Development of Deep Learning Based Prognostics

BY

JASON DEUTSCH
B.S., University of Illinois at Chicago, Chicago 2013
M.S., University of Illinois at Chicago, Chicago, 2017

THESIS

Submitted as partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Industrial Engineering and Operations Research
in the Graduate College of the
University of Illinois at Chicago, 2018

Chicago, Illinois

Defense Committee:

David He, Chair and Advisor
Lin Li, Mechanical and Industrial Engineering
Michael Brown, Mechanical and Industrial Engineering
Yayue Pan, Mechanical and Industrial Engineering
Eric Bechhoefer, GPMS Inc

ACKNOWLEDGEMENTS

I would first like to thank and express my sincere gratitude to my advisor Dr. David He for the many valuable discussions and meetings we've had over the course of my graduate studies. Much of this research would simply not have been possible without his guidance, expertise, and insight. Dr. He has not only helped guide my research by recommending valuable research papers, but has always encouraged me to continually improve on all of my experimental results. In addition, I thank Dr. Eric Bechhoefer of GPMS Inc. for funding my research and being a part of my thesis committee; without his support none of this research would be possible. Dr. Lin Li, Dr. Michael Brown, and Dr. Yayue Pan have provided great direction for my research and I thank all of them for being members of my thesis committee.

Furthermore, I would like to thank former Ph.D students Dr. Brandon Van Hecke, Dr. Jae Yoon, Dr. Miao He, and current Ph.D student Khaled Akhad of the Intelligent Systems Modeling & Development Laboratory for all of their assistance and fruitful conversations. I especially would like to thank Dr. Van Hecke and Dr. Yoon for encouraging me to pursue my Ph.D in Industrial Engineering and Operations Research.

Lastly, I thank all of my friends and family for all of their support during this entire process. My parents have been especially supportive during my time as Ph.D student.

I would also like to give a special thanks to Paula Dempsey for providing all of the gear run to failure test data, as well as very detailed photographs of the data collection process seen in Section 4.2

CONTRIBUTION OF AUTHORS

Section 2 is comprised of a combination of materials published in (Deutsch and He 2016), (Deutsch *et al.* 2017), and (Deutsch and He 2017). The majority of Sections 3.2-3.3 comes from the materials published in (Deutsch and He 2016). The description of the results based on Sections 3.2 can be found in Section 5.1, which come from the same paper. The majority of Sections 3.4-3.5 comes from the materials published in (Deutsch and He 2017). The results based on this section can be found in Section 5.2, which come from the same paper. The majority of Sections 3.6-3.7 comes from the materials published in (Deutsch *et al.* 2017). The results based on this section can be found in Section 5.3, which come from the same paper. Section 3.10 contains some work from (He and He 2018). Sections 4.1 and 4.2 contain materials mainly published in (Deutsch *et al.* 2017) and (Deutsch and He 2017) respectively. In section 4.2 the description of how the condition indicators are processed contains work previously published in (He *et al.* 2011), which my advisor Dr. David He was the primary author of. In all of the other mentioned papers, I was the primary author and I am solely responsible for all of the mathematical formulations, ideas, and theories.

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES (continued)

LIST OF FIGURES (continued)

LIST OF ABBREVIATIONS

BPA	Backpropagation Algorithm
CBM	Condition Based Maintenance
CI	Condition Indicators
DBN	Deep Belief Network
DFT	Discrete Fourier Transform
FFT	Fast Fourier Transform
FNN	Feedforward Neural Network
HI	Health Index
HUMS	Health and Usage Monitoring System
IDFT	Inverse Discrete Fourier Transform
IID	Independently and Identically Distributed
LRELU	Leaky Rectified Linear Unit
MAPE	Mean Absolute Percentage Error
MDN	Mixture Density Network
MSE	Mean Squared Error
ODM	Oil Debris Mass
PDF	Probability Density Function
PHM	Prognostic and Health Management
RBM	Restricted Boltzmann Machine
RELU	Rectified Linear Unit
RMS	Root Mean Square
RMSE	Root Mean Square Error

LIST OF ABBREVIATIONS (continued)

RUL	Remaining Useful Life
TSA	Time Synchronous Average
TSR	Time Synchronous Resampling

SUMMARY

This summary is mainly comprised of materials published in (Deutsch and He 2016)[1], (Deutsch et al. 2017), (Deutsch 2017)[2], and (Deutsch and He 2017).

Rotating components such as bearings and gears are one of the most critical components in many industrial machines. Predicting the remaining useful life (RUL) of these components has been an important task for condition-based maintenance of industrial machines. Accurately predicting the RUL of rotating components is of great interest to many industries as it reduces maintenance costs, improves efficiency by optimizing component life and in some areas, it may improve safety. Critical challenges for performing such a task in the age of Internet of Things and Industrial 4.0 are to automatically process massive data and to accurately predict the RUL of these components. Prognostic and health management (PHM) systems can be utilized to capture massive real-time data from mechanical equipment. Effectively mining features from this data and accurately predicting the RUL of the rotating components with new advanced methods become issues in PHM.

The limitations of current methods rely quite heavily on user expertise in signal processing and explicit model equations such as the state transition model and measurement distribution model and therefore are limited in today's age of big data. These approaches to modeling the RUL are known as the physics based approaches.

Among these physics based approaches, particle filters (Arulampalam *et al.* 2002*)* have emerged 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. Due to the inherent noisy structure of data, the particle filter approach requires probability density functions (PDFs) to model the uncertainty of the state of the system. These physics based approaches contrasts with data driven approaches which solely rely on data to build up models for prediction and are typically based on machine learning methods. There are also exists hybrid models that combine both the physics and data driven approach for estimating the RUL.

Somewhat recently, deep learning methods, which are a sub-branch of machine learning, have shown state of the art predictive performance in a wide array of different fields including PHM (Zhao *et al.* 2016). Many deep learning architectures have been developed for machine learning tasks including, deep neural networks, autoencoders, deep belief networks, convolutional neural networks, recurrent neural networks and more.

The objective of this research is to addresses the limitations of traditional prognostics by presenting new methods based on deep learning for RUL prediction of rotating

components. The deep learning based approaches have the ability to automatically extract important features that can be used for RUL predictions. Deep learning can in essence automatically learn the underlying physics of which describes the components

RUL based solely on the data given. The uncertainty of these predictions can be handled by automatically learning the probability distributions used by the particle filter approach by using a Mixture Density Network (Bishop 1994) or by an ensemble of neural networks. Real data collected from both gear test rig and bearing run-to-failure tests are used to test and validate the methods to be developed. The results have shown the promising RUL prediction performance of the deep learning based approaches.

The contributions of this research include:

(1) The development of a new data driven deep learning based approach for prognostic RUL estimation, which is designed by incorporating a Mixture Density Network (MDN) and a particle filter. The developed method can automatically learn the state transition function and the measurement distribution that is required to perform particle filtering without requiring the need of any a priori models. A tailored probability distribution function to model the state transition distribution for many industrial prognostic settings is proposed. In addition, the use of artificial measurements are incorporated into the traditional particle filter scheme to improve overall prognostic accuracy.

(2) A new integrated deep learning based signal processing approach for predicting the RUL. The data driven based approach utilizes a deep belief network (DBN) to automatically process and extract useful features and to directly model the state transition distribution of the RUL through an ensemble of DBNs. The resulting state transition distribution is then combined with a particle filter to directly predict the RUL, without the need of any further explicit models.

(3) Validation of the developed deep learning based prognostic methods using real bearing and gear run-to-failure test data resulting in state of the art results. Thus proving the efficacy of deep learning based approaches for prognostics with scalability and providing motivation for further research.

1. INTRODUCTION

1.1 Prognostics – A Brief Introduction

Rotating machinery is widely used in practically every industry. Both bearings and gears are used in almost all types of machinery. Unfortunately, these parts degrade over time and eventually need to be replaced. Bearings may fail due to variety of reasons, including lubrication failure, corrosion, normal fatigue, static overloads, and more. These effects typically occur on the bearing's inner race, outer race, and rolling element. Out of all these possible reasons for failure, lubrication failure is typically the most common reason for bearing failures which can result in metal to metal contact and overheating, causing premature wear. Gear failures may be due to pitting, spalling, faitigue cracks, tooth breakage, as well as other degradation mechanisms. (Cubillo *et al.* 2016). Tracking the degradation trajectories of these mechanisms is of utmost importance for predicting the RUL of these components and is vital for providing proper maintenance; this is what defines prognostics. Prognostics contrasts with another closely related field within PHM known as diagnostics, which focuses on determining whether a component is healthy or not, where as in prognostics the focus is on describing how long the component is healthy for.

Being able to accurately predict the RUL of various components has the potential for increased cost savings, due to lower maintenance costs. This is due in part to better productivity and minimization of machine downtime, as well as reducing the total life cycle cost of components when a prognostic's strategy is implemented (Elattar *et al.*

2016). In fact, it is estimated that maintenance costs alone account for approximately 15% of production costs in many industries (Song and Lee 2013). There are also of course safety implications in reliably predicting the RUL of both bearings and gears, as these components are extensively used in aircrafts. Prognostics to this day, remains an open and challenging area of research.

1.2 Condition-Based Maintenance on Rotating Machinery

There are two common types of maintenance strategies used for PHM. The first strategy is known as scheduled maintenance. This strategy sets a specific schedule based on time for providing maintenance for various rotating components without necessarily taking the condition of the component into account. This contrasts with what is known as condition based maintenance (CBM) which takes the condition of the component by possibly using several measurements into account. There consists of three main steps in a CBM program (Jardine *et al.* 2006):

1. Data Acquisition. This step involves obtaining data relevant to the system's health
2. Data Processing. This step involves analyzing and understanding the data collected.
3. Decision Making. This step involves taking preventive maintenance actions based on the collected data.

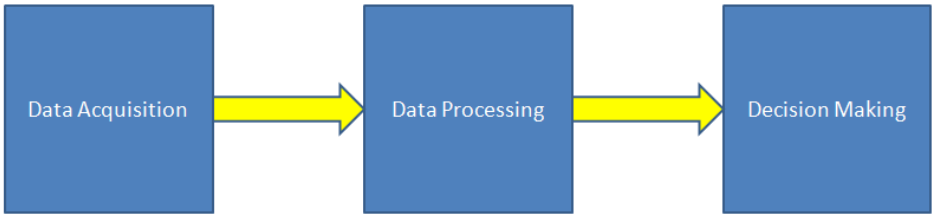

Figure 1.1. Three steps of CBM

In prognostics for rotating machinery, much of the data that is acquired is obtained from sensor data which typically captures the vibrations of the rotating components, as these vibrations correlate with the degradation state of the system. In the data processing step, much research has been developed in order to make sense of the collected data and heavily relies on techniques from signal processing. These techniques involve analyzing waveforms based on their time-domain and frequency domain, as well as performing a time-frequency analysis, which analyzes both the time and frequency domain which has been developed for non-stationary waveform signals (Jardine *et al.* 2006).

The next step for CBM in the prognostic setting is typically to develop a model using the extracted features in step two in order to provide an estimate for the RUL. The methods used to estimate the RUL can be classified as either physics based, data-driven, or a combination of the two (hybrid). 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 and are

based on statistical/machine learning techniques (Cubillo *et al.* 2016). Once an estimate is made for the RUL of a component, appropriate maintenance strategies and decisions can be made.

1.3 Objective

The objectives of this research includes: (1) Development of deep learning based methodologies for prognostics in order to accurately predict the RUL which can be applied towards rotating components using condition indicators. (2) Validating these approaches on real world data collected from run-to-failure tests on hybrid ceramic bearings and a spiral bevel gear test rig.

Various deep learning based approaches with varying degrees of complexity are investigated and are analyzed for their ability to accurately model the RUL. The main deep learning architecture that will be used throughout this research will be based on a restricted Boltzmann machine (RBM) (Smolensky 1986). The main benefit of using the RBM for prognostics is based on its power to automatically extract and process the data into meaningful features without the use of more complicated signal processing techniques. The extracted features are then ultimately useful for their ability to predict the degradation of rotating machinery.

The two most complex methodologies combine the deep learning architectures with the standard particle filter based approach. These approaches require the development of

PDFs, which are modeled using deep learning architectures in two different ways. The first method models the PDFs using resampling techniques and the other method uses a MDN to automatically model the PDFs. While these two methods differ in their modeling approach, both of these methods benefit from incorporating a particle filter since it allows for modeling of the inherent uncertainty that exists in RUL estimations.

The results for the deep learning based prognostic approaches show promising potential for RUL estimation and provide a more accurate estimation of the RUL when compared to a more traditional based particle filter.

1.4 Outline

The structure of this research is organized as follows:

Section 2 of this research details current methods and techniques used in prognostics and provides relevant background information regarding deep learning. Section 3 provides a detailed description of the deep learning methods and their specific application for prognostics. Section 4 describes the experimental run-to-fail tests used for validating the deep learning based methodologies. In Section 5, the validated results for each of the methodologies will be presented. Finally, the conclusions are provided in Section 6.

2. LITERATURE REVIEW

This section is mainly comprised of materials published in (Deutsch and He 2016), (Deutsch et al. 2017), (Deutsch 2017), and (Deutsch and He 2017).

2.1 Rotating Machinery Fault Prognosis

Remaining useful life has been used as an important parameter for condition-based maintenance decision making (Huynh *et al*. 2014). In recent years, many RUL prediction methods have been proposed for PHM. In comparison with model-based techniques, data-driven approaches can be designed and easily applied to systems when massive sensor data is available. A recent review of data-driven approaches can be found in (Si *et al* 2013). Traditionally, data-driven prognostics are largely dependent on signal processing and feature extraction techniques.

Over the past years, many prognostic methods that require explicit model equations have been developed (Vachtsevanos et al., 2006). For example, recurrent neural networks (Malhi *et al*. 2011), Heimes 2008), Kalman filter (Lim *et al*. 2014, Baraldi *et al*. 2012, Bechhoefer *et al*. 2010), dynamic Bayesian networks (Codetta-Raiteri and Portinale 2015), k-reliable decentralized prognosis (Yin and Li 2015), particle filter (Daigle and Goebel 2013, Baraldi *et al*. 2013a, Chen *et al*. 2011, He *et al*. 2011), and combined particle filter and neural networks (Daroogheh *et al*. 2016). In addition, some fuzzy systems-based approaches for prediction have been developed (Bououden *et al.* 2013). 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 (Arulampalam *et al*. 2002) have emerged in recent years as a comparatively good RUL prediction method. While similar to the Kalman filter, the particle filter is able to handle situations in which the state dynamics are not governed by Gaussian distributions. For example, (Yoon and He 2015) 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 the 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. For instance, in signal processing, there are a plethora of ways to extract features in the time domain alone. These might include computing the common statistical features such as the peak value, mean, skewness, kurotsis, crest indicator, shape indicator, impulse indicator, root mean square (RMS), and/or clearance indicator. There are also a wide variety of statistical features developed specifically for gear damage detection (Lei and Zuo 2009). There are also many other forms of analysis that may be performed using the Fast Fourier Transform (FFT), wavelet analysis, Hilbert–Huang Transform, as well other algorithms (Lee *et al.* 2014). 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. For instance, one may have to look at certain frequency

components of interest and extract useful features from that signal when performing a spectral analysis using a FFT (Jardine *et al.* 2006).

Various explicit state dynamic models have been proposed for prognostics. For instance, one might assume that the component crack growth follows the famous Paris-Erdogan model and the state dynamics can be described by the following equation (Myötyri *et al.* 2006):

$$x_t = x_{t-1} + e^{\omega_t} C(\Delta K)^n \tag{2.1}$$

Where x_t represents the crack depth at time t, ω_t represents independent and identically distributed Gaussian random variables, ΔK represents the stress intensity amplitude, and both C and n are material constants that can be estimated from experimental data. Equation (2.1) suggests that the crack growth follows a Markov process with independent and non-stationary increments. However, there are also other possibilities that can be used to describe the state dynamics of the crack growth (Myötyri *et al.* 2006). 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 for instance a traditional particle filter approach cannot be used (Baraldi *et al.* 2013a).

2.2 Deep Learning Based Methodology

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. These methods have the ability to automatically learn important relationships within the data and can for instance automatically learn the state dynamics of the systems without an explicit model equation. Since, the introduction of a deep belief network (DBN) (Hinton *et al.* 2006), 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 (Bengio *et al*. 2013). These methods have the capability of processing big data and mining hidden information due to their multiple layer structures and are able to perform highly nonlinear operations. This contrasts with more shallow architectures, which only have a single hidden layer and typically do not learn highly complex representations of the data. The recent success of AlphaGo by Google Deepmind has demonstrated the power of deep learning for big data processing and feature learning (Silver *et al.* 2016). 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.

There have been great successes in building deep neural network architectures in various domains such as image recognition, automatic speech recognition, and natural language processing (LeCun et al. 2015), and many more. Deep learning models have recently

shown promising results for machine fault diagnostics on extraction of raw vibration signals (Chen *et al*. 2016) as well as time domain features (Shao *et al*. 2015). 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 (Lv *et al*. 2015), weather (Hossain *et al*. 2015), wind speed (Tao *et al*. 2014), and internet traffic (Oliveira *et al*. 2014). There are many types of deep learning algorithms present including auto encoders, restricted Boltzmann machines, deep belief networks, convolutional neural networks, and more that can also be used for prediction problems. Deep learning represents an attractive option to process mechanical big data for RUL prediction as deep learning has the ability to automatically select features that otherwise require much skill, time, and experience.

In this research, four different experiments of varying degrees of complexity are used for prognostics. The first method utilizes a deep learning based approach based on a RBM for bearing remaining useful life prediction using vibration sensors. The second method is developed by using 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 approach is tested and validated using data collected from a gear test rig and bearing run-to-failure tests.

The last two experiments investigate a method that can use the strength of deep learning to overcome the limitations of the particle filter. The first approach uses 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. 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 a traditional based particle filter. The second approach combines a MDN with the particle filter algorithm for RUL estimation.

The presented approaches are tested and compared with existing PHM methods. The test results show that the presented methods and can overcome the above mentioned limitations of the traditional data-driven approaches without human intervention in the age of big data.

3. METHODOLOGY

This section is mainly comprised of materials published in (Deutsch and He 2016), (Deutsch et al. 2017), (Deutsch 2017), and (Deutsch and He 2017).

3.1 Overview of the Deep Learning Based Approaches for Prognostics

One of the goals that deep learning and to a larger extent machine learning algorithms attempts to solve is to learn a function that can describe an output. That is by having a database of inputs (independent variables) and desired outputs (dependent variables) these models attempt to learn their relationship; this process is known as supervised learning and requires training tuples of the form (inputs,outputs). In the case of prognostics, the goal is to predict the RUL at L steps ahead into the future. Supervised learning contrasts slightly with another technique used in machine learning known as unsupervised learning in which only the inputs are used. The idea behind unsupervised learning is to learn the distribution behind how the data (inputs) is generated. In the next sections relevant background information regarding the mechanics of how the deep learning models (specifically the RBM and DBN) perform these tasks and how it can be used for RUL estimation will be presented. This section will conclude with the combined MDN and particle filter-based approach for RUL prediction.

