

A STUDY ON DEEP LEARNING FOR PROGNOSTICS AND HEALTH MANAGEMENT APPLICATIONS:  
AN EVOLUTIONARY CONVOLUTIONAL LONG SHORT-TERM MEMORY DEEP NEURAL NETWORK  
DATA-DRIVEN MODEL FOR PROGNOSTICS OF AIRCRAFT GAS TURBINE

A Dissertation  
Submitted to the Graduate Faculty  
of the  
North Dakota State University  
of Agriculture and Applied Science

By  
Phattara Khumprom

In Partial Fulfillment of the Requirements  
for the Degree of  
DOCTOR OF PHILOSOPHY

Major Department:  
Industrial and Manufacturing Engineering

April 2022

Fargo, North Dakota

North Dakota State University  
Graduate School

---

**Title**

A Study on Deep Learning for Prognostics and Health Management Applications:  
An Evolutionary Convolutional Long Short-Term Memory Deep Neural Network  
Data-Driven Model for Prognostics of Aircraft Gas Turbine

---

**By**

Phattara Khumprom

---

The Supervisory Committee certifies that this *disquisition* complies with North  
Dakota State University's regulations and meets the accepted standards for the  
degree of

**DOCTOR OF PHILOSOPHY**

SUPERVISORY COMMITTEE:

Dr. David Grewell

---

Chair

Dr. Lokesh Karthik Narayanan

---

Dr. Chad Ulven

---

Dr. Supavich Pengnate

---

Approved:

April 12, 2022

---

Date

Dr. David Grewell

---

Department Chair

## ABSTRACT

The fundamental concept of prognostics and health management (PHM) within the scope of Condition-Based Maintenance (CBM) is to find an approach to evaluate the system health and predict its remaining useful life (RUL). Many methods and algorithms have been proposed for PHM modeling, most of which have been proven to perform relatively well. One of the leading algorithms in the current data-driven technology era is a deep learning approach, which is based on the concept of multiple hidden layers in a neural network. RUL prediction is an important part of PHM, which is the science that is aimed at increasing the reliability of the system and, in turn, reducing the maintenance cost and potential failure. The majority of the PHM models proposed during the past few years have shown a significant increase in the number systems that are data-driven. While more complex data-driven models are often associated with higher accuracy, there is a corresponding need to reduce model complexity. One possible approach is to reduce the complexity of the model is to use the features (attributes or variables) selection and dimensionality reduction methods before the model training process. In this work, the effectiveness of multiple search-based methods that seek for the best features set to perform model training, which included, filter and wrapper feature selection methods (correlation analysis, relief forward/backward selection, and others), along with Principal Component Analysis (PCA) as a dimensionality reduction method, was investigated. A basic algorithm of deep learning, Feedforward Artificial Neural Network (FFNN), was used as a benchmark modeling algorithm. It is believed that all of those approaches can also be applied to the prognostics of an aircraft engine. The aircraft engine data from NASA Ames prognostics data repository was used to test the effectiveness of the filter and wrapper feature selection methods. The findings show that applying feature selection methods helps to improve overall model accuracy by 3% to 5% compared to other existing works and significantly reduces the complexity by using 7 out of 21 less input nodes for the deep learning type of models.

## ACKNOWLEDGMENTS

Throughout my Ph.D. journey, there had been several changes and adjustments. There were both things I had to learn and unlearn through the years, but it was the support and guidance from many people that made all the difference. I could not have overcome many difficulties if it were not because of them.

Firstly, I definitely could not make it without the kind support and understanding from my advisor, Dr. David Grewell. I would like to express my appreciation and gratitude to him and to all my Ph.D. committee members. I would also like to thank the Industrial and Manufacturing Engineering department at NDSU for providing resources, opportunities, and flexibilities during the time that I attended the program. Much appreciation for the NDSU Graduate School for their help along the process and answers to all questions throughout my studies. I also would like to thank Dr. Nita Yodo for her guidance during my early years at North Dakota State University.

Additionally, I would like to extend my thanks and gratitude to my master's degree advisor at Robert Morris University, Dr. Sushil Acharya, for his sincerity and dedication in providing all the help I needed during my first journey in the U.S. His kindness and words are what I will always hold onto wholeheartedly. I wish to also thank my NDSU colleagues, especially Dr. Alex Davila Frias for his friendship and mentorship during our final years in the Ph.D. program. I appreciated the times that we hung out in his last year at NDSU.

Next, I would like to thank all my family members for their great patience and for trying their best to be understanding of the path I chose. Especially my aunt, Ms. Patchareeya Coomprom, the biggest sponsor who always stepped in whenever financial support was needed, not only for myself but for the whole family over the years, I am tremendously grateful for her help.

Lastly, many thanks go to my best friends, Mr. Kittiwate Dechrungruang and Mr. Narupot Piampanya, for bringing laughter and joy to my life and never criticizing whatever I do. Many seemingly trivial conversations with them actually meant a lot during my difficult times. I wish to also thank Dr. Sansiri Tampradab. Her inspiration and counsel both personally and academically are the crucial part of my success. I certainly cannot make it this far without her. I thank her for always being there every step of the way and staying with me through all my journeys.

## **DEDICATION**

To my parents and grandparents.

Thank you for all their life lessons that I have always brought along with me through my journey.

# TABLE OF CONTENTS

ABSTRACT .....	iii
ACKNOWLEDGEMENTS .....	iv
DEDICATION .....	v
LIST OF TABLES .....	ix
LIST OF FIGURES.....	x
LIST OF ABBREVIATIONS.....	xii
LIST OF APPENDIX TABLES.....	xv
LIST OF APPENDIX FIGURES .....	xvi
1. INTRODUCTION.....	1
1.1. Background .....	1
1.1.1. PHM Definition .....	2
1.1.2. Deep Learning Algorithms .....	10
1.1.3. Performance Measurement Metrics for PHM .....	15
1.1.4. Initial Framework for PHM Data Modeling Using Deep Learning .....	18
1.2. Research Objective and Contribution .....	19
1.2.1. Objectives and Tasks.....	20
1.2.2. Deliverables .....	21
1.2.3. Contributions .....	22
1.3. Chapter Summary .....	22
2. A SURVEY OF DEEP LEARNING APPROACH FOR PROGNOSTICS AND HEALTH MANAGEMENT APPLICATIONS .....	23
2.1. Deep Learning Paradigm in PHM Tasks.....	23
2.2. Research Application of PHM Domain Using Deep Learning Algorithms .....	25
2.2.1. PHM Models Using DNN .....	26
2.2.2. PHM Models Using CNN .....	27
2.2.3. PHM Models Using RNN-LSTM.....	28
2.2.4. PHM Models Using Hybrid Deep Learning Layers .....	30

2.3. Some of the Challenges of Deploying Deep Learning for PHM Applications .....	31
2.3.1. Data Challenges .....	31
2.3.2. Uncertainty .....	35
2.3.3. Difficulty to Train Models.....	36
2.3.4. The Proposed Framework for Deploying Deep Learning in the PHM Domain.....	38
2.4. Chapter Summary .....	38
<b>3. A DATA-DRIVEN PREDICTIVE PROGNOSTICS MODEL FOR LITHIUM-ION BATTERIES BASED ON A DEEP LEARNING ALGORITHM .....</b>	<b>39</b>
3.1. Data-Driven Prognostics Approach for Lithium-ion Battery .....	39
3.1.1. Overview of Data-Driven Prognostics.....	39
3.1.2. Prognostics of the Lithium-ion Battery .....	41
3.2. Data-Driven Predictive Prognostics Model for Lithium-ion Batteries Based on A Deep Learning Algorithm .....	45
3.2.1. Related Works.....	45
3.2.2. Deployment of Deep Learning for Prognostics Model of Lithium-ion Battery Data .....	46
3.2.3. The Proposed Deep Neural Network Prognostics Model for Lithium-ion Battery.....	47
3.2.4. Results for Model's SoH Estimation .....	48
3.2.5. Results for Model's RUL Estimation .....	53
3.3. Result Discussion.....	55
3.4. Chapter Summary .....	56
<b>4. DEEP NEURAL NETWORK FEATURE SELECTION APPROACHES FOR DATA-DRIVEN PROGNOSTIC MODEL OF AIRCRAFT ENGINES.....</b>	<b>58</b>
4.1. Deep Neural Feature Selection Approach for Modeling RUL Prediction of Aircraft Engines Data.....	58
4.1.1. Feature Selection Methods for Neural Network Architectures.....	59
4.1.2. C-MAPSS Aircraft Engines Data.....	61
4.1.3. Related Works.....	66
4.2. Methodology.....	67
4.2.1. Problem Statement .....	68
4.2.2. Deep Neural Network Architecture .....	68

4.3. Experiment and Result.....	72
4.3.1. Training Procedure and Hyperparameters Selection .....	72
4.3.2. Experiment Setup and Results .....	73
4.4. Result Discussion .....	77
4.5. Chapter Summary .....	87
5. AN EVOLUTIONARY CONVOLUTIONAL LONG SHORT-TERM MEMORY DEEP NEURAL NETWORK DATA-DRIVEN MODEL FOR PROGNOSTICS OF AIRCRAFT GAS TURBINE .....	88
5.1. Hybrid Deep Neural Network Layers Approach for Modeling RUL Prediction of Aircraft Engines Data .....	88
5.2. Methodology.....	89
5.2.1. Problem Statement .....	89
5.2.2. Sliding Time Window Processing.....	90
5.2.3. Defined Convolutional Neural Network.....	91
5.2.4. Defined Long Short-Term Memory Recurrent Neural Network .....	93
5.2.5. CNN-LSTM Hybrid Architecture.....	96
5.3. Experiment and Result.....	97
5.3.1. Evolutionary CNN Model.....	98
5.3.2. Hybrid Evolutionary CNN-LSTM Model .....	100
5.4. Result Discussion.....	106
5.5. Chapter Summary .....	107
6. GENERAL SUMMARY.....	108
REFERENCES.....	110
APPENDIX .....	128
A1. Adaptive Moment Estimation.....	128
A2. Pearson Correlation Matrix .....	129
A3. Principle Components Matrix.....	130
A4. Evolutionary DNN Model Description .....	131



## LIST OF TABLES

<u>Table</u>	<u>Page</u>
1. PHM performance metrics. ....	16
2. Summary of prognostics application research using deep learning. ....	32
3. Difference between data-driven and physics-based models for PHM. ....	40
4. RMSE results of each stacked hidden layer model. ....	48
5. RMSE results of 10 trials for a model with three stacked hidden layers. ....	48
6. RMSE of the SoH estimation by using DNN and traditional machine learning algorithms. ....	50
7. Models created from the lithium-ion battery training dataset. ....	52
8. The error of RUL estimation by using DNN and traditional machine learning algorithms. ....	53
9. C-MAPSS dataset description [38]. ....	61
10. Hyperparameters values evaluated in the proposed DNN model for C-MAPSS data. ....	73
11. C-MAPSS attribute values from different filter methods. ....	76
12. Best RMSE and prediction score results of RUL prediction from all DNN models. ....	77
13. Mean RMSE from all DNN models. ....	79
14. The best DNN Models for FD002 test data. ....	84
15. The notation of the symbol in the LSTM memory cell. ....	94
16. Hyperparameters values evaluated in the proposed CNN model. ....	99
17. RMSE and prediction score for RUL prediction from different CNN configurations. ....	99
18. Hyperparameters values evaluated in the proposed hybrid CNN-LSTM model. ....	101
19. RMSE and prediction score for RUL prediction from different CNN-LSTM configurations. ....	101

