Remaining Useful Life Prediction of a Turbofan Engine Using Deep Layer Recurrent Neural Networks

by

Unnati Thakkar

A thesis submitted to the Faculty of Graduate and Postdoctoral Affairs in partial fulfillment of the requirements for the degree of

Master of Applied Science

in

Electrical and Computer Engineering

Faculty of Engineering and Design
Department of Electronics
Carleton University
Ottawa, Ontario, Canada

Abstract

Turbofan engine is a pivotal component of the aircraft. Engine components are susceptible to degradation over the life of their operation which affects the reliability and performance of an engine. In order to direct the necessary maintenance behavior, remaining useful life prediction is the key. This thesis presents a prediction framework for the Remaining Useful Life (RUL) of an aircraft engine using the whole life cycle data and deterioration parameter data based on a machine learning (ML) approach. In specific, a Deep Layer Recurrent Neural Network (DL-RNN) model is proposed to address the problem of prognostic instability based on deep learning. In addition, for the aircraft engine, a new health indicator (HI) measure is implemented based on the pre-processing of raw data. The proposed method is compared against Multilayer-Perceptron (MLP), Non-linear Auto Regressive Network with Exogenous Inputs (NARX), Cascade Forward Neural Network (CFNN) and validated through the IEEE 2008 Prognostics and Health Management (PHM) conference Challenge dataset and Commercial Modular Aero-Propulsion System Simulation (C-MAPSS) dataset provided by NASA results reveal a better predictive precision with respect to other ML algorithms.

Acknowledgements

First and foremost, praises and thanks to the Almighty God, for the showers of blessings throughout my research work to complete the research successfully.

I would like to express my deep and sincere gratitude to my research supervisor Dr. Hicham Chaoui for giving me the opportunity to do research and providing invaluable guidance throughout this research. His dynamism, vision, sincerity and motivation have deeply inspired me. It was a great privilege and honor to work and study under his guidance. He is a great teacher, and his doors were always open when I had any question. His prompt replies to the emails even during holidays helped in the rapid progression of this work.

I would like to extend my profound gratitude to Department of Electronics and Prof. Chaoui for the financial assistance in the form of TA and RA funding.

I am extremely grateful to my family for their unconditional, emotional and financial support throughout my life and during this project. Without their love, sacrifice, patience and motivation, this study would not have been feasible.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

Abbreviations	Definition
AI	Artificial Intelligence
ANN	Artificial Neural Network
CFNN	Cascade Forward Neural Network
C-MAPSS	Commercial Modular Aero-Propulsion System Simulation
CNN	Convolutional Neural Network
DDA	Data-driven Approaches
DL	Deep Learning
DL-RNN	Deep Layer-Recurrent Neural Network
GB	Gradient Boosting
GA	Genetic Algorithm
HI	Health Indicator
HMMs	Hidden Markov Models
HPC	High Pressure Compressor
HPT	High Pressure Turbine
ISE	Integrated Square Error
LSTM	Long-short term memory
LPC	Low Pressure Compressor
LPT	Low Pressure Turbine
ML	Machine Learning
MLP	Multi-Layer Perceptron
MAE	Mean Absolute Error

MSE	Mean Square Error
NARX	Non-linear Auto Regressive Network with Exogenous Inputs
PHM	Prognostics and Health Management
ReLU	Rectifier Linear Unit
RMSE	Root Mean Square Error
RUL	Remaining Useful Life
RF	Random Forest
RVM	Relevance Vector Machine
RVR	Relevance Vector Regression
SRU	Statistical Recurrent Unit
SVM	Support Vector Machine
SVR	Support Vector Regression

Chapter 1

Introduction

Humans have had to deal with a variety of degradations and failures in their creations over the years due to the corrosion of materials, structural fatigue, or even just bad design choices. A washing machine, for example, may be used indefinitely before it stops working. Since a malfunction of such an asset rarely results in a substantial financial loss or a fatal outcome, almost no maintenance is performed beforehand. This type of maintenance is called Corrective Maintenance. On the other hand, an automobile, for example, must be maintained on a regular basis according to either time or mileage to ensure that it is in good working order; without proper maintenance, a vehicle may be unsafe. This type of maintenance is called Preventive Maintenance or Condition Based Maintenance. Condition-based maintenance helps an asset's availability and protection to be maximized while reducing maintenance costs. Condition-based maintenance considers not only the asset's past and current health problems, but also its predicted future health. Predictive maintenance systems and/or applications are commonly referred to as "Prognostics and Health Management," or "PHM." The findings of PHM strategies have been shown to be accurate and consistent. However, one of their major drawbacks is that each part studied must be approached as a unique task, as not all devices work in the same way. As a result, expert’s skill is needed to properly collect and interpret data from reliability tests for a given system in order to gain a broad understanding of its health status. Reliability engineers have recently tackled many of the drawbacks of conventional PHM approaches from a data-driven standpoint. Indeed, advances in computational hardware and the ongoing development of Machine Learning (ML) and

Deep Learning (DL) techniques have allowed engineers to research and implement such algorithms in a variety of fields.

This thesis presents a DL method for estimating the state of health of any system under uncertainty using big machinery data. The approach consists of Layer-Recurrent Neural Network as a novel topology to predict the RUL of the turbofan engine. Extensive comparison is also carried out against three topologies, i.e., Multilayer Perceptron, Non-linear auto regressive network with exogenous inputs and Cascade forward neural network. Two different datasets are used to train, test and validate the proposed architecture. These consist of a C-MAPSS dataset and a Challenge dataset of turbofan engine.

1.1 Motivation

The need to reliably forecast potential behaviour of variables that control complex machines has prompted the adoption of data-driven computer health monitoring systems approaches in the reliability community over the last decade [1] [2], [3]. The advancement of Machine Learning (ML) and Deep Learning (DL) techniques has increased the usage of Data-Driven Approaches (DDA) as a supplement to standard methods for estimating the RUL, with Neural Networks becoming the most widely employed methodology for this purpose. Nonetheless, as it will be discussed later, there are still many DL implementations in this field that have yet to be tested. The engine is a heart of an airplane. System faults and the deterioration process will eventually occur with the emergence of action cycles. It is also crucial to know how often such failures will occur to enhance system's performance and reliability. Prognostics and Health Management (PHM) is a framework that delivers integrated yet individualized approaches for health system management [4]. Aircraft engine prognostics determine an engine's actual state of health and is the most complex and technically demanding comprehensive technology in prognosis and health management. Remaining useful life (RUL) is defined as the length from the current time to the failure of a system. The RUL prognostics is now a universal practice for every system, and it is the primary concern of many organizations, including NASA and the Prognostics and Health Management Institute (PHM). These organizations promoted the field by publishing several datasets and amongst all the two most famous datasets are Challenge dataset and C-MAPSS dataset for RUL prediction of turbofan engine.

1.2 Objectives and Contribution

The main objective of this thesis is to develop and implement a deep learning-based framework for estimating the health status of physical assets using data from multisensorial measurements. We suggest a novel data-driven method for assessing remaining useful life. To predict the RUL, the approach learns the relationship between acquired sensor data and health indicator. This thesis focused on two separate datasets provided by NASA namely C-MAPSS dataset and Challenge dataset. The challenge dataset consists of two datasets: training and testing whereas the C-MAPSS dataset consists of four datasets and each dataset is further divided into training and testing. Each row is a snapshot of data acquired during a single cycle of operation; each column is a different variable. Each engine has 21 sensors which collects data for different measurements including the temperature, pressure, speed, bypass ration, cooling flow of each module and so on. The first two columns are the unit number and operational time in cycles whereas the column number three, four and five are operational settings. Rest of the columns are sensor measurements. The aim of this thesis is to propose and validate a deep learning framework for producing accurate prognostics on a system's health. Two separate datasets are used to test the proposed system, and the output results are correctly described in tables and figures.

1.3 Organization of the Thesis

The rest of the thesis is organized as follows. Chapter 2 presents the quick review on the definition and purposes of Machine Learning and Deep Learning. In this context, DL techniques are thoroughly explained, addressing their advantages and applications and it also reviews various techniques for RUL prediction and their classification in detail. Following this, the architecture and working procedures of the proposed techniques and data description are described in detail in Chapter 3. Chapter 4 reviews the proposed deep learning framework along with the data preprocessing in detail. Next, Chapter 5 presents the RUL prediction results obtained in the MATLAB software along with a comparison against different methods. Finally, Chapter 6 provides the concluding remarks and insights on future research opportunities.

Chapter 2

Literature Review

The following chapter details the necessary background to successfully achieve the objectives stated for this thesis wok. First, a quick review is given on the definition and purposes of Machine Learning and Deep Learning. In this context, DL techniques are thoroughly explained, addressing their advantages and applications. This chapter also reviews various techniques for RUL prediction and their classification in detail.

2.1 Machine Learning

Machine learning became popular in the early 90's. Before that, in order to describe a process or phenomenon it was necessary to study the behavior of a system under certain boundary conditions. Also, it was necessary to analyze different effects over the system functionality and the properties of the system on which the model is dependent. It could take months, years, or even decades to identify certain interactions that can reliably model measurable phenomena. Machine learning can overcome all these issues. Machine Learning is the branch of science that studies how machines can learn to perform a task without being directly trained to do so. ML algorithms use a variety of approaches to discover correlations in data based on a given dataset, which are then used to forecast potential values of determined variables or perform other decision-making tasks [5]. In a more thorough manner, a ML computer program (model) learns to complete a given task T (output) from experience E (data) if its success on T improves with experience E, where the performance is assessed by a chosen metric measure P [6]. That is, if the model can learn how to model or interpret the targeted output variable from the raw data by training itself with new data and producing better results for the estimation metric P, then the ML algorithm can learn how to model or interpret the preferred output variable from the input data. As a result, the success of any machine learning algorithm is highly dependent on the representation of the data it is given. There are plenty of machine learning methods developed for a variety of applications. However, the core structure of how the algorithm works remains same for all the models i.e., firstly a parametric ML algorithm is said to be learned to learn from data and perform a particular task, indicating that the software has a set of parameters that must be tuned based on the training data. To train the data by itself, ML uses the training dataset given each sample in the data corresponds to the training instance. Further the performance of a model is evaluated through the metrices such as Root mean square error or mean square error depending on the problem the algorithm is intended to solve.

The main benefit of machine learning approaches is that they efficiently handle to analyze data without requiring specific knowledge about the problem to be solved or specifically writing down the basic rules that control the algorithm. As a result, the framework is considerably shorter, simpler to operate, and certainly more reliable.

2.1.1 Deep Learning

Nowadays, machine learning methods can be used in a variety of implementations, ranging from a speech recognition device in a smartphone that converts a user's voice into text to advertisements served up by webpages to each specific user depending on their previously searched items. Processing data in its raw form, on the other hand, necessitates learning skills that are still constrained by traditional machine learning techniques. This is because it requires designing a feature extractor that can translate raw data into a suitable internal representation or feature vector for the ML and constructing the features that feed an ML algorithm takes more time, expertise, and effort than training the ML model itself [7]. To overcome all these issues, DL techniques are designed. Deep learning is a subset of machine learning which are created by composing simple non-linear modules that transform the starting raw input into a higher abstract level representation with multiple level of representation. One of the many approaches to Artificial Intelligence is Deep Learning (DL) that allows computer system to learn and improve with experience and data [8].

DL algorithms have been around for over two decades, but their deployment has been hampered by the high computing capacity needed to tune the large number of parameters in a DL architecture. In addition, in order to effectively train a model, vast datasets must normally be evaluated, resulting in extensive training and testing procedures. However, in the last decade, DL algorithms have increased in popularity as the cost of computational components has declined, allowing students, academics, and virtually all in the science world to access fast computers. This, along with the advancement of GPU-based computing, has accelerated the adoption of DL techniques.

2.1.2 Artificial Neural Networks

Artificial Neural Networks have been popular in recent years with their strong ability to approximate functions [9]. ANNs are biologically inspired networks that are closely equivalent to the behavior of neural networks in the brain and are used as machine learning applications that are made up of data processing neurons in the brain, which are neural network components [10] [11]. A biological neuron comprises of many branches classified as dendrites and axon [12]. The impulse signals are received by the dendrites, and when a certain threshold limit is reached, an acceptable response signal is produced [12]. With the aid of axon, this response signal is transmitted to other neurons through a complex branching mechanism [12]. After attaining the neuron body, the incoming signals (inputs) are weighted and summed, and then transformed into outgoing signals after passing through the transfer function [12]. The input layer, hidden layer/s, and output layer are the three layers that make up an ANN [13]. The input layer receives data from the network and transfers it as raw information to the hidden layer/s [13].. This data is processed by the hidden layer/s and transferred to the output layer [13]. There is an interconnected link established by a neuron between an input and a desired output. Inside an ANN, each layer performs a linear transformation by multiplying the data from the previous layer by a weights matrix and adding a bias term b. A non-linear activation function, such as the Rectifier Linear Unit (ReLU) or the hyperbolic tangent, is used to determine the output of this linear transformation (tanh) [14]. ANN uses data from continuous monitoring programs which includes samples from preparation [15]. Fig. 2.1 illustrates an example of an ANN architecture, where a three-layer ANN is presented. The number of nodes in the NN structure's hidden layer is determined by trial and error. The input layer feeds the signals to the hidden layer nodes, and the output layer produces the output after processing them with hyperbolic tangent activation functions. There are various training functions to train the NN and amongst all Levenberg-Marquardt is most used as it takes less computing time.