3.2 The Restricted Boltzmann Machine Structure

A RBM (Smolensky 1986) 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 3.1. 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 3.1, a_i represents the bias term in the visible layer, and b_j in the hidden layer.

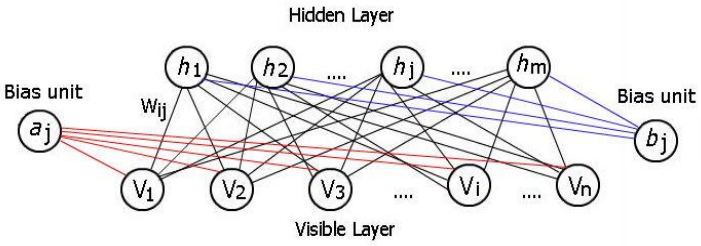

Figure 3.1. A restricted Boltzmann machine

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

$$P(\boldsymbol{v}) = \frac{1}{Z}\sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})} \tag{3.1}$$

where Z is the normalization constant that can be obtained by summing over all the possible pairs of visible and hidden vectors:

$$Z = \sum_{\boldsymbol{v}} \sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})} \tag{3.2}$$

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

$$E(\boldsymbol{v}, \boldsymbol{h}) = -\boldsymbol{a}^T \boldsymbol{v} - \boldsymbol{b}^T \boldsymbol{h} - \boldsymbol{v}^T \boldsymbol{W} \boldsymbol{h} \tag{3.3}$$

Theoretically, the problem of maximizing Equation (3.1) can be solved by taking its partial log derivative with respect to its parameters $\boldsymbol{W}$, $\boldsymbol{a}$, and $\boldsymbol{b}$:

$$\frac{\partial \log\big(P(\boldsymbol{v})\big)}{\partial \boldsymbol{W}, \boldsymbol{a}, \boldsymbol{b}} = \sum_{\boldsymbol{v},\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h}) - \sum_{\boldsymbol{h}} E(\boldsymbol{v}, \boldsymbol{h}) \tag{3.4}$$

Normally Equation (3.4) can also be written as:

$$\frac{\partial \log\big(P(\boldsymbol{v})\big)}{\partial \boldsymbol{W}, \boldsymbol{a}, \boldsymbol{b}} = < v_i h_j >_{data} - < v_i h_j >_{model} \tag{3.5}$$

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 (Hinton 2002).

$$
\begin{aligned}
W_{ij}{}^{k} &= W_{ij}{}^{k} + \gamma\left(< v_i{}^{k} h_j >_{data} - < v_i{}^{k} h_j >_{T}\right) \\
&= W_{ij}{}^{k} + \gamma\left[P\left(h_j = 1 \middle| \boldsymbol{v}\right) v_i - v_i{}^{k} P\left(h_j = 1 \middle| \boldsymbol{v}^{k}\right)\right] \\
a_i{}^{k} &= a_i{}^{k} + \gamma(< v_i{}^{k} >_{data} - < v_i{}^{k} >_{T}) \\
&= a_i{}^{k} + \gamma[P(v_i = 1 | \boldsymbol{h}) - v_i{}^{k}] \\
b_j{}^{k} &= b_j{}^{k} + \gamma\left(< h_j >_{data} - < h_j{}^{k} >_{T}\right) \\
&= b_j{}^{k} + \gamma\left[P\left(h_j = 1 \middle| \boldsymbol{v}\right) - P\left(h_j = 1 \middle| \boldsymbol{v}^{k}\right)\right]
\end{aligned}
\tag{3.6}
$$

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:

$$
P\left(h_j = 1 \middle| \boldsymbol{v}\right) = \sigma\left(b_j + \sum_{i=1}^{n} W_{ij} v_i\right) \tag{3.7}
$$

$$
P(v_i = 1 | \boldsymbol{h}) = \sigma\left(a_i + \sum_{j=1}^{m} W_{ij} h_j\right) \tag{3.8}
$$

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:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3.9}$$

3.3 Deep Learning Based Bearing Prognostics Using a Restricted Boltzmann Machine

In order to use a RBM as a prognostics model, the weights and biases can first be learned in the unsupervised stage of learning illustrated in the previous section. Once the optimal parameters have been determined the output of the last layer (hidden features learned) can be used as an input to a supervised learning algorithm; a linear regression layer was used as the last layer to make the L-step ahead predictions. The RMS values are used as the fault feature to determine the components degradation over time. This feature serves as the input into the RBM. The RMS at each time interval (denoted as x_t) can be calculated as follows:

$$x_t = \sqrt{\frac{1}{n}\sum_{i=1}^{n} f_{ti}^2} \tag{3.10}$$

where f_{ti} represents the ith raw vibration data point at time interval t and n is the length of the signal.

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:

$$[x_t, x_{t-1}, \ldots., x_{t-d+1}], \ \epsilon \mathbb{R}^d \tag{3.11}$$

and the output as:

$$[x_{t+L}, x_{t+L+1}, \ldots, x_n] \tag{3.12}$$

where d represents the embedding dimension, and determines the size of the first visible layer in the RBM. Thus, yielding training tuples of $([x_t, x_{t-1}, \ldots., x_{t-d+1}], x_{t+L}\ )$, $t = d - 1, \ldots, n - L - 1$.

Equations (3.11) and (3.12) define a method often called the sliding window technique (Frank *et al.* 2001). This windowing approach is best illustrated below in Fig. 3.2. The first plot in Fig. 3.2 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,..., 98, 99 (first row of features) and the

second window would contain points from t = 1, 2, 3,..., 99, 100 (second row of features) and so on.

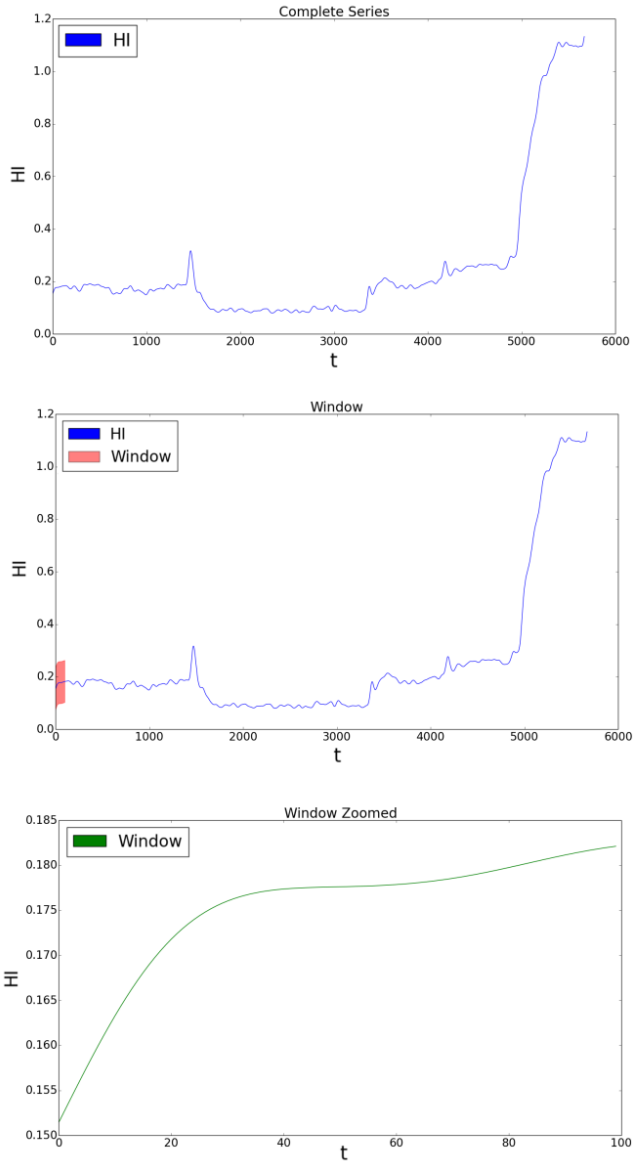

Figure 3.2. The windowing approach

Once the data has been constructed the RBM can essentially perform automatic feature engineering in order to better capture the dependency of the lagged RMS values onto the future RMS values and thus avoiding the use of some more complex manual feature extractions of the data. The predicted RUL can then be computed by using the predicted RMS values and the time of the bearing's failure. One can estimate the predicted RUL by the following equation:

$$\widehat{RUL_t} = t_{life} - \phi(F_t) \tag{3.13}$$

where $\widehat{RUL_t}$ is the predicted remaining useful life at some time t, $t_{t_{life}}$ is the total time of the bearing's life, and $\phi(F_t)$ is a function that maps F_t, the predicted RMS value, to an estimated point in time of the bearing's life. This is done by simply taking a polynomial curve fit of the RMS values (as a function of time) and solving for the value of t given the predicted RMS value.

3.4 The Deep Belief Network Structure

A DBN is formed by stacking multiple RBMs on top of each other (Figure 3.3). in order to create high representations of data that can be used for classification, regression (continuous output) tasks as well as unsupervised learning.

The RBM becomes a building block for forming a deep belief network. The DBN can be trained in greedy-layer-wise fashion by stacking RBMs on top of each other (Bengio *et al.* 2007). 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.

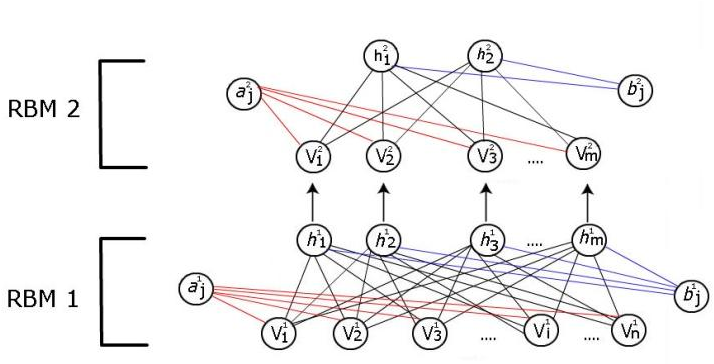

Figure 3.3. A deep belief network with two hidden layers

3.5 Deep Learning Based Bearing Prognostics using a Deep Belief Network

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 3.4. The parameters of

the entire network are then updated using the back-propagation algorithm in the same way as a 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 (Erhan *et al.* 2010).

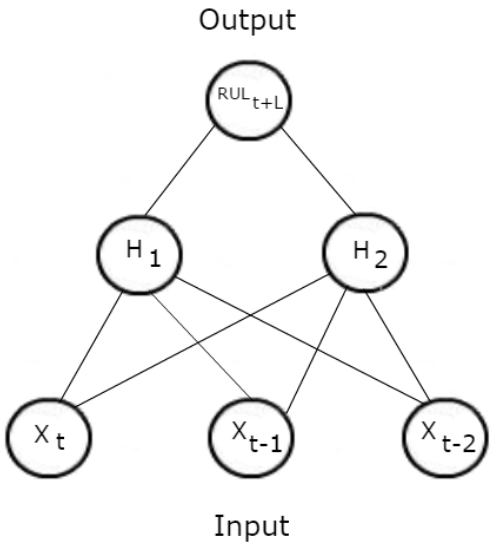

Figure 3.4. A feedforward neural network with $d = 3$ and a single hidden layer.

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. The same process of establishing explicit input and outputs from Equations (3.11) and (3.12) is used in the DBN based approach, however, Equation (3.12) is replaced with the actual RUL in our training set; i.e. training tuples are of the form $([x_t, x_{t-1}, \dots, x_{t-d+1}], RUL_{t+L})$, $t = d-1, \dots, n-L-1$, as it has provided better empirical results. It should be noted that the embedding dimension d defines the input size of the first layer in the DBN. The predicted RUL can then be directly estimated as a function of the fault feature by the following equation:

$$\widehat{RUL_{t+L}} = \phi(X_t, X_{t-1}, \dots, X_{t-d+1}) \tag{3.14}$$

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

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, which also may be used for confidence bounds (Efron and Gong 1983). The method is described next

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

$$\overline{\phi(.)} = \frac{1}{n}\sum_{i=1}^{n} \phi(.)_{-i} \tag{3.15}$$

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

$$\hat{\sigma} = \sqrt{\frac{n-1}{n}\sum_{i=1}^{n}\left[\overline{\phi(.)} - \phi(.)_{-i}\right]^2} \tag{3.16}$$

The confidence interval with $N-1$ degrees of freedom and with $100(1-\alpha)\%$ confidence can then be calculated as following:

$$\overline{\phi(.)} \pm t_{1-(\alpha/2),N-1}\hat{\sigma} \tag{3.17}$$

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 (BPA).

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.

3.6 The 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:

$$\boldsymbol{x}_t = f(\boldsymbol{x}_{t-1}, \boldsymbol{\omega}_{t-1}) \tag{3.18}$$

$$\boldsymbol{z}_t = h(\boldsymbol{x}_t, \boldsymbol{v}_t) \tag{3.19}$$

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. 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:

$$\begin{aligned} &\int P(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{z}_{1:t-1}) P(\boldsymbol{x}_{t-1}|\boldsymbol{z}_{1:t-1}) d\boldsymbol{x}_{t-1} \\ &= \int P(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) P(\boldsymbol{x}_{t-1}|\boldsymbol{z}_{1:t-1}) d\boldsymbol{x}_{t-1} \end{aligned} \tag{3.20}$$

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:

$$P(\boldsymbol{x}_t|\boldsymbol{z}_{1:t}) = \frac{P(\boldsymbol{x}_t|\boldsymbol{z}_{1:t-1})\ P(\boldsymbol{z}_t|\boldsymbol{x}_t)}{\int P(\boldsymbol{x}_t|\boldsymbol{z}_{1:t-1})P(\boldsymbol{z}_t|\boldsymbol{x}_t)d\boldsymbol{x}_t} \tag{3.21}$$

Typically, the calculations for Equations (3.20) and (3.21) are intractable and the particle filter sampling approach is used to approximate them. This can be accomplished by using a set of samples/particles $\{\boldsymbol{x}_t{}^i, w_t{}^i\}_{i=1}^{N_p}$, where $w_t{}^i$ is the ith weight at time t and N_p is the user-specified number of particles generated. The weights can be updated by using the bootstrap filtering algorithm (Gordon *et al.* 1993) and samples/particles are drawn from the proposal distribution:

$$\boldsymbol{x}_t^i \sim p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}^i) \tag{3.22}$$

With weight:

$$w_t^i = \frac{p(\boldsymbol{z}_t|\boldsymbol{x}_t^i)}{\sum_{i=1}^{N} p(\boldsymbol{z}_t|\boldsymbol{x}_t^i)} \tag{3.23}$$

Finally, the posterior distribution $P(\boldsymbol{x}_t|\boldsymbol{z}_{1:t-1})$ is obtained by resampling from $\{\boldsymbol{x}_t^1, \boldsymbol{x}_t^2, \ldots\ \boldsymbol{x}_t^{N_p}\}$ where $\boldsymbol{x}_t{}^i$ (predicted estimate) is drawn with probability w_t^i. This resampling step can be performed by using the multinomial resampling method (Douc

and Cappé 2005). The initial particles, $\boldsymbol{x}_0^i$, are drawn from the prior probability $P(\boldsymbol{x}_0)$ and the weights are initialized from the discrete uniform distribution, that is $w_0^i = \frac{1}{N_p}\ for\ i = 1,2, \ldots N$. It should also be noted that other particle filtering algorithms besides the bootstrap filtering algorithm may be used. However, this algorithm does not suffer from the weight degeneracy problem, in which typically after only few iterations of the particle filtering process, only a few particles contain a large weight and other particles contain a very small weight.

For purposes of terminology, $p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}^i)$ will be denoted as the state transition distribution and $p(\boldsymbol{z}_t|\boldsymbol{x}_t^i)$ will be denoted as the measurement distribution.

The entire particle filtering is best illustrated below in Figure 3.5. In Figure 3.5, each particle's color ranges from light to dark depending on its weight; the darker the color the higher the weight.

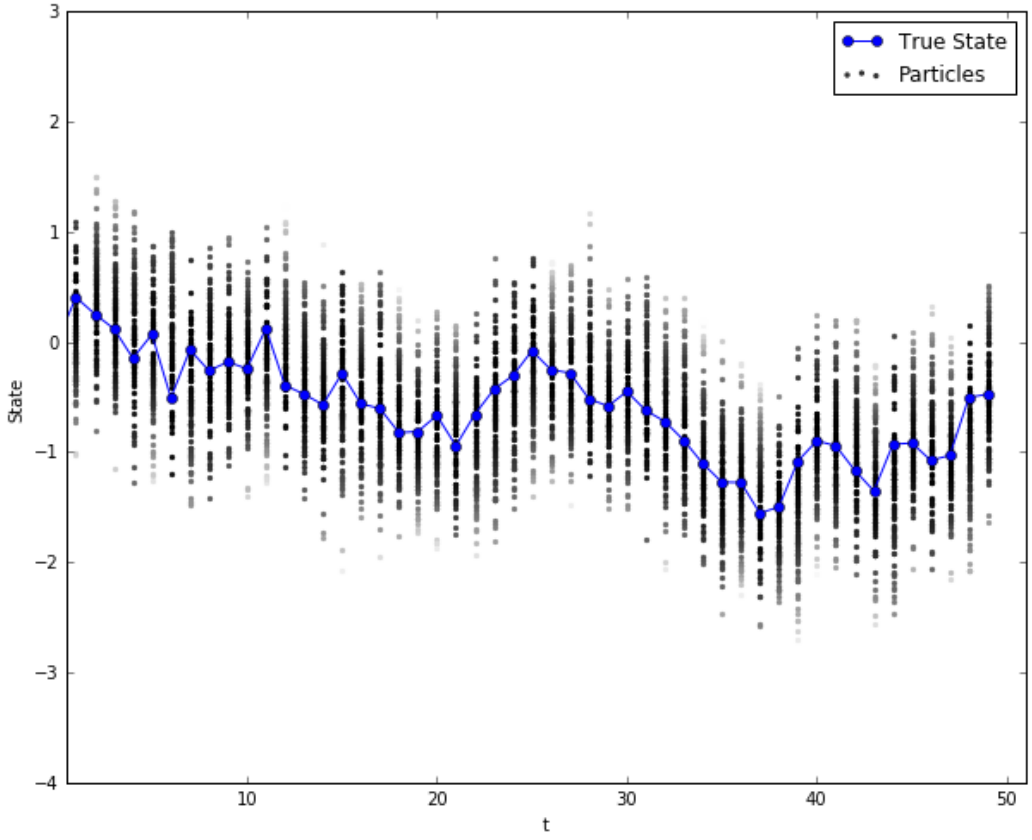

Figure 3.5. Particle filtering process

3.7 Combined Deep Belief Network and Particle Filter-Based Approaches for Remaining Useful Life 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 $L \epsilon \{1,2,\dots N-t\}$ 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 $\{0,1,\dots t\}$ may be used for the prediction.