## LIST OF FIGURES

<u>Figure</u>	<u>Page</u>
1. Roles of diagnostics and prognostics scheme in the maintenance scenarios [22]. .....	4
2. The phase of prognostics and its transition from diagnostics [22]. .....	5
3. Illustration of some important prognostics time definitions and concepts [27]. .....	8
4. Trajectory prediction that can be modified corresponding to the estimation of RUL [27]. .....	9
5. Comparison of RUL predictions from ground truth [27]. .....	10
6. Example of standard Artificial Neural Network structure. ....	11
7. Convolutional layers. ....	14
8. Unfolded Recurrent Neural Network [34]. .....	14
9. The process of the prognostics framework using machine learning in general [48]. .....	20
10. Tasks in PHM [50]. .....	23
11. Proposed prognostics framework using a deep learning algorithm. ....	37
12. The schematic diagram of the tested battery. ....	42
13. The current and voltage during the discharging and charging of battery No. 05. ....	43
14. Predicted state of health of battery No. 05. ....	44
15. Dropout in deep neural network model. ....	47
16. The proposed Deep Neural Network model for lithium-ion battery data. ....	50
17. The SoH estimation with all algorithms for battery No. 06, 07, and 18. ....	51
18. The RUL estimation of battery No. 05 using different learning algorithms. ....	54
19. Role of feature extraction and feature selection in the prognostics modeling process. ....	61
20. Engine and sensor points and engine parts modules connections [172]. ....	63
21. Example of sensor signals (NRc and Ps30) and all feature descriptions. ....	63
22. Example of before and after z-normalization. ....	63
23. RUL curve of all testing engines FD002 and FD004. ....	64
24. Example of RUL curve of one testing engine FD002 and FD004. ....	65
25. Auto-encoder Deep Neural Networks construction. ....	68
26. The proposed Deep Neural Networks model architecture for C-MAPSS data. ....	71

27.	Validation result using evolutionary selection from C-MAPSS data. ....	75
28.	All RUL prediction curves for FD002.....	80
29.	All RUL prediction curves for FD004.....	81
30.	RUL prediction points for one engine of FD002 test data.....	82
31.	RUL prediction points for one engine of FD004 test data.....	83
32.	RMSE fluctuation for FD002 and FD004 test data. ....	85
33.	Prediction error distributions from feature selection methods.....	86
34.	The defined Convolutional Neural Network architecture. ....	91
35.	LSTM memory cell [190].....	94
36.	The defined Long Short-Term Memory Network architecture.....	95
37.	The proposed evolutionary hybrid CNN-LSTM deep neural network model architecture. ....	97
38.	Evolutionary CNN RUL prediction curves for FD002 test data.....	102
39.	Evolutionary CNN RUL prediction curves for FD004 test data. ....	103
40.	Hybrid evolutionary CNN-LSTM RUL prediction curves for FD002 test data. ....	104
41.	Hybrid evolutionary CNN-LSTM RUL prediction curves for FD004 test data. ....	105

## LIST OF ABBREVIATIONS

PHM .....	Prognostics and Health Management.
CMS .....	Condition Monitoring System.
RUL .....	Remaining Useful Life.
ANN.....	Artificial Neural Network.
FFNN.....	Feedforward Artificial Neural Network
SVM.....	Support Vector Machine.
LoR.....	Logistic Regression.
EOL .....	End-Of-Life.
EOP .....	End-Of-Prediction.
UTT .....	Unit Under Test.
DNN.....	Deep Neural Network.
CNN.....	Convolutional Neural Network.
RNN.....	Recurrent Neural Network.
LSTM.....	Long Short-Term Memory network.
ROC .....	Receiver Operating Characteristic.
ROI.....	Return Of Investment.
FP .....	False Positives.
FN.....	False Negatives.
MAPE .....	Mean Absolute Percentage Error.
ACC.....	Anomaly Correlation Coefficient.
MSE.....	Mean Squared Error.
MAE.....	Mean Absolute Error.
RMSE .....	Root Mean Squared Error.
RMSPE.....	Root Mean Squared Percentage Error.
SSD .....	Sample Standard Deviation
AD .....	Mean Absolute Deviation from the sample median
MAD .....	Median Absolute Deviation from the sample median.

CRISP-DM.....	Cross-Industry Standard Process for Data Mining.
AI .....	Artificial Intelligence.
GA .....	Genetic Algorithm.
DBN.....	Deep Belief Networks.
MLP .....	Multi-Layer Perceptron.
SVR .....	Support Vector Regression.
RVR.....	Relevance Vector Regression.
PCoE .....	NASA Ames Prognostics Center of Excellence.
C-MAPSS .....	Commercial Modular Aero-Propulsion System Simulation.
PSO.....	Particle Swarm Optimization.
MDSVC .....	Multimodal Deep Support Vector Classification.
GDBM.....	Gaussian-Bernoulli Deep Boltzmann Machine.
DRFF.....	Deep Random Forest Fusion.
MHMS .....	Machine Health Monitoring Systems.
CBLSTM.....	Convolutional Bi-directional Long Short-Term Memory networks.
GRU .....	Gated Recurrent Units.
BiGRU .....	Bidirectional Gated Recurrent Units.
LFGRU .....	Local Featured-based Gated Recurrent Units.
CNNHMMs .....	Convolutional Neural Network-based Hidden Markov Models.
t-SNE.....	t-distributed Stochastic Neighbor Embedding.
CNC.....	Computer Numerical Control.
MAR .....	Missing at Random.
ARMA.....	Auto-Regressive Moving Average.
AANN .....	Auto-Associative Neural Networks.
ELM.....	Extreme Learning Machine.
ESNs .....	Echo-State Networks.

MODE..... Multi-Objective Differential Evolution.

SaNSDE ..... Self-adaptive Differential Evolution with Neighborhood Search.

TOPSIS ..... Technique for Order of Preference by Similarity to Ideal Solution.

SoC ..... State of Charge.

SoH ..... State of Health.

EIS..... Electrochemical Impedance Spectroscopy.

AIC ..... Akaike Information Criterion.

LR ..... Linear Regression.

k-NN ..... k-Nearest Neighbors algorithm.

GHPF ..... Gauss-Hermite Particle Filter.

ReLU ..... Rectified Linear Unit.

tanh ..... Hyperbolic Tangent.

**LIST OF APPENDIX TABLES**

<u>Table</u>		<u>Page</u>
A1.	The Pearson correlation matrix (for C-MAPSS dataset).....	129
A2.	The Principle Components (PC) matrix (for C-MAPSS dataset). ....	130

## LIST OF APPENDIX FIGURES

<u>Figure</u>	<u>Page</u>
A1. The proposed evolutionary DNN model description (for C-MAPSS dataset). .....	131



# 1. INTRODUCTION

## 1.1. Background

In an era of information technology, data are being generated, collected, and accumulated across all fields at an incredible rate. To extract useful knowledge of these rapidly growing volumes of data, computational tools that can analyze big data are needed. This requirement has led to developments in data mining, for various fields of applications.

Data mining is the process of discovering and extracting a pattern from data [1], which generally involves characterization, generalization, classification, clustering, association, pattern matching, and visualization of large quantities of data [2]. The area of Prognostics and Health Management (PHM) is included as a part of a Condition Monitoring System (CMS) systems, which usually collects massive data from equipment during system operations. This large collection of data in PHM makes it beneficial to employ data mining approaches. Additionally, the prognostics domain also involves forecasting and predicting failure precursors to determine the time span of a system until it reaches the system's end-of-life. The operational life of a system until its end-of-life in PHM is usually known as the Remaining Useful Life (RUL) of the system, which is one of the key indicators of the system's health status.

Currently, there are challenges in predicting RUL using traditional approaches, such as the traditional regression model or statistical analysis approach. First, the traditional approaches are incapable of accurately predicting the RUL when having little or no prior knowledge of the system's physical behaviors. Second, the traditional approaches often fail to analyze the system that has complex or multiple fault conditions and/or features. Third, the accuracy of traditional methods, which provided the prediction accuracy around 45% to 50%, is still unsatisfactory. And fourth, is the lack of general performance metrics for PHM that can be used in a standardized manner in comparing multiple PHM models. The data mining methodology is grounded by multi-disciplines such as probability [3], statistics [4], machine learning [5], and artificial intelligence (AI) [6]. However, of these, the data mining disciplines that are believed to address the major challenges in PHM are machine learning and artificial intelligence algorithms. The traditional machine learning methods also called shallow learning models, such as an artificial neural network (ANN) [7], support vector machine (SVM) [8], logistic regression (LoR) [9], and others. Though these traditional methods have already been shown to have great performance over the

years, they still fail to overcome those challenges in estimating RUL. This opens an opportunity for a modern machine learning approach namely, deep learning, to be adopted in PHM applications.

The early concept of deep learning was first initiated by Geoffrey Hinton in 2006 as the product of improving the higher dimensionality of an existing ANN [10]. Deep learning has the potential to overcome many shortcomings of the shallow learning methods, as it uses multiple non-linear transformation functions to capture more complex representative information from the raw data [11]. Deep learning approaches also have been proven to successfully construct good prediction models that use smaller knowledge of past behaviors and yet are still able to provide an acceptable level of prediction accuracy [12]. The only major challenge that possibly remains as an area of development is the standardized metric for evaluating the performance of employing deep learning in PHM, which will also be addressed in this dissertation as well as machine learning and deep learning approach in PHM applications.

### **1.1.1. PHM Definition**

This section describes the development of PHM ideology. Some predefined terms and definitions of PHM will be discussed to establish a fundamental understanding of the PHM area and its predecessor methodology as part of the literature survey. This section also provides initial links between PHM and deep learning approaches that will be discussed in the later parts of this thesis.

#### **1.1.1.1. General Ideology of PHM**

Activities of the system's health analysis are essential to support the critical decision-making process in engineering systems. Most of the engineering systems are composed of complex, interconnected, multiple components and materials that must be maintained and/or replaced within a certain operating time. To maintain the operation of a system, the run to failure scenario of the system should be avoided, in most cases. One of the solutions to this scenario is by performing maintenance of the system's components while the operation is still running. This maintenance scheme not only guarantees that the operation is performing properly, but also has a great economic impact in terms of reducing the operation, production, and support costs. Diagnostics usually plays a role in identifying the types of a particular failure that might occur, while prognostics is used to evaluate the health state of the system. The operational time of the system's components is usually called time until end-of-life. As the main focus of this dissertation is regarding prognostics of the system, the term EOL, RUL, and system

health projection or prediction will occur multiple times. As the root of prognostics came from aspects of system diagnostics, some parts and applications of diagnostics that relate to prognostics will also be mentioned and discussed in this introduction part.

The term PHM was introduced and developed as a method to enable proactive maintenance decisions that involved not only monitoring the health condition, but also predicting the RUL of the system [13-17]. PHM technology can transform the passive system's reliability into adaptive while also reducing the system's life cycle cost [18, 19]. This makes prognostics, a growing field, as one of the core components within CBM as part of the CMS of critical systems. The prognostics of a system are defined by the ability to evaluate and estimate the RUL of the system. Traditionally, the prognostics method for the prognostics system focuses on only the accuracy of RUL estimation for certain applications [20]. However, after many studies have been employed by PHM over the years, some other challenges are to be found in PHM as well. This is because prognostics is not fully developed compared to diagnostics. Additionally, prognostics studies have put more effort into developing and employing new methods to fulfill only requirements of particular end-users [21], rather than evaluating and comparing performances among those methods from wide perspectives. This leaves gaps for new methods as well as precise standard performance metrics to be deployed in PHM. Note that in this dissertation, the terms prognostics and PHM might be used interchangeably from time to time.

To illustrate the importance of CBM, a proposed system's operation maintenance procedure is seed as in Figure 1. Diagnostics and prognostics are essentially involved in CBM. As mentioned previously, diagnostics is the process of identification of faults or failures, whereas prognostics is the process of identifying health states and making a prediction of the time to failure in the system [22]. The time left before observing a failure is described as the RUL of the system [23].

The prognostics process also involves two phases as shown in Figure 2, fault prognostic and diagnostics. The current system's health state is assessed and evaluated in the first phase. This phase of prognostics overlaps with diagnostics. Again, diagnostics and prognostics are both critical in system health analysis, and in many cases, do not distinguish from each other. Classification or clustering techniques are utilized within this phase for pattern recognition. This is the part that data mining techniques, such as machine learning, and deep learning can be employed. The goal of the second

phase is to predict the time to failure of a system or a component by identifying, forecasting, or projecting the RUL. Projection trends, tracking techniques, and time series analysis are utilized in this phase.