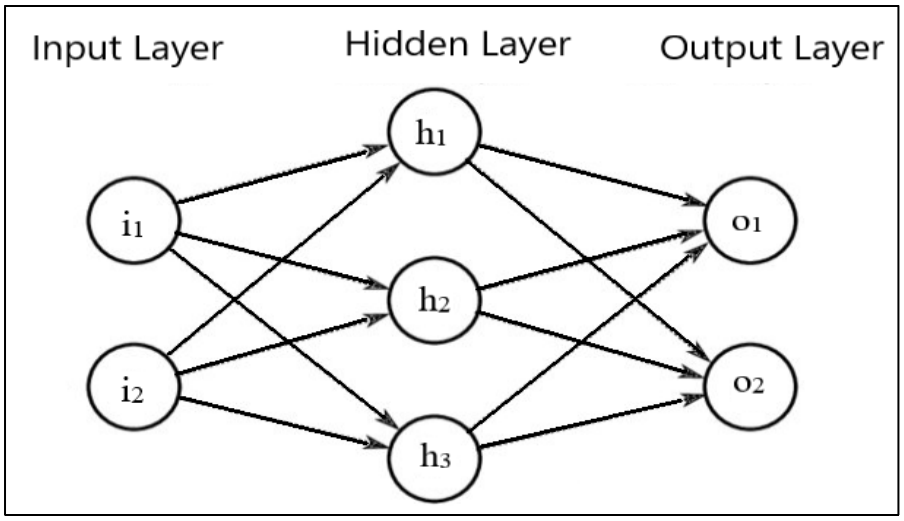

Fig. 2.1: Artificial Neural Network Architecture [16]

Since deep learning architectures typically offer a lot of flexibility, it is simple for overfitting to happen during the training phase. That is, it is possible to achieve an over adjustment of the model's weights and prejudices to the training data as a result of the training phase, resulting in poor generalization results to unseen data (test data). To overcome this issue, regularization techniques are applied. Regularization is a strategy for enhancing the generalization of a learning algorithm by making small changes to it. One more common regularization technique is early stopping which ends the training loop when training and validation failures begin to diverge.

ANN is the most popular approach for estimating useful life of turbofan engine. An advanced neural network training is performed in [17] to classify the difference between a good and a bad system. Recurrent Neural Network (RNN) is proposed in which to create an accurate and compact algorithm, the recurrent neural network is trained with back-propagation by time gradient calculations, an Expanded Kalman Filter training process, and evolutionary algorithms. Initially, a Multi-Layer Perceptron was used to train the algorithm, but the output was very noisy and the MLP network is reasonably accurate for predicting remaining useful life at the end of the run when the machine was close to failure. However, the job of forecasting RUL is much more challenging during the early and middle sections of the race, and the performance is not as good. Long Short-Term Memory (LSTM) which is an artificial neural network architecture used in the field of deep learning is proposed in [18] for RUL prediction of turbofan engine. LSTM can learn how long the old data should be memorized, when it should be forgotten and when new data should be used. An initial RUL was set to improve the estimation accuracy and the initial RUL of 97 showed better performance compared to initial RUL of 130. The performance of LSTM was compared with other methods such as Multilayer Perceptron (MLP), Support Vector Regression (SVR), Relevance Vector Regression (RVR) and Convolutional Neural Network (CNN) and it outperformed all the methods. LSTM performs well with time series data, but it takes a very long time to converge [19] [20]. In order to predict the RUL of the same engine, Convolution Neural Networks (CNN) is proposed [21]. For sample processing, it adopts a time window strategy, which may provide more detail on deterioration. As a result, the dimension of model inputs becomes greater, so the creation of a neural network model is a daunting task: how to set up network nodes. A Neural Network based on Statistical Recurrent Unit (SRU) is proposed in [22] for RUL prediction of turbofan engine. This method can derive hidden partners from multivariate time series sensor data with various operation state faults and errors and outperforms other dynamic methods of deep learning, since it obtains a plurality of historical perspectives by linear combinations of few averages.

2.2 RUL Prediction Techniques

Remaining useful life (RUL) is defined as the length from the current time to the failure of a system and its prediction techniques are classified in the following ways: model-based technique (also called physics of failure technique), data driven based technique and hybrid-based technique [23]. The methods are listed in Fig. 2.2. The detailed explanation of each technique is given below.

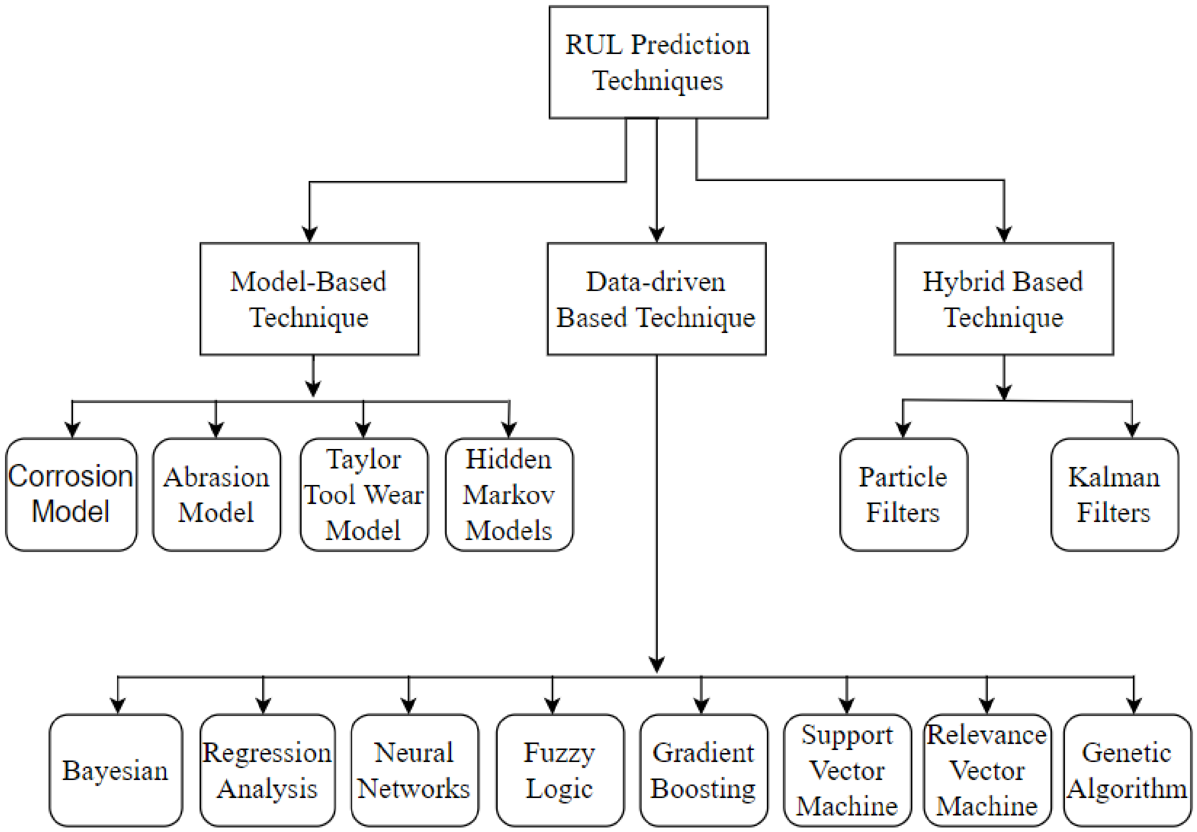

Fig. 2.2: RUL prediction techniques

2.2.1 Model-Based Technique

Model-based technique also known as physics of failure technique relies on accurate physical model that represents the system's dynamics and integrates the physical model with calculated data to define the model parameters and forecast potential behavior [24]. It is derived from the model and uses the data on "run-to-failure" and is applicable to decision-making for maintenance [25]. RUL is often estimated using a model-driven approach, which guides maintenance decisions based on failure threshold. Model based technique includes Corrosion Model, Abrasion Model, Taylor Tool Wear Model and Hidden Markov Models (HMMs) [26] [27]. The model-based technique is easy and fast to implement, and the model can be reused. However, model development requires a thorough understanding of the system and high-fidelity models can be computationally intensive [28] [29].

2.2.2 Data-Driven Based Technique

The data-driven method uses data from previously acquired data (training data) to analyze the characteristics of the damage situation currently analyzed and to model the future trend. Data-driven methods are determined by analyzing frequent condition-monitoring data from device metrics on daily basis [30] [31] [32]. Learning approaches from machine learning and pattern analysis are used in data-driven prognostics. Examples of the data driven techniques are Regression Analysis, Neural Networks, Fuzzy Logic, Bayesian, Random Forest (RF), Gradient Boosting (GB), Support Vector Machine (SVM), Genetic Algorithm (GA), and Relevance Vector Machine (RVM) [33] [34]. Data driven technique is easy and fast to implement and it may identify relationships that were not previously considered. On the other hand, it requires lots of data, a balanced approach and there is a risk of overlearning and over-generalizing the data [35]. Bayesian networks also called as belief networks is a probabilistic model that depicts a series of random variables as well as their conditional interdependencies [36]. In different modes, the Markov model is represented as a Markov mechanism with unobserved (hidden) state identification. For turbofan engines with powerful nonlinear and time-dependent characteristics, the SVM and RF are not appropriate, because the dynamic interaction between sensor data and RUL cannot be captured within a comparatively simple framework. Among all the data driven techniques, Artificial Neural Networks are commonly used for prognostics.

2.2.3 Hybrid based Techniques

Hybrid based technique also known as fusion-based technique is the combination of model based and data-driven technique. It uses the data to learn the model parameters and the knowledge about the physical process to determine the type of regression analysis to apply (linear, polynomial, exponential, etc.). It forecasts RUL independently and, using probability theory-based approaches, allows the combination of two or more RUL prediction outcomes to produce a new RUL. The hybrid-based technique combines the strength of each approach and it uses the data where system knowledge is lacking, and it uses physics where data is lacking. On the other hand, it requires in-depth knowledge of the system and meaningful data is still needed [37] [38]. The examples of hybrid-based technique are Particle filters, Kalman filters etc. [39].

Next chapter provides the architecture and working procedures of the proposed techniques and theories are described in detail. It also provides the data description in detail.

Chapter 3

3.1 Application: Gas Turbine Engine

A gas turbine engine, also known as a jet engine, is a type of turbine engine that spins pressurized gas to generate electricity or to provide an airplane with kinetic energy [40]. There are mainly five types of gas turbine engines, namely, turbojet, turbofan, turboprop, turboshaft and ramjet engine [40]. The turbofan engine or fanjet is the modern variant of gas turbine engine that is widely used for aircraft propulsion. Propulsion is the net force produce by the thrust that push the object forward [40]. So, because of their high thrust and good efficiency, almost all airliners use turbofan engines. The engine refers to the mechanism that transforms the energy of the fuel into shaft power and the power of the shaft power to drive it. The engine is composed of air intake, a compressor (low and high pressure), a combustor or combustion chamber, a turbine (high and low pressure) and an outlet or exhaust nozzle. The air enters and then travels through the compressor to reach high pressure through the air inlet. Further, increases the temperature of the high-pressure air burning (mixing) it in the combustor with the gasoline. The combustion results in high levels of hot velocity gases that move through the turbines, producing energy to power the turbines from a compressor. Such high-speed hot gases are exhausted from the compressor to the turbine. The high velocity gas coming from turbine is drained out through the jet nozzle which produces a thrust to drive the engine forward outside through the outlet. Some common advantages of turbofan engine are it is fuel efficient and it produces less noise than turbojet engine. On the other hand, being massive than turbojet and inefficient at very high altitudes are the two disadvantages [40].