There are two functions that are of interest in modeling; the state transition distribution Equation (3.22) (sometimes referred to as the proposal distribution) and the measurement distribution Equation (3.23). Both of these functions can be modeled by the DBN, since neural networks in general are universal function approximators (Hornik *et al.* 1989).

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 $t = 1,2,\dots N$) into a matrix as was done in the previous approaches. Formally, the input can be denoted as:

$$[\boldsymbol{z}_t, \boldsymbol{z}_{t-1}, \dots., \boldsymbol{z}_{t-d+1}], \ \epsilon \mathbb{R}^d \tag{3.24}$$

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

$$[\boldsymbol{x}_t, \boldsymbol{x}_{t+1}, \dots, \boldsymbol{x}_N] \tag{3.25}$$

Thus, the network requires training tuples of the form $([\boldsymbol{z}_t, \boldsymbol{z}_{t-1}, \dots, \boldsymbol{z}_{t-d+1}], x_t)$, $t = d-1, \dots, N$. Using this training set, the state transition model is instead actually modeling the pdf of $p(\boldsymbol{x}_{t+L}^i | \boldsymbol{z}_t, \boldsymbol{z}_{t-1}, \dots, \boldsymbol{z}_{t-d+1})$. The L step ahead prediction, $\boldsymbol{x}_{t+L}^i$, is simply made by subtracting the network's output $\boldsymbol{x}_t$ by L. The advantages of treating the state $\boldsymbol{x}_t$ (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 $\boldsymbol{z}_t$ (often a multi dimensional vector) can be constructed into a mono-dimensional feature vector by setting the input of the DBN as $\boldsymbol{z}_t$ 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 $\boldsymbol{z}_t$.

The state transition model and its respective variance in Equation (3.22) $\boldsymbol{\omega}_{t-1}$ 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:

$$ST_b(\boldsymbol{z}) = F(\boldsymbol{z}; ST_b^*) \tag{3.26}$$

where, $\boldsymbol{z}$ represents the generic testing input vector of the past d measurements Equation (3.24) and $F(\boldsymbol{z}; ST_b^*)$ represents the DBN-FNN output Equation (3.25) with respect to the bth bootstrap. The mean and variance are simply then calculated as:

$$ST_{avg}(\boldsymbol{z}) = \frac{1}{B}\sum_{b=1}^{B} ST_b(\boldsymbol{z}) \tag{3.27}$$

$$\hat{\sigma}_{ST}^2(\boldsymbol{z}) = var(ST_b(\boldsymbol{z})) \tag{3.28}$$

and then samples $\boldsymbol{x}_t^i$ can be generated from the following Gaussian distribution:

$$\mathcal{N}\big(ST_{avg}(\boldsymbol{z}), \hat{\sigma}_{ST}^2(\boldsymbol{z})\big) \tag{3.29}$$

The measurement distribution can be modeled in a somewhat similar approach to the state transition distribution. The technique (Baraldi *et al.* 2013a) requires a dataset of tuples of $(\boldsymbol{x}_t, \boldsymbol{z}_t)_{t=1}^{N_{training}}$ 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:

$$\varphi_{avg}(x) = \frac{1}{B}\sum_{b=1}^{B}\varphi(x; T_b^*) \tag{3.30}$$

where, T_b^* is the bth bootstrap sampled from the dataset and $\varphi(x; T_b^*)$ represents bth bootstrap's output given an input vector x. The measurement model in Equation (3.19) can be hypothesized as:

$$z(x) = f(x) + v(x) \tag{3.31}$$

From Equation (3.31), $\varphi_{avg}(x)$ can be subtracted from both sides such that:

$$z(x) - \varphi_{avg}(x) = [f(x) - z(x)] + v(x) \tag{3.32}$$

The error term $z(x) - \varphi_{avg}(x)$ is then a function of $[f(x) - z(x)]$ and $v(x)$, which will be called 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:

$$\varphi_{com}^m(x) = \frac{1}{K}\sum_{i=1}^{K}\varphi_i(x) \tag{3.33}$$

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:

$$\hat{\sigma}_m^2(x) = \frac{1}{P}\sum_{p=1}^{P} var(\Psi_p) \tag{3.34}$$

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

$$\hat{\alpha}_t^2 = max\left\{\left(\boldsymbol{z}_t - \varphi_{avg}(\boldsymbol{x}_t)\right)^2 - \hat{\sigma}_m^2(\boldsymbol{x}_t), 0\right\} \tag{3.35}$$

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 $p(\boldsymbol{z}_t|x_t)$ can be approximated by:

$$p(\boldsymbol{z}_t|x_t) \approx \mathcal{N}\left(\varphi_{avg}(x), \hat{\sigma}_m^2(x) + \hat{\alpha}_t^2(x)\right) \tag{3.36}$$

The confidence intervals for a generic prediction $\hat{x}$ can then be obtained by the following formula (Khosravi *et al.* 2011):

$$\hat{x} \pm t_{1-(\alpha/2),B-1}\sqrt{\frac{\hat{\sigma}_{ST}^2(x)}{B-1}} \tag{3.37}$$

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.

Approximate the state transition model:

Step 2

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 $t = d-1, \dots, N_{training} - L - 1$.

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 $\boldsymbol{x}_{t+L}$. This is accomplished by subtracting the network's output $\boldsymbol{x}_t$ by L, i.e., $\boldsymbol{x}_{t+L} = x_t - L$.

f Let $t = t + 1$, and update the training vector with input features $(\boldsymbol{z}_t, \boldsymbol{z}_{t-1}, \ldots., \boldsymbol{z}_{t-d+1})$.

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 $(\boldsymbol{x}_t, \boldsymbol{z}_t)_{t=1}^{N_{training}}$ Generate B bootstraps from this training dataset and apply Equations (3.30)–(3.36) to obtain the measurement distribution. The DBN can be used for estimates in Equations (3.30) and (3.35).

Step 4 Estimate the state. The predicted state $\widehat{\boldsymbol{x}}_{t+L}$ can be obtained by sampling $\{\boldsymbol{x}_{t+L}^1, \boldsymbol{x}_{t+L}^2, \ldots \boldsymbol{x}_{t+L}^B\}$ with probability w_t^i [from Equation (3.23)].

This method allows a complete data-driven approach to be developed in conjunction with a particle filter for RUL estimation. Equation (3.18), the proposal distribution is built up using training tuples of the form $([\boldsymbol{z}_t, \boldsymbol{z}_{t-1}, \ldots., \boldsymbol{z}_{t-d+1}], \boldsymbol{x}_t)$. The bootstrapping approach

is then used to collect the statistics for the mean and variance. Equation (3.19), the measurement model is then modeled by Equations (3.30)–(3.36). Here the DBN can serve two purposes. It can be used to reduce the dimensionality of the measurement $\boldsymbol{z}_t$ and it can also to be used in order to initialize the weights of the neural network.

3.8 Mixture Density Network

As seen in the previous section, the combined DBN and particle filter-based approach requires training two neural networks to model the measurement distribution and assumes and that the final measurement distribution $p(\boldsymbol{z}_t|\boldsymbol{x}_t)$ follows a normal distribution. These two problems however, can be rectified by using a MDN.

The MDN is capable of modeling a probability distribution for a specific target variable y given an input vector $\boldsymbol{x}$**.** This contrasts with a typical neural network in that a standard FNN's output for predicting a continuous target variable y estimates the conditional averages $\mathrm{E}(y|\boldsymbol{x})$ instead of modeling the distribution $p(y|\boldsymbol{x})$. As suggested by (Bishop 1994), these conditional averages provide only a limited description of the target variables.

In the past few years, MDNs have become a somewhat popular technique in supervised learning tasks. MDNs have shown promising results in statistical parametric speech synthesis (Zen *et al.* 2014) and have also been used for the same task by combining MDNs with a recurrent autoregressive model (Wang *et al*. 2017). MDNs have also been used to model short-term wind speeds and power projections (Men *et al*. 2016) by

employing an ensemble of MDNs, which have shown superior performance to common forecasting methods. It also been shown to outperform deep neural networks (FNNs with many hidden layers) in cleaning noisy speech observations (Kinoshita *et al.* 2017). As suggested by (Kinoshita *et al.* 2017) one reason that MDNs may outperform other methods is that more traditional neural network approaches falsely assume a one to one mapping between the noisy input and output space without any uncertainty and hence, the inverse mapping can't be achieved deterministically.

In prognostics and specifically in determining the RUL, the problem is generally not a one to one mapping. Typically, the input space from signal features is highly noisy and these features may not deterministically map to a given RUL or state. If one is to make these assumptions on this type of problem using a typical neural network, the result will be an average of several correct target values (since it minimizes the sum of squared errors), which may not necessarily be a correct prediction (Bishop 1994). This can be rectified by employing a MDN to model the probabilities over a set of target values instead of simply predicting the averages (see figures below). To date, a MDN has not been used for prognostics nor has a MDN combined with a particle filter been employed.

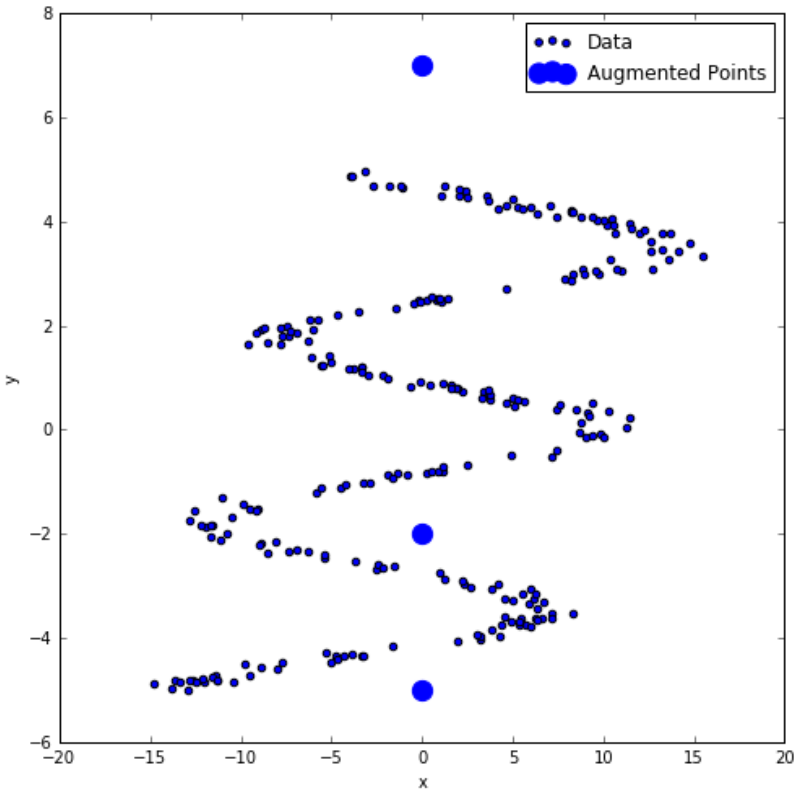

Figure 3.6. Plot of $Y = x + 10\cos(1.8x) + \epsilon$

In Figure 3.6, the plot $Y = x + 10\cos(1.8x) + \epsilon$, where ϵ represents Gaussian white noise with variance one, has been augmented with three data points centered at $x = 0$, with three different Y values to help illustrate the problems a standard neural network has in dealing with learning a function in non one to one mapping situations. After training a FFN on values of x to predict Y and using the Mean Squared Error (MSE) loss function to train the network, the network is unable to capture the information needed to accurately predict Y and as seen below in Figure 3.7, the network is predicting an average over the several possible correct values of Y. This is clearly evident by the big bump for the predicted values centered around $x = 0$. As can be seen in Figure 3.8, the MDN,

which is trained on the same data, is able to accurately predict the target variable Y. The MDN's predictions are made by computing the PDF $p(Y|x)$ across a range of Y values, and sampling each of these values with probability $p(Y|x)$.

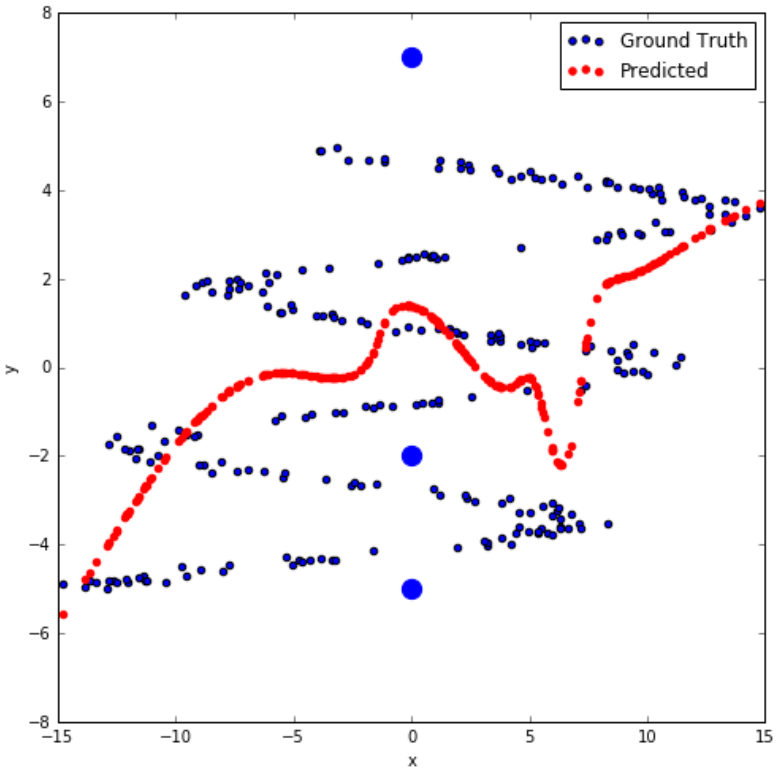

Figure 3.7. Standard feedforward neural network output

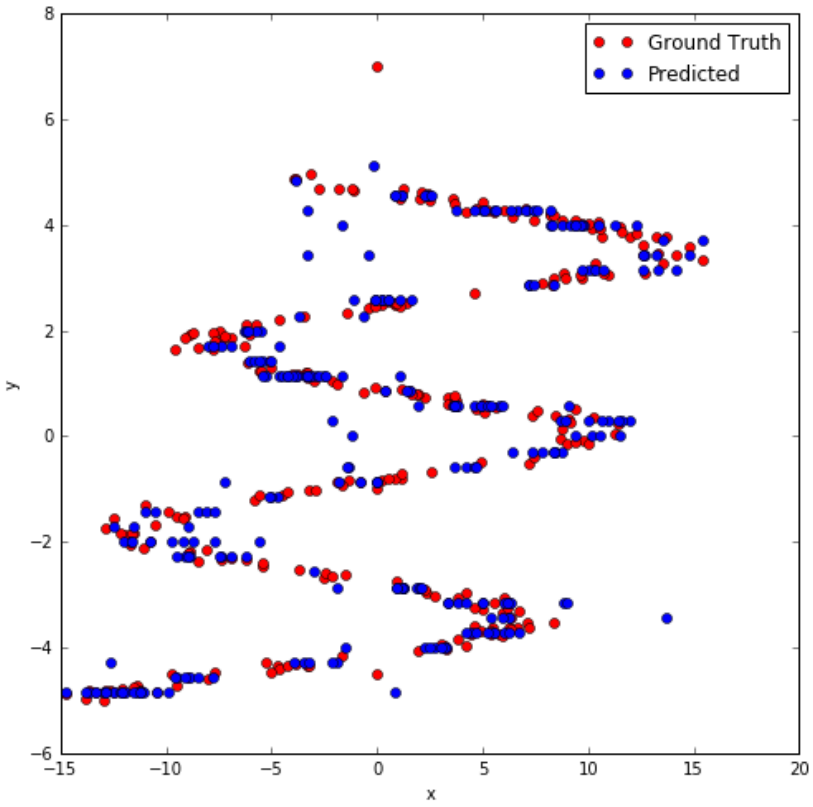

Figure 3.8. Mixture Density Network output

3.9 Mixture Density Network Methodology

In a MDN the probability distribution $p(y|\boldsymbol{x})$ can be modeled by the following equation (y is assumed to be mono-dimensional for simplicity, but the multidimensional case can be easily extended):

$$p(y|\boldsymbol{x}) = \sum_{k=1}^{K} \alpha_k(\boldsymbol{x}) \phi_k \left(y, \mu_k(\boldsymbol{x}), \Sigma_k(\boldsymbol{x})\right) \tag{3.38}$$

And subject to the constraint:

$$\sum_{k=1}^{K} \alpha_k(\boldsymbol{x}) = 1 \tag{3.39}$$

It should be noted that the standard deviation must also be positive. These constraints can be satisfied by the following equations:

$$\alpha_k = \frac{e^{z_k{}^{\alpha}}}{\sum_{j=1}^{K} e^{z_j{}^{\alpha}}} \tag{3.40}$$

$$\Sigma_k = e^{z_k{}^{\Sigma}} \tag{3.41}$$

Equation (3.38) allows for the modeling of an arbitrary probability distribution by modeling it as a weighted sum of Gaussian distributions, where α_k represents the weight of the kth Gaussian distribution $\phi_k(.)$ centered around mean μ_k and standard deviation Σ_k with K mixing components. The mean μ_k can be represented by the networks output, that is $\mu_k = z_k{}^{\mu}$. The terms, $z_k{}^{\alpha}$, $z_k{}^{\Sigma}$,$z_k{}^{\mu}$ represent the last output layer in the network

corresponding to the parameters α, Σ, and μ respectively. The topology of the MDN can be shown below in Figure 3.9.

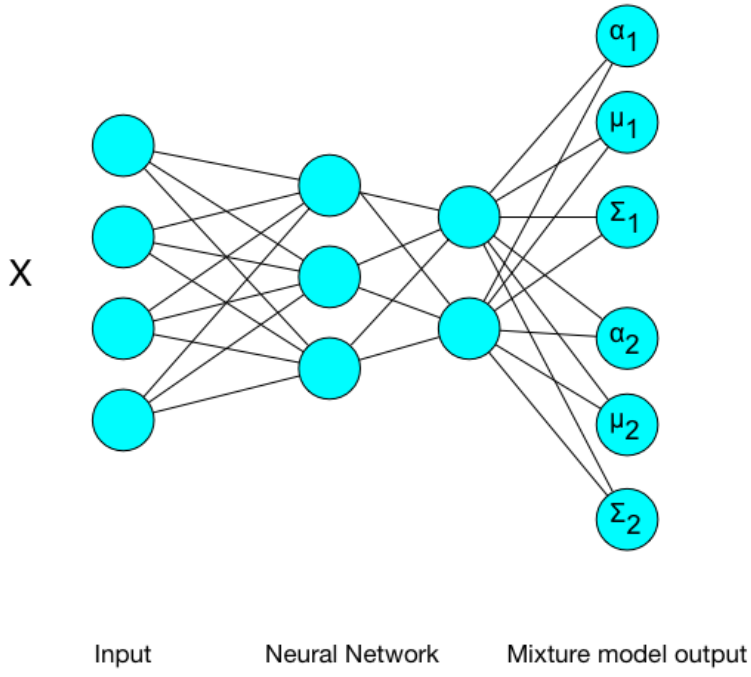

Figure 3.9. Topology of a Mixture Density Network with two hidden layers and two mixing components

Finally, the parameters of the MDN can be obtained by minimizing the negative log likelihood loss function given by:

$$L(\boldsymbol{x}, y) = -ln\left(\sum_{k=1}^{K} \alpha_k(\boldsymbol{x})\phi_k\left(y, \mu_k(\boldsymbol{x}), \Sigma_k(\boldsymbol{x})\right)\right) \tag{3.42}$$

The errors of the mixture model output can then be backpropagated towards the last output layer in the network. The gradients are then of the following form:

$$\frac{\partial L(\boldsymbol{x}, y)}{\partial {z_k}^{\alpha}} = \alpha_k - \pi_k \tag{3.43}$$

$$\frac{\partial L(\boldsymbol{x}, y)}{\partial {z_k}^{\Sigma}} = -\pi_h\left\{\frac{\|y - \mu_k\|^2}{{\Sigma_k}^2} - 1\right\} \tag{3.44}$$

$$\frac{\partial L(\boldsymbol{x}, y)}{\partial {z_k}^{\mu}} = \pi_k\left\{\frac{\mu_k - y}{{\Sigma_k}^2}\right\} \tag{3.45}$$

Where π_k is defined as:

$$\pi_k(\boldsymbol{x}, y) = \frac{\alpha_k\phi_k}{\sum_{j=1}^{K}\phi_j\alpha_j} \tag{3.46}$$

The gradient computations for all of the other layers remain unchanged.