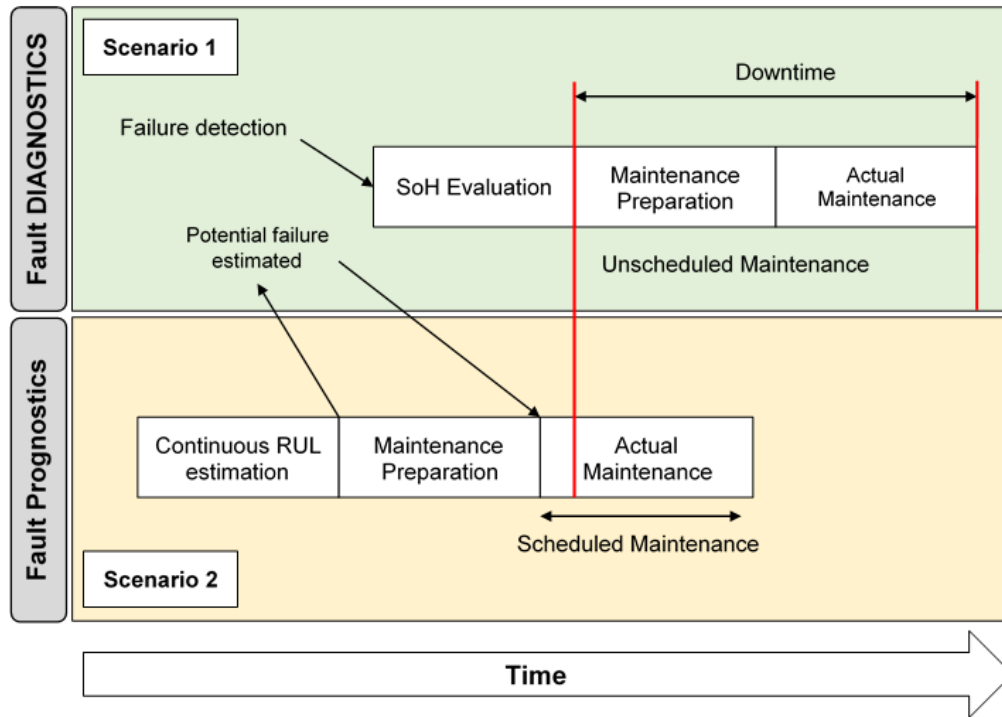


Figure 1. Roles of diagnostics and prognostics scheme in the maintenance scenarios [22].

In addition to the phase and roles of prognostics, modeling techniques must be established to analyze the health state of the system and RUL. There are two types of modeling techniques generally used for prognostics data. The first is, the data-driven model, and the second is, the physics-based model. To construct a data driven model, requires sufficient run to failure samples data from the system, whereas the physics-based model considers the physics of failure progression in the system must be well understood. Some detailed definitions of data-driven and physics-based models for prognostics analysis are as described next.

The data-driven model aims to model system behavior using collections of censoring data instead of relying on system physicality or domain experts that understand the fundamentals of the systems [24]. Data-driven approaches are generally categorized into two types. The first one is from the statistical approach and the second one is from the machine learning approach. The models using statistical approaches are generally built upon a probabilistic and deterministic method from the available data. While the goal of machine learning approaches is to recognize and capture complex patterns based on

historical data. The deep learning technique that will be the main discussion of the dissertation falls into the machine learning approach.

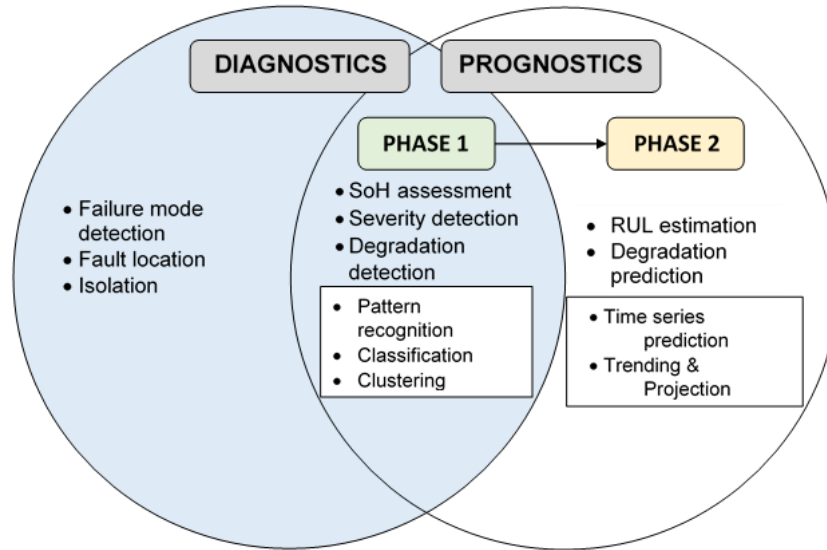


Figure 2. The phase of prognostics and its transition from diagnostics [22].

The physics-based model requires knowledge or understanding of the system’s fundamental physics to generate a model that can estimate the system’s RUL. The degradation or run to failure data does not play an important role in the physics-based model. Therefore, in this case, the physical rules within the system must be substantially known beforehand, which means, the domain expert is essentially required to generate the physics-based model. The early step in a physics-based model is to exploit the known relationship of physical parameters that represent the health state of the systems [25].

In physics-based prognostics models, two challenges must be considered: 1) the physics of degradation of some systems may be very difficult to determine or in many cases can be unknown, and 2) the value of parameters in such a system might be impossible to obtain or evaluate. A physics-based model requires vast information and a deep understanding of the failure mechanism of the system as well as a high level of quality control. Therefore, skilled-well-knowledge personnel of particular systems or subject matters is essentially required to construct and determine the physics-based model [26]. These challenges of the physics-based model led to the development of a data-driven model that will come to replace the physics-based model in the future.

One of the data-driven models approaches for prognostics that will be the main discussion is machine learning approaches. The advancement of computational tools has largely impacted the

development of this approach. The complex machine learning algorithm requires capable computational power. Because of the better computational power of a modern computing machine, the machine learning algorithms have been proven to be able to empirically learn and recognize more complex patterns of the system's data. In most cases, it is believed it will replace the existing physics-based model. Deep learning is a machine learning algorithm that has been proposed and proven to outperform other traditional learning algorithms. Deep learning algorithms will be discussed more in detail in further sections. While some terms and definitions of prognostics must be described to have a clear understanding of PHM before any deeper discussion in the next section.

#### **1.1.1.2. Prognostics Terms and Definitions**

All commonly used terms in prognostics found in the literature are described in this section. Many similar terms have been interchangeably used within the PHM research community. As well as in some cases, the same terms have been referred to different notions. Thus, this list aims to help in clarifying some discrepancies that might have been caused by some non-standardized use cases outside of the PHM community [27].

##### Assumptions

- Prognostics can detect the failure precursors and predict RUL. The prediction of RUL is heavily based on the current state of health and the future operating conditions of the system.
- A health index is defined as the identification of the health state of a system. Health index can be considered because of the aggregated from a system's features and conditions.
- RUL estimation is a forecasting, prediction, and extrapolation procedure.
- For comparison, the employed algorithms can generate a single RUL value for each prediction. Algorithms that produce RUL distributions can also be compressed to one estimated number.
- All systems usually stay under continuous monitoring activities as part of the Condition Monitoring System and have the capability to measure and acquire data from the system as the system's fault evolves.

### Common Terminologies

- EOL: End-Of-Life, is the time index of the actual end of life.
- RUL: Remaining Useful Life, is the amount of time left before system health drops under the failure threshold.
- EOP: End-Of-Prediction, is the earliest time index,  $i$ , when the prediction has reached the defined failure threshold.
- UUT: Unit Under Test.
- $l$ , is the index for time instant  $t_l$ .
- $O$ , is the time index for the time of the birth of the system,  $t_0$ .
- $F$ , is the time index for the time when a fault starts to occur,  $t_F$ .
- $D$ , represents the time index when the diagnostics system detects the fault within the system,  $t_D$ .
- $P$ , represents the time index when the prognostics system makes the first prediction,  $t_P$ .
- $f_n^l(i)$ , is the value of the  $n^{th}$  feature for the  $l^{th}$  UUT at time index  $i$ .
- $c_n^l(i)$ , is the value of the  $n^{th}$  operational condition for the  $l^{th}$  UUT at time index  $i$ .
- $r_n^l(i)$ , is the RUL estimation at time  $t_i$ , when given that the data is available up to time  $t_i$  for the  $l^{th}$  UUT.
- $\pi^l(i | j)$ , is the prediction at time index  $i$  for the given data up to time  $t_j$  for the  $l^{th}$  UUT. Prediction can be made in any domain.
- $\Pi^l(i)$ , represents the trajectory of predictions at time index  $i$  for the  $l^{th}$  UUT.
- $h^l(i)$ , represents the health of a system for the  $l^{th}$  UUT.

Figure 3 illustrates the common time terminologies used in prognostics. These terms will be constantly used for the rest of the paper. The red-dotted line represents the actual behavior or the prognostics system while the blue line represents the prediction value from the prognostics model.

### Terms Definitions

- Time index: In a prognostics application, time can be either discrete or continuous. Somehow, time index  $i$  is used instead of the actual time, likewise, considering time index as a discrete measurement. This is useful in cases when the sampling time does not fall into the same pattern. Time indexes describe in this paper also do not depend on or vary to time scales.

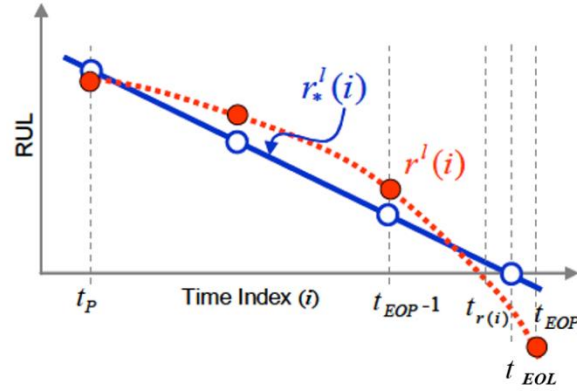


Figure 3. Illustration of some important prognostics time definitions and concepts [27].

- Health index:  $h^l(i)$  is a health index at the time  $i$  for UUT, when  $l = 1, 2, \dots, L$ . Normalized aggregate of health indicators in operational conditions is also considered as  $h$ .
- Time of detection of fault:  $D$  is the time index ( $t_D$ ) when faults are detected in the system. This triggers the algorithm to begin performing RUL predictions as soon as enough data is collected from the system's fault state. For the applications, such as battery PHM, the prognosis is employed as a degradation or decaying process. The faulty state may not be detected. For this case,  $t_D$  might be considered to be equal to  $t_0$ .
- Time to start prediction: Normally, there is a difference between the time when the system starts making a prediction ( $t_P$ ) and the time when a fault state is detected ( $t_D$ ). Generally,  $t_P \geq t_D$  as the algorithms required some time to fine-tune and adjust with additional input data before starting to predict RUL. However, for some cases that data is collected in the system continuously before faults are detected. There might be enough available data to start predicting RUL anytime, hence  $t_P = t_D$ .
- Prognostics features: Given that  $f_n^l(i)$  is a feature at the time  $i$ , where  $n = 1, 2, \dots, N$  is the feature number, and  $l = 1, 2, \dots, L$  is the UUT index. In most of the prognostics domains, features usually change their behaviors over time. Features can take many forms such as system parameters, system attributes, component variables, or other quantities that can measure, calculate, or compute from any aspects of the system's operating conditions that relate to the system's prognostics. Features can also be referred to as a feature vector  $F(i)$  of the  $l$ th UUT at the time  $i$ .



- Operational conditions: Given that  $c^l_m(i)$  is one of the operational conditions at the time  $i$ , where  $m = 1, 2, \dots, M$  is the condition number, and  $l = 1, 2, \dots, L$  is the UUT index. The load can also be referred to as one of the operational conditions, which can also be referred to as a vector  $C^l(i)$  of the  $l$ th UUT at the time  $i$ .
- Point prediction: Given that  $\pi^l(i | j)$  is a point of prediction at the time  $i$ . Normally, the information is provided at until time  $t_j$ , where  $t_j \leq t_i$ . This case  $\pi_l(i | j)$  for  $i = EOL$ , is a health indicator at a critical threshold. In some cases, the first step is to extrapolate the features, then, aggregate those features to calculate the health status of the system. For other cases, the features are aggregated to the health state to predict RUL.
- Trajectory prediction:  $\Pi_l(i)$  is the trajectory of RUL predictions at time  $i$ .  $\Pi_l(i) = \{\pi_l(i | i), \pi_l(i+1 | i), \dots, \pi_l(EOL | i)\}$
- RUL estimation: Given that  $r^l(i)$  be the remaining useful life estimation at the time  $i$ . The system's conditions and system's features are provided until the time  $i$  assume that the future operational profile of the system is available. As illustrated in Figure 5, the prediction has drawn at time  $t_i$  and it predicts the RUL from the given information until the time  $i$ . For the  $l$ th UUT, RUL will be predicted as  $r^l = \arg\{h(z) = 0\} - i$ . Note that the subscript of the star symbol (\*) indicates the best belief to be the true value of the system variable. This condition is described as Ground Truth in the literature [27].