3.1.1 Components of Gas Turbine Engine

The working principal of turbofan engine is explained in detail below. There are five main parts of the gas turbine engine which plays a very important role in the operation of an engine.

a) Air inlet

The air travels through the engine from left to right passing through the fan. The fan oversees producing most of the turbofan engine-generated thrust. The fan is interconnected to the low-pressure compressor and the low-pressure turbine via the shaft. Most of the air passing through the fan travels around the core of the engine which is known as bypass air and it never interacts with other components of engine. Infact it is directly accelerated from the rear of an engine [41]. The leftover air enters an engine's heart and then enters the low-pressure compressor.

b) Compressor

The main function of the compressor is to add energy to the air in the form of pressure and heat which makes it ready for combustion. As the pressure increases, temperature of the air also increases. There are primarily two types of compressor: the low-pressure compressor (LPC) and the high-pressure compressor (HPC). The low-pressure compressor is coupled with the fan and the low-pressure turbine via the low-pressure shaft. Whereas, directly downstream of the LPC and directly upstream of the combustor is the high-pressure compressor located which is connected to high-pressure turbine through high pressure shaft [41].

c) Combustor

The hot pressure air travelling through the compressor mixes with the fuel and burned in combustion chamber. This is where the reaction takes place, and it results in very high temperature and high velocity gases which is used to run the turbine [41]. Combustion chamber is placed just after the compressor and they are made with the materials that can withstand high temperature resulting from combustion. It is found in an engine's heart.

d) Turbine

Turbine consists of row of blades which spins. The high temperature and high velocity gas from the combustor pass through the turbine which extracts the energy and transfers it to the compressor and fan through connecting shaft. This energy helps compressor and fan to spin. The temperature and pressure of the air leaving the turbine is relatively low due to the energy extracted by the turbine. Two types of turbines primarily exist: the high-pressure turbine (HPT) and the low-pressure turbine (LPT). The high-pressure turbine is located to the downstream of the combustor or burner and to the upstream of low-pressure turbine. Whereas the low-pressure turbine is located to the downstream of high-pressure turbine and to the upstream of exhaust nozzle [41].

e) Exhaust nozzle

Exhaust nozzle is the component located at the right most of the engine and just after the turbine. Also, it is the last component the air passes through before leaving the engine. It is a tube-shaped component which helps to produce additional thrust which helps the engine move forward [41]. Moreover, it also helps regulating pressure within the engine which helps other components functioning properly

3.1.2 Failure of Turbofan Engine

Just like the life of human being is affected by their health same as all systems deteriorate as a result of time, usage, and environmental conditions. Gas-turbine engines may be subjected to severe environmental and operating conditions such as corrosion, wear, buckling, erosion etc. which eventually lead to costly and catastrophic failures if a run-to-failure philosophy is adopted [42]. Hence, in order to monitor the state of gas turbine engine more than two hundred sensors are installed in each gas turbine. An engine health deteriorates substantially over time. Certain engine variables which are also known as health parameters have significant impact on engine health. Different engines have different health parameters. Hence it becomes important to monitor and evaluate these health parameters in order to improve life, performance, reliability, etc. of engine. However, due to the complexity of aircraft and sometimes due to unavailability of devices to measure certain parameters it is not possible to directly measure the health parameters [42] .Hence it is of great importance to have fault diagnosis and prognostics and health management (PHM) on engine. The prognostics on remaining useful life (RUL) of an engine is the most difficult and challenging part among PHM.

3.2 C-MAPSS Dataset for Turbofan Engines RUL Prediction

The first dataset used to test the proposed deep learning architecture is the C-MAPSS data competition benchmark data [43].

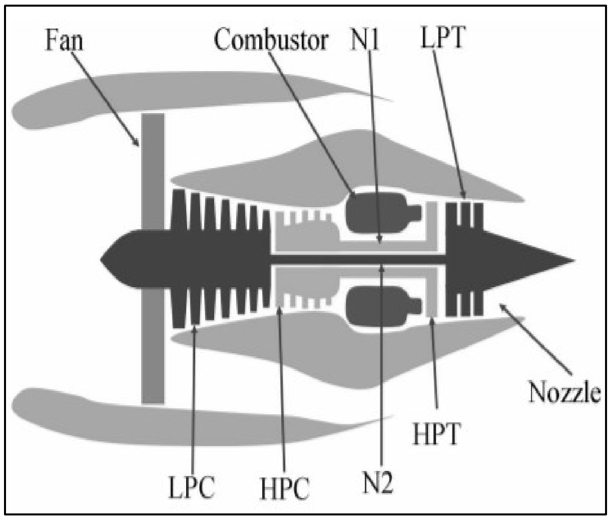

Fig. 3.1: Simplified diagram of engine simulated C-MAPSS [44]

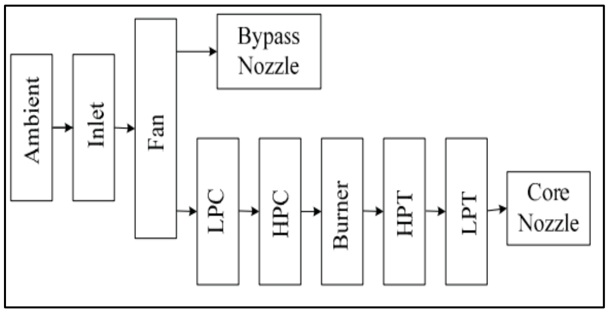

Fig. 3.2: C-MAPSS modular components [44]

The engine diagram in Fig. 3.1 shows the main elements of the engine model and the modular components of the simulation software and their connections are shown in Fig. 3.2. Commercial Modular Aero-Propulsion System Simulation (C-MAPSS) simulates the 90000lb thrust level commercial high bypass ratio turbofan engine by MATLAB/Simulink, which built the engine and relative control system model [44]. The PHM08 Challenge dataset and Turbofan Engine Degradation Simulations were implemented using the C-MAPSS software [44] .Training and test trajectories are two types of datasets, with the latter being a subset of the former. Operational cases of full run-to-failure data are generated in the training trajectories, and these are intended to be used to train the algorithms. Test trajectories, on the other hand, can only be set up using shorter instances of data up to a certain point before the method fails. Four different operating conditions are simulated using a collection of trajectories, as shown in Table 3.1. To model uncertainty in parameter readings during operation, the sensors have been polluted with operating regimes and noise. In addition, each trajectory has its own degree of initial wear and manufacturing variance, which is considered natural and unnoticed by the user. Each dataset is stored in a CSV file, with each row representing one time-step (measured in cycles) and containing 21 sensor measurements, three operating conditions, and other useful information (see Table 3.2). Since each time series is from a different engine, the data from the C-MAPSS dataset may be considered to come from a fleet of similar engines (Unit Number).

Table 3.1: C-MAPSS Dataset

Dataset	Train Trajectories	Test Trajectories	Conditions	Fault Mode
FD001	100	100	One (sea level)	One (HPC)
FD002	260	259	Six	One (HPC)
FD003	100	100	One (sea level)	Two (HPC, Fan)
FD004	248	249	Six	Two (HPC, Fan)

Table 3.2: Variables C-MAPSS Dataset

Index	**Data description**
1	Unit number
2	Time (in cycles)
3	Operational setting 1
4	Operational setting 2
5	Operational setting 3
6	Sensor measurement 1
7	Sensor measurement 2
8	…
9	Sensor measurement 21

The engine's performance is greatly affected by the three operational settings used in the dataset. The engine begins working normally with an unknown initial deterioration for each trajectory in the training sets before failure is detected. Sensor noise has also contaminated the data. The time series in the test set ends just before the device fails, and the RUL for each test engine is given in a separate CSV file. In this thesis, only the first sub-dataset is evaluated i.e., FD001.

3.3 Challenge Dataset for Turbofan Engines RUL Prediction

Over two hundred multivariate time-series datasets were provided as a part of PHM 2008 prognostics data challenge [45]. Each dataset is unique with unknown initial wear and manufacturing variation. The datasets consist of three operational settings and twenty-one sensor measurements that are contaminated with sensor noise. Three datasets are provided namely "training" and "testing" and "final". Training trajectories are used to train the algorithm to predict remaining useful life and is formed of complete run-to-failure data. Each time series is from a different engine, i.e., the information can be of the same kind from a fleet of engines. The trained algorithm is then feed to test trajectories which are formed of short instance time-series that ends some time prior to degradation. Users will implement their methods to the final test dataset and submit the vector of expected RULs to the Prognostics Center of Excellence for review after their algorithms have been trained to their satisfaction. There is a web-based framework for uploading test data and calculating an overall score feedback. Each row is a snapshot of data acquired during a single cycle of operation; each column is a different variable. Each engine has 21 sensors which collects data for different measurements of engine. The minimum unit length is 127 cycles whereas maximum unit length is 356 cycles. The number of trajectories for training, testing and final test are 218, 218 and 435, respectively. The first two columns are the unit number and operational time in cycles whereas the column number three, four and five are operational settings. Rest of the columns are sensor measurements as shown in Table 3.3.

Table 3.3: Sensor Measurements

	Symbol	Description	Units
Sensor 1	T2	Total temperature at fan inlet	°R
Sensor 2	T24	Total temperature at LPC outlet	°R
Sensor 3	T30	Total temperature at HPC outlet	°R
Sensor 4	T50	Total temperature at LPT outlet	°R
Sensor 5	P2	Pressure at fan inlet	psia
Sensor 6	P15	Total pressure in bypass-duct	psia
Sensor 7	P30	Total pressure at HPC outlet	psia
Sensor 8	Nf	Physical fan speed	rpm
Sensor 9	Nc	Physical core speed	rpm
Sensor 10	epr	Engine pressure ratio	-
Sensor 11	Ps30	Static pressure at HPC outlet	psia
Sensor 12	phi	Ratio of fuel flow to Ps30	pps/psi
Sensor 13	NRf	Corrected fan speed	rpm
Sensor 14	NRc	Corrected core speed	rpm
Sensor 15	BPR	Bypass ratio	-
Sensor 16	farB	Burner fuel-air ratio	-
Sensor 17	htBleed	Bleed Enthalpy	-
Sensor 18	Nf_dmd	Demanded fan speed	rpm
Sensor 19	PCNfR_dmd	Demanded corrected fan speed	rpm
Sensor 20	W31	HPT coolant bleed	lbm/s
Sensor 21	W32	LPT coolant bleed	lbm/s

Table 3.4 below shows the number of trajectories, faults modes and unit length of all three trajectories.

Table 3.4: Challenge Dataset

	Train Trajectories	Test Trajectories	Final Test Trajectories
Number of Trajectories	218	218	435
Fault Modes	One (HPC)	One (HPC)	One (HPC)
Maximum Unit Length	357	364	298
Minimum Unit Length	128	15	20

Chapter 4

Deep Learning Framework

Many deep learning techniques have been used for RUL prediction of turbofan engine. Recurrent Neural Network trained with back-propagation through time gradient calculations showed a great performance for RUL prediction. On the other hand, LSTM neural network outperformed all other methods such as MLP, SVR, RVR and CNN and the prediction accuracy was improved by setting an initial RUL of 97. This chapter will discuss the deep learning techniques used in this thesis.

4.1 Deep Learning in Prognostics for Mechanical Components

4.1.1 Multilayer Perceptron Neural Networks

The multilayer perceptron neural network is made up of several layers, as its name suggests. Multilayer perceptron (MLP) is a supplement of feedforward artificial neural networks having one or more hidden layers. They are widely used for pattern recognition, input pattern classification, prediction based on input data, and approximation [46].The connection structure of a feedforward neural network is unidirectional multi-layer, with each layer consisting of many neurons. Each neuron receives the output of the previous layer's neurons and produces the output for the next layer's neurons. Generally Multilayer Perceptron Neural Network consists of input layer, hidden layer and an output layer [47]. Every perceptron sends outputs to all perceptron’s on the second layer i.e., the hidden layer from the first layer on the left i.e., the input layer and all perceptrons on the second layer transmit outputs to the final layer on the right i.e., the output layer. Every perceptron transmits multiple signals which passes through the next layer. Moreover, the perceptron sets unique weights for every signal it transmits. In order to train the algorithm for prediction remaining useful life, multilayer perceptron was used. Multilayer perceptron algorithm was implemented using MATLAB with ten inputs and one target for the challenge data and fourteen inputs and one target for the C-MAPSS dataset. The inputs and the output (target) were all normalized between the range of [0,1] using min-max. The inputs were the normalized raw sensor data of useful sensors and the target was HI. Four hidden nodes were used in first layer whereas five hidden nodes were used in second layer. The training function used to train the algorithm was Levenberg-Marquardt which takes less time compared to Bayesian regularization but more memory to train and it avoids overfitting. The performance function was set to mean square error (MSE). The algorithm was trained for around 58 seconds to 1 minute 26 seconds with 18000 epochs and then the same training raw data was incorporated into the received network feature to verify that the learned neural network precision is adequate. The validation resulted into the similar pattern but when compared to the original target the results were not satisfactory. The MLP network is reasonably accurate for predicting remaining useful life at the end of the run when the machine was close to failure. However, the job of forecasting RUL is much more challenging during the early and middle sections of the race, and the performance is not as good. Various changes were made to improve the performance of the network such as different training functions were used namely Bayesian regularization, Scaled conjugate gradient backpropagation, RPROP backpropagation and so on. Also, different performance functions and transfer functions were implemented in order to improve accuracy of the function, but the performance did not improve.