3.10 Mixture Density Network Integrated Methodology for Gear Remaining Useful Life Estimation

The entire process of predicting the RUL using the integrated approach is illustrated below in Figure 3.10.

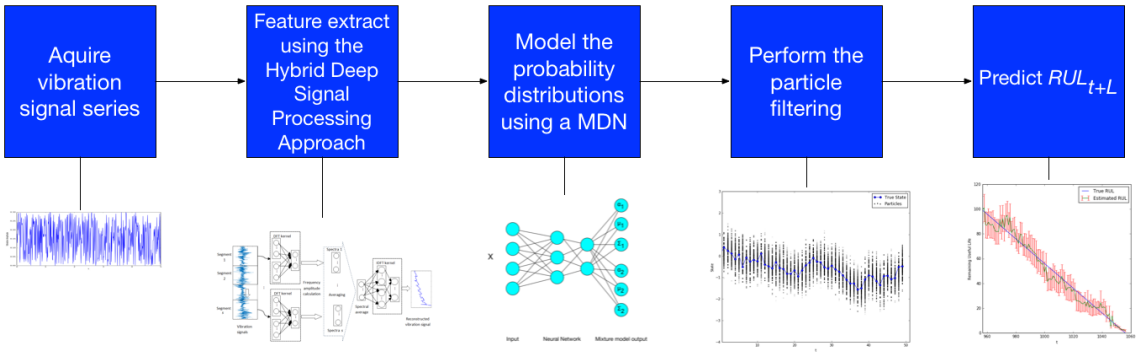

Figure 3.10. Methodology of predicting the RUL

The first two steps of Figure 3.10 pertain to using a deep learning based approach to extract features from the vibration signals. Once the vibration signals are acquired, these signals (raw vibration signals in the time domain) can be fed into the hybrid deep signal processing approach to extract the measurements, $\boldsymbol{z}_t$, which are then used in the MDN.

The third and fourth step requires the use of a MDN to model the state transition distribution and the measurement distribution. The ultimate objective of the particle filter approach is to estimate the pdf of $P(\boldsymbol{x}_{t+L}|\boldsymbol{z}_{1:t})$, where $L \epsilon \{1,2, \dots N-t\}$ 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 $\{0,1, \dots t\}$ may be used for the prediction. Both distributions (state transition and measurement) can be modeled by the MDN in order to estimate the pdf of $P(\boldsymbol{x}_{t+L}|\boldsymbol{z}_{1:t})$ and predictions can be made using the bootstrap filtering algorithm.

Training tuples of the form $(\boldsymbol{x}_{t-1}, \boldsymbol{x}_t)_{t=2}^{N_{training}}$ can be used to build up the state transition distribution and the measurement distribution can be modeled by using training tuples of the form $(\boldsymbol{x}_t, \boldsymbol{z}_t)_{t=1}^{N_{training}}$.

In the field of prognostics, the state of interest $\boldsymbol{x}_t$ is typically non negative, as the state is usually represented by measurements that are non negative (crack depth of a gear, oil debris mass, battery capacity, elongation (Baraldi *et al.* 2013b), etc.). Thus, the state transition distribution should not generate any particles that are negative (i.e. the domain of the distribution should be $\{y \in \mathbb{R} \mid y > 0\}$). Therefore, for the state transition distribution, a slight change is proposed to the standard MDN proposed in (Bishop 1994). Instead of setting the function $\phi_k(.)$ to the Gaussian distribution, the

function $\phi_k(.)$ is set to a log-normal distribution:

$$\phi_k\big(y, \mu_k(\boldsymbol{x}), \Sigma_k(\boldsymbol{x})\big) = \frac{e^{\frac{-[ln(y)-\mu_k(\boldsymbol{x})]^2}{2\Sigma_k(\boldsymbol{x})^2}}}{\Sigma_k(\boldsymbol{x}) y\sqrt{2\pi}} \tag{3.47}$$

The gradient computations in equations (3.42) and (3.43) need only to be modified slightly to reflect this change and are as follows:

$$\frac{\partial L(\boldsymbol{x}, y)}{\partial {z_k}^{\Sigma}} = -\pi_h \left\{ \frac{\|ln\,(y) - \mu_k\|^2}{{\Sigma_k}^2} - 1 \right\} \tag{3.48}$$

$$\frac{\partial L(\boldsymbol{x}, y)}{\partial {z_k}^{\mu}} = \pi_k \left\{ \frac{\mu_k - ln\,(y)}{{\Sigma_k}^2} \right\} \tag{3.49}$$

The modification to the gradients is only slight as it only requires taking the natural log of the target output variable y. The gradient computation in equation (3.43) remains unchanged. By using a log-normal distribution, a small degree of a priori knowledge is implemented in the network, which typically leads to better network generalization and optimization of the network parameters (Joerding and Meador 1991). For the measurement distribution however, the function $\phi_k(.)$ is simply set to the Gaussian distribution originally used in (Bishop 1994) as the measurement $\boldsymbol{z}_t$ may be any real

value.

In the state transition distribution samples can be drawn by first sampling a single log-normal distribution ϕ_k with probability α_k and then drawing particles $\boldsymbol{x}_t^i$ from the distribution $\phi_k \left(x_t, \mu_k\left(\boldsymbol{x}_{t-1}^i\right), \Sigma_k\left(\boldsymbol{x}_{t-1}^i\right)\right)$. Once, a particle has been drawn it is then resampled (filtered) based on the particle weights determined from equation (3.23). The RUL can then be determined from these resampled particles by propagating these particles forward in time by recursively using the state transition distribution in equation (3.22). This process allows one to model the posterior probability $p(RUL|\boldsymbol{z}_{1:t})$. The RUL is then determined by the number of steps taken for the particle to meet or exceed a predefined threshold value. The threshold value may be set to indicate when the system exhibits complete failure or more typically it is set to a lower value than at the point of failure as means of acting as a margin of safety (Saxena *et al.* 2010).

In a similar fashion to the approach used in (Liu *et al.* 2012) and (Chen *et al.* 2012), the particle filtering process can still be applied to determine the RUL after the last measurement $\boldsymbol{z}_t$ is observed. In (Liu *et al.* 2012), predicted measurements are used to update the parameters of the state transition distribution and in (Chen *et al.* 2012) they are used in conjunction with a dual particle filter for state estimation. These predicted measurements can be utilized in a traditional particle filter to help guide the trajectory of the particles after the last measurement $\boldsymbol{z}_t$ is observed in order to improve the

forecasting accuracy of the RUL. The measurement $\boldsymbol{z}_{t+1}$ can be written as:

$$\boldsymbol{z}_{t+1} = g(\boldsymbol{z}_{1:t}) + \boldsymbol{u}_{t+1} \tag{3.50}$$

Where, $g(.)$ represents the measurement prediction function (outputs the predicted measurement) based on the past history of observed measurements and $\boldsymbol{u}_t$ represents the noise vector at time t representing the prediction error. A growth model of the noise vector $\boldsymbol{u}_t$ established in (Chen *et al.* 2012) may be utilized to help model the uncertainty of the forecast as a function of the prediction horizon.

During the propagation of the particles to determine the RUL of the system, the predicted measurements can then be used in the particle filtering process to help filter out unlikely particles. The predicted measurements are used to determine the particle's weight and thus particles can then be resampled in which particles with higher weights are more likely to be propagated further.

Certainly, as the prediction horizon for the measurements increase the uncertainty of the prediction grows. It is for this reason, that when propagating particles in order to determine the RUL, only a few steps in time, denoted as N_s (number of steps), are used for predicting the measurement values; N_s in this case is analogous to the prediction horizon for the measurements.

The pseudocode for determining the RUL (assuming the state and measurement are mono-dimensional) based on this approach is provided below in Figure 3.11.

Algorithm 1 Determine Remaining Useful Life

procedure DETERMINERUL($P_s, z_{1:t}, x_t^{i\ (i=1:N_p)}, threshold, g, P_m, N_s, t$)
  $RUL \leftarrow []$
  $counter \leftarrow 0$
  while $(True)$ **do**
    if $counter < N_s$ **then**
      $z_{t+1} \leftarrow g(z_{1:t}) + u_{t+1}$ ▷ Predict the future measurement
    end if
    for $i \leftarrow 1, N_p$ **do**
      if $x_t^i \geq threshold$ **then**
        $RUL[i] \leftarrow counter$
        $x_t^i.remove(i)$ ▷ Remove corresponding particle i
      else
        $x_{t+1}^i \sim P_s(x_{t+1}|x_t^i)$ ▷ State transition model
        if $counter < N_s$ **then**
          $w_{t+1}^i \leftarrow P_m(z_{t+1}|x_{t+1}^i)$ ▷ Measurement model
        end if
      end if
    end for
    if $x_t^{i\ (i=1:N_p)} = \emptyset$ **then** ▷ If all the particles have been removed
      return RUL
    end if
    if $counter < N_s$ **then**
      for $i \leftarrow 1, N_p$ **do**
        $w_{t+1}^i \leftarrow w_{t+1}^i / \sum_{i=1}^{N_p} w_{t+1}^i$ ▷ Normalize
      end for
      $x_{t+1}^{i\ (i=1:N_p)} \leftarrow resample(\{x_{t+1}^i, w_{t+1}^i\}_{i=1}^{N_p})$
    end if
    $counter \leftarrow counter + 1$
    $t \leftarrow t + 1$
  end while
end procedure

Figure 3.11. Psuedocode for determining RUL

During the propagation of particles to determine the RUL of the system, a parameter update is not performed, simply due to the fact that a potentially large number of parameters involved in a MDN.

Finally, confidence intervals of the RUL with $100(1-\alpha)\%$ confidence of the set $\{RUL_t^i, w_t^i\}_{i=1}^{N_p}$, ($RUL_t^i$ represents the RUL calculation based on particle i at the current system operating time t) can be obtained by the following:

$$\overline{RUL_t} \pm t_{1-(\alpha/2),N_p-1}\frac{s}{\sqrt{N_p}} \tag{3.51}$$

Where $t_{1-(\alpha/2),N_p-1}$, represents the Student t distribution with quantile $1-(\alpha/2)$, N_p-1 degrees of freedom, and

$$\overline{RUL_t} = \sum_{i=1}^{N_p} w_t^i RUL_t^i \tag{3.52}$$

$$s = \sqrt{\sum_{i=1}^{N_p} w_t^i \left(\overline{RUL_t} - RUL_t^i\right)^2} \tag{3.53}$$

3.11 Prognostic Feature Extraction Using a Hybrid Deep Signal Processing Approach

In order to obtain accurate RUL forecasts by using the methodology in Section 3.10, the measurement $\boldsymbol{z}_t$ should reflect the system's state of health accurately. In addition, the algorithm in Figure 3.11 depends on being able to accurately predict future measurements based on the past history of observed measurements. Hence, the signal processing technique should yield a clear monotonic trend with as little noise as possible. The technique used in (He and He 2018), which is based on a deep learning approach to performing Time Synchronous Resampling (TSR), is used to process measurements $\boldsymbol{z}_t$. The advantages of the approach are that it eliminates the challenge of performing Time Synchronous Averaging (TSA) by not requiring the number of data points in each revolution to stay constant, does not require the use of a tachometer, and is able to find the optimal signal segmentation via an adjustable filter.

The signal processing approach begins first by segmenting raw vibration signals into s different segments. The optimal parameter s is then obtained via a gradient based optimization routine using a convolutional neural network with variable filter size s.

The segmented vibration signals are then fed into an autoencoder which imitates the process of performing a Discrete Fourier Transform (DFT) and an Inverse Discrete Fourier Transform (IDFT). The DFT process of the approach encodes the frequency

components of the segmented vibration signals into the network's hidden layer. Spectral averaging is then performed and the IDFT process then decodes these averaged components. After the signal is processed the RMS of the signal is taken as the measurement value. The complete process of this approach is shown in Figure 3.12.

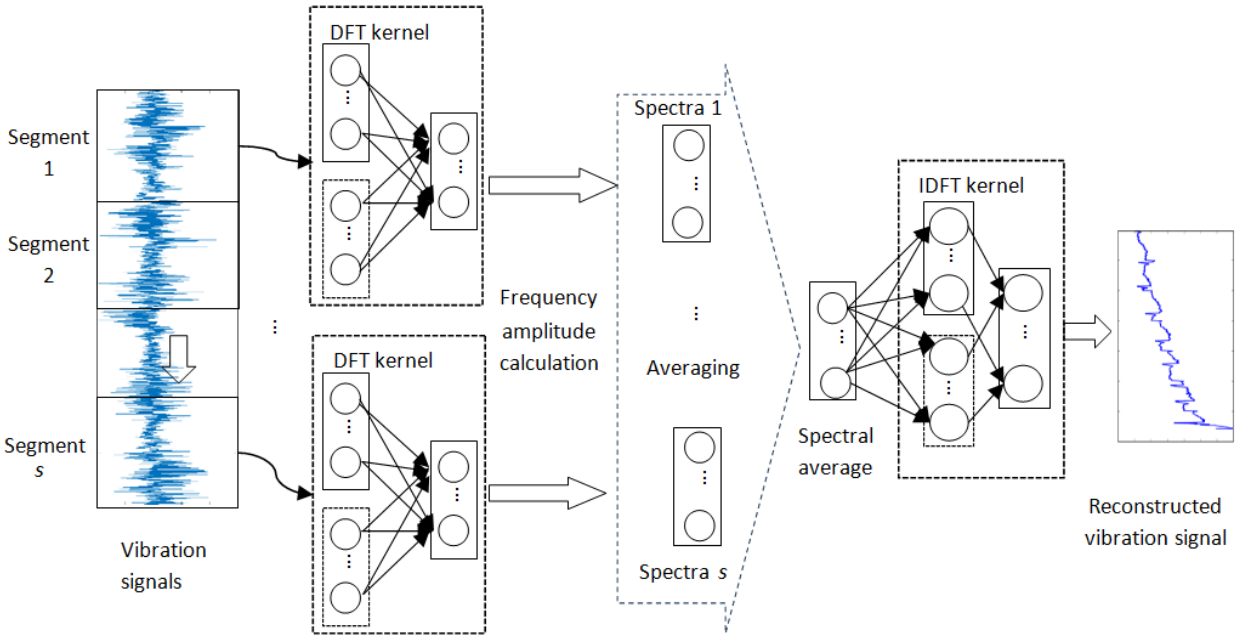

Figure 3.12. Prognostic feature extraction autoencoder

4. EXPERIMENTAL SETUP

This section is mainly comprised of materials published in (Deutsch and He 2016), (Deutsch et al. 2017), (He, et al. 2011)[3]*, and (Deutsch and He 2017).*

In this section, the validation of the deep learning based approaches for prongostics are validated using vibration data collected at the NASA Glenn Spiral Bevel Gear Test Facility and data collected from hybrid ceramic bearing run-to-failure tests are used.

4.1 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. 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.

[3] *He et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 United States License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited*

The bearing run to failure test rig used to collect the data is shown below in Figure 4.1.

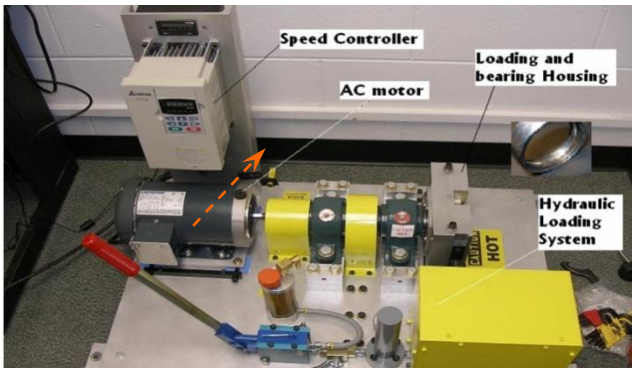

Figure 4.1. Bearing run-to-failure test rig

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 minute gap between any two sampling points. At the end of the test, the test bearings were disassembled, checked, and photographed. Two bearing run-to-fail tests were performed. The first bearing (B1) dataset recorded contained a total of 255 data files with a length of 21.25 hours and the second bearing (B2) dataset contained a total of 849 data files with a length of approximately 71 hours.

TABLE I describes the run-to-failure test setting. Table II provides the specifications of the tested bearing.

TABLE I. THE RUN-TO-FAILURE TEST SETTING

Name	**Type**	**Load (psi)**	**Input Shaft Speed (Hz)**
B1	Hybrid Ceramic Bearing	600	30
B2	Hybrid Ceramic Bearing	600	30

TABLE II. 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 (B1)	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 Deg C	121
Weight (g)	110.32

4.2 Spiral Bevel Gear Run-to-Failure Test Setup

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 total of eight

experiments were performed on the spiral bevel gears (36 teeth on the gears and 12 teeth on the pinions). The rig as shown in Figure 4.2 was used to determine the level of damage with respect to time. Vibration condition indictors (CIs) and oil debris mass (ODM) data were used during the tests to detect and quantify pitting damage on the gears.

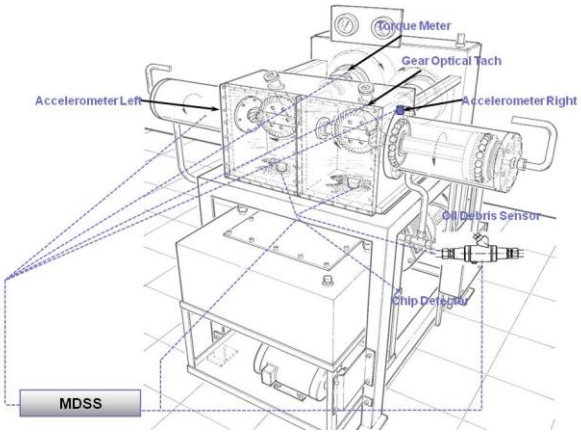

Figure 4.2. The bevel gear test rig

The tests consisted of placing the gears under load. Vibration data was collected once every minute with a sampling rate of 300 kHz for one second in length, generating TSA data on the shaft of the gear. The experiments were performed either until the occurrence of surface fatigue or until a specified number of hours elapsed. As show in Figure 4.3, destructive pitting could be observed on the teeth of the pinions.

