 Input simultaneous anti-aliasing filters; (3) Software-configurable AC/DC coupling and IEPE conditioning; (4) Vibration analysis functions such as envelope analysis, cepstrum analysis, and so on for computing necessary condition indicators.

The tested hybrid ceramic ball bearing was a SMR6205C-ZZ/C3 #3 L55/MG2 type bearing by Boca Bearing Company (Boca Bearings, Boynton Beach, FL, USA). It consisted of stainless steel inner outer races, and ceramic balls. The bearing was mounted on the test rig. Two accelerometers were stunt mounted on the bearing housing in the direction perpendicular to the shaft. During the tests, the rig was run at a speed of 1800 rpm (30 Hz) and was subjected to a radial load of 600 psi. Vibration data were collected with a sampling rate of 102.4 kHz for two seconds at each sampling point. There was a 5 min gap between any two sampling points. At the end of the test, the test bearing was disassembled, checked, and photographed. The bearing data contained a total of 849 data files with a length of approximately 71 hours. Table 1 describes the run-to-failure test setting. Table 2 provides the specifications of the tested bearing.

Table 1. The settings of the run-to-failure test.

Test Bearing Name	Type of Bearing	Load (psi)	Input Shaft Speed (hz)
B2	Hybrid ceramic bearing	600	30

Table 2. Specifications of the test bearing.

Parameter	Specification
Bearing material	Stainless steel 440c
Ball material	Ceramic SI3N4
Inner diameter (d)	25 mm
Outer diameter (D)	52 mm
Width	15 mm
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 (×1000 rpm)	19
RPM oil (×1000):	22
Dynamic load (kgf)	1429
Basic load (kgf)	804
Working temperature (°C)	121
Weight (g)	110.32

The root mean square (RMS) of the vibration signals was computed to represent the degradation of the bearing over time during the run to failure tests. The RMS at each time interval (denoted as RMS_t) can be calculated as follows:

$$RMS_t = \sqrt{\frac{1}{n}\sum_{i=1}^{n} f_{ti}^2} \tag{30}$$

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 for the bearing data can be seen in Figure 6.

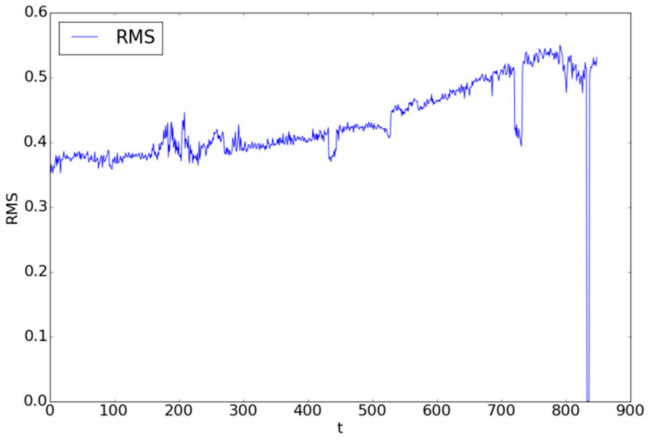

Figure 6. The bearing RMS values.

For the bearing data, the RUL_t (remaining useful life at time t) was calculated simply by taking the time index of the maximum recorded RMS value as the point of failure denoted as $RMS_{T_{end}}$ and subtracting it from each time step:

$$RUL_t = RMS_{T_{end}} - t \tag{31}$$

The raw vibration signals were preprocessed using the fast Fourier transform (FFT), and the FFT values were used as the only signal feature input into the integrated DBN-particle filter model to predict the RUL of the bearing. The FFT, which is an efficient algorithm for computing the discrete Fourier transform (DFT) at a time interval t, can be calculated as follows:

$$DFT_{tn} = \sum_{k=0}^{N-1} f_{tk} e^{\frac{-2\pi ikn}{N}} \tag{32}$$

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

3.2. 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. As seen from Figure 6, a rather large dip in the RMS values occurred 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 Figures 7 and 8 for $L = 1$ and $L = 10$, respectively. The error metrics and hyperparameters of the DBN with $L = 1$ and $L = 10$ are provided in Tables 3 and 4, respectively. Table 4 shows the hyperparameters of the DBN for the state transition model which were determined using a grid search. Input data was also scaled to be in [0, 1], $d = 100$ was set as the embedding dimension, B was set as 50, $M = 10$, $K = 5$, $P = 1000$ and 50 particles were used for both $L = 1$ and $L = 10$ predictions.