4.1.2 Nonlinear Autoregressive Network with Exogenous Inputs

The non-linear autoregressive network with exogenous inputs is a persistent hierarchical network, with feedback relations encompassing many layers of the network. A NARX network consists of a multilayer perceptron taking the variables of the input state as a window for past inputs and the values of the output and determines the current output. This implies that the current value of a time series is related by the model to both: values from the history of the same series; and present and historical values of the (exogenous) driving sequence [48] [49]. The NARX model can be considered as a dynamic recurrent neural network and is based on the ARX linear model, which is widely used in the simulation of time series [48] [49]. In several dynamic systems with complex nonlinearities, NARX has been used. NARX neural networks are not only computationally efficient in theory, but they also have a range of practical advantages. Gradient-descent learning, for example, has been shown to be more efficient in NARX networks than in other neural network architectures with "hidden states" [50]. Despite the NARX network's benefits, its potential as a nonlinear method for univariate time series modelling and prediction has yet to be thoroughly investigated [51]. After the poor accuracy of MLP neural network, the NARX algorithm was implemented using MATLAB. A NARX network can be trained in two ways [52]:

a) Series-parallel mode sequence or open loop mode NARX network in which the actual output is directly used instead of the approximate output being fed back.
b) The parallel mode of the NARX network or the closed loop network is to feed the expected performance back to the network.

Input and output sequence initialization is the first step in the training of the network structure. Inputs were configured as a "10*time" cell of usable raw sensor data for the Challenge data and "14*time" cell for C-MAPSS dataset, while the target series was configured in the previous segment as a 1*time HI cell. The input and feedback delays were set to two along with two hidden layers in total. The first hidden layer has four nodes, and the second hidden layer has five nodes. Both the inputs and the target were mapped in the range of [0,1] using min-max. The algorithm was trained initially in an open loop mode and then changed to closed loop mode. Levenberg-Marquardt was used as a training function and the convergence time was around 1 minute 4 seconds to around 1 minute 50 seconds with tansig as a transfer function. After training the algorithm, the validation was carried out on same raw inputs to verify the accuracy of trained neural network. The validation resulted a very poor performance of the network. Smooth data was used to filter the noise and smooth the output.

4.1.3 Cascade Forward Neural Network

Cascade-forward neural networks are similar to feed-forward networks in that they have a connection from the input and every previous layer to the subsequent layers [53]. In a three-layer network, the output layer is connected directly to the input layer as well as the secret layer. Provided enough hidden neurons, a two-or more layer cascade-network, like feed-forward networks, can learn any finite input-output relationship arbitrarily well [53]. For any form of input to output mapping, a cascade-forward neural network can be used. This approach has the advantage of accommodating the nonlinear relationship between input and output while leaving the linear relationship intact [54]. We use the network in the field of time series in this analysis. In both the input and hidden units, the optimal architecture was computed using the incremental search process. The basic one was built first, and then the more complicated one was built by one-by-one adding the modules. The best option is then selected based on the mean square error criterion. To make predictions of time series data, a CFNN model that is nonlinear-nonparametric and more versatile can be used. This allows for more accurate results [54].

The issue with CFNN modelling is the input selection process, which must be followed in order to obtain the best architecture. The initialization of the input and output sequences is the first step in the training of the network structure. For the Challenge data, inputs were configured as a "10*time" cell of accessible raw sensor data and "14*time" cell for the C-MAPSS dataset, while the goal series was configured as a 1*time HI cell. Two input and feedback delays, as well as two hidden layers, were used in total. With tansig as a transition tool, the convergence period was around 1 minute 8 seconds to around 1 minute 35 seconds with Levenberg-Marquardt as a training function. After training the algorithm, validation was performed on the same raw inputs to ensure that the trained neural network was accurate. The network's success was severely harmed as a result of the validation.

The proposed deep learning framework is based on Layer-Recurrent Neural Network. An exponential RUL target is proposed as an objective for C-MAPSS dataset. Raw data is normalized with a Min-Max scaler in the range [0,1] before training the resulting model, and variables from the dataset that do not contribute useful information to the model or were used for label generation are dropped. Data pre-processing is performed on Challenge dataset to remove senor noise and to construct health indicator as a target for neural network training. For FD001, dataset of C-MAPSS health indicator is constructed through an exponential degradation model by using the initial RUL value. Once the health indicator is constructed for both the datasets, the proposed architecture is trained and tested through the validation datasets described in Chapter 3.

4.2 Data Preprocessing

The pre-processing of raw data is a required step in any analysis that uses data-driven techniques [55]. NASA published a dataset with 21 sensor signal variables. The spectrum of these signals may be vastly different, having a significant effect on model training. Therefore, signals must be normalized. There are several normalization techniques like scaling to range, clipping, log scaling and z-score. To normalize the data and prevent overfitting the model over variables with a high order of magnitude, a scaling to range normalization, also known as Min-Max normalization, is applied to both training and testing trajectories of C-MAPSS dataset along each variable. Equation describes how to evaluate the scaled value $Xscaled$ of a vector X. The minimum and maximum values of each column function are represented by $Xmin$ and $Xmax$, respectively.

$$Xscaled = \frac{X - Xmin}{Xmax - Xmin} \tag{4.1}$$

In addition, variables that do not alter over time and thus could affect the results are removed. In this thesis fourteen useful sensor measurements 2,3,4,7,8,9,11,12,13,14,15,17,20,21 are selected, and the irregular/unchanged sensor data are abandoned, as [56] [57] [58].

A similar approach is taken to deal with the challenge dataset. A data analysis is used in this work to explain feature extraction, data cleaning, and feature collection in order to improve the multi regime normalization tool and obtain a useful health index for prognosis. A data processing approach that involves feature extraction, data cleaning and feature selection is necessary to have an HI that is useful for prognosis. Feature extraction is linked to data cleaning, which is the way of detecting and likely correcting faulty or inaccurate measurements from data. It involves gathering field information to locate inaccurate or irrelevant parts for correction or standardization. Then, to pick the appropriate features for use in RUL prediction construction, a feature selection procedure is carried out. Without a validation method, it is impossible to find the best attainable high precision model in real-world data. Such data is often inconsistent, incomplete, and deficient habits, as well as containing multiple errors. Data preprocessing entails converting these data into a readable format. As a result, the approach starts by resolving issues related to organizing condition monitoring data, such as reducing data reliability and enhancing data integrity. The dataset provided by NASA has 21 sensors measuring parameters which includes temperature, pressure, bypass ratio. The range of the variables may be very different which urge the need of normalization as it may have great impact on the training of algorithm. Moreover, the sensor data of training dataset is contaminated with lot of noise. As can be seen from the Fig. 4.1, the raw data directly comes from the engine source and is not subjected to any processing or cleaning methods. Also, each trajectory has specific initial wear and is contaminated with noise. Hence pre-processing of the raw data is needed which includes regime clustering, regime normalization and sensor selection. The pre-processing was performed on raw data in order to transform into an understandable data before switching on to the remaining useful life. The steps of pre-processing include regime clustering, regime normalization, sensor selection, trendability analysis and constructing health indicator [59]. The pre-processing was performed on both training and test trajectories. The pre-processing steps are shown below:

a) Regime Clustering

This is the very first step of data pre-processing which is used to identify the operational regimes. The operational settings in the engine data have a direct effect on the sensors, and several regimes can be discovered by determining the optimal number of clusters in these settings. It is unlikely that all operational settings in different data would respond to a clustering approach in the same way. Some people may react better than others, while others will not respond at all. Clustering, on the other hand, should not be a one-size-fits-all approach to operational settings, but rather as a general problem to solve. Clustering is a method used for large datasets to identify and group similar data points. The number of regimes can be identified by determining the number of clusters. There are various methods of regime clustering such as k-means clustering, hierarchical clustering, grid-based clustering etc. This thesis uses k-means algorithm. After applying k-means algorithm, it was found that there are six cluster points which means there are six operational regimes of three operational settings as shown in Fig. 4.2 [59].

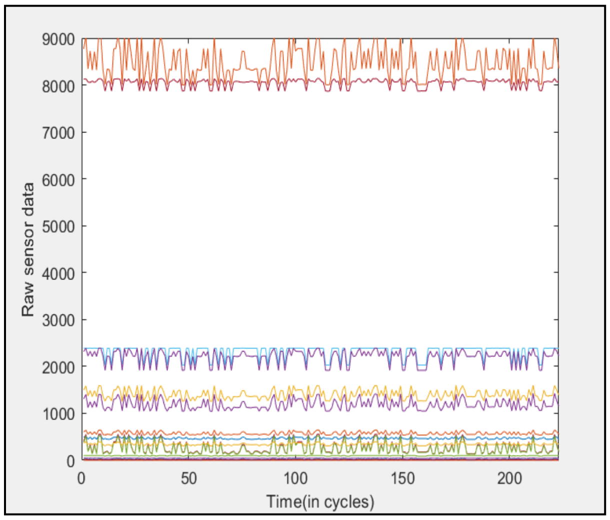

Fig. 4.1: Raw data from the engine source

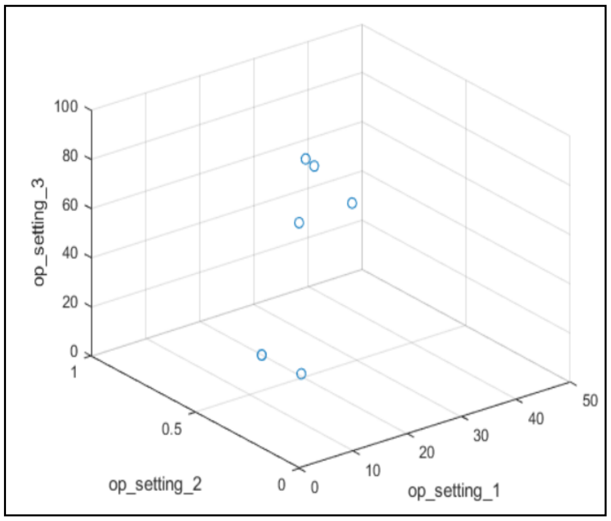

Fig. 4.2: Operational regimes for three operational settings

b) Regime Normalization

The dataset provided by NASA includes raw values that are multivariate, incompatible with one another, and work under various regimes. The raw data must be scaled down from their original scales to something more common. Normalization is the method carried out on raw data so that all the data are on a common scale and normalized. By returning raw values into a single domain, a normalization is a method for performing these changes. This includes calculating z-score which is nothing but subtracting each regime's population average from each individual raw sensor and then dividing the difference by the standard deviation of the population. The formula to calculate z-score normalization is given as:

$$Z - score = \frac{x - \mu}{\sigma} \tag{4.2}$$

Where x is the raw data value of the regimes, μ is the population average and σ is the standard deviation of the population. The population mean is given by:

$$\mu = \frac{1}{n}\sum_{i=0}^{n} x \tag{4.3}$$

Where n is the number of observations and standard deviation is given as:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x-\mu)^2}{n}} \tag{4.4}$$

After applying this equation to each regime separately, the standardized sensors are used to reassemble the normalized dataset, considering the sensor locations at the outset in order to allocate them to a common scale while preserving the initial wear characteristics within each trajectory. The result of normalization for the first five sensors i.e., sensor 1, sensor 2, sensor 3, sensor 4 and sensor 5 are shown in Fig. 4.3 [59].

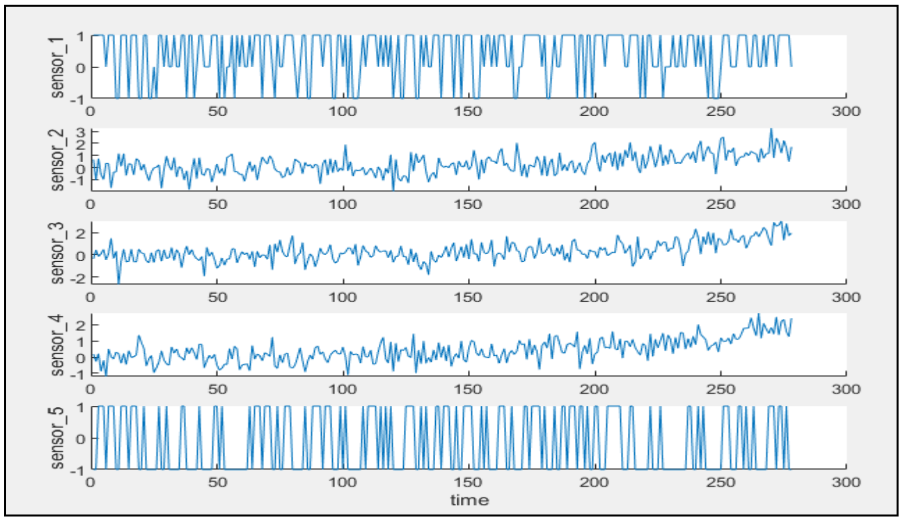

Fig. 4.3: Result of normalization for first five sensors.

c) Sensor Selection

After performing regime normalization, we can see the trend of all 21 sensors. It is important to exclude those sensors that do not accurately reflect continuous normalized values exhibiting a monotonic exponential trend during the operational lifetime of the training units in order to determine how well the sensors represent the degradation pattern [59]. For doing this, this study uses the prognostic parameter-choosing measures "Monotonicity," "Prognosability," and "Trendability" [60] to allocate the useful (and meaningful) sensors to be used in subsequent stages. Monotonicity is a simple metric for determining if a sensor has the same underlying positive or negative patterns that are needed for prognosis. This metric defines the general increasing or decreasing existence of a sensor, showing the system's degradation process. It is defined as follows:

$$Monotonicity = mean\left(\left|\frac{\#pos\frac{d}{dx}}{n-1} - \frac{\#neg\frac{d}{dx}}{n-1}\right|\right) \tag{4.5}$$

where n is the number of trajectories. The average difference of the fraction of positive and negative derivatives for each trajectory is used to determine the monotonicity of a sensor population. This metric detects whether the sensor has a characteristic underlying positive or negative trend because device deterioration follows monotonic trends in all trajectories in a dataset.