Figure 4.3. Damaged spiral bevel gear

CIs obtained from vibration data are typically used to analyze and diagnose mechanical faults. For instance, the health and usage monitoring system (HUMS) that are currently established in helicopters use a considerable number of vibration based CIs. Unfortunately, there is not a single CI that is sensitive and able to capture every failure mechanism of a gear (Bechhoefer *et al*. 2011). Thus, the development of a one dimensional health index (HI) to quantify the gear's state of health can be very advantageous for gear health prognostics. The HI was processed using following approach developed in (He, *et al.* 2011):

"TSA data was processed with gear CI algorithms presented in (Zakrajsek and Townsend 1993) and (Wemhoff *et al*. 2007). A total of 6 CIs were used for computing a one-dimensional HI as the fault feature in order to predict the RUL: residual RMS, energy operator RMS, FM0, narrowband kurtosis, amplitude modulation kurtosis, and frequency modulation RMS. To compute the one-dimensional 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 $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{L}^*$, where $\boldsymbol{L}$ is a lower triangular, and $\boldsymbol{L}^*$ is its conjugate transpose. Let $\boldsymbol{F}$ be a set of correlated CIs. It then follows that:

$$\boldsymbol{L}\boldsymbol{L}^{*} = \Sigma^{-1} \tag{4.1}$$

and

$$\boldsymbol{Y} = \boldsymbol{L} \times \boldsymbol{F}^{T} \tag{4.2}$$

where $\boldsymbol{Y}$ is a vector of *n* independent CIs with unit variance and *correlation*($\boldsymbol{Y}$) = 0. Eq. (4.2) creates the necessary IID conditions required to define the health index 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 (Bechhoefer *et al*. 2011):

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

$$\boldsymbol{Y} = \boldsymbol{L} \times (\boldsymbol{F}^{T} - \boldsymbol{m}) \tag{4.3}$$

$$HI = \frac{(max\{\boldsymbol{Y}\} + .34) * .50}{(3.41 + 0.34)} \tag{4.4}$$

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

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

$$\boldsymbol{Y} = \boldsymbol{L}\, x\, \boldsymbol{F}^{T}$$

$$HI = \frac{0.5}{(8.352 - 0.15)(-0.15 + \sum_{i=1}^{n} \boldsymbol{Y}_i)} \tag{4.5}$$

When the HI is defined as the total energy of *n* Gaussian, it can be computed as following:

$$\boldsymbol{Y} = \boldsymbol{L}\, x\, \boldsymbol{F}^{T}$$

$$HI = \frac{0.5}{3.368\sqrt{\sum_{i=1}^{n} \boldsymbol{Y}_i{}^2}} \tag{4.6}$$

The HI used for RUL prediction was the order statistic defined by Equation (4.4) as the order statistic gave a more consistent trending of the HI than other statistics."

To predict the RUL of the gear, the health index 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 $X_{T'_{end}}$ and is denoted as T_{end}. This process is shown below in Figure 4.4 and Figure 4.5.

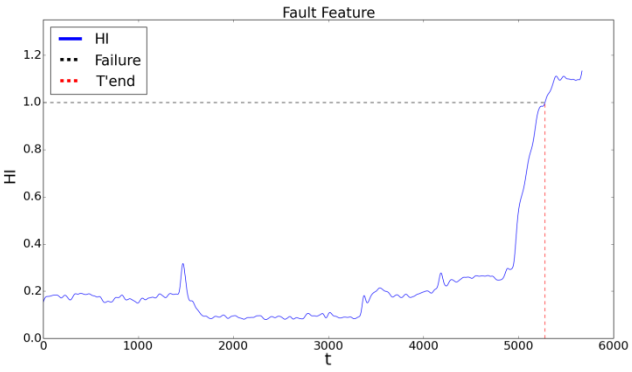

Figure 4.4. Determining T'_{end} from the fault feature

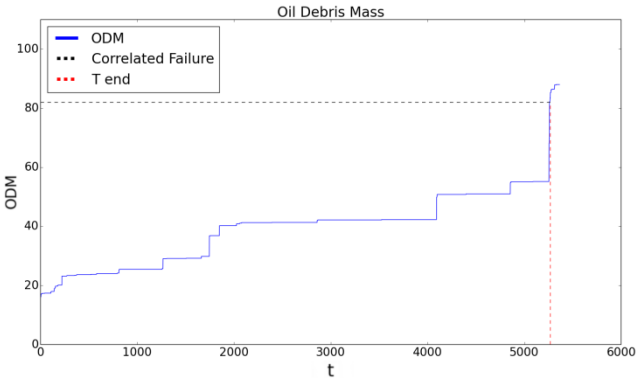

Figure 4.5. Determining T_{end} from the ODM

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

$$RUL_t = T_{end} - t\left(\frac{T_{end}}{T'_{end} - 1}\right) \tag{4.7}$$

Since the fault feature and the ODM are not a one to one mapping, Equation (4.7) allows one to map the RUL by the length of the fault feature.

5. RESULTS

This section is mainly comprised of materials published in (Deutsch and He 2016), (Deutsch et al. 2017), and (Deutsch and He 2017).

In this section each of the methodologies presented in Section 3, are used on the run-to-fail experiment data and the results are compared to a traditional based particle filter approach for modeling the RUL.

5.1 Validation Results Using a Restricted Boltzmann Machine

In this section the methodology presented using a RBM for prognostics (Section 3.3) are validated using both bearing run-to-fail test sets B1 and B2. A total of 804 RMS values were utilized at an interval t of 5 minutes based on 2 seconds of data collection at a sampling rate of 102.4 kHz for bearing B2 and a total of 192 RMS values were utilized for bearing B1. A RBM with a linear regression layer as the last layer of the network was developed to model the RMS values. Since the RBM assumes a binary input or a real valued input between [0, 1] the input values were scaled to be between [0,1]. An embedding dimension d = 50 and a training size of 250 examples was empirically found to yield good results in the model for bearing B2 and just 33 examples for bearing B1. The hyperparamters were found by using a grid search (exhaustive search). The root mean squared error (RMSE) was used as metrics to determine the most appropriate model. The mean absolute percentage error (MAPE) was also recorded for each model. The MAPE and RMSE are defined by the following equations:

$$MAPE = \left(\frac{1}{n_{pr}} \sum_{t=1}^{n_{pr}} \frac{A_t - F_t}{A_t} \right) * 100\% \tag{5.1}$$

$$RMSE = \frac{1}{n_{pr}} \sum_{t=1}^{n_{pr}} (A_t - F_t)^2 \tag{5.2}$$

In Equations (5.1) and (5.2), A_t = actual value, F_t = predicted value (from the model), and n_{pr} = the number of predicted points. Two step values of L = 1 and $L = 10$ were used to predict 5 minutes and 50 minutes respectively into the future for both bearings. Figure 5.1 and Figure 5.2 show the plots of the RBM's predicted RMS values vs. the actual RMS values for bearing B2 with L = 1 and L = 10, respectively.

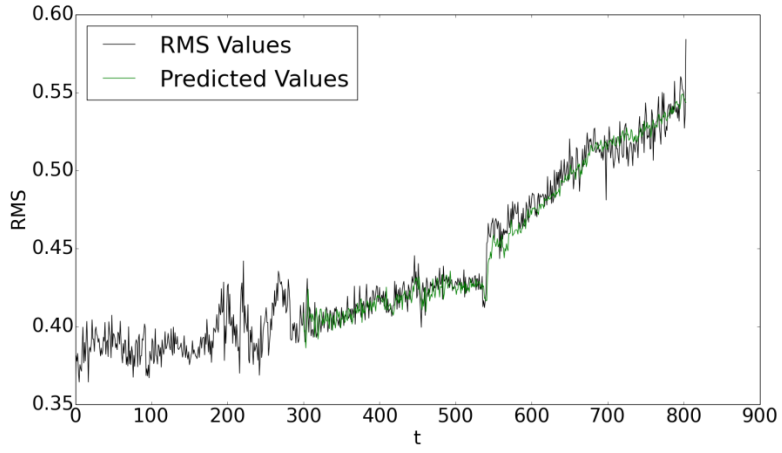

Figure 5.1. Plot of RBM predicted RMS values vs. actual RMS for bearing B2 with L = 1

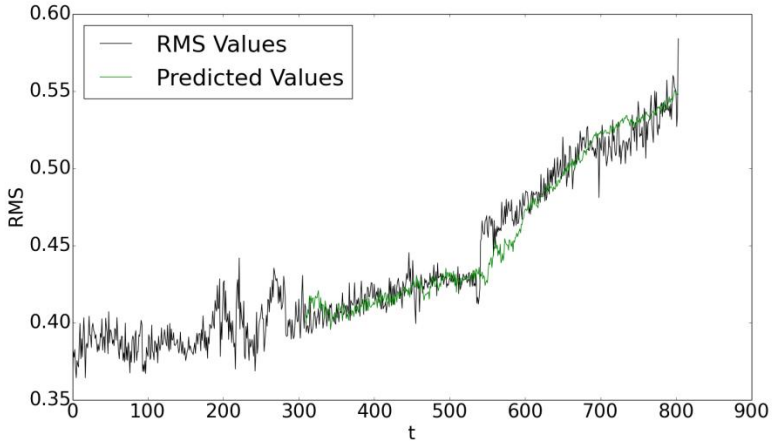

Figure 5.2. Plot of RBM predicted RMS values vs. actual RMS for bearing B2 with L = 10

Figure 5.3 and Figure 5.4 show the plots of the RBM's predicted RMS values vs. the actual RMS values for bearing B1 with L = 1 and L = 10, respectively.

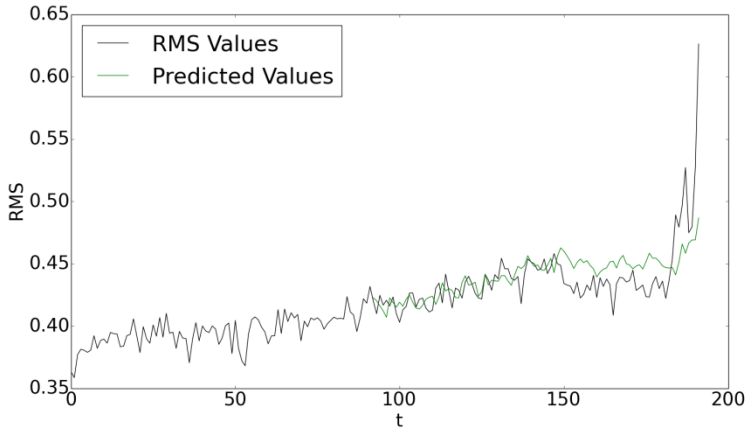

Figure 5.3. Plot of RBM predicted RMS values vs. actual RMS for bearing B1 with L = 1

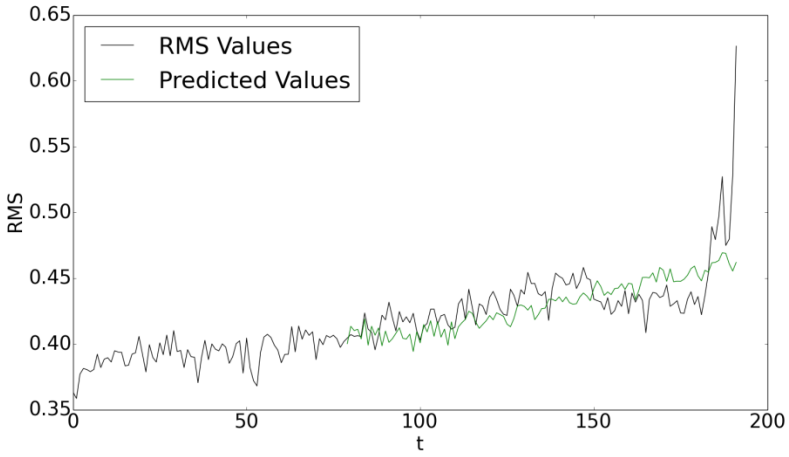

Figure 5.4. Plot of RBM predicted RMS values vs. actual RMS for bearing B1 with L = 10

From the above figures, it can be seen that the RBM model is able to capture much of the dynamics of the vibration data well throughout the predictions, stay within most of the noise in the data, and is able to capture the overall trend of the data.

To evaluate the RUL prediction performance of the RBM model, the last 100 testing points over a time period of 500 minutes (about 8 hours) were used to estimate the bearing RUL.

Figure 5.5 and Figure 5.6 show the plots of the estimated $\widehat{RUL_t}$ vs. the true RUL for bearing B2 with L = 1 and L = 10, respectively.

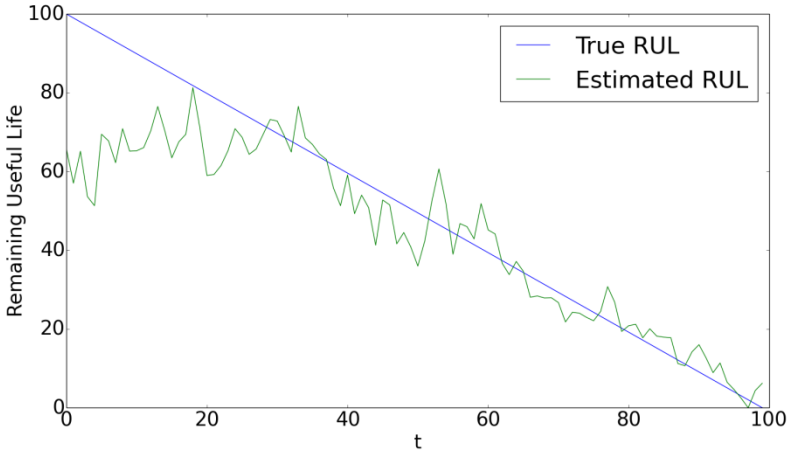

Figure 5.5. Plot of $\widehat{RUL_t}$ values of bearing B2 with L = 1

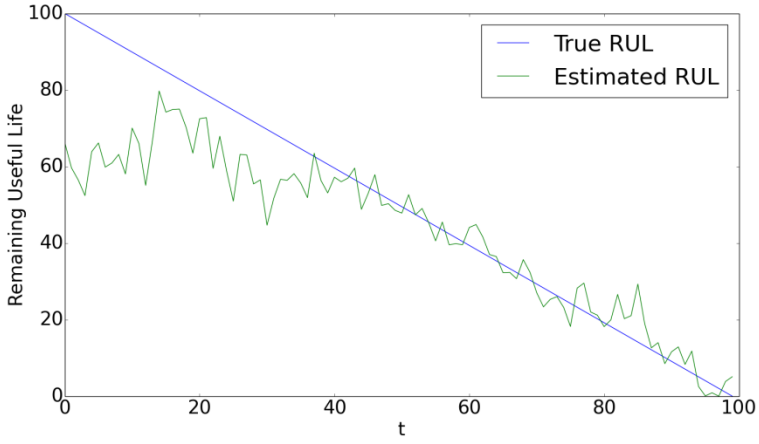

Figure 5.6. Plot of $\widehat{RUL_t}$ values of bearing B2 with L = 10

Figure 5.7 and Figure 5.8 show the plots of the estimated $\widehat{RUL_t}$ vs. the true RUL for bearing B1 with L = 1 and L = 10, respectively.

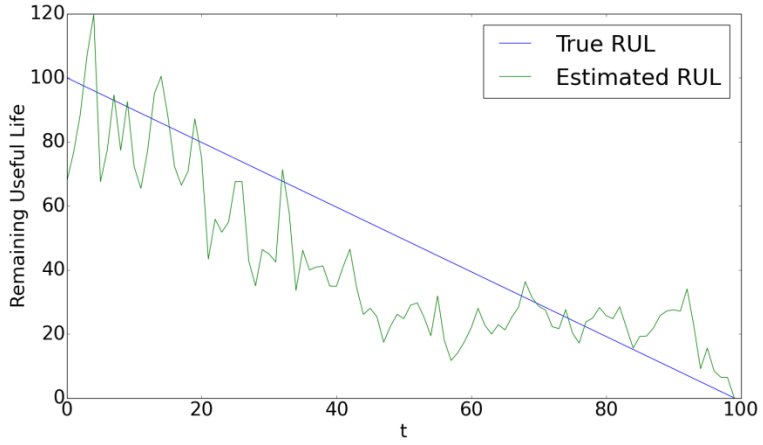

Figure 5.7. Plot of $\widehat{RUL_t}$ values of bearing B1 with L = 1

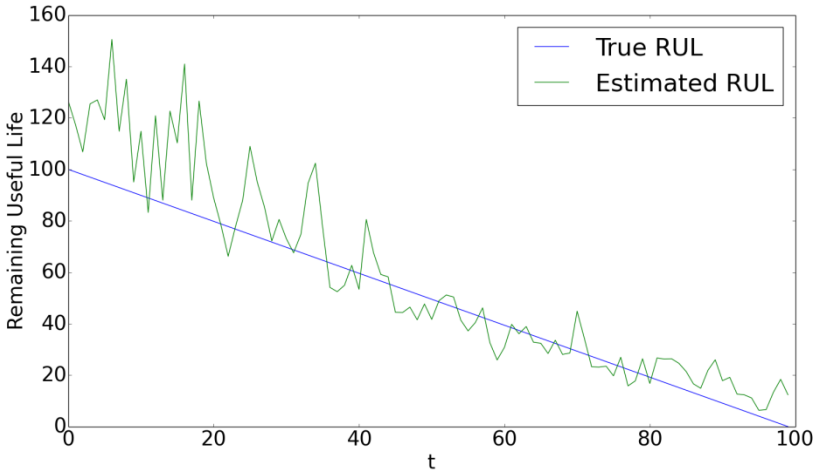

Figure 5.8. Plot of $\widehat{RUL_t}$ values of bearing B1 with L = 10

From the above figures, it can be seen that the RBM deep learning based approach gives good RUL predictions especially when it approaches the end of the bearing's life. The figures also show that as more data points were fed into the deep learning model, the more accurate the RUL predictions become. In addition, the RMSE and MAPE were computed for the RUL predictions obtained by the deep learning model and compared with those obtained by the particle filter based approach (Li *et al*. 2010). TABLE III AND TABLE IV show the RMSE and MAPE values of the predictions (comparing against the true RUL) obtained by the deep learning based and particle filter based approaches for bearing B2 and B1, respectively.