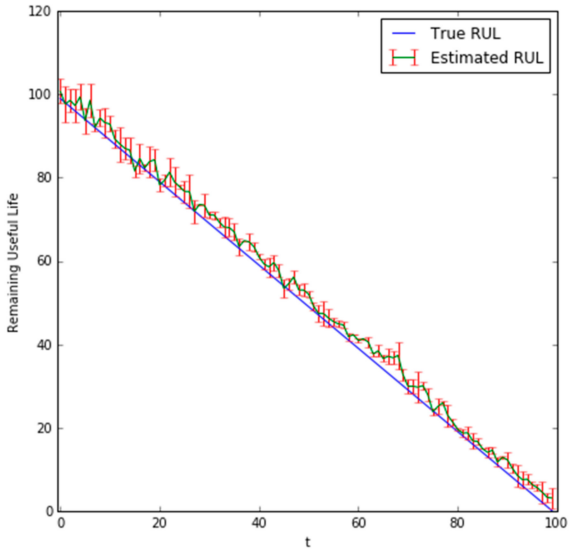

Figure 7. Plot of bearing $\widehat{RUL_t}$ values with $L = 1$.

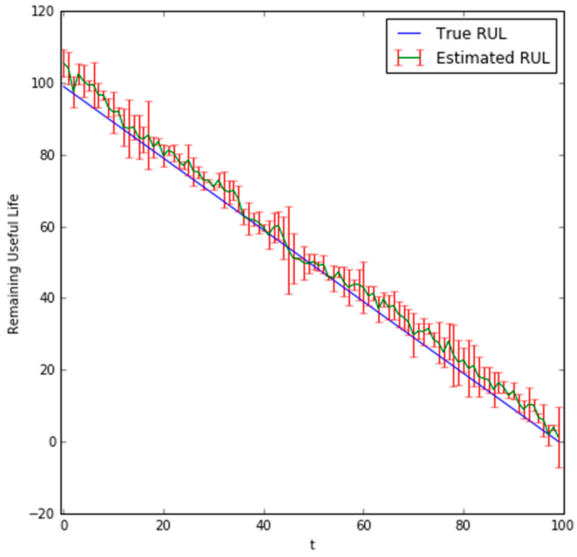

Figure 8. Plot of bearing $\widehat{RUL_t}$ values with $L = 10$.

Table 3. Root mean squared error (RMSE) and mean absolute percentage error (MAPE) results.

Combined DBN and Particle Filter-Based Approach			
L	**RMSE**	**MAPE**	**$\alpha - \lambda$ Accuracy (α = 10%)**
1	2.04	7.33%	0.80
10	3.52	8.68%	0.61
Particle Filter-Based Approach			
L	**RMSE**	**MAPE**	**$\alpha - \lambda$ Accuracy (α = 10%)**
1	2.53	7.47%	0.71
10	3.65	8.73%	0.53

Table 4. Hyperparameters of the deep belief network (DBN).

L	**DBN Learning Rate**	**DBN Epochs**	**Hidden Layer Structure**	**FNN Learning Rate**	**FNN Epochs**
1	0.002	74	[146, 53]	0.0017	176
10	0.0023	82	[120, 54]	0.0014	92

In Figures 7 and 8, the green color represents the average predicted RUL values across the bootstrapped samples. The red error bars represent the 95% bounds. The predicted results show for both $L = 1$ and $L = 10$ that it can accurately predict the true RUL, and as the bearing approaches the point of failure, the accuracy of the predictions tends to increase.

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 common error metrics used in Table 3 are the root mean squared error (RMSE), $\alpha - \lambda$ metric [37,38], and the mean absolute percentage error (MAPE). The MAPE, RMSE, and $\alpha - \lambda$ metric are defined by the following equations:

$$MAPE = \left(\frac{1}{n} \sum_{t=1}^{n} \frac{A_t - F_t}{A_t} \right) * 100\% \tag{33}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(A_t - F_t)^2} \tag{34}$$

$$(\alpha - \lambda)_{t_\lambda} = \begin{cases} 1 & if\ (1-\alpha)RUL_{t_\lambda} \leq \widehat{RUL}_{t_\lambda} \leq (1+\alpha)RUL_{t_\lambda} \\ 0 & otherwise \end{cases} \tag{35}$$