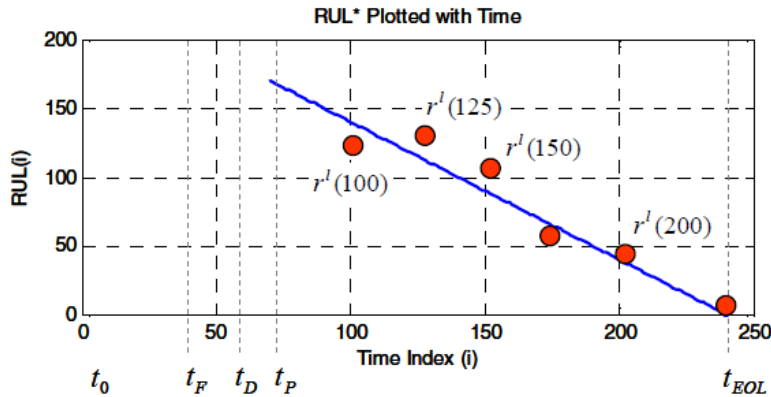


Figure 4. Trajectory prediction that can be modified corresponding to the estimation of RUL [27].

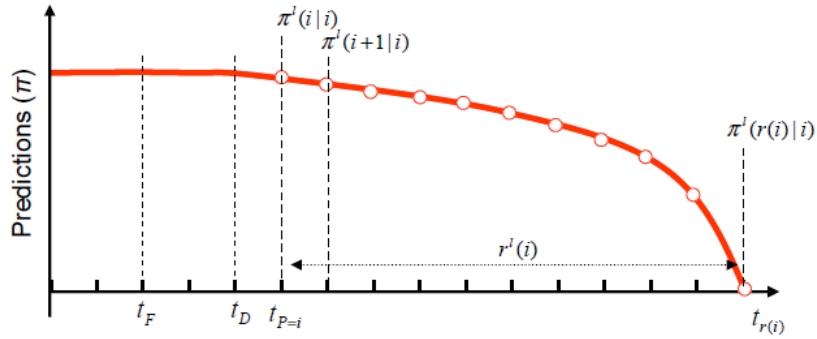


Figure 5. Comparison of RUL predictions from ground truth [27].  
When,  $t_p [70, 240]$ ,  $t_{EOL} = 240$ ,  $t_{EOP} > 240$ .

### 1.1.2. Deep Learning Algorithms

The reason that the machine learning approach works well with the prognostics data, in general, is because it is possible to collect the massive data when performing system's condition monitoring. Although there are multiple machine learning algorithms that have been used in the PHM model over the years, the main focus of this thesis is the concept of deep learning which will be discussed in this section. As the deep learning concept was developed based on the Artificial Neural Network [7, 28-31], the first discussion in this section will be an initial description of ANN. After that, the deep learning algorithms and how they apply to prognostics prediction will be discussed.

#### 1.1.2.1. Artificial Neural Networks

An ANN or a "neural network", is a computational model inspired by the structural and functional aspects of biological neural networks [7]. A single neural network, or a perceptron, has an interconnected group of artificial neurons, which processes computational information through inter-connections approaches from node to node. An ANN is an adaptive system, which means, it can change its connections based on the different information during the learning phase. There are two configuration modes in ANN. The first configuration mode is the feed-forward, and the second is the backpropagation. For the feed-forward network, the connections between the units or nodes do not form a completed back-and-forth cycle. Instead, the information in the network moves only one way forward from the input units, through the hidden units, to the output units. While backpropagation moves the information backward to update weights and connections in the network.

Backpropagation is a supervised learning method that has two phases, propagation phase and weight update phase [28]. These two phases are repeatedly performed until the performance requirement of the model is fulfilled. In backpropagation algorithms, the output values from the network are compared to the actual or correct value through the calculation of the error-function value. This error-function value is fed back through the network as a reference to make an appropriate adjustment of the weights of each connection. The goal is to reduce the value of the error function by selecting proper weights. This process is repeatedly performed in the training cycle until the condition is satisfied. Usually, the network will converge to a certain state in which the calculated error is sufficiently small. This scenario can be considered as if the network can learn a certain target function.

A sigmoid function is usually used as the activation function in ANN. The activation function is basically the function that ‘activate’ the learning capability of the neural network. However, other activation functions can be implemented in ANN, for example, linear or identity function, binary step functions, hyperbolic tangent function, sigmoid function. A sigmoid function is a type of Gaussian spheroid function, expressed as follows:

$$y(x) = e^{-\left(\frac{\|x-c\|^2}{2\sigma^2}\right)} \tag{1}$$

The output of the hidden neuron gives a measure of distance between the input vector  $x$  and the centroid  $c$  of the data cluster. The  $\sigma$  parameter represents the radius of the hypersphere, which is normally determined by using the iterative process of selecting the optimum width. The general structure of the ANN is as described in Figure 6.

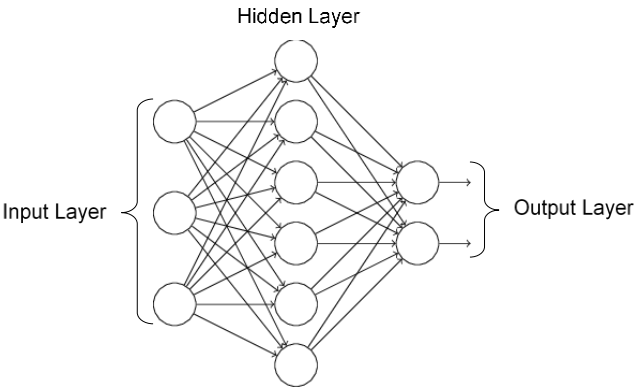


Figure 6. Example of standard Artificial Neural Network structure.

In addition to the activation function of the neural network, another condition that needs to be considered to construct the classification model is the learning or training algorithm of the neural network. A learning algorithm is a systematic step-by-step procedure through which the connection weights among neurons are adjusted to minimize the difference between the predicted and actual values of an output variable [29]. This adjustment was performed in this study using the most popular method of training, known as the back-propagation learning algorithm. In addition to its broad employments in various applications, the backpropagation learning algorithm is more efficient than other learning algorithms for solving most of the regression problems [30].

There are three advantages to the backpropagation learning algorithm. First, this learning algorithm is straight forward and easy to program. Second, the backpropagation learning algorithm can provide reasonably accurate results for complicated applications in which the input and output relationships are nonlinear [31]. Finally, and most importantly, the backpropagation learning algorithm has revealed an acceptable level of generalization ability. The performance of neural networks trained by the backpropagation learning algorithm is usually controlled by mainly two parameters, namely, learning rate and momentum. At the start of the learning process, the learning rate will be varied in a range between 0 and 1. The learning rate is a parameter that affects how connection weights within a network are updated. These updates also include a portion of the last weight change to accelerate the training convergence and improve the training precision. This portion is defined by the second key parameter of the learning algorithm, which is, momentum. Momentum is variant of the stochastic gradient descent that helps the weight update to avoid getting stuck in local minima. This consequently helps to accelerate the learning speed. Similar to the learning rate, the momentum will generally be varied in a range of 0–1. A specific rule that determines the best values for the learning rate and momentum has not yet been proposed in the literature. One of the common practices is by examining different values from 0–0.9 with a constant step size of 0.1. Equation 2 and 3 describe the process of momentum and weight update in the neural network. Equation 2 is the initialize step of identifying weight. While equation 3 shows the effect of the momentum in the process. Where,  $\Delta W_{ij}$  represents weight increment and  $\Delta W_{ij}^{t-1}$  represents the weight increment from the previous iteration,  $\eta$  is the learning rate,  $\frac{\partial Y}{\partial w_{ij}}$  is the weight gradient varied by the output  $Y$ , and  $\gamma$  represents momentum.

$$\Delta W_{ij} = \eta \times \frac{\partial Y}{\partial w_{ij}} \quad (2)$$

$$\Delta W_{ij} = \left( \eta \times \frac{\partial Y}{\partial w_{ij}} \right) + \left( \gamma \times \Delta W_{ij}^{t-1} \right) \quad (3)$$

### 1.1.2.2. Overview of Deep Learning Concept

The concept of deep learning suggested by Geoffrey Hinton in 2006 has gained interest by academia and industry [10]. Deep learning is based on ANN with a strong power of representation, which holds the potential to overcome the deficiencies in traditional intelligent methods [32]. The representation in neural network means the ability or how well that the output of network can mimic or 'represent' the pattern of the actual data. The prominent advantage of deep learning is being able to capture the information from raw data through multiple non-linear transformations and approximate complex non-linear functions.

The main algorithms of deep learning include the Deep Neural Network (DNN), the Convolutional Neural Network (CNN), the Recurrent Neural Network (RNN), the expansion of CNN and RNN, such as Long Short-Term Memory network (LSTM), and the hybrid network which is the combination of different type of stacked layers [11]. The following are the characteristics of each deep learning algorithm:

A Deep Neural Network is generally a stack of multiple hidden layers instead of only one hidden layer in the standard ANN architecture. The DNN hidden layers are the multiple feed-forward layers that are trained with back-propagation stochastic gradient descent (SGD). The hidden layers consist of neurons nodes with hyperbolic tangent activation function (tanh), rectified linear unit activation function (ReLU), and maxout function. DNN has additional parameters to the vanilla ANN, such as adaptive learning rate, rate annealing, dropout, and regularization. These parameters can be fine-tuned and believed to enable higher predictive accuracy than the vanilla ANN.

Convolutional Neural Network is layers of convolution function consisting of neurons with tanh, ReLU applied to the results. CNN uses convolutions over the input layer to compute the output. An individual layer of CNN applies different types of filters. The edges of layers capture the shape of data and then use these shapes to deter higher-level features. The last layer classifies the output using these high-level features. The general idea of convolutional layers in CNN is as shown in Figure 7.

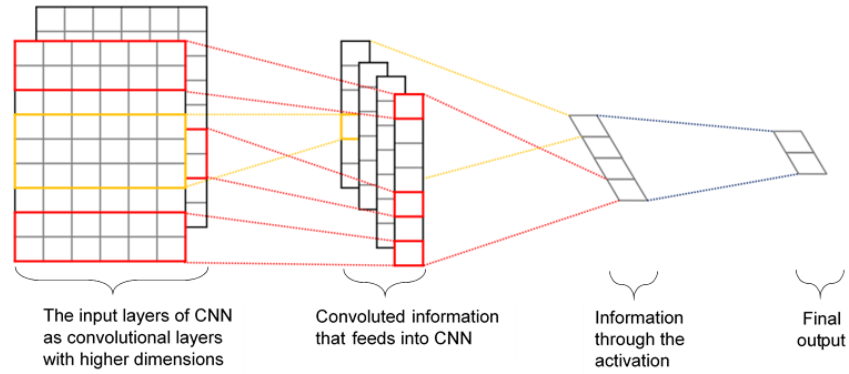


Figure 7. Convolutional layers.

Recurrent Neural Network makes use of sequential information. RNN defines input and output as dependent variables based on a time sequence. RNN performs the same task for every element of a single sequence, with output at the end of the last time step depending on the previous computations. RNN may consider having “memory” as it can capture information about the calculation in the past sequence. However, RNN has a limitation in capturing the length of data. This leads to the development of the LSTM network which can capture the longer sequence of information [33]. Figure 8 shows an RNN being or unfolded into a full network [34]. The formulas that are used for RNN computation are as follows:

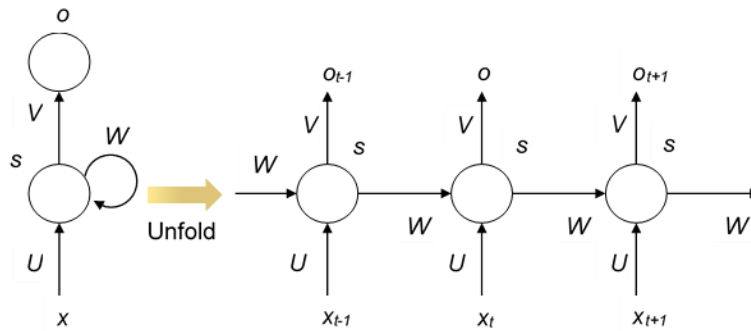


Figure 8. Unfolded Recurrent Neural Network [34].