TABLE III. RMSE AND MAPE RESULTS OF BEARING B2

Deep learning based approach				
L	MAPE	RMSE	Learning rate	Hidden layer size
1	21.62%	12.85	0.120	50
10	23.24%	13.68	0.130	50
Particle filter based approach				
L	MAPE	RMSE		
1	7.47%	2.53		
10	8.73%	3.65		

TABLE IV. RMSE AND MAPE RESULTS OF BEARING B1

Deep learning based approach				
L	MAPE	RMSE	Learning rate	Hidden layer size
1	34.99%	15.86	0.001	11
10	43.65%	20.79	0.195	91
Particle filter based approach				
L	MAPE	RMSE		
1	10.56%	5.87		
10	12.42%	7.21		

From TABLE III and TABLE IV, it can be seen that the deep learning based approach achieved a lower level of accuracy as the particle filter based approach as evidenced by the MAPE and RMSE of the deep learning based approach being higher than those of the particle filter based approach. However, given that the deep learning based approach doesn't require 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 approach has shown its potential for bearing RUL prediction with big data.

5.2 Validation Results Using a Deep Belief Network

In this section the methodology presented using a DBN for prognostics (Section 3.5) are validated using both bearing run-to-fail test B2 and the spiral bevel gear run-to-fail data.

5.2.1 Spiral Bevel Gear Remaining Useful Life Prediction Results

A total of 5670 time steps were extracted from experiment 5 of the gear data, in which all the data up until T'_{end} = 5280 was used. The last 100 points of the gear's life was set 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. (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 Figure 5.9 and Figure 5.10, respectively. The error metrics and hyperparamters for predictions with L = 1 and L = 10 are provided in TABLE V and VI, respectively. The L step ahead predictions for $L = 1$ predict approximately 1 minute into the future and 10 minutes into the future for $L = 10$.

In Figure 5.9 and Figure 5.10, 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.

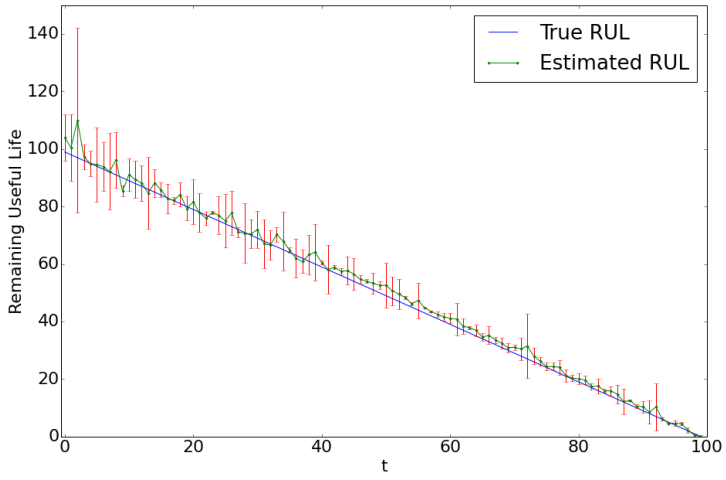

Figure 5.9. Plot of gear RUL values with $L = 1$

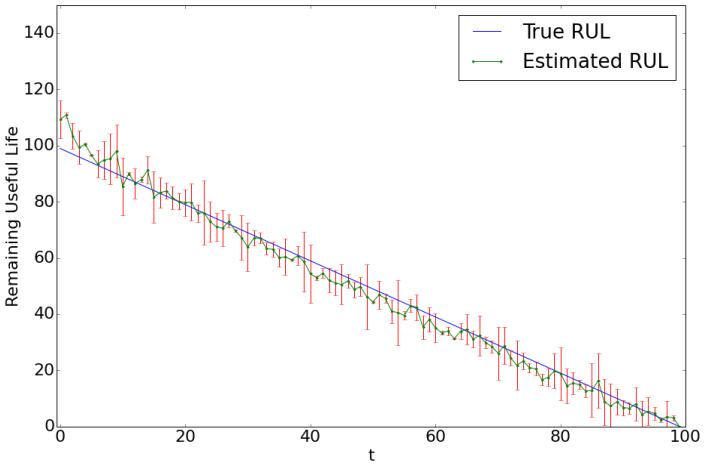

Figure 5.10. Plot of gear RUL values with L = 10

TABLE V. 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 VI. DBN-FNN HYPERPARAMETERS FOR GEAR DATA

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

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. The hyperparameters chosen were simply those that minimized the RMSE on the testing set.

The best results were found by using the MAPE as a loss function instead of the more typical MSE loss function used for regression problems. The Adam optimizer (Kingma and Ba 2014), 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 the 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 non-linear activation functions can cause (Glorot *et al*. 2011). The RELU activation function for the input x of a neuron is defined as:

$$RELU(x) = max\{0, x\} \tag{5.3}$$

For L = 10 on the gear data, the best results were found using a Leaky RELU (LRELU) (Mass *et al*. 2013) defined as:

$$LRELU(x) = \begin{cases} x, & x > 0 \\ x\alpha, & x \leq 0 \end{cases} \tag{5.4}$$

where α is some value which allows for a small non-zero gradient when the neuron is saturated and not active. The best results were found 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 V. 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, whereas 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.

5.2.2 Hybrid Ceramic Bearing Remaining Useful Life Prediction Results

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 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 plot for the bearing data (B2) can be seen in Figure 5.11.

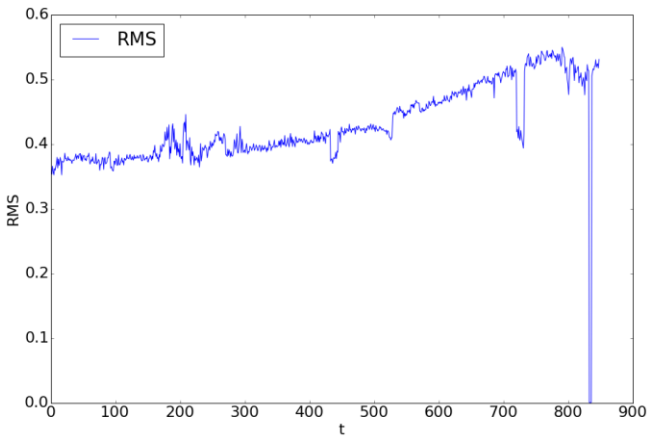

Figure 5.11. The bearing RMS values

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

$$DFT_{tn} = \sum_{k=0}^{N-1} f_{tk} e^{\frac{-2\pi ikn}{N}} \tag{5.5}$$

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 (5.5) transforms the vibration signals from a time domain to a frequency domain in which eight equal bands were extracted 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 $RMS_{T_{end}}$ and subtracting it from each time step:

$$RUL_t = RMS_{T_{end}} - t \tag{5.6}$$

As seen above from Figure 5.11, 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.

The predicted RUL values for the last 100 steps can be seen in Figures 5.12 and 5.13 for L = 1 and L = 10, respectively. The error metrics and hyperparameters for prediction with L = 1 and L = 10 are provided in TABLE VII and Table VIII, 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 Figure 5.12 and Figure 5.13, 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 of failure, the accuracy of the predictions tends to increase and converge.

Similar to the L = 10 prediction for the gear data, a LRELU was used for both the hidden layers in the bearing data, with α = 0.0085 and α = 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.

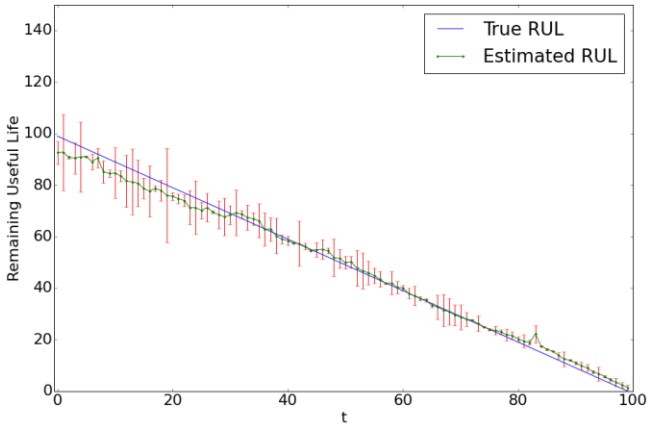

Figure 5.12. Plot of bearing RUL values with $L = 1$

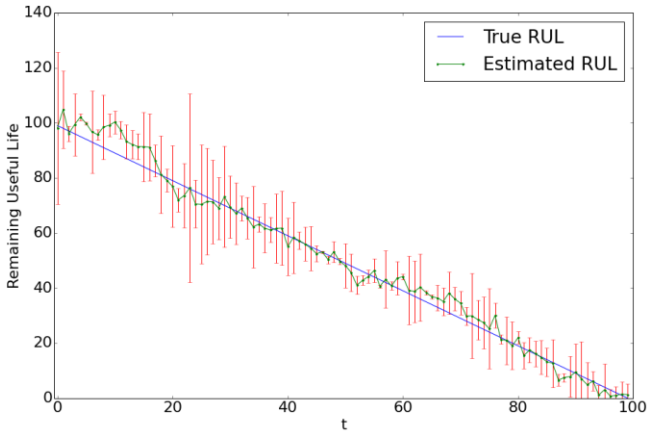

Figure 5.13. Plot of bearing RUL values with $L = 10$

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 VII. 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.

TABLE VII. 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 VIII. DBN-FNN HYPERPARAMETERS FOR BEARING DATA

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

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 comparable 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.

5.3 Validation Results Using a Combined Deep Belief Network and Particle Filter

The RMS of the vibration signals was computed to represent the degradation of the bearing (B2) over time during the run to failure tests. The same data preprocessing steps used in the previous Section (5.2.2) were used. The same FFT computation was also taken to represent the measurements.

The predicted RUL values for the last 100 steps can be seen in Figures 5.14 and 5.15 for L = 1 and L = 10, respectively. The error metrics and hyperparameters of the DBN with L = 1 and L = 10 are provided in TABLE IX and TABLE X, respectively. TABLE X 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.

In Figure 5.14 and Figure 5.15, 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.

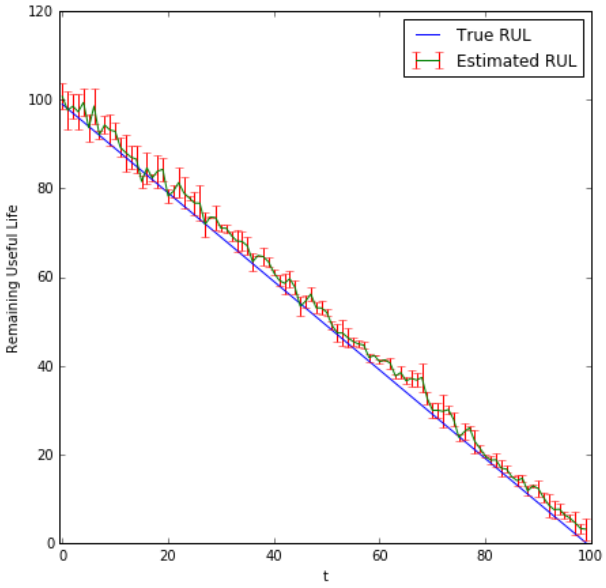

Figure 5.14. Plot of bearing RUL values with $L = 1$

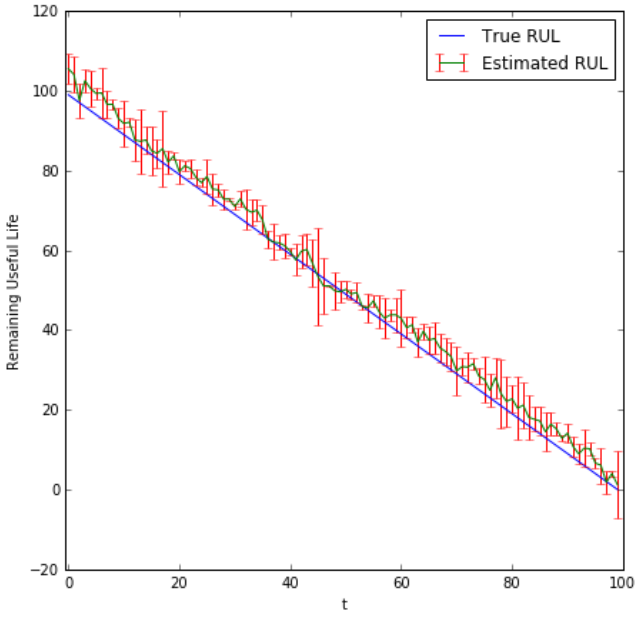

Figure 5.15. Plot of bearing RUL values with L = 10

TABLE IX. RMSE AND 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 X. HYPERPARAMETERS OF THE 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

The error metrics used in TABLE IX are the RMSE, $\alpha-\lambda$ metric (Al-Dahidi *et al.* 2016, Saxena *et al*. 2010), MAPE. In TABLE IX, the metrics were computed based on the average value of the predicted RUL. The $\alpha-\lambda$ metric is defined by the following equation:

$$(\alpha-\lambda)_{t_\lambda} = \begin{cases} 1 & if\ (1-\alpha)RUL_{t_\lambda} \leq \widehat{RUL}_{t_\lambda} \leq (1+\alpha)RUL_{t_\lambda} \\ 0 & otherwise \end{cases} \quad (5.7)$$

In Equation (5.7), 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 IX represents the average $\alpha-\lambda$ metric across the all testing samples (values 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 Equation (3.30) and Equation (3.35) 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 (Bergstra *et al.* 2012*).* 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 RELU function.

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 IX. 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. It can be considered that the integrated method presented here is better than the DBN-based approach used on the same bearing 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.

5.4 Validation Results Using an Integrated Mixture Density Network and Particle Filter

Experiments three, four, six, and all but the last 100 time steps of experiment seven of the gear run-to-failure data were utilized to serve as the training set. The last 100 time steps of experiment seven were used as the testing set. Hyperparamters for both the state transition and the measurement model were found by using a grid search and validating these using a ten fold cross validation. The same training data was utilized to determine these optimal hyperparameters with the exception that only the first 300 time steps were utilized of experiment 7.

These hyperparameters can be viewed in TABLE XI. The adagrad optimizer (Duchi *et al.* 2013) was used to obtain the optimal parameters for both the state transition distribution and measurement distribution. The total number of particles N_p = 75 were used for all predictions. The ODM was treated as the value of the state.

TABLE XI. HYPERPARAMETERS OF THE MDN

	MDN learning rate	MDN epochs	Hidden layer structure	Activation function	Mixing components (K)	MDN probability density function
State transition	1e-3	949	[21]	Tanh	3	Log-normal
Measurement model	1e-3	37	[299]	Tanh	3	Normal

The learned state transition distribution model from the MDN can be seen below in Figure 5.16 and the MDN measurement model distribution can be seen in Figure 5.17. As can be seen, the actual one step ahead state value lies for the most part in high dense regions of the state transition distribution model. The red line in these plots indicates the true state $\boldsymbol{x}_t$.

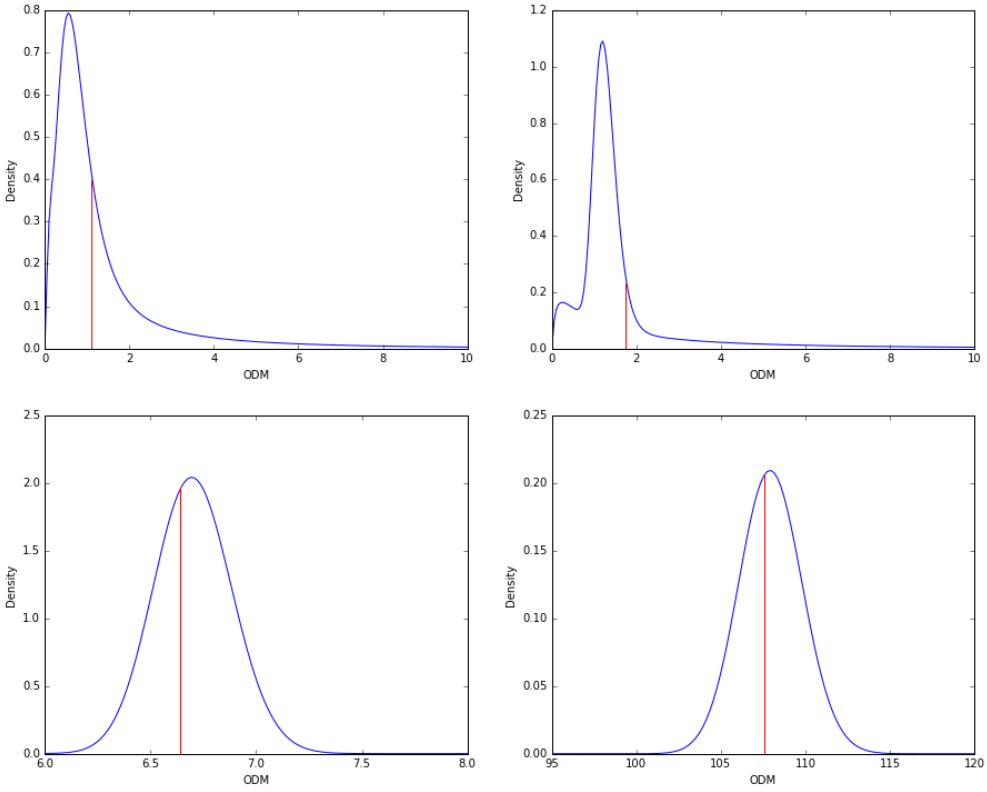

Figure 5.16. PDF plots of the state transition model for time steps $t = 1$ (upper left), $t = 101$ (bottom left), $t = 11$ (upper right), $t = 1001$ (bottom right) for experiment 7

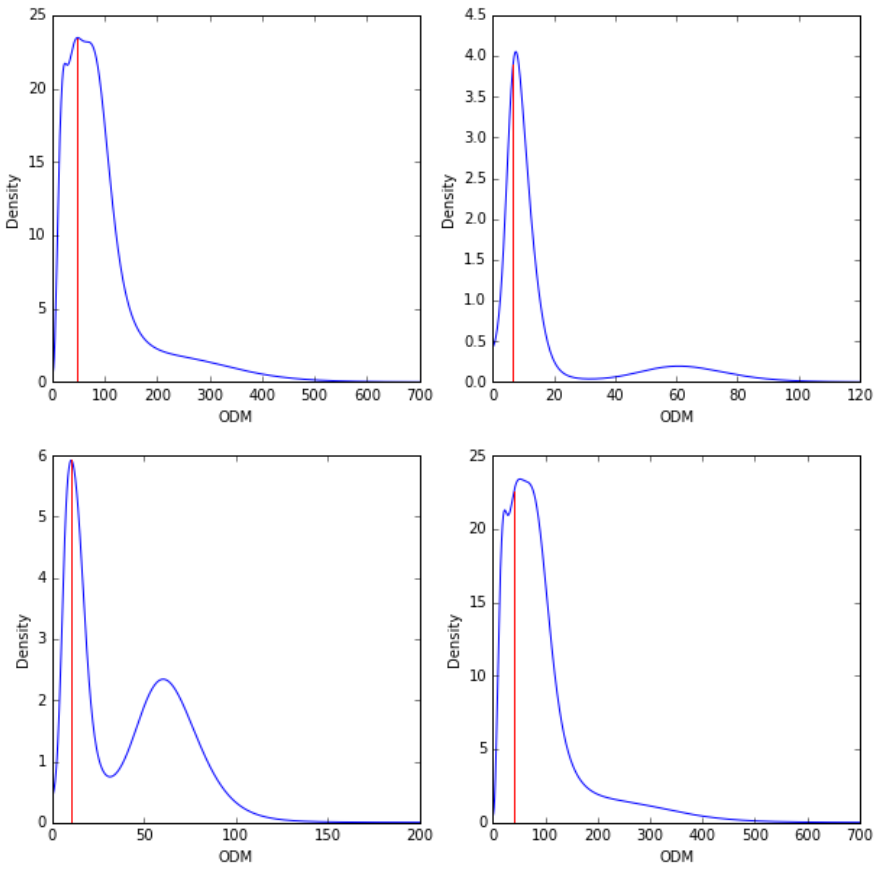

Figure 5.17. PDF plots of the measurement model for time steps t = 957 (upper left), t = 451 (bottom left), t = 101 (upper right), t = 951 (bottom right) for experiment 7.