The deviation of the failure threshold points for each trajectory is divided by the average variance of the sensor over its entire operating period to measure prognosability [60]. The main aim of this metric is to quantify the distribution of a sensor's failure value (or threshold point) over a population of trajectories. To achieve the desired 0 to 1 scale, the calculation is exponentially weighted using the following equation:

$$Prognosability = exp\left(-\frac{\sigma failure}{\mu failure - \mu healthy}\right) \tag{4.6}$$

Prognosability measures near 1 indicate that the failure points in all trajectories are identical domains and that the sensors are useful for prognosis, while measures near 0 indicate that the failure thresholds do not fit and that the sensors are unable to provide meaningful prognostic calculations [60].

The minimum absolute correlation computed among all trajectories is used to determine trendability. The metric primarily determines if the sensor for a group of trajectories has the same underlying trend pattern and, as a result, can be represented by the same parametric function. Trendability analysis includes the selection of most trendable sensors measurements in order to construct health indicator (HI) and the degradation pattern of each sensor is estimated [59]. Mathematical expression for this is given as:

$$Trendability = min(|\, corrcoeffij|) \tag{4.7}$$

where $corrcoeffij$ is the correlation coefficients computed among all the training trajectories.

Hence, by performing the trendability analysis ten sensors are selected that has the same degradation pattern that are 2,3,4,7,11,12,15,17,20,21. These ten sensors were found useful for the calculation of health indicator. The degradation pattern of five useful sensors out of ten are shown in Fig. 4.4.

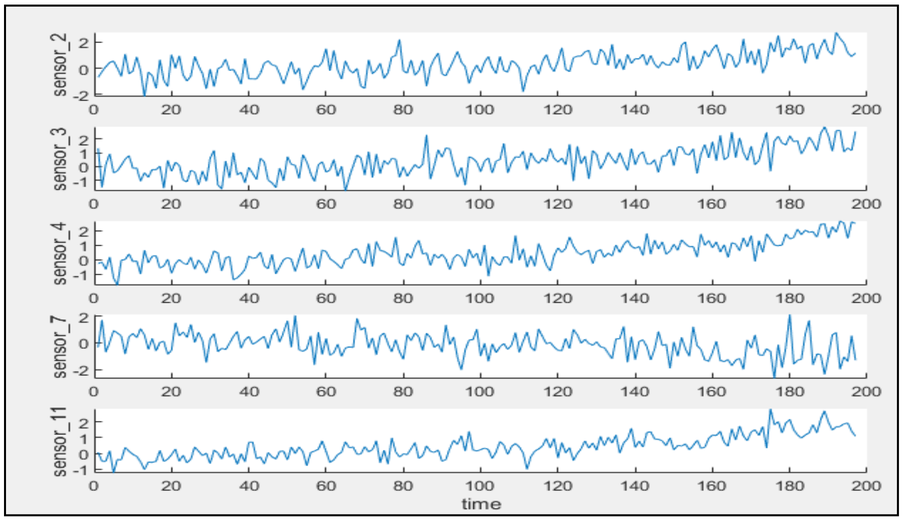

Fig. 4.4: Degradation pattern of 5 useful sensors

4.2.1 Health Indicator Assessment

The actual RUL values in C-MAPSS dataset is given only for test data hence it is important to label the training data correctly. Setting the RUL to be the form of (4.8) is an easy and convenient procedure [61]. The health equation can be written as:

$$h(t) = 1 + d - \exp\{a * t^b\} \tag{4.8}$$

where d is the non-zero initial degradation, index t is the number of cycles, a and b are the coefficients from which b is set to 1 as a default value and a can be written as (4.9) at the end of cycle when $h(t)$ is 0.

$$a = \log(1 + d)\,/t \tag{4.9}$$

The deterioration of an engine system, on the other hand, is negligible before the engine has been running for a while and the malfunction starts. As a result, the RUL of the engine system should start at a constant value and gradually decrease as the failure progresses. But how do you work out what RUL is worth in the early stages? It was estimated to be about 130 cycles in literature [17]. The reason for selecting 130 as the initial RUL, on the other hand, is seldom clarified. We suggested a method for selecting an acceptable initial RUL in this thesis. The coefficient a is tuned in such a way that at zero time cycle it gives an initial RUL value of that engine and at the end of the cycle it gives zero RUL. Fig. 4.5 below shows the RUL of engine no.1 which shows when the time cycle is zero the RUL value is 112 which is the initial RUL of engine no.1and it degrades as the time cycles increase and at the end of cycle or when the maximum threshold is reached the RUL is zero. This method is adopted for all 100 engines in FD001, and the constructed health indicator is further used as a target to train the algorithm.

For the Challenge dataset initial RUL values are yet not revealed. So, the above method cannot be applied for challenge dataset for health indicator assessment. Instead, a health indicator is constructed by fusing all 21 sensors. The health indicator is assumed to degrade over the time exponentially from 1 to 0. Hence the health situation at the beginning is assumed to be 1 and at failure it is assumed to be 0 [59]. Moreover, a fit linear regression model was constructed for the ten useful sensors as regressors. The fused health indicator for training data is shown in Fig. 4.6 which is further used as an output of neural network. This step was performed for both training and test trajectories.

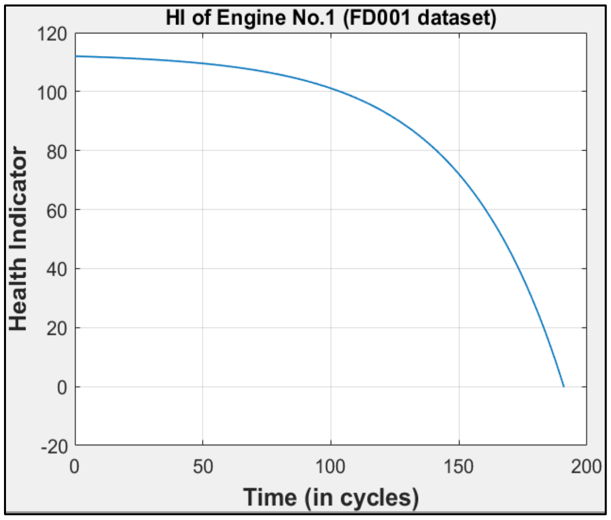

Fig. 4.5: Health Indicator of engine no.1 of FD001 dataset.

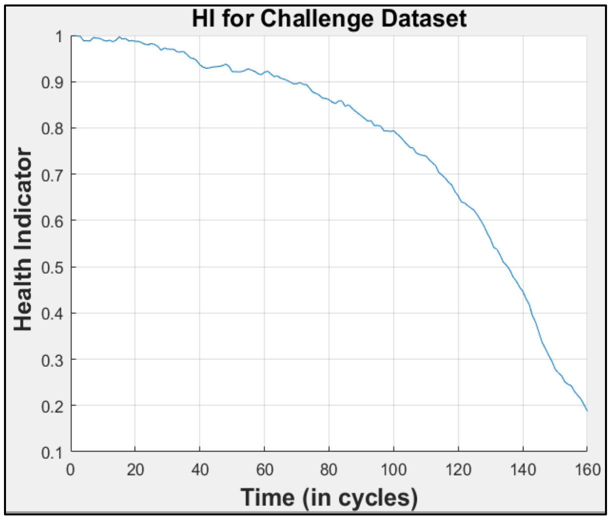

Fig. 4.6: Fused health indicator for Challenge dataset.

4.3 Deep Layer-Recurrent Neural Network

A Deep Layer-Recurrent Neural Network (DL-RNN) module uses within-layer recurrence to learn contextual knowledge adaptively. This approach is inspired by the ReNet architecture of Visin et al. [62]. A DL-RNN module is a combination of 1D RNNs that can learn contextual knowledge adaptively, with the successful receptive field able to reach across the entire function map or picture if necessary, for the task [62]. Recurrent neural networks with layers are analogous to feedforward networks, except that each layer has a recurrent relation associated with a tap delay. This enables the network to have an infinite dynamic response to input data from the time series [63]. In the training phase, DL-RNN used to consider the previous values. It consists of two parts recurrent layers and feedforward part. With just a few parameters, the DL-RNNs can learn long-range dependencies across a layer efficiently. Fully recurrent networks, recursive neural networks, Hopfield networks, Elman and Jordan neural networks, continuous-time RNNs, and bidirectional RNNs are few examples of DL-RNNs. The Fully recurrent network was established in the 1980s and is a network of neuron-like units. Each unit is linked to the others in a guided manner. Each connection has a real-valued weight that can be modified. Hopfield neural network is a form of recurrent artificial neural network that was first developed in 1982. John Hopfield, the inventor, was given the name. It has an important function in that it ensures that all its dynamics can converge to the local minimum. It does, however, often converge to a false local minimum. It is a unique neural network since it does not use sequences of patterns. Same sets of weights are recursively distributed over a structure in a recursive neural network, as the name suggests [64]. Elman neural network is distinguished by the presence of local memory and feedback connections. J.L Elman was the one who first suggested it in 1990. It is both a two-layer neural network and a back propagation neural network [65]. The basic DL-RNN network's architecture is shown in Fig. 4.7. It has an input layer denoted by $u(k)$, two hidden layers denoted by $v(k)$ and an output layer $y(k)$ [66]. In the DL-RNN, certain node's output is used as an input to the other nodes to construct an iterated feedback loop. The DL-RNN uses feedback data to recall the values from the previous step. Therefore, based on the previous and current input, a new output will be produced [66].

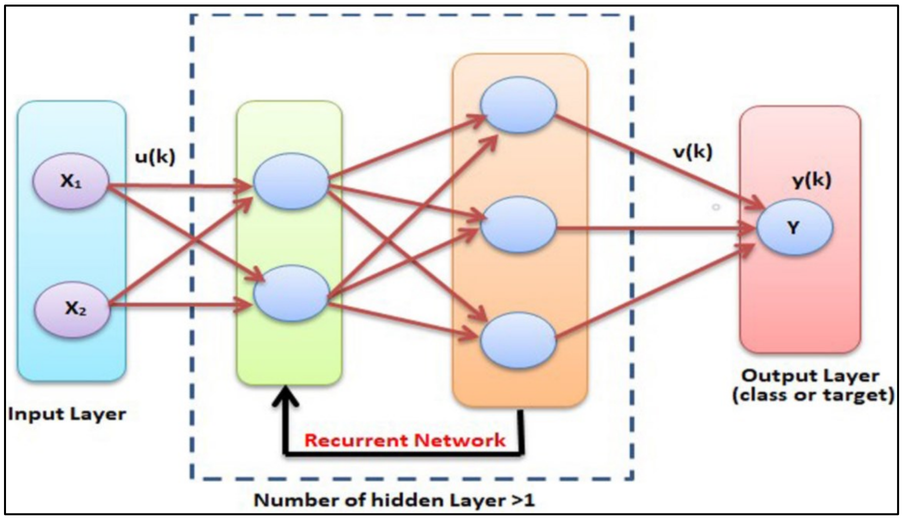

Fig. 4.7: Basic architecture of DL-RNN [66]

4.4 Neural Network Training

The first step in training the network is to configure the input-output series after the useful health features have been extracted from raw data. The dynamic data from 14 different variables of raw useful sensor data is expressed in the inputs as a “14*t” cell of a matrix for C-MAPSS dataset. The goal series is a matrix with a “1*t” cell array that only reflects “t” time steps of an output element, which is the measured HI from the health equation. This configuration's purpose is to provide a neural network mapping between a collection of raw numeric inputs and a set of measured health indicators. From the training set, only 70% of the training dataset is used for training, while 15% is used for validation, and another 15% for testing. The layer-recurrent neural network is designed in a triple loop design with an increasing number of hidden layers in the outer loop along with a default value of layer delays and scaled conjugate gradient is used as a training function. In short, the hidden layers are not set to a default number. During the neural network learning stage, the inputs are the raw sensors values for the same trajectory and the target is the HI constructed for the same trajectory. The same raw data is applied to the obtained network function to receive a neural network prediction, ensuring that the trained neural network can achieve high accuracy for an output estimate. As seen in Fig. 4.8, the results show a pattern that is very similar to the fitted HI i.e., target used in network training, but it was contaminated with noise which was filtered using smooth data the network was able to achieve good accuracy after 57 seconds of training. Both patterns have exponential curves and scales that are identical. As a result, it can be concluded that the network can efficiently process raw data and standardize outputs without needing all the dataset's trajectories. However, in the training of multilayer neural networks, due to overfitting and computational overheads, the network can result in poor network calculations. As a result, the validate network function must be used on new inputs of test trajectories to ensure that the algorithm can memorize the samples for the training data and correctly generalize the upcoming testing cases. The model was able to correctly describe the HI curve of the inserted data despite differences in duration and characteristic trajectory features (such as initial wear level and deterioration pattern) as seen from the Fig. 4.9.