- $x_t$ : input at the time step  $t$ .
- $s_t$ : hidden state at the time step  $t$ . This might be considered as ‘memory’ of the network.  $s_t$  can be calculated based on the previous hidden state and the input of the current step:  $s_t = f(Ux_t + Ws_{t-1})$ . The function  $f$  usually is normally non-linear such as tanh or ReLU.
- $s_{-1}$ : an initial hidden state that is required for the first hidden state typically initialized to zero.
- $o_n$ : the output at step  $n$

### 1.1.3. Performance Measurement Metrics for PHM

Many PHM performance metrics have been reported in the literature. A particular metric was employed based on different prognostics application domains and algorithms that were used to construct the model as described in Table 1. Part of the dissertation will only focus on what is called “algorithmic performance metrics” as machine learning and deep learning approaches are based on this type of performance evaluator [27]. There are mainly three categories of algorithmic performance metrics found in the literature, 1) accuracy-based metrics, 2) precision-based metrics, and 3) robustness-based metrics. The majority of the literature employed accuracy-based and precision-based metrics. Several cases used robustness-based metrics but there is a limited formal definition found for robustness-based metrics, which are Brier Score, Receiver Operating Characteristic (ROC), and sensitivity, as described in Table 1. However, the robustness metrics may lead to the measurement of model stability against the other metrics. The measurement of robustness and stability of the PHM model is needed to be explored in the future because there are limited works that have performed a detailed study on the subject.

Generally, algorithmic performance can be evaluated by measuring the difference of errors between the actual and predicted RUL. Other performance characteristics such as statistical, convergence, moment, etc., can also use errors to quantify. However, one of the most important notions is that the errors can be calculated if there is actual data available, which might not be the case. When actual data is unavailable, historical, or past data may be generated to draw corresponding inferences. However, this is only valid for the case when assuming that the current process similar distribution compared to past data.

The list and descriptions of the three types of metrics are included in Table 1. It was found that trajectory prediction metrics were not explicitly well defined in much literature. Some literature discussed other types of metrics such as, “similarity measure” [20] and “prediction behavior error” [35]. These metrics might be possibly adapted as trajectory prediction metrics. However there is no clear discussion regarding trajectory prediction performance evaluations found [27].

Machine learning and deep learning approaches are heavily computational. It was suggested that some computational metrics such as complexity [36], specificity [37], as well as computational time and memory space, might need to be measured and included in the final results. The popular term to measure

computational performance is called “Big O” [38]. This notion describes the amount of time algorithm needed to run the relative size of the input data or function.

In addition to PHM metrics related to the deep learning approach, other PHM performance metrics that might be beneficial for further discussion as well is, “cost-benefit metrics” [39-41]. This group of metrics focuses on how operation cost can be reduced if RUL can be predicted beforehand and how the accuracy of RUL prediction impacts the cost to operate the system. The measurements are mostly found to inform of return of investment (ROI) per operation. More details can be found in references work by D. L. Goodman, S. Vohnout, and S. M. Wood [39-41]. Table 1 summarized most of the PHM matrices having been found in the literature that well suited the deep learning approach and use cases.

Table 1. PHM performance metrics.

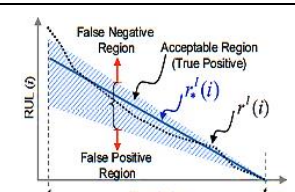
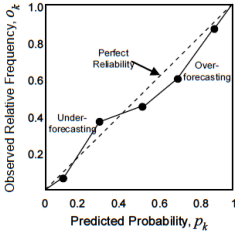

	Metric	Definition	Description	Reference
Accuracy-Based Metrics	Error	$\Delta^l(i) = r_*^l - r^l(i)$	The error is the basic notion defined as the difference or diversion of the prediction result when compared to actual data. The majority of the accuracy-based metrics are directly or indirectly an error measurement.	[20]
	Average scale-independent error	$A(i) = \frac{1}{L} \sum_{l=1}^L \exp \left\{ -\frac{ \Delta^l(i) }{D_0} \right\}$ where $D_0$ is a normalizing constant value	The weights exponentially scale the errors of RUL predictions and average RUL result over several UUT	[20, 42]
	Average bias	$B_l = \frac{\sum_{i=P}^{EOP} \{\Delta^l(i)\}}{(EOP - P + 1)}$	The averages of the prediction errors can be made at all and any time from the beginning of the prediction at the $l^{th}$ UUT. This metric can also be extended to calculate the average of biases overall UUT index and total bias in an application.	[20]
	Timeliness	$A(i) = \frac{1}{L} \sum_{l=1}^L \varphi\{\Delta^l(i)\}$ where, $\varphi(z) = \begin{cases} \exp\{ z /a_1\} - 1, & \text{if } z < 0 \\ \exp\{ z /a_2\} - 1, & \text{if } z \geq 0 \end{cases}$ and $a_1 > a_2 > 0$	The exponentially weights calculate RUL prediction errors via a type of asymmetric weighting function. This usually penalizes late predictions rather than early predictions.	[20]
	False Positives (FP)	$FP(r_*^l(i)) = \begin{cases} 1, & \text{if } \Delta^l > t_{FP} \\ 0, & \text{otherwise} \end{cases}$ where, $t_{FP}$ = user-defined acceptable early prediction	 The FP aims to determine unacceptable early predictions, while FN is used to determine unacceptable late predictions. Acceptable ranges ( $t_{FN}$ and $t_{FP}$ ) must be set before prediction. This is to address the early predictions that are the result of the redundant lead time that usually causes unnecessary corrections. The prediction that comes later after critical threshold time units ( $t_c$ ) will not be considered as a prediction result.	[43]



Table 1. PHM performance metrics (continued).

	Metric	Definition	Description	Reference
Accuracy-Based Metrics	False Negatives (FN)	$FN(r_s^l(i)) = \begin{cases} 1, & \text{if } -\Delta^l > t_{FN} \\ 0, & \text{otherwise} \end{cases}$ <p>where, <math>t_{FN}</math> = the defined acceptable point in time for an early prediction</p>		
	Mean absolute percentage error (MAPE)	$MAPE(i) = \frac{1}{L} \sum_{l=1}^L \left  \frac{100\Delta^l(i)}{r_s^l(i)} \right $	The averages of the absolute percentage errors in the predictions are calculated for multiple UUT at the same prediction time. Instead of the mean, median, this metric can also be used to compute the Median absolute percentage error (MdAPE).	[42-44]
	Anomaly correlation coefficient (ACC)	$ACC = \frac{\sum(\pi^l(i j) - z_{\#}(i))(z_s(i) - z_{\#}(i))}{\sqrt{\sum(\pi^l(i j) - z_{\#}(i))^2 \sum(z_s(i) - z_{\#}(i))^2}}$ <p>where, <math>z_s(i)</math> is a prediction variable, and <math>z_{\#}(i)</math> is the corresponding history data.</p>	The ACC measures the difference between the prediction and observations phase. This can be done by subtracting the historical mean at each prediction point. The advantage of ACC is that it is not sensitive to bias or error. However, a good anomaly correlation does not guarantee prediction accuracy. ACC is normally computed over a few time-steps after $t_p$ . This can be used to modify long-term predictions. Note that the computation of baseline from historical data is required for ACC.	[45]
	Mean squared error (MSE)	$MSE(i) = \frac{1}{L} \sum_{l=1}^L \Delta^l(i)^2$	The averages of the squared prediction calculate the error for multiple UUT at the same prediction horizon. A derivative of MSE is Root Mean Squared Error (RMSE) which is also popular among many PHM application	[42]
	Mean absolute error (MAE)	$MAE(i) = \frac{1}{L} \sum_{l=1}^L  \Delta^l(i)^2 $	The averages of the absolute prediction also calculate the error for multiple UUT at the same prediction horizon but use median instead of mean.	[42]
	Root mean squared percentage error (RMSPE)	$RMSPE(i) = \sqrt{\frac{1}{L} \sum_{l=1}^L \left  \frac{\Delta^l(i)}{r_s^l(i)} \right ^2}$	The square root of the average percentage calculates the error of the prediction from multiple UUT. A similar metric is the Root median squared percentage error (RMSPE).	[42]
Precision-Based Metrics	Sample standard deviation (SSD)	$SSD(i) = \sqrt{\frac{\sum_{l=1}^n (\Delta^l(i) - M)^2}{n - 1}}$ <p>where <math>M</math> is the sample mean of error</p>	The sample standard deviation measures the spread of the error with respect to the sample mean. This metric is restricted to the normal distribution of the error. The result of this error is usually recommended to be plotted when reporting this type of error.	[20]
	Mean absolute deviation from the sample median (AD)	$AD(i) = \frac{1}{n} \sum_{l=1}^n  \Delta^l(i) - M $ <p>where <math>M = \text{median}(\Delta^l(i))</math> and the median is the <math>\left(\frac{n+1}{2}\right)^{th}</math> order statistic</p>	This is an estimator of the spread of error. It is normally being used when the plot of error is not a normal distribution, and when there is a small number of UUT.	[46]
	Median absolute deviation from the sample median (MAD)	$MAD(i) = \text{median}( \Delta^l(i) - M )$ <p>where <math>M = \text{median}(\Delta^l(i))</math> and the median is the <math>\left(\frac{n+1}{2}\right)^{th}</math> order statistic</p>	This is also an estimator of the spread of error. It is also normally used when the plot of error is not a normal distribution, and when there is a small number of UUT or single UTT but focuses more on median measurement rather than average of error.	[46]

Table 1. PHM performance metrics (continued).

	Metric	Definition	Description	Reference
Robustness-Based Metrics	Reliability diagram, Brier Score	 <p>The Brier Score computed as <math>BS = \frac{1}{K} \sum_{k=1}^K (p_k - o_k)^2</math> is a measure of the deviation from diagonal</p>	The reliability diagram plots are used to observe the prediction frequency against the predicted probability of the RUL of a system in the condition that RUL must be within a given interval, or the health index is crossing a threshold. The prediction of RUL is not considered as an event (in probability). In this case, the prognostics problem must be converted to the classification domain. The range of probabilities that prognostics event occurs is divided into $K$ probabilities. The proximity of the plotted curve to the diagonal indicates "reliability". The deviation from the diagonal indicates conditional bias. The curve is below the diagonal line, which means, it is over-forecasting or too high of prediction probabilities, in opposition, points above the line indicate under-forecasting or too low prediction probabilities.	[45]
	Receiver Operating Characteristic (ROC)	 <p>The area under the ROC curve can be used as a score</p>	The ROC provides an overview tradeoff between false positives (FP) and false negatives (FN). That curve that has zero false positives and zero false negatives is an ideal curve. Such a curve may not be able to achieve. As a result, tuning the prognostics algorithm based on only ROC may not be practical in practice.	[45, 47]
	Sensitivity	$S(i) = \frac{1}{L} \sum_{l=1}^L \left\{ \frac{\Delta M^l(i)}{\Delta_{input}} \right\}$ <p>where,  <math>\Delta M</math> is the distance measure between two successive outputs for metric <math>M</math>'s value and  <math>\Delta_{input}</math> is a distance measure between two successive inputs</p>	Sensitivity measures how sensitive a prognostics algorithm is to the variations of input or disturbance signal. Sensitivity can be evaluated along with any performance metrics.	[20]

#### 1.1.4. Initial Framework for PHM Data Modeling Using Deep Learning

In general, a machine learning framework for PHM data was developed based on a cross-industry standard process for data mining (CRISP-DM) [48] and consisted of five phases: definition states phase, pre-processing phase, training phase, testing phase, and evaluating phase. The standard framework is illustrated in Figure 9. However, because of the high complexity of deep networks, it has been often found that deep networks are harder to train to compare to vanilla neural networks or other machine learning algorithms [49]. To guarantee that the deep networks can be successfully trained and overcome the learning difficulty, additional steps (such pre-training, feature selection, dimensionality reduction, and etc.) might need to be added into the standard machine learning framework. These issues will be further addressed in more detail in the next chapter and one of the possible new frameworks of PHM application using deep learning will be initially proposed.

Most of the prognostics models that employed machine learning were developed based on the aforementioned framework. The experiment with data was mostly constructed by varying the hyper-parameters depending on the algorithm. The hyperparameters are the modeling parameters that can be fine-tuned. For deep learning algorithms, the hyper-parameters to experiment are the number of layers, type of activation functions, momentum, epoch, batch size, etc. All these hyperparameters can cause the variation of the deep learning architecture configurations for each PHM application domain.