The learned measurement distribution model also appears to yield decent results, with the true state lying in areas of high density. As evidenced by Figure 5.17, the nondeterministic nature of the system is directly visible and apparent by the bimodal shape of some of the distributions; a similar observed measurement may be associated with multiple values of the ODM.

By recursively using the state transition distribution, particles are propagated to model the system state over time. Figure 5.18, provides an example of particles being

propagated for 50 time steps into the future in order to track the ODM using the learned state transition distribution from the MDN.

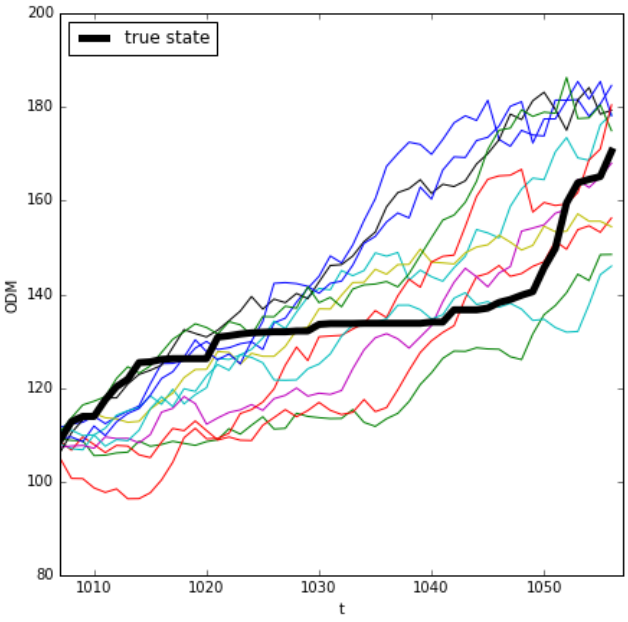

Figure 5.18. Particle propagation 50 time steps ahead using 11 particles

Measurement data $\boldsymbol{z}_t$ was processed into a mono-dimensional feature using the aforementioned hybrid deep signal processing technique described in Section 3.11. By utilizing this approach, a very clear monotonically increasing trend and be seen in Figure 5.19, which closely matches the true state of the system (Figure 5.20). A comparison was also performed on the same measurement data using a DBN. However, as can be seen in Figure 5.21, the processed measurements performed via the DBN do not show any noticeable trend and very poorly correlate with the true state of the system. The output of

the DBN only shows some slightly higher energies near the gear's end of life and utilizing the features extracted from the DBN would provide very poor prognostic results.

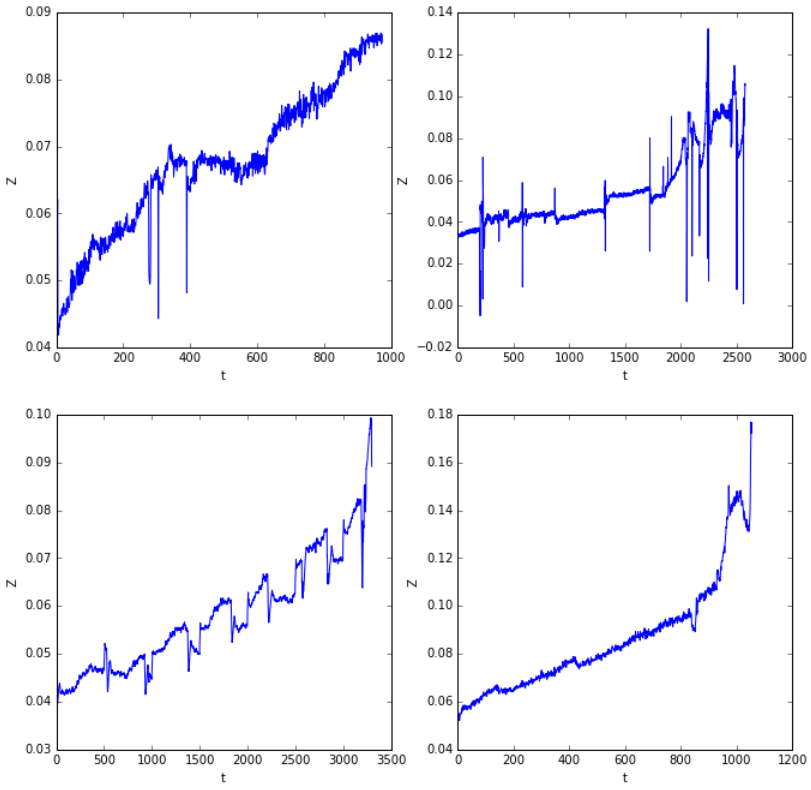

Figure 5.19. Plots of the measurement value $\boldsymbol{z_t}$ over time for experiment 3 (upper left), experiment 6 (bottom left), experiment 4 (upper right), and experiment 7 (bottom right)

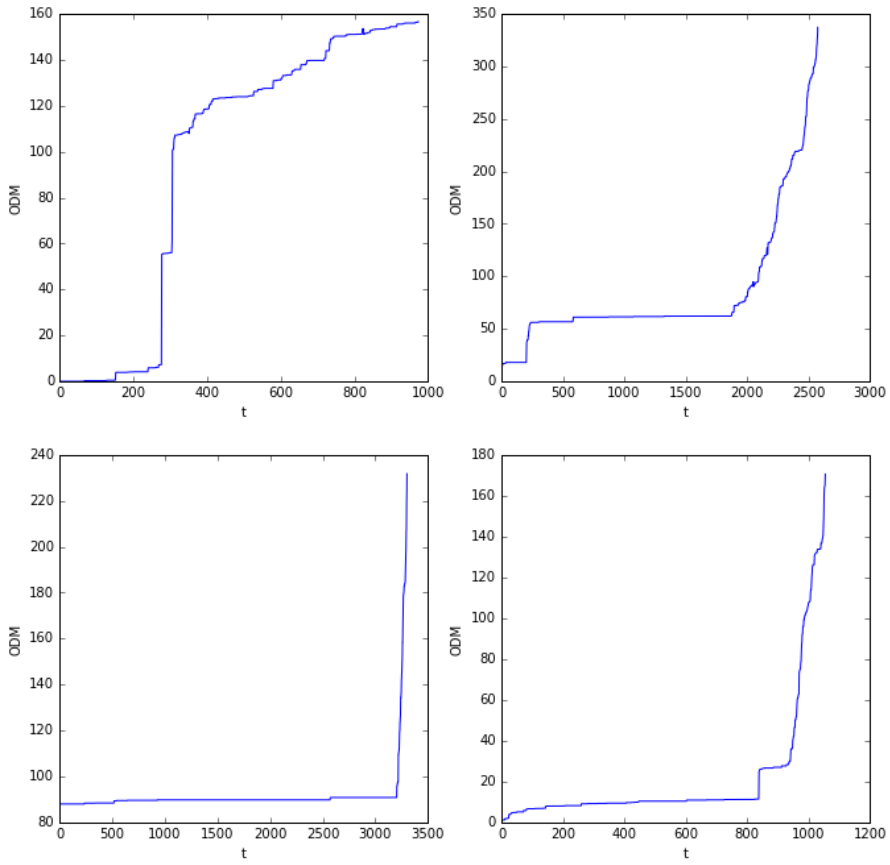

Figure 5.20. Plots of the state value over time for experiment 3 (upper left), experiment 6 (bottom left), experiment 4 (upper right), and experiment 7 (bottom right)

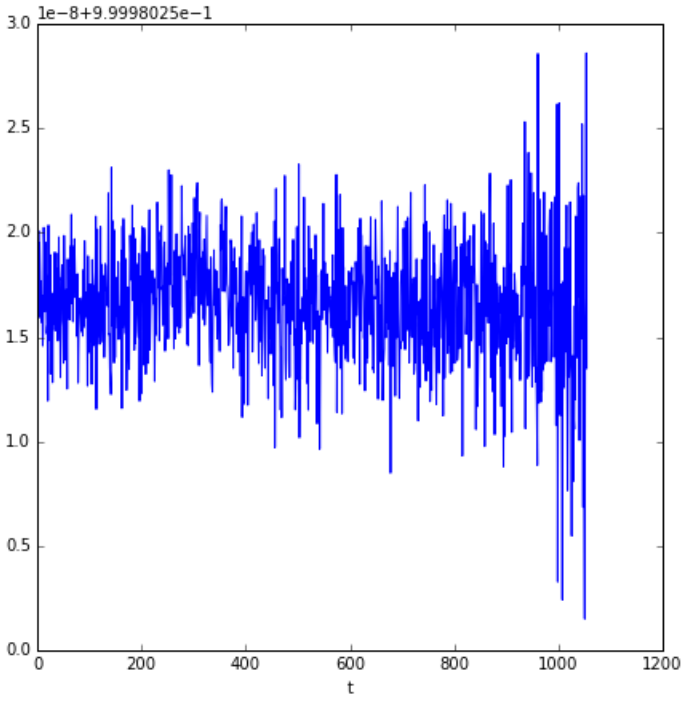

Figure 5.21. DBN output for experiment 7

TABLE XII. HYPERPARAMETERS OF THE DBN FOR EXPERIMENT 7

DBN learning rate	DBN epochs	Hidden layer structure	Activation function
1e-3	300	[500,100,50,1]	Sigmoid

The measurement data was then processed further by smoothing out the data using a simple polynomial curve fit. Likewise, the function $g(.)$ was built up using a polynomial curve fit on the training measurement data in order to predict the future values of the measurements. Other more elaborate functions may of course be used, however, for the

sake of simplicity a polynomial of degree four provided satisfactory results. The number of steps N_s to project the particle filter into the future was set to a short forecast horizon of six steps.

Finally, as can be seen in Figure 5.22 and 5.23, the predicted RUL results follow very closely with the actual RUL, with nearly all of the true RUL points lying within the confidence bounds. In Figures 5.22-25 the green line represents $\overline{RUL_t}$ and the red lines represents the confidence bounds.

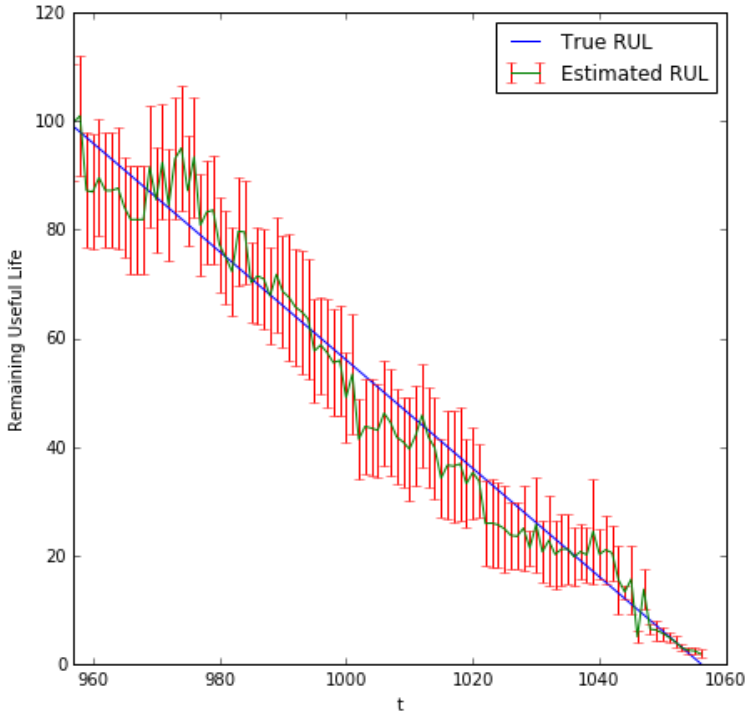

Figure 5.22. L = 1, Plot of the RUL with 99% confidence bounds

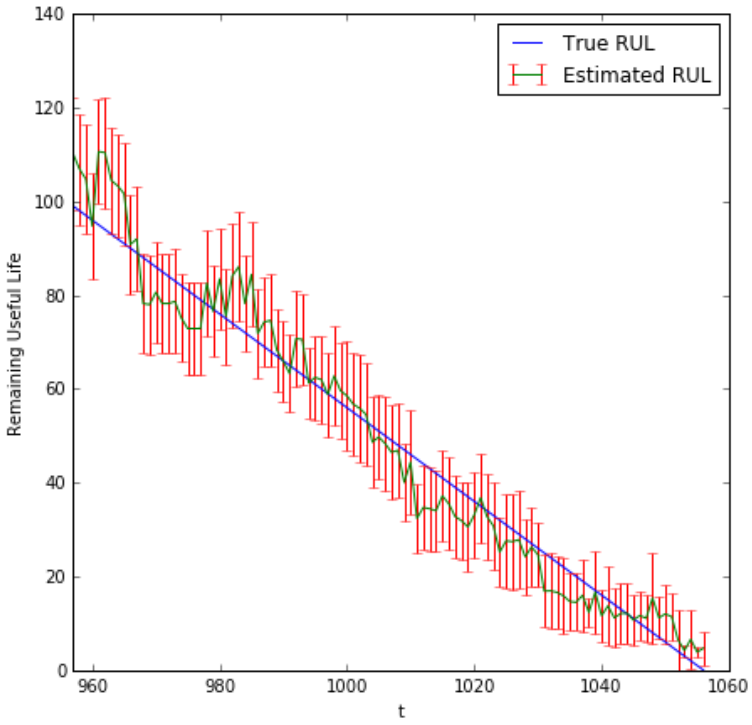

Figure 5.23. L = 10, Plot of the RUL with 99% confidence bounds

Results obtained by using the traditional particle filter based approach (without using future measurements) and using the same learned distributions via the MDN also yielded good results and are shown below in figures 5.24 and 5.25. However, towards the end of the gear's life, the accuracy of the RUL predictions decrease slightly and are a bit optimistic.

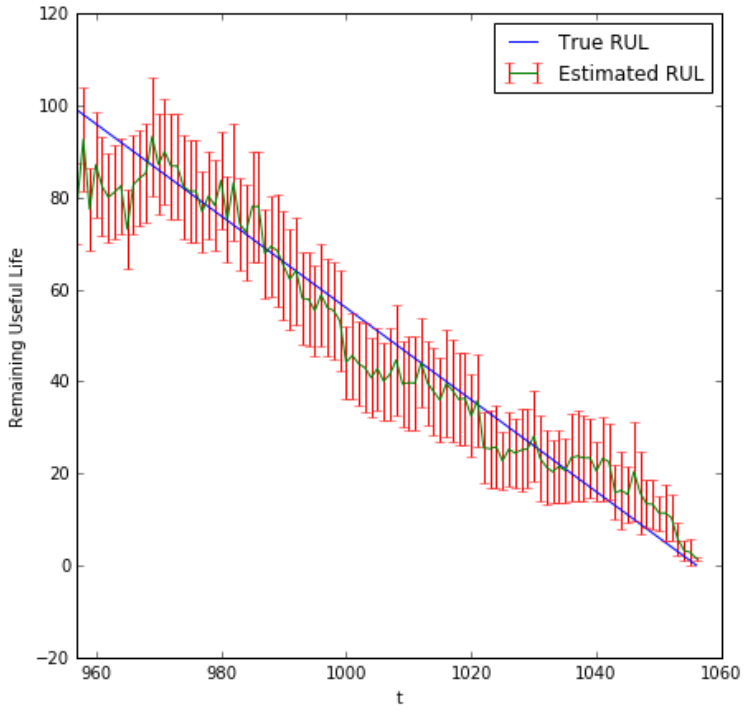

Figure 5.24. L = 1, Plot of the RUL with 99% confidence bounds without using future measurements

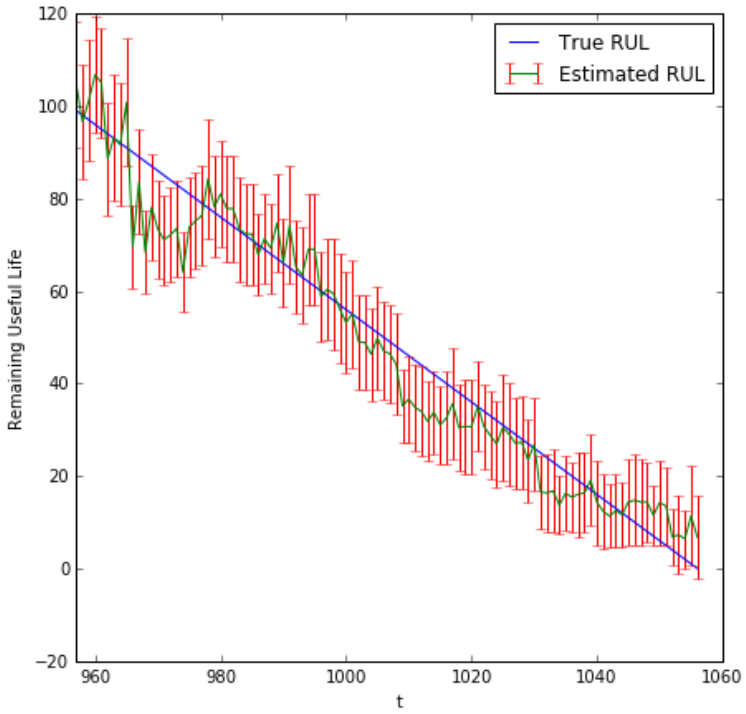

Figure 5.25. L = 10, Plot of the RUL with 99% confidence bounds without using future measurements

These results were compared to the methodology presented in Section 3.7. The comparison of the results can be found below in Table XIII.

TABLE XIII. RESULTS OF THE MDN APPROACH

Method	*L*	RMSE	MAPE
MDN (w/future measurements)	1	5.10	12.14%
MDN (w/future measurements)	10	6.10	19.79%
MDN (w/o future measurements)	1	6.58	21.29%
MDN (w/o future measurements)	10	6.73	30.28%
Integrated Deep Learning and Particle Filter Approach	1	6.76	23.36%
Integrated Deep Learning and Particle Filter Approach	10	15.56	32.50%

In TABLE XIII, the metrics were computed based on the average value of the predicted RUL. Both MDN approaches provided the best overall accuracy. The MDN using future measurements yielded the best overall results across both *L* step ahead predictions. It can be hypothesized that this is due to the fact that the future measurements can effectively decrease the projection horizon of the particles when propagating these particles until they meet or exceed the system's safety threshold.