In Equations (33) and (34), A_t denotes the actual value, F_t represents the predicted value, and n equals the number of points to be predicted. In Equation (35), α represents a user specified bound level, RUL_{t_λ} represents the actual RUL at time t_λ, and $\widehat{RUL}_{t_\lambda}$ represents the predicted RUL at time t_λ. λ is given as a percentage of RUL for a given equipment (i.e., $t_{\lambda=50\%}$ represents the time index where half the RUL is left). The reported $\alpha - \lambda$ in Table 3 represents the average $\alpha - \lambda$ metric across the all testing samples. In the $\alpha - \lambda$ metric, results closer to one indicate better performance. The loss functions used to train the state transition models for $L = 1$ and $L = 10$ were the MSE ($RMSE^2$) and the MAPE respectively. Satisfactory hyperparameters for building up Equations (22) and (27) for both $L = 1$ and $L = 10$ were set as [105, 59] for the hidden layer structure using 122 epochs with a learning rate of 0.00085. The hyperparameters for the reconstructed mono-dimensional measurement data were set as [142, 87] for the hidden layer structure using a small learning rate of 0.0094 and 212 epochs. The hyperparameters for each built network was done by employing a grid search and evaluating candidate hyperparameters on the MSE using a 10 fold cross validation on the training set. Cross validation is a widely used method to avoid overftting when selecting hyperparameters [39]. The chosen hyperparameters were obtained by those that minimized the MSE during cross validation. For all of these networks, the activation function was set as the rectified linear unit (ReLU) function. It solves the vanishing gradient problem that other non-linear activation functions can cause [40]. The ReLU activation function for the input x of a neuron is defined as:

$$ReLU(x) = max\{0, x\} \tag{36}$$

Since the particle filter is the most competitive RUL prediction method for bearings, for a comparison purpose, the RMSE and MAPE of the RUL predictions obtained by the particle filter-based approach are also provided in Table 3. In comparison with the results obtained using the particle filter-based approach, the RMSE and MAPE values of the integrated approach were slightly lower and the $\alpha - \lambda$ metric values were higher. Note that since in [28], it was reported that using the same bearing data, the particle filter gave a better RUL prediction accuracy than a DBN based approach did, we consider the integrated method presented here as better than the DBN-based approach used on the same dataset. The comparison results showed the promising performance of combining the deep learning based approach with particle filter for hybrid ceramic bearing RUL prediction.

Given that the integrated approach did not require explicit model equations like the particle filter-based approach and is scalable for big data applications, the RUL prediction performance achieved by the integrated approach has shown great potential for bearing RUL prediction with big data.

4. Conclusions

Predicting the remaining useful life of bearings has been an important task for condition-based maintenance of industrial machines, as bearings are one of the most critical components in these machines. To meet the challenge of automatically processing massive data and accurately predicting RUL of the bearings in the era of the Internet of Things and Industrial 4.0, this paper addressed the limitations of traditional data-driven prognostics by presenting a new method that integrates a deep belief network and a particle filter for RUL prediction of hybrid ceramic bearings. Real vibration data collected from hybrid ceramic bearing run-to-failure tests were used to test and validate the integrated method. The performance of the integrated method was also compared with DBN and particle filter-based approaches. In comparison with the RUL prediction results obtained using the

particle filter-based method, the prediction accuracy measured by RMSE and MAPE of the integrated method is slightly lower. Also, based on the previous comparison between DBN and the particle filter methods for the same dataset reported in [28], the RUL prediction performance of the integrated method is considered better than the DBN method. The validation and comparison results showed the promising RUL prediction performance of the integrated method.

Since the integrated approach was a purely data-driven approach, future work should focus on increasing user confidence in the method. This may be accomplished by combining the integrated approach with other techniques such as employing the use of fuzzy similarity [41]. This would require a large database of run to fail trajectories that can be compared based on their similarity with the observed data. This large database should not only increase the accuracy of the integrated approach, but also further validate it. Additional work may also include the processing of the raw vibration signals at each sampling interval by a DBN into a single dimension signal feature that can be used for prognostics.