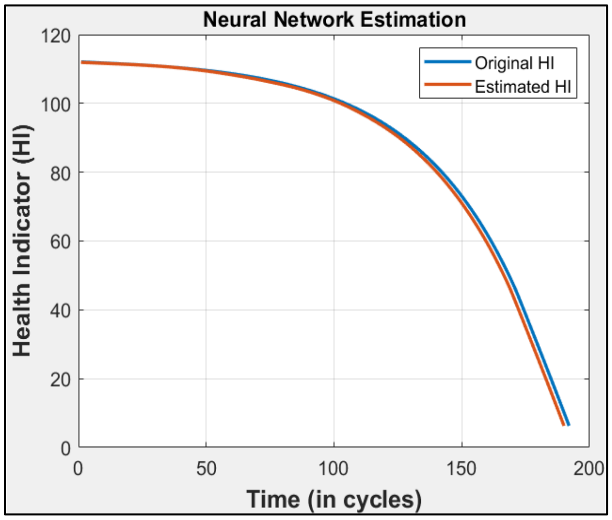

Fig. 4.8: Neural Network validation with the same input for FD001 dataset.

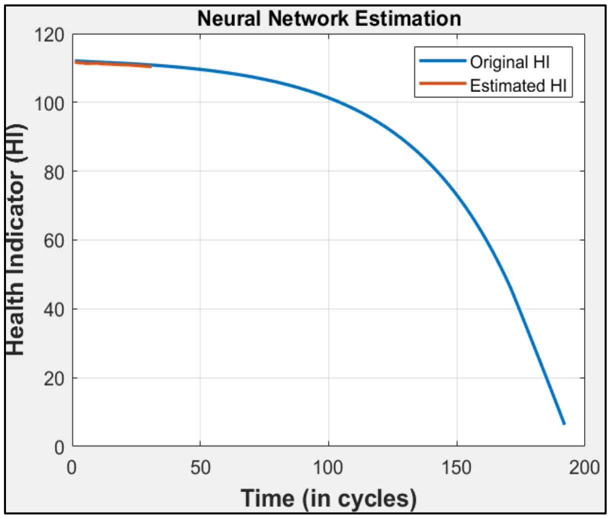

Fig. 4.9: Neural Network validation with the different input for FD001 dataset.

The training is performed on the entire training set of 100 engines. The raw test input shown in Fig. 4.9 has shorter instances.

For Challenge dataset, the training is performed with “10*t” cell of a matrix as an input and the target is a matrix with a “1*t” cell array that only reflects “t” time steps of an output element, which is the measured HI after multi regime normalization and function extraction. Both the inputs and a target are normalized using a Min-Max normalization to map in the range of [0,1]. The hidden layers and the layers’ delays are same as the one used for C-MAPSS dataset and the training function used is scaled conjugate gradient. During the neural network learning stage, the raw sensor values of same trajectory are taken as an input and HI value as a target of same trajectory. To obtain a neural network prediction, the same raw data is applied to the obtained network function, ensuring that the trained neural network can achieve high accuracy for an output estimate. As can be seen from the Fig. 4.10, a similar trend culminated in confirmation, but it was contaminated with noise for which smooth data was used to filter noise and the network was able to achieve good accuracy after 1 minute 19 seconds of training. Now to validate, the network function is used on new inputs of test trajectories to ensure that the algorithm can memorize the samples for the training data and correctly generalize the upcoming testing cases. Despite differences in length and characteristic trajectory features (such as initial wear level and deterioration pattern), the model was able to correctly describe the HI curve of the inserted data, as shown in the Fig. 4.11.

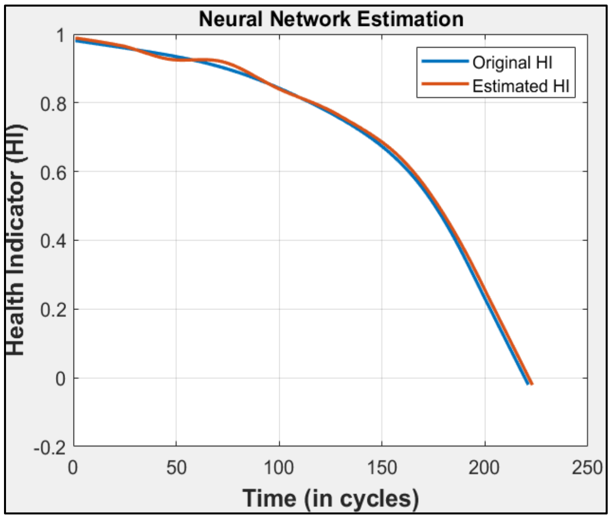

Fig. 4.10: Neural Network validation with the same input for Challenge dataset.

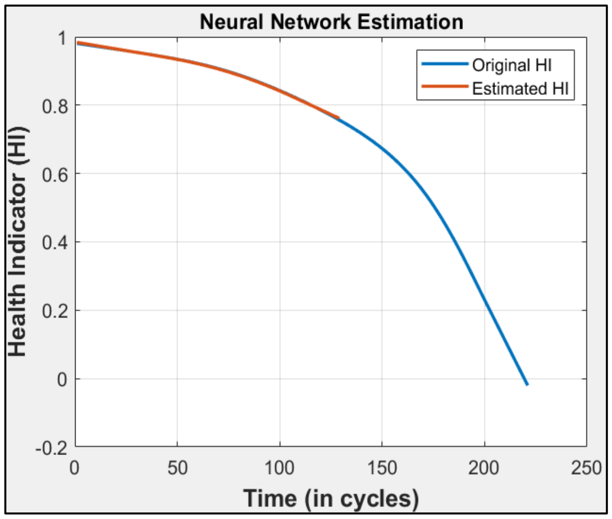

Fig. 4.11: Neural Network validation with the different input for Challenge dataset.

Chapter 5

5.1 RUL Prediction

The prognostic performance of the proposed method is discussed in this chapter. The trained network functions can be used to filter all HIs for both training and test trajectories within the same domain once the neural network has provided a competent generalization of the input-output relationship. The comparisons with other ensemble methods including Multi-Layer Perceptron (MLP), Nonlinear Autoregressive Network with Exogenous Inputs (NARX) and Cascade forward neural network (CFNN) are carried out to show the superiority of Deep Layer-Recurrent Neural Network (DL-RNN). All the experiments are conducted on a PC with Intel Core i5 CPU, 8-GB RAM and in the environment of MATLAB. For the FD001 dataset the plot of error and a table of comparison of different methods are shown below.

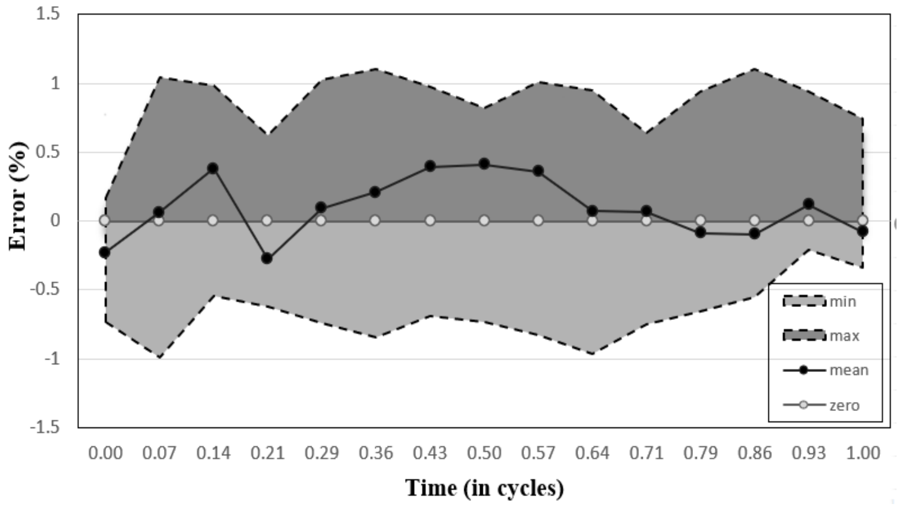

Fig. 5.1: The error plot showing the variation of the mean, maximum, and minimum values of DL-RNN (FD001 dataset).

Fig. 5.1 shows the randomly selected 15 series and plots their variation of the mean, maximum, and minimum values of Yt − Rt over time. Where Yt and Rt is the estimated and predicted RUL at a time point. As can be seen from the plot, the upper bound denotes the maximum error, lower bound denotes the minimum error and line in black line denotes mean of the error. It can be clearly seen that the error is within the acceptable range whereas for other three topologies the error plot is beyond the range. The error plot of one topology out of three i.e., multi-layer perceptron is shown in Fig. 5.2.

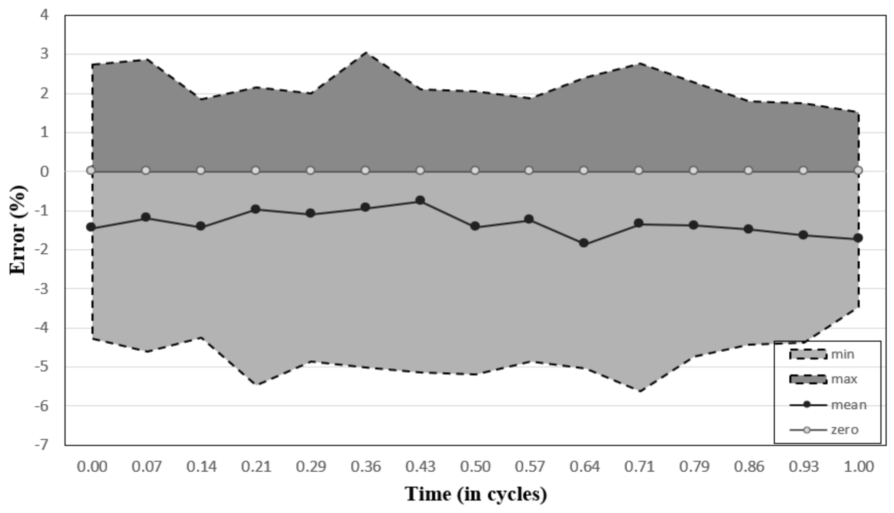

Fig. 5.2: The error plot showing the variation of the mean, maximum, and minimum values of MLP (FD001 dataset).

For the performance evaluation, the metrics and their formula shown in the table below are used.

Table 5.1: Prognostic Metrics

Metric	Formula	
Mean Absolute Error	MAE = $\frac{1}{n}\sum_{i=1}^{n}\lvert Yt - Rt\rvert$	
Mean Square error	MSE = $\frac{1}{n}\sum_{i=1}^{n}(Yt - Rt)^2$	Yt = Predicted RUL

Root Mean Square Error	RMSE= $\sqrt{\frac{\sum_{i=1}^{n}(Yt-Rt)^2}{n}}$	Rt= True RUL $n = number\ of\ trajectories$
Integral Square Error	ISE = $\int_0^\infty (Yt - Rt)^2 dt$	

In order to check whether the training and testing is performed properly, the Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) are used for performance evaluation and is calculated both for training and testing for all four topologies. The lower the values of MAE and RMSE, better the performance of a network. As can be seen from the table, MAE and RMSE of Layer-Recurrent Neural Network is lower compared to other three topologies. The performance of Cascade Feedforward Neural Network is poor compared to all other methods. We can note from Table 5.2 that best results were obtained using DL-RNN model.

Table 5.2: Performance comparison of different methods (FD001 dataset)

	Training		Testing	
	MAE (%)	RMSE (%)	MAE (%)	RMSE (%)
DL-RNN	0.174	0.199	0.177	0.121
NARX	0.771	1.04	0.775	1.08
MLP	1.724	2.54	1.754	2.295
CFNN	2.32	3.20	2.77	4.54

Experiments were conducted based on testing the four topologies separately using Levenberg-Marquardt (LM), Bayesian Regularization (BR) and Scaled Conjugate Gradient (SCG). Compared to the LM and BR, SCG takes less time to converge, and it provides better result as compared to other two training algorithms.

Table 5.3 shows the MSE and ISE calculation along with the number of iterations when trained with SCG. As can be seen from the table the MSE and ISE values are lower for DL-RNN compared to NARX, MLP and CFNN. The number of iterations for all four topologies is set to 18000.