6. CONCLUSION

Deep learning is a very powerful tool and thus, has become a very popular tool for solving many tasks across a wide spectrum of domains. Prognostics are no exception to this. Accurate forecasts can be obtained to determine the RUL without requiring much of the skill, expertise, and domain knowledge used for signal processing with deep learning based approaches. Specific models such as the state transition distribution and measurement model can be automatically built up using deep learning approaches that could be deployed where models may be unavailable or very complex to build. However, this is not to say that domain knowledge is of little use when employing deep learning based approaches for prognostics. The black-box approach to utilizing deep learning for prognostics did not seem to outperform the approaches in which at least some a prior knowledge was incorporated into the modeling procedure. This is certainly illustrated by the results from Section 5.4, which yielded the best overall prognostic forecasts and utilized the most of amount of domain knowledge into the prognostic forecasts. Therefore, it seems logical that further research should be developed based on hybrid models that incorporate the physics behind the system into the deep learning methodologies.

CITED LITERATURE

Al-Dahidi, S., Di Maio, F., Baraldi, P. and Zio, E. (2016). "Remaining useful life estimation in heterogeneous fleets working under variable operating conditions", *Reliability Engineering & System Safety*, vol. 156, pp.109-124.

Arulampalam, M.S., Makell, S., Gordon, N., Clapp, T. (2002). "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking", *IEEE Transactions on Signal Process*, vol. 50, no. 2, pp. 174-188.

Baraldi, P., Compare, M., Sauco, S., and Zio, E. (2013a). "Ensemble neural network-based particle filtering for prognostics", *Mechanical Systems and Signal Processing*, vol. 41, no. 1, pp. 288–300.

Baraldi, P., Mangili, F., and Zio, E. (2012). "A kalman filter-based ensemble approach with application to turbine creep prognostics", *IEEE Transactions Reliability*, vol. 61, pp. 966 – 977.

Baraldi, P., Mangili, F., and Zio, E. (2013b). "Investigation of uncertainty treatment capability of model-based and data-driven prognostic methods using simulated data", Reliability Engineering & System Safety, vol. 112, pp. 94-108.

Bechhoefer, E., Clark, S., and He, D. (2010). "A state space model for vibration based prognostics", *Proceedings of the 2010 Annual Conference of the Prognostics and Health Management Society*, Portland, OR, October 10 – 16.

Bechhoefer, E., He, D., and Dempsey, P. (2011). "Gear health threshold setting based on a probability of false alarm", *Proceedings of the 2011 Annual Conference of the Prognostics and Health Management Society*, Montreal, Quebec, Canada, September 25 – 29.

Bengio, Y., Courville, A., and Vincent, P. (2013). "Representation learning: a review and new perspectives", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798-1828.

Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. (2007). "Greedy layer-wise training of deep networks", *Advances in neural information processing systems*, vol. 19, pp. 153, MIT Press: Cambridge, MA.

Bergstra, J. and Bengio, Y. (2012). "Random search for hyper-parameter optimization", *Journal of Machine Learning Research*, vol. 13, pp.281-305.

Bishop, C.M., (1994). "Mixture density networks". Technical report.

Bououden, S. Chadli, M. Allouani, F. Filali, S. (2013). "A new approach for fuzzy predictive adaptive controller design using particle swarm optimization

algorithm", *International Journal of Innovative Computing, Information and Control*, vol. 9, no. 9, pp. 3741–3758.

Chen, C., Zhang, B., Vachtsevanos, G., and Orchard, M. (2011). "Machine condition prediction based on adaptive neuro–fuzzy and high-order particle filtering", *IEEE Transactions on Industrial Electronics*, vol. 58, no. 9, pp. 4353–4364.

Chen, X., Yu, J., Tang, D., and Wang, Y. (2012). "A novel pf-lssvr-based framework for failure prognosis of nonlinear systems with time-varying parameters", *Chinese Journal of Aeronautics*, vol. 25, pp.715-724.

Chen, Z., Zeng, X., Li, W., and Liao, G. (2016). "Machine fault classification using deep belief network", *2016 IEEE International Instrumentation and Measurement Technology Conference Proceedings*, Taipei, Taiwan, pp. 1-6.

Codetta-Raiteri, D. and Portinale, L. (2015). "Dynamic Bayesian networks for fault detection, identification, and recovery in autonomous spacecraft", *IEEE Transactions on Systems, Man, Cybernetics: Systems*, vol. 45, no. 1, pp. 13 – 23.

Cubillo, A., Perinpanayagam, S. and Esperon-Miguez, M. (2016). "A review of physics-based models in prognostics: Application to gears and bearings of rotating machinery", *Advances in Mechanical Engineering*, vol. 8, no. 8, DOI: 10.1177/1687814016664660.

Daigle, M. J. and Goebel, K. (2013). "Model-based prognostics with concurrent damage progression processes", *IEEE Transactions on Systems, Man, Cybernetics*, vol. 43, no. 3, pp. 535–546.

Daroogheh, N., Baniamerian, A., Meskin, N., and Khorasani, K. (2016). "Prognosis and health monitoring of nonlinear systems using a hybrid scheme through integration of PFs and neural networks", *IEEE Transactions on Systems, Man, Cybernetics: Systems*, DOI: 10.1109/TSMC.2016.2597272.

Deutsch, J. (2017). "Development of deep learning based prognostics for rotating components", *Prognostics and Health Management Annual Doctoral Symposium*, St. Petersburg, FL, Oct. 2 – 5.

Deutsch, J. and He, D. (2016). "Using Deep Learning Based Approaches for Bearing Remaining Useful Life Prediction", *Annual Conference of the Prognostics and Health Management Society 2016*, Denver, CO, Oct. 2 – 8.

Deutsch, J. and He, D. (2017). "Using Deep Learning Based Approach To Predict Remaining Useful Life of Rotating Components", *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 48, no. 1, pp. 11-20.

Components, IEEE Transactions on Systems, Man, and Cybernetics: Systems, and May 2017])

Deutsch, J., He, M., and He, D. (2017). “Remaining Useful Life Prediction of Hybrid Ceramic Bearings using an Integrated Deep Learning and Particle Filter Approach”, *Applied Sciences*, vol. 7, no. 7.

Douc, R. and Cappé, O. (2005). “Comparison of resampling schemes for particle filtering”, Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis, Zagreb, Croatia, Croatia, September 15-17, pp. 64-69.

Duchi, J., Hazan, E., and Singer, Y. (2011). “Adaptive subgradient methods for online learning and stochastic optimization”, *Journal of Machine Learning Research*, vol. 12, pp. 2121-2159.

Efron, B. and Gong, G. (1983). "A leisurely look at the bootstrap, the jackknife, and cross-validation" *The American Statistician*, vol. 37, no. 1, pp. 36-48.

Elattar, H.M., Elminir, H.K. and Riad, A.M. (2016). “Prognostics: a literature review”, Complex & Intelligent Systems, vol. 2, no. 2, pp.125-154.

Erhan, D., Bengio, Y., Courville, A., Manzagol, P., Vincent, P., and Bengio, S. (2010). “Why does unsupervised pre-training help deep learning?”, *J. Mach. Learn. Res.* 11, pp.625-660.

Frank, R. J., Davey, N., and Hunt, S. P. (2001). “Time series prediction and neural networks”, *Journal of Intelligent Robotics Systems*, vol.31, no. 1-3, pp. 91-103.

Glorot, X., Bordes, A., and Bengio, Y. (2011) “Deep sparse rectifier networks”, *AISTATS*, pp. 315–323.

Gordon, N., Salmond, D., Smith, A. (1993). “Novel approach to nonlinear/non-Gaussian Bayesian state estimation”, IEE *Proceedings* F Radar Signal Processing, vol. 140, no. 2, pp. 107–113.

He, D., Bechhoefer, E., Ma, J., and Li, R. (2011). “Particle filtering based gear prognostics using one-dimensional health index”, *Proceedings of the 2011 Annual Conference of the Prognostics and Health Management Society*, Montreal, Quebec, Canada, September 25 - 29.

He, M. and He, D., (2018). “A new hybrid deep signal processing approach for bearing fault diagnosis using vibration signals”, Working Paper#9, Department of Mechanical and Industrial Engineering, The University of Illinois at Chicago, Chicago, IL.

Heimes, F. (2008). “Recurrent neural networks for remaining useful life estimation”, *Proceedings of the 2008 IEEE International Conference of Prognostics and Health Management*, Denver, CO, pp. 1–6.

Hinton, G. E, Osindero, S., and Teh, Y.-W. (2006). "A fast learning algorithm for deep belief nets", *Neural Computation,* vol. 18, no. 7, pp.1527-1554.

Hinton, G. E. (2002). "Training products of experts by minimizing contrastive divergence", *Neural Computation,* vo. 14, no. 8, pp. 1711–1800.

Hornik, K., Stinchcombe, M. and White, H. (1989). "Multilayer feedforward networks are universal approximators", *Neural networks,* vol.*2,* no. 5, pp.359-366.

Hossain, M., Rekabdar, B., Louis, S. J., and Dascalu, S. (2015). "Forecasting the weather of Nevada: a deep learning approach", *International Joint Conference on Neural Networks (IJCNN),* Killarney, Ireland, pp. 1-6.

Huynh, K. T., Castro, I. T., Barros, A., and Be´renguer, C. (2014). "On the use of mean residual life as a condition index for condition-based maintenance decision-making", *IEEE Transactions on Systems, Man, Cybernetics: Systems,* vol. 44, no. 7, pp. 877 – 893.

Jardine, A.K., Lin, D. and Banjevic, D. (2006). "A review on machinery diagnostics and prognostics implementing condition-based maintenance", *Mechanical systems and signal processing,* vol. *20,* no. 7, pp.1483-1510.

Joerding, W.H., and Meador, J.L. (1991). "Encoding a priori information in feedforward networks", Neural Networks, vol. 4, no. 6, pp. 847-856.

Khosravi, A., Nahavandi, S., Creighton, D. and Atiya, A.F. (2011). "Comprehensive review of neural network-based prediction intervals and new advances", *IEEE Transactions on neural networks,* vol. *22,* no. 9, pp.1341-1356.

Kingma, D. and Ba, J. (2014). "Adam: a method for stochastic optimization", *Proceedings of the 3rd International Conference for Learning Representations,* San Diego, CA, USA, pp. 1-15.

Kinoshita, K., Delcroix, M., Ogawa, A., Higuchi, T. and Nakatani, T. (2017). "Deep mixture density network for statistical model-based feature enhancement", *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference,* New Orleans, LA, USA, March 5-9, pp. 251-255.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). "Deep learning", *Nature,* vol. 521, no. 7553, pp. 436-444.

Lee, J., Wu, F., Zhao, W., Ghaffari, M., Liao, L. and Siegel, D. (2014). "Prognostics and health management design for rotary machinery systems—Reviews, methodology and applications", *Mechanical systems and signal processing,* vol. *42,* no. 1, pp.314-334.

Lei, Y. and Zuo, M.J. (2009). "Gear crack level identification based on weighted K nearest neighbor classification algorithm", *Mechanical Systems and Signal Processing,* vol. *23,* no. 5, pp.1535-1547.

Li, R., Ma, J., Panyala, A., and He, D. (2010). "Hybrid ceramic bearing prognostics using particle filtering" *Proceedings of the 2010 Conference of the Society for Machinery Failure Prevention Technology*, Huntsville, AL, April 13 – 15, pp. 57 – 69.

Lim, P., Goh, C. K., Tan, K. C., and Dutta, P. (2014). "Estimation of remaining useful life based on switching Kalman filter neural network ensemble", *Proceedings of the 2014 Annual Conference of the Prognostics and Health Management Society*, Fort Worth, TX, pp. 2–9.

Liu, J., Wang, W., Ma, F., Yang, Y.B., and Yang, C.S. (2012). "A data-model-fusion prognostic framework for dynamic system state forecasting", *Engineering Applications of Artificial Intelligence*, vol. 25, no. 4, pp. 814-823.

Lv, Y., Duan, Y., Kang, W. Li, Z., Wang, F.-Y. (2015). "Traffic flow prediction with big data: a deep learning approach", *IEEE Transactions on Intelligent Transportation Systems*, vol.16, no.2, pp.865-873.

Maio, F. and Zio, E. (2013). "Failure prognostics by a data-driven similarity-based approach", *International Journal of Reliability, Quality and Safety Engineering*, vol. 20, no.1, pp. 1-17.

Malhi, A., Yan, R., and Gao, R. X. (2011). "Prognosis of defect propagation based on recurrent neural networks", *IEEE Transactions on Instrument and Measurement*, vol. 60, no. 3, pp. 703–711.

Mass, A., Hannun, A., and Ng, A. (2013). "Rectifier nonlinearities improve neural network acoustic models", *Proceedings of the 2013 International Conference on Machine Learning*, Atlanta, GA.

Men, Z., Yee, E., Lien, F.S., Wen, D. and Chen, Y. (2016). "Short-term wind speed and power forecasting using an ensemble of mixture density neural networks", *Renewable Energy*, vol. 87, pp.203-211.

Myötyri, E., Pulkkinen, U. and Simola, K. (2006). "Application of stochastic filtering for lifetime prediction", *Reliability Engineering & System Safety*, vol. 91, no. 2, pp.200-208.

Oliveira, T. P., Barbar, J. S., and Soares, A. S. (2014). "Multilayer perceptron and stacked autoencoder for Internet traffic prediction", *Network and Parallel Computing*, vol. 8707 of the series Lecture Notes in Computer Science, pp. 61-71.

Pourazarm, S., Farahmand A., Nikovski, D. (2017). "Fault Detection and Prognosis of Time Series Data with Random Projection Filter Bank", In Proceedings of the 2017 Prognostics and Health Management (PHM) Conference, St. Petersburg, Fl, USA, 2-5 October 2017.

Saxena, A., Celaya, J., Saha, B., Saha, S., Goebel, K. (2010). "Metrics for offline evaluation of prognostic performance", *International Journal of Prognostics and Health Management*, vol.1, no. 1, pp.4-23.

Shao, H., Jiang, H., Zhang, X., and Niu, M. (2015). "Rolling bearing fault diagnosis using an optimization deep belief network" *Measurement Science and Technology*, vol. 26, no. 11, pp. 115002-115018.

Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dielman, S. (2016). "Mastering the game of Go with deep neural networks and tree search", Nature vol. 529, pp. 484–489.

Smolensky, P. (1986). *"Chapter 6: Information Processing in Dynamical Systems: Foundations of Harmony Theory", Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, vol. 1. Cambridge, MA, USA: MIT Press, pp. 194–281.

Song, B.L. and Lee, J. (2013). "Framework of designing an adaptive and multi-regime prognostics and health management for wind turbine reliability and efficiency improvement", *International Journal of Advanced Computer Science and Applications*, vol. 4, no. 2, pp. 142-149.

Tao, Y., Chen, H., and Qiu, C. (2014). "Wind power prediction and pattern feature based on deep learning method", *Power and Energy Engineering Conference (APPEEC), 2014 IEEE PES Asia-Pacific* , vol., no., pp.1-4.

Vachtsevanos, G., Lewis, F. L., Roemer, M., Hess, A., and Wu, B. (2006). *Intelligent fault diagnosis and prognosis for engineering system*. Hoboken, NJ: John Wiley & Sons, Inc.

Wang, X., Takaki, S. and Yamagishi, J., (2017). "An autoregressive recurrent mixture density network for parametric speech synthesis", *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference,* New Orleans, LA, USA, March 5-9, pp. 4895-4899.

Wemhoff, E., Chin, H., and Begin, M. (2007). "Gearbox diagnostics development using dynamic modeling", *AHS 63rd Annual Forum*, Virginia Beach, VA.

Yin, X. and Li, Z. (2015). "Reliable decentralized fault prognosis of discrete-event systems", *IEEE Transactions on Systems, Man, Cybernetics: Systems*, vol. 46, no. 10, DOI: 10.1109/TSMC.2015.2499178.

Yoon, J. and He, D. (2015). "Development of an efficient prognostic estimator", *Journal of Failure Analysis and Prevention*, vol. 15, no. 1, pp.129-138.

Zakrajsek, J. J. and Townsend, D. P. (1993). "An analysis of gear fault detection method as applied to pitting fatigue failure damage", *NASA Technical Memorandum 105950*.

Zen, H. and Senior, A. (2014). “Deep mixture density networks for acoustic modeling in statistical parametric speech synthesis”, *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference*, Florence, Italy, May 4-9, pp. 3844-3848.

Zhao, G., Zhang, G., Ge, Q. and Liu, X. (2016). “Research advances in fault diagnosis and prognostic based on deep learning”, *Prognostics and System Health Management Conference*, Chengdu, China, October 19-21, pp. 1-6.

Zio, E. and Peloni, G. (2011). “Particle filtering prognostic estimation of the remaining useful life of nonlinear components”, *Reliability Engineering & System Safety*, vol. 96, no. 3, pp. 403-409.

APPENDIX

Some of the research presented in this thesis contains substantial material previously published in other journals. The following reuse policies for each of the journals/societies are provided below:

THE PROGNOSTIC AND HEALTH MANAGEMENT SOCIETY

APPLIED SICENCES AN OPEN ACCESS JOURNAL FROM MDPI

IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS: SYSTEMS

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

VITA

Jason Michael Deutsch

EDUCATION

B.S., Industrial Engineering
University of Illinois at Chicago, Chicago, IL, 2013

M.S., Computer Science
University of Illinois at Chicago, Chicago, IL, 2017

Ph.D., Industrial Engineering and Operations Research
University of Illinois at Chicago, Chicago, IL, 2014-present
Thesis title: "Development of Deep Learning Based Prognostics"

ACADEMIC EMPLOYMENT

Graduate Teaching Assistant, Department of Mechanical and Industrial Engineering, University of Illinois at Chicago, 2015-present. Responsibilities include: assisting students with understanding the course material, facilitating lectures, grading homework, quizzes and exams.

PUBLICATIONS

Journal Publications

Deutsch, J. and He, D. (2017). "Using Deep Learning Based Approach To Predict Remaining Useful Life of Rotating Components", *IEEE Transactions on Systems, Man, and Cybernetics: Systems*.

Deutsch, J., He, M., and He, D. (2017). "Remaining Useful Life Prediction of Hybrid Ceramic Bearings using an Integrated Deep Learning and Particle Filter Approach", *Applied Sciences*, vol. 7, no. 7.

Qu, Y., He, M., Deutsch, J., and He, D. (2017). "Detection of Pitting in Gears using A Deep Sparse Autoencoder", *Applied Sciences*, vol. 7, no.5.

Conference Publications

Deutsch, J. and He, D. (2016). "Using Deep Learning Based Approaches for Bearing Remaining Useful Life Prediction", *Annual Conference of the Prognostics and Health Management Society 2016*, Denver, CO, Oct. 2 – 8.