Author Contributions: Jason Deutsch conceived the idea and formulated the mathematical models behind the provided approach. In addition, he was responsible for all parameter tuning, data processing, figures, and calculations. Miao He was responsible for the run to fail trajectories of the hybrid ceramic bearings in the experimental setup. David He was responsible for research idea generation, the introduction, abstract, conclusions, and for the final editing and reviewing of the provided approach.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Huynh, K.T.; Castro, I.T.; Barros, A.; Bérenguer, C. On the use of mean residual life as a condition index for condition-based maintenance decision-making. *IEEE Trans. Syst. Man Cybern. Syst.* **2014**, *44*, 877–893. [CrossRef]
2. Vachtsevanos, G.; Lewis, F.L.; Roemer, M.; Hess, A.; Wu, B. *Intelligent Fault Diagnosis and Prognosis for Engineering System*; John Wiley & Sons, Inc.: Hoboken, NJ, USA, 2006.
3. Malhi, A.; Yan, R.; Gao, R.X. Prognosis of defect propagation based on recurrent neural networks. *IEEE Trans. Instrum. Meas.* **2011**, *60*, 703–711. [CrossRef]
4. Heimes, F. Recurrent neural networks for remaining useful life estimation. In Proceedings of the 2008 IEEE International Conference of Prognostics and Health Management, Denver, CO, USA, 6–9 October 2008; pp. 1–6.
5. Lim, P.; Goh, C.K.; Tan, K.C.; Dutta, P. Estimation of remaining useful life based on switching Kalman filter neural network ensemble. In Proceedings of the 2014 Annual Conference of the Prognostics and Health Management Society, Fort Worth, TX, USA, 29 September–2 October 2014; pp. 2–9.
6. Baraldi, P.; Mangili, F.; Zio, E. A Kalman filter-based ensemble approach with application to turbine creep prognostics. *IEEE Trans. Reliab.* **2012**, *61*, 966–977. [CrossRef]
7. Bechhoefer, E.; Clark, S.; He, D. A state space model for vibration based prognostics. In Proceedings of the 2010 Annual Conference of the Prognostics and Health Management Society, Portland, OR, USA, 10–16 October 2010.
8. Codetta-Raiteri, D.; Portinale, L. Dynamic Bayesian networks for fault detection, identification, and recovery in autonomous spacecraft. *IEEE Trans. Syst. Man Cybern. Syst.* **2015**, *45*, 13–24. [CrossRef]
9. Yin, X.; Li, Z. Reliable decentralized fault prognosis of discrete-event systems. *IEEE Trans. Syst. Man Cybern. Syst.* **2016**, *46*, 1598–1603. [CrossRef]
10. Daigle, M.J.; Goebel, K. Model-based prognostics with concurrent damage progression processes. *IEEE Trans. Syst. Man Cybern. Syst.* **2013**, *43*, 535–546. [CrossRef]
11. Baraldi, P.; Compare, M.; Sauco, S.; Zio, E. Ensemble neural network-based particle filtering for prognostics. *Mech. Syst. Signal Process.* **2013**, *41*, 288–300. [CrossRef]
12. Chen, C.; Zhang, B.; Vachtsevanos, G.; Orchard, M. Machine condition prediction based on adaptive neuro–fuzzy and high-order particle filtering. *IEEE Trans. Ind. Electron.* **2011**, *58*, 4353–4364. [CrossRef]