Table 5.3: Results of testing (SCG algorithm, FD001 dataset)

Scaled Conjugate Gradient			
	Iterations	MSE (%)	ISE (%)
DL-RNN	18000	0.00002	0.0004
NARX	18000	0.0002	0.0038
MLP	18000	0.0814	0.0156
CFNN	18000	0.0155	0.081

For the Challenge dataset, the error plot for DL-RNN is shown below (Fig. 5.3). Similar to C-MAPSS dataset, the error range for this dataset is also within the acceptable bounds which proves that the accuracy of the model is good compared to other three topologies.

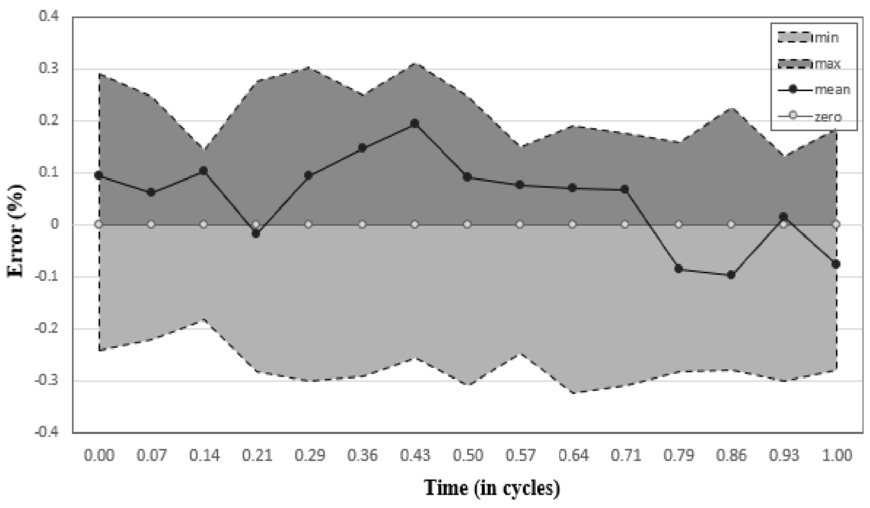

Fig. 5.3: The error plot showing the variation of the mean, maximum, and minimum values of DL-RNN (Challenge dataset).

For comparison, the error plot of NARX is shown below (Fig. 5.4). As can be seen from the plot, the minimum and maximum error is in the range of -2 and 2 whereas the minimum and maximum range of DL-RNN is within -0.3 to 0.3. This clearly shows the superiority of DL-RNN model to other three topologies.

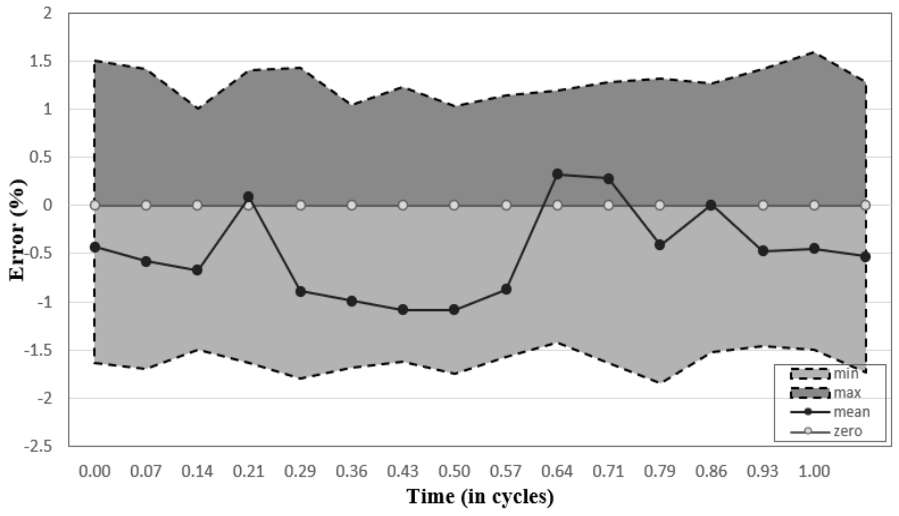

Fig. 5.4: The error plot showing the variation of the mean, maximum, and minimum values of NARX (Challenge dataset).

Same as C-MAPSS, the comparison table of training and testing are shown below. MSE and RMSE are evaluated for all four topologies and are presented in Table 5.4. As can be seen, the RMSE and MAE values of DL-RNN is lowest compared to other three. Therefore, it can be said that for both the datasets, DL-RNN topology outperformed other three topologies in terms of performance.

Table 5.4: Performance comparison of different methods (Challenge dataset)

	Training		Testing	
	MAE (%)	RMSE (%)	MAE (%)	RMSE (%)
DL-RNN	0.159	0.203	0.127	0.135
NARX	0.293	0.455	0.514	0.597
MLP	1.459	1.625	1.516	1.772
CFNN	1.44	2.39	1.36	2.45

Table 5.5 shows the comparison of SCG algorithm for all four methods. Using SCG algorithm the network gives better estimation result and it converge fast compared to other algorithms. The number of iterations is set to 18000 to train the algorithm.

Table 5.5: Results of testing (SCG algorithm, Challenge dataset)

Scaled Conjugate Gradient			
	Iterations	MSE (%)	ISE (%)
DL-RNN	18000	0.00008	0.0018
NARX	18000	0.0004	0.0095
MLP	18000	0.0036	0.0804
CFNN	18000	0.0053	0.1187

The remaining useful life of 100 testing engines is estimated after successfully training the model. The estimated RUL is compared with the actual RUL values and are shown in Fig. 5.5 below where the blue line indicates the actual RUL, and the red line is the predicted RUL. As can be seen, the expected values closely match the true values. Most of the prediction errors are within appropriate bounds.

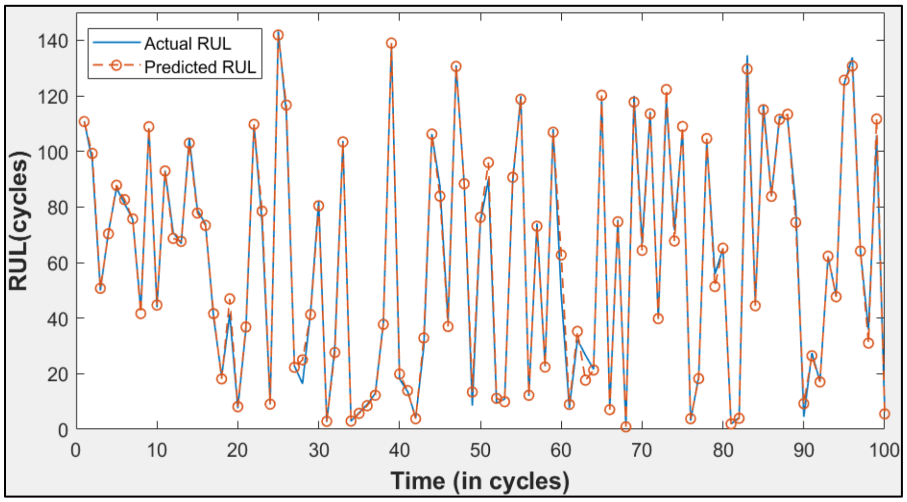

Fig. 5.5: RUL of 100 test engines

For better observation, the real RUL of 100 research engine instances is sorted from small to large. Fig. 5.6 shows the sorted real RUL outcomes in 100 testing engine cases, along with the estimated RUL. The RUL decreases as the horizontal axis value decreases. The expected value becomes increasingly reliable as the engine degrades, as seen in Fig. 5.6.

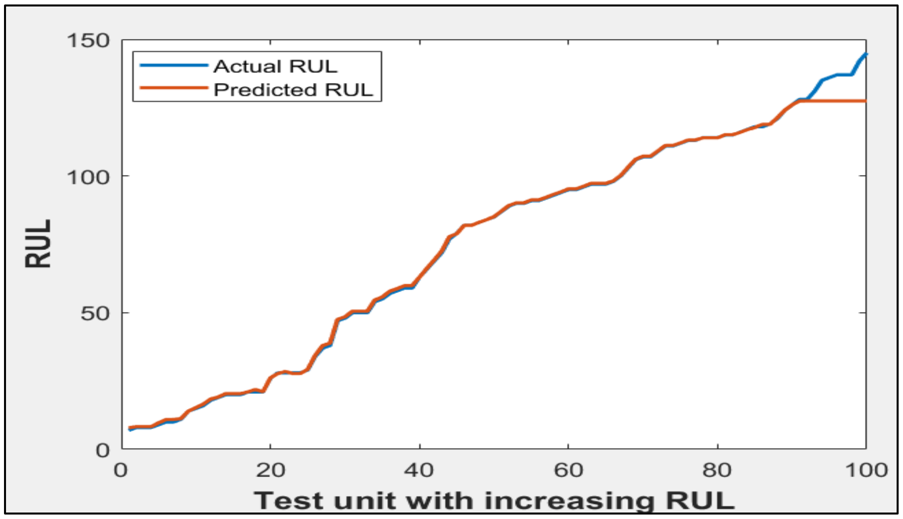

Fig. 5.6: Sorted prediction for 100 testing engine instances.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

Deep learning methods have gained popularity in engineering applications over the last decade, especially in data analysis for reliability evaluation, which was previously inefficient due to the requirement of expert expertise on the studied system, as well as the limitations of conventional PHM techniques. Many challenges still exist for reliability-related, data-driven applications in order to keep improving the estimation of health state parameters that can guarantee an accurate diagnosis for systems, facilities. This thesis presents a deep learning method for estimating the health state of complex systems using big machinery data. The framework is validated through two different datasets and consists of a novel topology layer-recurrent neural network to train the algorithm. The C-MAPSS dataset and the Challenge dataset are used to validate the proposed framework through the training and testing of different models. The proposed framework shows robustness for prognosis of remaining useful life of both datasets. For the C-MAPSS dataset, the training and testing is performed on all 100 engines of first subset. The average result obtained for both the metrics, MAE and RMSE were compared with other three topologies, where the proposed DL-RNN based model outperformed all three topologies. For the Challenge dataset, error plots of DL-RNN is presented along with three other topologies, where for the proposed topology the error is within the acceptable range whereas for other three topologies the error is beyond the acceptable range.

6.2 Future Work

The results presented and discussed above provide convincing evidence of the framework's ability to extract features and conduct sequential analysis on raw data. The system is adaptable to a variety of situations and produces reliable results. However, there are still some areas that could be improved. One of the key disadvantages of the topologies used in this thesis is that the training process is fully supervised. As a result, while the model can be directly applied to many problems, there are still many reliability-related phenomena where the required amount of labelled data to train a fully supervised model cannot be produced. Therefore, a next step for the proposed work would be to introduce an unsupervised data preprocessing technique capable of creating labels for the training data, such as clustering.

References

[1] L. Yao and Z. Ge, "Deep Learning of Semi supervised Process Data with Hierarchical Extreme Learning Machine and Soft Sensor Application," *IEEE Transactions on Industrial Electronics,* vol. 65, no. 2, pp. 1490-1498, Feb. 2018.

[2] O. Costilla-Reyes, P. Scully and K. B. Ozanyan, "Deep Neural Networks for Learning Spatio-Temporal Features from Tomography Sensors," *IEEE Transactions on Industrial Electronics,* vol. 65, no. 1, pp. 645-653, Jan. 2018.

[3] J. Pan, Y. Zi, J. Chen, Z. Zhou and B. Wang, "Lifting Net: A Novel Deep Learning Network with Layer wise Feature Learning from Noisy Mechanical Data for Fault Classification," *IEEE Transactions on Industrial Electronics,* vol. 65, no. 6, pp. 4973-4982, June. 2018.

[4] Goebel K, Saha B, Saxena A, "A comparison of three data driven techniques for prognostics," in *62nd Meeting of the Society For Machinery Failure Prevention Technology (MFPT)*, 2008.

[5] K. P. Murphy, Machine Learning: A Probabilistic Perspective, vol. 4, London: The MIT Press Cambridge, Massachusetts, 1991.

[6] Géron, A., Hands-On Machine Learning with Scikit-Learn and TensorFlow, Sebastopol: O'Reilly Media,Inc, 2017.

[7] Y. LeCun, B. Yoshua, and H. Geoffrey, "Deep learning," *Nature,* vol. 521, no. 2015, pp. 436-444, May. 2015.

[8] A. YoshuaBengio, Ian J. Goodfellow, Deep Learning, Cambridge: MIT Press, 2016.

[9] J. Deutsch and D. He, "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, Jan. 2018.

[10] Bishop, C.M., Neural networks for pattern recognition, New York: Oxford University Press, Inc., 1995.

[11] Kevin Gurney, An introduction to neural networks, London and New York: UCL Press Limited, 1997.

[12] B.G.M. Vandeginste, D.L. Massart, L.M.C. Buydens, S. De Jong, P.J. Lewi, J. Smeyers-Verbeke, "Chapter 44 - Artificial Neural Networks, Data Handling in Science and Technology," *Elsevier,* vol. 20, pp. 649-699, part 2,1998.