The performance metrics are calculated and measured first during the training phase until the performance result is satisfied and again during the testing phase to measure model performance against unknown or new data or a set of validation data. Multiple performance metrics might be used to draw a more complete assumption of the experiment. As briefly discussed in the previous section, the stability of these models have not been widely addressed in many published pieces of literature. The stability of the model might be improved by evaluating the uncertainties associated with the training parameters. This might involve additional tasks within or before the preprocessing phase. While this issue is not the main focus of this dissertation, it will be briefly discussed in the next chapter and one of the possible new frameworks that might help to overcome some aspects of stability will be introduced.

## **1.2. Research Objective and Contribution**

In this research work, the goal was to develop a modeling algorithm to accurately predict the lifespan of certain PHM applications, in this case is, the gas turbine aircraft engines. The lifespan prediction can be used to determine when maintenance should be performed. This not only saves the cost of operating and maintenance of such systems, but also to help save the lives of the people by avoiding a catastrophic failure. In addition, this dissertation also defines a baseline or generic results for implementing and developing predictive models for PHM applications in general. Thus, the dissertation also includes the experiment results for additional deep learning predictive models of another popular PHM application, which is, a PHM of lithium-ion battery as well as proposing a general modeling framework for using deep learning in PHM. This promotes the concept of using deep learning algorithms that can be generalized across multiple PHM applications and provide a benchmark for PHM modeling scheme in general. To achieve all those goals, below is the list of objectives, tasks, and contributions that this dissertation aims to fulfill.

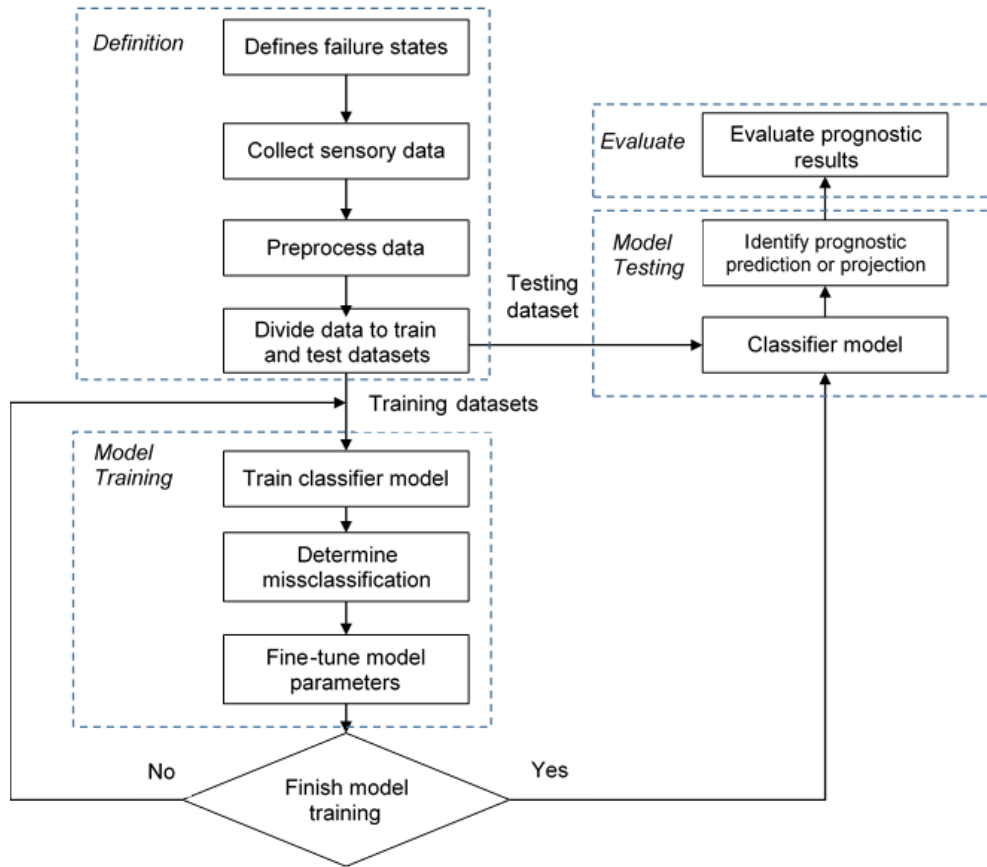


Figure 9. The process of the prognostics framework using machine learning in general [48].

### 1.2.1. Objectives and Tasks

Objective 1: Perform a literature survey for the PHM applications that deployed deep learning algorithms.

- Review comparative literature in the ongoing PHM areas that applied deep learning or neural networks as a modeling algorithm.
- Suggest a possible general framework to deploy deep learning algorithms for PHM applications.

Objective 2: Perform preliminary data-driven model experimentation on PHM data.

- Generate PHM predictive models with multiple data mining or machine learning algorithms and compare their performance with the model constructed with deep learning algorithms in other existing works and the experiments using basic or vanilla deep learning or neural network algorithms.

- Use the most popular PHM dataset—lithium-ion battery data, as a benchmark dataset to test the assumption and develop preliminary deep learning data-driven predictive models for the PHM dataset.

Objective 3: Extract meaningful features for deep learning data-driven models from the aircraft gas turbine engine dataset to improve neural network-based feature selection method for aircraft gas turbine engines RUL prediction.

- Evaluate the features/attributes from the dataset first. Not only to develop high accuracy models but also to reduce the complexity of the models by selecting only meaningful features/attributes from the dataset.
- Use evolutionary selection (applied Genetic Algorithm (GA)) as a feature selection algorithm to extract the meaningful feature from the aircraft gas turbine engines dataset based on deep neural network modeling algorithms.
- Compare results from evolutionary selection to other models using other neural network-based feature selection approaches in other existing works and the experiments using basic or vanilla deep learning or neural network algorithms.

Objective 4: Develop/Propose a data-driven model using deep learning algorithms for the prognostics of aircraft gas turbine engines.

- Use the Hybrid Convolutional Deep Neural Network as a modeling algorithm
- Compare results from the Hybrid Convolutional Deep Neural Network to other models using other neural network-based algorithms

### **1.2.2. Deliverables**

The final model(s) that was developed demonstrated improvement of deep learning model by reduced complexity or increased accuracy over existing models in other literature.

Deliverable 1: Reduce complexity; by performing features selection, dimensionality reduction, and data compression.

Deliverable 2: Increase accuracy; by using suitable deep learning algorithms, and hybrid modeling algorithm schemes.

Deliverable 3: Faster convergence; by using optimized hyper-parameters tuning.

### **1.2.3. Contributions**

Contribution 1: Introduce a better prognostics model(s) of aircraft gas turbine engines to the PHM research communities and/or aerospace communities.

Contribution 2: Draw a conclusion on which features/attributes in the aircraft gas turbine engines dataset that increase the efficiency of implementing data-driven deep learning prognostics models.

Contribution 3: Validate the developed methodology can improve the RUL prediction model for gas turbine engines by comparing its performance/error and complexity to the model derived from original features and existing models in other literature.

Contribution 4: Provide baseline/benchmark results for predicting the lifespan of PHM data for such aircraft engines.

### **1.3. Chapter Summary**

In this first chapter of the dissertation, this work looked at presenting the concept of applying a new deep learning science and approaches to PHM applications. There are attempts in the past to emerge those two fields with limited success. Thus, more studies are needed in this field. This dissertation serves as a part of the studies in the aforementioned topic by focusing on developing a deep learning model for a specific PHM application, which in this case is, the PHM of aircraft gas turbine engines. In addition, the goal is to provide the general outlook for using deep learning within the PHM scheme. This will be achieved by providing a general concept of using deep learning for PHM application along with some preliminary experiments as well as fine-tuning the deep learning data-driven predictive models for aircraft gas turbine engines. The following chapters of the dissertation (Chapter 2-5) are organized by the four specific objectives mentioned previously in Chapter 1. The dissertation concludes in Chapter 5 with the end results and finally deep learning PHM model of aircraft gas turbine engines along with the summary and future directions of Deep Learning for Prognostics and Health Management Applications topic.

## 2. A SURVEY OF DEEP LEARNING APPROACH FOR PROGNOSTICS AND HEALTH MANAGEMENT APPLICATIONS

### 2.1. Deep Learning Paradigm in PHM Tasks

PHM is a computation-based application that elaborates on physical knowledge, information, and data [50] of structures, systems, and components operation and maintenance, to detect anomalies of equipment and process, diagnose and evaluate degradation states and faults, as well as predict the progression of degradation to failure and estimate the remaining useful life. The PHM task is illustrated in Figure 10. As mentioned in more detail in Chapter 1, the outcomes of the PHM elaboration are used to support condition-based and predictive maintenance decisions for efficient, reliable, and safe operations. The ability to deploy these maintenance strategies provides the opportunity of setting efficient, just-in-time, and just-right maintenance strategies [51, 52]. This can help to maximize the production profits and minimize all costs and losses. As a result, PHM research and development has intensified, both in academia and industry, involving various disciplines of mathematics, computer science, operation research, physics, chemistry, materials science, engineering, etc.[53, 54].

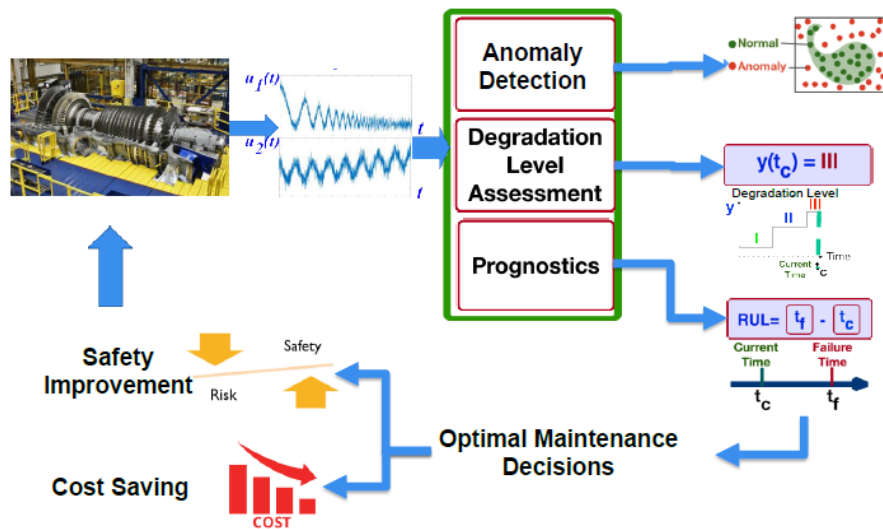


Figure 10. Tasks in PHM [50].

The practical implementation of PHM includes data acquisition to enable detection, diagnostics and prognostics tasks, and maintenance decision-making [55]. The supporting PHM framework and its requirements must be properly defined to perform well in real industrial scenarios. Given the increasing

complexity, integration, and informatization of modern engineering, PHM can no longer be an isolated addition for supporting maintenance but must be closely linked to the other structure, power, electromechanical, information and communication technology, control parts of the systems. PHM must be included at the beginning of the system conceptualization and carried through its design and development in an integrated framework capable of satisfying the overall operation and performance requirements [56, 57]. The development of PHM in practice also involves other aspects, including design (e.g. the use of smart components may lead to different reliability allocation solutions), and impacts various work units involved in maintenance decisions and actions (e.g., workers can use smart systems, maintenance engineers can analyze big data), including the supporting logistics (spare parts availability and warehouse management can be driven by the PHM results) [58]. This is where new techniques such as machine learning and deep learning paradigm can improve and make PHM smarter and predict the outcomes more accurately and more reliably.

Methods of fault detection, fault diagnostics, and failure prognostics within the PHM framework are continuously being developed. The advance 'smarter' data analytics (also including image processing and text mining) are mostly based on the newly developed artificial intelligence, machine learning, and deep learning paradigms with the adoption of more computational power systems. The objective of performing each task in diagnostic and prognostic is different. The objective of fault detection is to recognize anomalies behavior. The objective of fault diagnostics is to identify the degradation states and the causes of degradation. Prognostics aims at predicting the RUL, which is the main focus of the experiments and models proposed in this dissertation. All fault detection and diagnostics, and failure prognostics combined can be an enabler of condition-based and predictive maintenance, which offers major opportunities for Industry 4.0 and smart structures, systems, and components operation and maintenance, as they can allow reducing failures, increasing infrastructure usage, and reducing operation and maintenance costs, with tangible benefits of reduction of production downtime, risk and asset losses, and consequent increase of production profit [59]. All these are where machine learning and deep learning paradigms can be integrated into PHM systems to improve the main 3 tasks; anomaly detection, degradation level assessment, and prognostics, in PHM as described in Figure 10. The new deep learning technique has proven to be able to handle all these tasks but still needs more studies.