13. He, D.; Bechhoefer, E.; Ma, J.; Li, R. Particle filtering based gear prognostics using one-dimensional health index. In Proceedings of the 2011 Annual Conference of the Prognostics and Health Management Society, Montreal, QC, Canada, 25–29 September 2011.
14. Daroogheh, N.; Baniamerian, A.; Meskin, N.; Khorasani, K. Prognosis and health monitoring of nonlinear systems using a hybrid scheme through integration of PFs and neural networks. *IEEE Trans. Syst. Man Cybern. Syst.* **2016**, *PP*, 1–15. [CrossRef]
15. Bououden, S.; Chadli, M.; Allouani, F.; Filali, S. A new approach for fuzzy predictive adaptive controller design using particle swarm optimization algorithm. *Int. J. Innov. Comput. Inf. Control* **2013**, *9*, 3741–3758.
16. Arulampalam, M.S.; Makell, S.; Gordon, N.; Clapp, T. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Trans. Signal Process.* **2002**, *50*, 174–188. [CrossRef]
17. Yoon, J.; He, D. Development of an efficient prognostic estimator. *J. Fail. Anal. Prev.* **2015**, *15*, 129–138. [CrossRef]
18. Hinton, G.E.; Osindero, S.; Teh, Y.-W. A fast learning algorithm for deep belief nets. *Neural Comput.* **2006**, *18*, 1527–1554. [CrossRef] [PubMed]
19. Bengio, Y.; Courville, A.; Vincent, P. Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.* **2013**, *35*, 1798–1828. [CrossRef] [PubMed]
20. Silver, D.; Huang, A.; Maddison, C.J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. Mastering the game of Go with deep neural networks and tree search. *Nature* **2016**, *529*, 484–489. [CrossRef] [PubMed]
21. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [CrossRef] [PubMed]
22. Chen, Z.; Zeng, X.; Li, W.; Liao, G. Machine fault classification using deep belief network. In Proceedings of the 2016 IEEE International Instrumentation and Measurement Technology Conference, Taipei, Taiwan, 23–26 May 2016; pp. 1–6.
23. Shao, H.; Jiang, H.; Zhang, X.; Niu, M. Rolling bearing fault diagnosis using an optimization deep belief network. *Meas. Sci. Technol.* **2015**, *26*, 115002. [CrossRef]
24. Lv, Y.; Duan, Y.; Kang, W.; Li, Z.; Wang, F.-Y. Traffic flow prediction with big data: A deep learning approach. *IEEE Trans. Intell. Transp. Syst.* **2015**, *16*, 865–873. [CrossRef]
25. Hossain, M.; Rekabdar, B.; Louis, S.J.; Dascalu, S. Forecasting the weather of Nevada: A deep learning approach. In Proceedings of the 2015 International Joint Conference on Neural Networks (IJCNN), Killarney, Ireland, 12–17 July 2015; pp. 1–6.
26. Tao, Y.; Chen, H.; Qiu, C. Wind power prediction and pattern feature based on deep learning method. In Proceedings of the 2014 IEEE PES Asia-Pacific Power and Energy Engineering Conference (APPEEC), Hong Kong, China, 7–10 December 2014; pp. 1–4.
27. Oliveira, T.P.; Barbar, J.S.; Soares, A.S. Multilayer perceptron and stacked autoencoder for Internet traffic prediction. In Proceedings of the 11th IFIP International Conference on Network and Parallel Computing (NPC), Ilan, Taiwan, 18–20 September 2014; Springer: Berlin/Heidelberg, Germany, 2014; pp. 61–71.
28. Deutsch, J.; He, D. Using deep learning based approach to predict remaining useful life of rotating components. *IEEE Trans. Syst. Man Cybern. Syst.* **2017**, *PP*, 1–10. [CrossRef]
29. Smolensky, P. Information Processing in Dynamical Systems: Foundations of Harmony Theory. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations*; MIT Press: Cambridge, MA, USA, 1986; Chapter 6, pp. 194–281.
30. Hinton, G.E. Training products of experts by minimizing contrastive divergence. *Neural Comput.* **2002**, *14*, 1711–1800. [CrossRef] [PubMed]
31. Bengio, Y.; Lamblin, P.; Popovici, D.; Larochelle, H. Greedy layer-wise training of deep networks. In Proceedings of the 19th International Conference on Neural Information Processing Systems, Vancouver, BC, Canada, 4–7 December 2006; MIT Press: Cambridge, MA, USA, 2006; Volume 19, pp. 153–160.
32. Erhan, D.; Bengio, Y.; Courville, A.; Manzagol, P.; Vincent, P.; Bengio, S. Why does unsupervised pre-training help deep learning? *J. Mach. Learn. Res.* **2010**, *11*, 625–660.
33. Gordon, N.; Salmond, D.; Smith, A. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proc. F Radar Signal Process.* **1993**, *140*, 107–113. [CrossRef]
34. Hornik, K.; Stinchcombe, M.; White, H. Multilayer feedforward networks are universal approximators. *Neural Netw.* **1989**, *2*, 359–366. [CrossRef]

35. Frank, R.J.; Davey, N.; Hunt, S.P. Time series prediction and neural networks. *J. Intell. Robot. Syst.* **2001**, *31*, 91–103. [CrossRef]
36. Khosravi, A.; Nahavandi, S.; Creighton, D.; Atiya, A.F. Comprehensive review of neural network-based prediction intervals and new advances. *IEEE Trans. Neural Netw.* **2011**, *22*, 1341–1356. [CrossRef] [PubMed]
37. Al-Dahidi, S.; Di Maio, F.; Baraldi, P.; Zio, E. Remaining useful life estimation in heterogeneous fleets working under variable operating conditions. *Reliab. Eng. Syst. Saf.* **2016**, *156*, 109–124. [CrossRef]
38. Saxena, A.; Celaya, J.; Saha, B.; Saha, S.; Goebel, K. Metrics for offline evaluation of prognostic performance. *Int. J. Progn. Health Manag.* **2010**, *1*, 4–23.
39. Bergstra, J.; Bengio, Y. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.* **2002**, *13*, 281–305.
40. Glorot, X.; Bordes, A.; Bengio, Y. Deep sparse rectifier neural networks. In Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS), Ft. Lauderdale, FL, USA, 11–13 April 2011; pp. 315–323.
41. Maio, F.; Zio, E. Failure prognostics by a data-driven similarity-based approach. *Int. J. Reliab. Qual. Saf. Eng.* **2013**, *20*, 1–17.