[13] M. Imran, S. A. Alsuhaibani, "A Neuro-Fuzzy Inference Model for Diabetic Retinopathy Classification," in Intelligent Data-Centric Systems, Intelligent Data Analysis for Biomedical Applications, Dammam: Academic Press, 2019, pp. 147-172.

[14] Uhrig, R. E., "Introduction to artificial neural networks," *Proceedings of IECON '95 - 21st Annual Conference on IEEE Industrial Electronics, Orlando, FL, USA,* vol. 1, pp. 33-37, 1995.

[15] Zhigang Tian, Lorna Wong, Nima Safaei, "A neural network approach for remaining useful life prediction utilizing both failure and suspension histories," *Mechanical Systems and Signal Processing,* vol. 24, no. 5, pp. 1542-1555, July. 2010.

[16] N. Budhani, C. K. Jha and S. K. Budhani, "Prediction of stock market using artificial neural network," in *International Conference of Soft Computing Techniques for Engineering and Technology (ICSCTET)*, Bhimtal, India, 2014.

[17] Heimes, F. O., "Recurrent neural networks for remaining useful life estimation," in *International Conference on Prognostics and Health Management,*, Denver, CO, USA, 2008.

[18] G. Lan, Q. Li and N. Cheng, "Remaining Useful Life Estimation of Turbofan Engine Using LSTM Neural Networks," in *IEEE CSAA Guidance, Navigation and Control Conference (CGNCC)*, Xiamen, China, 2018.

[19] M. Yuan, Y. Wu and L. Lin, "Fault diagnosis and remaining useful life estimation of aero engine using LSTM neural network," in *IEEE International Conference on Aircraft Utility Systems (AUS)*, Beijing, China, 2016.

[20] R. Wu and J. Ma, "An Improved LSTM Neural Network with Uncertainty to Predict Remaining Useful Life," in *CAA Symposium on Fault Detection, Supervision and Safety for Technical Processes (SAFEPROCESS)*, Xiamen, China, 2019.

[21] X. Li, Q. Ding, J. Q. Sun, "Remaining Useful Life Estimation in Prognostics Using Deep Convolution Neural Networks," *Reliability Engineering and System Safety,* vol. 172, pp. 1-11, 2018.

[22] Ribeiro de Miranda, T. M. G. de Andrade Barbosa, A. G. Scolari Conceição and S. G. Soares Alcalá, "Recurrent Neural Network Based on Statistical Recurrent Unit for Remaining Useful Life Estimation," in *8th Brazilian Conference on Intelligent Systems (BRACIS)*, Salvador, Brazil, 2019.

[23] Okoh C, Roy R, Mehnen J, et al., "Overview of Remaining Useful Life Prediction Techniques in Through-life Engineering Services," *Procedia CIRP,* vol. 16, pp. 158-163, 2014.

[24] Zaidan, M. A., *"Bayesian Approaches for Complex System Prognostics",* PhD thesis, University of Sheffield, 2014.

[25] Eker, O. F., Camci, F., and Jennions, I. K., "A similarity-based prognostics approach for remaining useful life prediction," in *2nd European Conference of the Prognostics and Health Management Society, Nantes, France*, France, July. 2014.

[26] H. Chaoui, M. Kandidayeni, L. Boulon, S. Kelouwani, and H. Gualous, "Real-Time Parameter Estimation of a Fuel Cell for Remaining Useful Life Assessment," *IEEE Transactions on Power Electronics, (in press),* vol. 36, no. 7, pp. 7470-7479, July. 2021.

[27] A. El Mejdoubi, H. Chaoui, J. Sabor, and H. Gualous, "Remaining Useful Life Prognosis of Supercapacitors Under Temperature and Voltage Aging Conditions," *IEEE Transactions on Industrial Electronics,* vol. 65, no. 5, pp. 4357-4367, May. 2018.

[28] Jianhui Luo, M. Namburu, K. Pattipati, Liu Qiao, M. Kawamoto and S. Chigusa, "Model-based prognostic techniques [maintenance applications]," in *Proceedings AUTOTESTCON 2003. IEEE Systems Readiness Technology Conference.*, Anaheim, CA, 2003.

[29] B. Lamoureux, J. Massé and N. Mechbal, "Improving Aircraft Engines Prognostics and Health Management via Anticipated Model-Based Validation of Health Indicator," *Prognostics Journal,* vol. 2, no. 1, pp. 18-38, 2014.

[30] Schwabacher, M., "A survey of data-driven prognostics," in *Proceedings of the AIAA Infotech@Aerospace Conference: American Institute for Aeronautics and Astronautics, Inc.*, Reston, VA, 2005.

[31] Schwabacher, M. and Goebel, K., “A Survey of Artificial Intelligence for Prognostics.”, AAAI Fall Symposium: Artificial Intelligence for Prognostics., 2007.

[32] Peng, Y., Dong, M., and Zuo, M. J., "Current status of machine prognostics in condition-based maintenance: a review," *The International Journal of Advanced Manufacturing*

Technology, vol. 50, pp. 297-313, 2010.

[33] H. Chaoui, and C. Ibe-Ekeocha, "State of Charge and State of Health Estimation for Lithium Batteries using Recurrent Neural Networks," *IEEE Transactions on Vehicular Technology,* vol. 66, no. 10, pp. 8773-8783, October. 2017.

[34] H. Chaoui, C. Ibe-Ekeocha, and H. Gualous, "Aging Prediction and State of Charge Estimation of a LiFePO4 Battery using Input Time-Delayed Neural Networks," *Electric Power Systems Research, Elsevier,* vol. 146, pp. 189-197, May. 2017.

[35] A. Mosallam, K. Medjaher and N. Zerhouni, "Data-driven prognostic method based on Bayesian approaches for direct remaining useful life prediction," *Journal of Intelligent Manufacturing,* vol. 27, no. 5, pp. 1037-1048, 2014.

[36] D. A. Tobon-Mejia, K. Medjaher, N. Zerhouni and G. Tripot, "A Data-Driven Failure Prognostics Method Based on Mixture of Gaussians Hidden Markov Models," *IEEE Transactions on Reliability,* vol. 61, no. 2, pp. 491-503, June. 2012.

[37] Y. Song, C. Yang, T. Wang, D. Liu and Y. Peng, "Hybrid approach of iterative updating for lithium-ion battery remaining useful life estimation," in *Prognostics and System Health Management Conference (PHM-Chengdu)*, Chengdu, China, 2016.

[38] B. Saha, K. Goebel, and J. Christophersen, "Comparison of prognostic algorithms for estimating remaining useful life of batteries," *Transactions of the Institute of Measurement & Control,* vol. 31, pp. 293-308, June. 2009.

[39] A. El Mejdoubi, A. Oukaour, H. Chaoui, Y. Slamani, J. Sabor, and H. Gualous, "Online Supercapacitor Diagnosis for Electric Vehicle Applications," *IEEE Transactions on Vehicular Technology,* vol. 65, no. 6, pp. 4241-4252, June. 2016.

[40] V. Fornlöf, D. Galar, A. Syberfeldt, T. Almgren, M. Catelani and L. Ciani, "Maintenance, prognostics and diagnostics approaches for aircraft engines," in *IEEE Metrology for Aerospace (MetroAeroSpace)*, Florence, Italy, 2016.

[41] McClung, M. Enright and R., "A Probabilistic Framework for Gas Turbine Engine Materials With Multiple Types of Anomalies," *Journal of Engineering for Gas Turbines and Power,* vol. 133, no. 8, 2011.

[42] NAEEM, M, "Implications of engine deterioration for operational effectiveness of a military aircraft," *Applied Energy,* vol. 60, no. 3, pp. 115-152, 1998.

[43] K. Goebel and A. Saxena , "Turbofan Engine Degradation Simulation Dataset," *NASA Ames Prognostics Data Repository,* 2008.

[44] D. Frederick, DeCastro, and Litt, "User's Guide for the Commercial Modular Aero-Propulsion System Simulation (C-MAPSS)," *NASA/ARL,* 2007.

[45] Saxena A, Goebel K, "PHM08 challenge dataset.," *NASA Ames Prognostics Data Repository, NASA Ames Research Center,* 2008.

[46] F. Smach, "Design of a Neural Networks Classifier for Face Detection," *Journal of Computer Science,* vol. 2, no. 3, pp. 257-260, 2006.

[47] B. Zhang, D. Wang, W. Song, S. Zhang and S. Lin, "An Interval-Valued Prediction Method for Remaining Useful Life of Aero Engine," in *2020 39th Chinese Control Conference (CCC)*, Shenyang, China, 2020.

[48] A. Tatli, S. Kahveciolu, "NARX neural networks-based time series prediction for amount of airworthiness time, in electrical, electronics and biomedical engineering (Elec)," *Elektronika Ir Elektrotechnika,* vol. 26, no. 5, pp. 28-32, 2020.

[49] H. Tang. H. Xie, Y. H. Liao, "Time series prediction based on NARX neural networks: an advanced approach, in: machine learning and cybernetics," in *2009 International Conference on Machine Learning and Cybernetics*, Baoding, China, 2009.

[50] S. Chen, S. A. Billings, and P. M. Grant, "Non-linear system identification using neural networks," *International Journal of Control,* vol. 51, no. 6, pp. 1191-1214, 1990.

[51] Parthasarathy, K. S. Narendra and K., "Identification and control of dynamical systems using neural networks," *IEEE Transactions on Neural Networks,* vol. 1, no. 1, pp. 4-27, 1990.

[52] S. M. Zainorzuli, S. Afzal Che Abdullah, R. Adnan and F. A. Ruslan, "Comparative Study of Elman Neural Network (ENN) and Neural Network Autoregressive with Exogenous Input (NARX) For Flood Forecasting," in *IEEE 9th Symposium on Computer Applications & Industrial Electronics (ISCAIE)*, Malaysia, 2019.

[53] Serackis, M. Tamulionis and A., "Comparison of Multi-Layer Perceptron and Cascade Feed-Forward Neural Network for Head-Related Transfer Function Interpolation," in *Open Conference of Electrical, Electronic and Information Sciences*, Vilnius, Lithuan, 2019.

[54] B. Warsito, R. Santoso, Suparti and H. Yasin, "Cascade Forward Neural Network for Time Series Prediction," in *Journal of Physics: Conference Series*, Semarang, Indonesia, 2017.

[55] Diallo, O. N., "A data analytics approach to gas turbine prognostics and health management.," 2010.

[56] C. Zhang, P. Lim, A. K. Qin, K. C. Tan., "Multiobjective Deep Belief Networks Ensemble for Remaining Useful Life Estimation in Prognostics," *IEEE Transactions on Neural Networks and Learning Systems,* vol. 28, no. 10, pp. 1-13, Oct. 2017.

[57] P. Lim, C. K. Goh, K. C. Tan, "A Time Window Neural Network Based Framework for

Remaining Useful Life Estimation," in *2016 International Joint Conference on Neural Networks (IJCNN)*, Vancouver, BC, Canada, 2016.

[58] S. Zheng, k. Ristovski, A. Farahat, C. Gupta, "Long Short-Term Memory Network for Remaining Useful Life Estimation," in *2017 IEEE International Conference on Prognostics and Health Management (ICPHM)*, Dallas, TX, USA, 2017.

[59] T. Wang, Jianbo Yu, D. Siegel and J. Lee, "A similarity-based prognostics approach for Remaining Useful Life estimation of engineered systems," in *2008 International Conference on Prognostics and Health Management*, Denver, CO, USA, 2008.

[60] Coble, J. and Hines, J. W., "Identifying optimal prognostic parameters from data: a genetic algorithms approach," in *In Annual conference of the prognostics and health management society*, San diego, September. 2009.

[61] A. Saxena, K. Goebel, D. Simon and N. Eklund, "Damage propagation modeling for aircraft engine run-to-failure simulation," in *2008 International Conference on Prognostics and Health Management*, Denver, 2008.

[62] G. Li, Z. Yan and J. Wang, "A one-layer recurrent neural network for constrained nonsmoothed invex optimization," *Neural networks : the official journal of the International Neural Network Society,* vol. 50, pp. 79-89, 2014.

[63] X. Zhang, Y. Dong, L. Wen, F. Lu and W. Li, "Remaining Useful Life Estimation Based on a New Convolutional and Recurrent Neural Network," in *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*, Vancouver, BC, Canada, 2019.

[64] M. Kobayashi, "Noise Robust Projection Rule for Hyperbolic Hopfield Neural Networks," *IEEE Transactions on Neural Networks and Learning Systems,* vol. 31, no. 1, pp. 352-356,

Jan. 2020.

[65] Guanghua Ren, Yuting Cao, Shiping Wen, Tingwen Huang, Zhigang Zeng, "A modified Elman neural network with a new learning rate scheme," *Neurocomputing,* vol. 286, pp. 11-18, 2018.

[66] H. Turabieh, A. A. Salem, N. Abu-El-Rub, "Dynamic L-RNN recovery of missing data in IoMT applications," *Future Generation Computer Systems,* vol. 89, pp. 573-583, Dec. 2018.