In this chapter, a summary of research works in PHM as part of the survey and address some of the challenges within this domain are provided. Additionally, an improved frameworks (initial framework mentioned in Chapter 1 section 1.1.4) for using Deep Learning in PHM applications will be proposed.

## **2.2. Research Application of PHM Domain Using Deep Learning Algorithms**

Deep learning is considered a relatively new approach for research application in the PHM area, with promise to improve current outcomes of PHM. There are limited recent works that employed deep learning to PHM data. The majority of the works were published between 2014 to 2019. Table 2 provides a summary list of prognostics and some related diagnostics research using deep learning based on the application area.

The brief discussions of the applied research for each deep learning algorithm are described next. However, there is one deep learning algorithm, namely, Deep Belief Networks (DBN), that is discussed in this chapter because of its long history of being studied. DBN is not considered to be a recent algorithm for the deep learning approach. The new trend of deep learning recently leans toward more on newly developing types of deep layers, such as CNN, RNN-LSTM, and hybrid methods. Additionally, the improved DBN has already appeared within the hybrid methods area. Most of the interesting improved versions of DBN will also be discussed in hybrid allocation approaches. More information on PHM research applications using DBN can be found in [32].

Comparing the advantage and limitations between these deep learning algorithms (DNN, CNN, and RNN-LSTM), it was discovered that DNN is more suitable to tackle single-dimensional constructed data. While CNN is suitable to deal with multidimensional data, such as image data, or data with two/three-dimension (2D or 3D) construction. This is because a convolutional technique that has been used in CNN layers. This is due to the fact that CNN can help to expand the feature of the data in the higher level to better recognize patterns from those higher-level features. It is worth to also notice that DNN is usually employed to extract the global feature from fault data, while, CNN has outstanding performance for local feature extraction. The local feature is feature from the higher dimension which usually represent what was obtained from the raw data. Though there is limited works that have used RNN-LSTM, it has been shown that RNN-LSTM is efficient at handling the degradation dataset with a timespan and can recognize and capture a repetitive degradation pattern related to time better than other

deep learning algorithms. Also, CNN and RNN-LSTM algorithms are more complex than DNN. This causes CNN and RNN-LSTM to require more training the final model and require more computational resources compared to DNN. Group of prognostics models using each deep learning algorithm are as described in the next subsection. This is to determine the research gap in each prognostics area and provide a general idea of where this work can contribute to the PHM community.

### **2.2.1. PHM Models Using DNN**

In 2003, Samanta, B., et al. is one of the early adopters employing vanilla artificial neural network to estimate the fault state of the bearing elements. This work did not aim to improve the prediction accuracy but only aimed to set an example of how ANN can be employed (train and test) for fault diagnostic and prognostics application [60]. Ma, Y., et al. (2014), a proposed architecture for fault diagnosis of visual images and structured data based on the deep auto-encoder neural network [61]. Fink, O., et al. (2014) used multilayer feedforward neural networks based on multi-valued neurons for the railway turnout application, multilayer feedforward neural networks confirms the good performance in the long-term prediction of degradation and does not show accumulating errors for multi-step ahead predictions [62]. Weining Lu et al. (2015) developed a feature extraction method based on DNN using the bearing system as a case study [63]. Jingwei Qiu et al. (2015) proposed a full features extraction method using DNN combined with state analysis of the hidden Markov model to improve the diagnostics and prognostics analysis with the inseparable fault. This development leads to the innovation of the early-warning fault model. This method can effectively deal with the multi attributes-multi features prognostics data [64]. Li, K., & Wang, Q. (2015) designed a multi-class classification model using the auto-encoder stacked neural networks. The proposed method can identify fault characteristics for various diagnosis and prognosis issues [65]. Yaguo, L., et al. (2015) employed DNN to mine available fault characteristics [66]. Sarkar, S., et al. (2015) used DNN to classify greyscale flame images from the combustion chamber of the gas turbine engine whether the engine is in a stable stage or unstable regions [67]. Feng Jia et al. (2016) used DNN to develop an intelligent method for fault diagnosis diagnosing through the prognostics of rotating machinery dataset [68]. Liu, H., et al. (2016) proposed a new rolling bearing fault diagnosis method using the sound signal. The modeling method was based on short-time Fourier transform and stacked sparse auto-encoder DNN [69]. Sun, W., (2016) presented a DNN approach for fault diagnosis

of an induction motor by utilizing the sparse auto-encoder to extract features [70]. Lei, Y., et al used two stages-two layers DNN to develop a classification model for fault diagnostic of bearing equipment based on the vibration signal data of the bearing system [71]. Zhou, F., et al. (2017) propose a multimode classification method based on deep learning by constructing a hierarchical DNN model for mode partition of bearing [72]. Ma, K., et al. (2017) trained auto-encoder DNN model using fiber-optic acoustic data from a pipeline system and developed a multi-step method for pipeline anomaly detection [73]. Jiang, G., et al. (2017) proposed a fault detector model based on an unsupervised learning method, denoising auto-encoder deep network, which can capture nonlinear data patterns against noise and input fluctuation [74]. Bangalore, P., et al. applied an artificial neural network with Mahalanobis distance to develop a model for anomaly detection in wind turbine gearboxes [75]. Zhao, Z., et al. used an improved back propagation neural network as a modeling algorithm for predicting the RUL of multiple aircraft engines [76]. Xiao, H., et al. (2017) proposed a fault diagnosis framework using auto-associative neural networks for wastewater process. The framework has been validated by process data collected from two wastewater treatment plants with different dynamic characteristics [77]. In 2018, Chemali, E., et al. successfully employed DNN to predict the multi-state of charge or health state of the lithium-ion battery system [78]. In 2019, Tolo, S., et al., developed a robust on-line fault detection tool for the early accident detection for nuclear powerplant heavy-water reactor with artificial neural network architectures through the use of Bayesian statistics as a modeling algorithm [79]. Pliego, A., et al. (2019) used two real dataset from a wind turbine to train the false alarm model. Their proposed model was compared against the prediction result using fuzzy logic model [80].

### **2.2.2. PHM Models Using CNN**

Generally, new configurations or new constructions of the proposed neural network layers, including CNN, is rather new and can help to improve some aspects of the models over the years. Each configuration of the CNN layer provides different end results that might rather better in terms of prediction accuracy, complexity, or robustness of the models. Chen Zhiqiang et al. (2015) suggested a configuration of a deep Convolutional Neural Network (CNN) for fault identification and classification. The experiment had shown that the suggested model had outstanding performance compared to base algorithms [81]. Babu, G. S., et al. (2016) proposed another deep CNN based on the regression approach. The proposed

CNN model was compared to three regression algorithms including Multi-Layer Perceptron (MLP), Support Vector Regression (SVR), and Relevance Vector Regression (RVR). Results showed that the proposed CNN could deliver better results across multiple datasets [82]. Janssens, O., et al (2016) proposed a feature learning model for condition monitoring based on convolutional neural networks to autonomously learn useful features for bearing fault detection from several types of bearing faults data such as outer-raceway faults and lubrication degradation [83]. Ince, T., et al. (2016) developed a fast and accurate motor condition monitoring model as an early fault-detection system using one-dimension CNN with an adaptive design for the feature extraction and classification of the motor fault detection in a single learning body [84]. Dong, H.Y., et al (2016) developed a small fault diagnosis method using CNN trained by vibration data under several different small fault patterns of front-end controlled wind generators [85]. Gibert, X., et al. (2016) used CNN to generate a multi-task learning framework for detecting possible different failure modes of railway track from its image [86]. Lu, C. et al. (2017) also employed hierarchical CNN to develop a fault classification model of rolling bearings. Their model reduced learning computation requirements in the temporal dimension, and an invariance level of working condition fluctuation and ambient noise was provided by identifying the elementary features of bearings [87]. Xia, M., (2017) developed a CNN-based approach for fault diagnosis of rotating machinery. This work took advantage of the CNN structure to achieve higher and more robust diagnosis accuracy [88]. Janssens, O, et al. (2017) investigated if and how CNN can be applied to the infrared thermal video data to automatically determine the condition of the servo-motor [89]. In 2018, Xiang Li, et al. applied CNN as a time window approach to generate a feature extraction model of C-MAPSS aero-engine data [90].

### **2.2.3. PHM Models Using RNN-LSTM**

Xuhong, W., et al (2005) presented a diagonal RNN based approach for detecting fault in the induction motors. In this work, two diagonal recurrent neural networks were employed to detect turn fault of the induction motors. Turn fault is a type of electrical fault occur in the induction motor that might happen either in stator or rotor in the induction motor. One was used to estimate the fault severity, the other was used to determine the exact number of faults turns [91]. In 2007, Qingpei Hu, et al. applied RNN for software reliability. RNN applied to model fault detection, and fault correction process within the software packet [92]. Oliver Obst (2014) deployed an RNN that can learn spatiotemporal correlations

between different sensors and made use of the learned model to detect faulty sensors within a system [93]. Yuan, M. et al. (2016) applied the RNN-LSTM network for fault diagnosis and estimation of the remaining useful life of aircraft engines. Results showed that the performance of the LSTM model with modifications outperformed other peer algorithms [94]. Tim de Bruin, et al. (2016) used LSTM to learn dependencies within railway track circuits data to timely detect and identify the faults in railway track [95]. Guo, L., et al. (2017) proposed a data-driven model using RNN for remaining useful life prediction of bearing dataset [96]. Zhang, S. et al. (2017) develop a method for data-based line trip fault prediction in power systems using LSTM compared against support vector machine [97]. Zhang, Y. et al. (2017) used a simple one-layer RNN-LSTM to construct the RUL prediction model for lithium-ion battery from NASA Ames Prognostics Center of Excellence (PCoE) and compare the result with Support Vector Machine [98]. In 2018, Zhang, Y. et al. used RNN-LSTM to make predictions of the remaining useful life of the lithium-ion battery dataset. The experiment demonstrated that LSTM was able to capture the underlying long-term dependencies or variables among the degraded capacities lithium-ion battery datasets. The experiment was contrasted to the support vector machine model, the particle filter model, and the simple RNN model for RUL prediction. RNN-LSTM was, again, outperformed every other baseline models [99]. Nguyen, K., et al. developed the prognostics step, based on the Long/Short-Term Memory network, oriented towards the requirements of operation planners. Their approach provided the probabilities that the system can fail in different time horizons to decide the moment for preparing and performing maintenance activities. The proposed framework was validated on a real application case study using the C-MAPSS aircraft dataset [100]. Again, a popular C-MAPSS aircraft dataset has been used by da Costa, P., et al. (2020) to develop the RUL prediction model. In this work, a Domain Adversarial Neural Network (DANN) approach was applied to training LSTM models. The results showed that their method provided more reliable RUL predictions than models trained only on source data for varying operating conditions and fault modes [101]. Recently in 2021, Shi, Z., et al. proposed a new dual-LSTM framework ideology to predict the life span of an aircraft engine using a popular C-MAPSS dataset. They used the LSTM network in two training states (change point prediction state and prediction state.) The result using their framework has proven to achieve higher precision compared to the existing benchmark results using the traditional LSTM framework [102].

#### 2.2.4. PHM Models Using Hybrid Deep Learning Layers

Wang, P., et al. (2015) improve the accuracy of fault recognition by developing a new identification method called, PDBN, which is a hybrid method that combines the particle swarm optimization (PSO) algorithm with the Deep Belief Network (DBN) [103]. Shao, H., et al. (2015) employed the optimization DBN model for fault diagnosis and prognosis, where PSO was additionally used to decide the optimal structure of DBN. The results confirmed that the suggested hybrid method was better performed in terms of accuracy when compared to SVM, ANN, and Boosting methods [104]. Zhiqiang, C., et al. (2015) presented multi-classifier models using a multi-layer neural network (MLNN) for fault diagnosis of vibration signals. The new learning architecture using Deep Belief Network (MLNNDNB) was proposed and tested in their work [105]. Chuan Li et al. (2015) experimented on a multimodal deep support vector classification (MDSVC) model to diagnose faults by employing Gaussian-Bernoulli Deep Boltzmann Machine (GDBM) [106]. Sanchez et al. (2016) addressed the use of the Deep Random Forest Fusion (DRFF) method for fault diagnosis and prognosis performance of gearboxes. They employed two independent DBNs to extract the features of an acoustic emission sensor and an accelerometer. This was an improvement of the vanilla random forest method [107]. Jha, D.K., et al. (2016) used CNN to extract spatial features from individual combustion wind turbine images and employed a Gaussian process to model the temporal dynamics of the spatial features extracted from CNN [108]. Zhao, L. et al. (2017) combined the CNN and LSTM to generate Machine Health Monitoring Systems (MHMS) model. In this work, Convolutional Bi-directional Long Short-Term Memory networks (CBLSTM) have been particularly designed to handle raw sensory data from MHMS. The first step of CBLSTM was to use a regular CNN layer to exploit local features from raw data that were robust and informative features. Next was using a bi-directional LSTM layer to encode temporal information. In this work, LSTM was used to capture long-term dependencies and sequential data, while bi-directional network layer is capable to capture the information within the past and future contexts from the raw data [109]. Zhao, R. et al. (2017) employed multiple extensions of RNN, including, Gated Recurrent Units (GRU), and Bidirectional Gated Recurrent Units (BiGRU), against their proposed local featured-based Gated Recurrent Units (LFGRU), again, for Machine Health Monitoring Systems. LFGRU proposed in this work was a hybrid approach that combined handcrafted feature design with automatic feature learning for machine health monitoring. First, features

from windows of input time series were extracted. Then, an enhanced bi-directional GRU network was designed and applied to the generated sequence of local features to learn the representation. A supervised learning layer was finally trained to predict machine conditions [109]. Wang, S., et al. (2018) used convolutional neural network-based hidden Markov models (CNNHMMs) to classify multi-faults in mechanical systems. In this work a CNN and the t-distributed stochastic neighbor embedding (t-SNE) technique was first employed to learn data features from raw vibration signals data, then, HMMs was employed as a tool to classify faults [110]. Ellefsen, A., et al. (2019) used Convolutional Neural Network and Long-Short Term Memory with the utilization of Genetic Algorithm (GA) for fine-tuning hyper-parameters to generate the RUL prediction model for C-MAPSS aircraft engine data [111]. Li, X., et al. (2019) also deployed a hybrid CNN and LSTM layer to implement the multi-scale feature extraction model for bearing data [112].

### **2.3. Some of the Challenges of Deploying Deep Learning for PHM Applications**

As mentioned in the previous chapter, prognostics aim at predicting RUL, i.e., the time left before the system or components can no longer be able to perform its intended function. This prediction fits into a regression scheme when deploying deep learning. However, many challenges remain and are still in discussion within the PHM communities. The challenges arise from the complexity of the physics, the data available, and requirements to the PHM for practical solutions. These are only the challenges within the applications themselves, some of which, are directly related to the challenges of using deep learning in the PHM domain as well.

After a literature survey, we detail such challenges into three categories: 1) Data challenges, 2) Uncertainty and 3) Difficulty to train models. This section will go through these challenges. Additionally, we also propose one of the possible new frameworks for deploying deep learning in the PHM domain which can possibly address these issues based on those challenges.

#### **2.3.1. Data Challenges**

Prognostics is concerned with the prediction of the future evolution to the failure state. It involves the processing of data to predict the future degradation and functional attributes, based on estimation of failure probability and RUL. The prognostic outcomes are used for the health management of which, using the RUL prediction to decide on and actuate operational actions and maintenance interventions.

Table 2. Summary of prognostics application research using deep learning.

Algorithm	Application field	Reference
DNN	Bearing components	[60, 63, 72]
	Power transformers	[61]
	Railway	[62]
	Pipeline monitoring system	[64, 73]
	Spacecraft	[65]
	Multistage gear system	[66]
	The combustion chamber of a gas turbine	[67]
	Rotating machinery	[68]
	The sound signals from the rolling bearing	[69]
	Induction motor	[70]
	Vibration from bearing equipment	[71]
	Wind turbine	[74, 80]
	Gearboxes in wind turbine	[75]
	Aircraft engine	[76]
	Wastewater process	[77]
Lithium-ion battery	[78]	
Nuclear powerplant reactor	[79]	
CNN	Gearboxes	[81]
	Aircraft engine	[82, 90]
	Rolling bearing	[83, 87]
	Motor	[84]
	Front-end controlled wind generator	[85]
	Railway track inspection	[86]
	Rotating machinery	[88]
Servo-motor	[89]	
RNN-LSTM	Induction motor	[91]
	Software reliability	[92]
	Sensor system	[93]
	Aircraft engine	[94, 100-102]
	Railway track circuits	[95]
	Bearing	[96]
	Power system	[97]
Lithium-ion battery	[98, 99]	
Hybrid	Industrial fault	[103]
	Bearing	[104, 112]
	Gearbox	[105-107]
	Wind turbine	[108]
	Computer Numerical Control (CNC) milling machine	[109]
	Rolling bearing	[110]
	Aircraft engine	[111]

It is important to note there are uncertainties within the data available from the sensors. For example, the uncertainties in the data collected from sensors, such as noise, interference of the environment, some operational errors, and etc. affect the degradation state prediction. This makes it practically impossible to precisely predict the future evolution of the state of health and it is necessary to account for the different sources of uncertainty that affect prognostics [113].



Traditional fault prognostics methods face the challenge of dealing with incomplete and noisy data collected at irregular time steps in correspondence with the occurrence of triggering events in the system. For example, for monitoring the degradation and failure processes of bearings in large turbine units, signal measurements collection (e.g., vibration signals measured) is only triggered by abnormal behaviors of the units, such as large noise and anomalous vibration behavior. These “snapshot” datasets are often encountered in industrial applications, dominated by the necessity of cost-saving in storing and managing the databases, and of reducing energy consumption and bandwidth resources.

Because failure events are rare event-based datasets, these datasets are dominated by missing measurements, where the values of all signals are often missing at the same time. With these characteristics, traditional methods for missing data management, such as case deletion [114], imputation [115-117], and maximum likelihood estimation [118], are difficult to apply. For instance, case deletion methods discard patterns, whose information is incomplete, they are not useful in the case of event-based datasets where the pattern is either present or absent for all signals [118]. Imputation techniques, which are based on the idea that a missing value of a signal can be replaced by a statistical indicator of the probability distribution generating the data, such as, the signal mean value [119] or a value predicted by a multivariable regression model [114], have been shown inaccurate in case of large fractions of missing values in the dataset [120, 121]. Maximum Likelihood methods use the available data to identify the values of the probability distribution parameters with the largest probability of producing the sample data [120]. They normally require the Missing At Random (MAR) assumption [122]. Few research works have considered fault prognostics in presence of missing data. A model based on Auto-Regressive Moving Average (ARMA) and Auto-Associative Neural Networks (AANN), has been developed for fault diagnostics and prognostics of water process systems with incomplete data [77]. An integrated Extreme Learning Machine (ELM)-based imputation-prediction scheme for prognostics of battery data with missing data [117] and a hybrid architecture of physics-based and data-driven approaches have been proposed to deal with missing data in a rotating machinery prognostics application [123]. In the medical field, a Bayesian simulator has been used to generate missing data for developing prognostics models [124] and a Multiple Imputation approach has been embedded within a prognostics model for assessing overall

survival of ovarian cancer in presence of missing covariate data [125]. It is important to note that all these methods are based on the two successive steps of missing data reconstruction and prediction.

Advancements in technology and new methods are still needed to enable predicting the RUL based on measurements collected when only triggering events occur, such as system faults or extreme operational conditions, and providing an estimate of the uncertainty affecting the RUL prediction. As an example, M. Xu, et al. [126] had developed a method based on Echo-State Networks (ESNs) to directly predict the RUL without requiring reconstructing the missing data. ESNs are considered to use for the experiment because of their ability to maintain information about the input history. The main difficulty is that, contrarily to the typical applications of ESNs, the time intervals at which the data become available are irregular. Two different strategies have been considered to address the event-based data collection. In one strategy, the ESN receives an input pattern only when an event occurs. The pattern is formed by the measured signals and the time at which the event has occurred. In a second strategy, the reservoir states are stimulated at each time step. If an event has occurred, the reservoir states are excited both by the previous reservoir states and the measured signals, whereas, if an event has not occurred, they are excited only by the previous reservoir states. By so doing, the connection loops in the reservoir allow reconstructing the dynamic degradation behavior at those time steps in which events do not occur. Multi-Objective Differential Evolution (MODE) algorithm based on a Self-adaptive Differential Evolution with Neighborhood Search (SaNSDE) [127] is used to optimize the ESN hyper-parameters. The Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS) [128] is, then, used to select the optimal solution from the obtained Pareto solutions. Furthermore, a bootstrap aggregating (Bagging) ensemble method is applied to improve the RUL prediction accuracy and estimate the RUL prediction uncertainty. Given that ESNs cannot be fed by random sequences of patterns, the traditional Bagging sampling mechanism used to create the bootstrap training sets has been modified. In the proposed solution, the bootstrap training sets are obtained by concatenating entire run-to-failure trajectories, randomly sampled with replacement. The benefits of the proposed methods are shown by application to the prediction of the RUL of a sliding bearing of a turbine unit.

### **2.3.2. Uncertainty**

Uncertainty is intrinsically present in the PHM tasks of detection, diagnostics, and prognostics, and may adversely affect their outcomes, so to lead to an imprecise assessment of the state and prediction of the behavior of such systems, which could lead to wrongly informed system health management decisions with possibly costly, if not catastrophic, consequences. For practical deployment, it is necessary to be able to estimate the uncertainty and confidence in the outcomes of detection, diagnostics, and prognostics activities, for quantifying the risk associated with the PHM decision-making on the operation of engineering systems. Despite the recognition of the importance of uncertainty in PHM [129], work is still needed to concretely address the impact of uncertainty on the different PHM tasks and to effectively manage it [130]. Not only uncertainties from the occurrence of abnormality from PHM tasks itself but also the uncertainties from the data collected as mentioned in the previous section.

The challenge comes from the fact that there are different sources of uncertainty that affect PHM, whose interactions are not fully understood and, thus, it is difficult to systematically account for them in the PHM tasks. While some sources are internal, others are external, and all must be accounted for in the different activities of PHM. There is uncertainty in the physical behavior of ones' system and epistemic uncertainty in the model of them, which developed based on sensors data, and the associated parameters. As mentioned earlier, there is uncertainty in the sensor's measurements and their processing tools. For the prognostics task, there is also uncertainty on the future system's operation profile and state evolution.

Given the relevance of uncertainty in the PHM tasks, it becomes necessary to develop systematic frameworks to account for such uncertainty in practical applications, to enable the robust verification and validation of the solutions developed, concerning the requirements for their use for decision-making and their contribution to the risk involved in such decisions. Such frameworks must enable the systematic identification, representation, quantification, and propagation of the different sources of uncertainty. Therefore, any PHM outcome is also provided with uncertainty, which needs to be considered for robust decision-making [131].

### 2.3.3. Difficulty to Train Models

Even though the deep learning approach can be perfectly fit for the PHM task, there are a few potential issues that could cause difficulties for deep learning to be implemented in current prognostics works. The most significant issue that has been addressed in the literature so far is the complexity of training the deep network. As commonly known, the goal of training the network is to enable its learning behavior so that the network can capture the pattern data and perform the prediction task effectively. The shallow network was found to be easier to be trained compared to the deep network. For most cases, the deep network is unable to learn or recognize the pattern of data with the simple training scheme while the shallow network can learn when the same task is assigned. To solve this issue, it has been suggested that one extra step might need to be performed before effectively generating the prediction model with the deep network. This step has been suggested in the literature called “unsupervised pre-training” [49] which is the method to train (or pretrain) the training data without using the target variable prior to the actual model training phase. This is to ensure that the deep architecture network can adaptively learn and successfully perform the prediction task effectively. It has also been believed that the pre-training process can help to improve the robustness of deep learning architecture as well as its performance, especially, when increasing the depth of the network. The result from the literature shows that the increasing depth of the network also causes a higher probability of finding poor local minima [49]. The pre-training step can solve this issue and gives consistently better generalization to the deep network. The better generalization is also believed to help when some of the new input attributes of the prediction task are not within the range of the training data which is a common problem found in PHM data as mentioned in the previous section as well.

Another issue found when employing a deep learning algorithm is the instability of prediction results. This can be observed when the prediction result has a wide range of distribution. The unstable behavior of the prediction results is believed to be caused by the randomness of initialization when the deep layer model is constructed. It has also been shown in the experiment result that the more layers added to the deep architecture, the more unstable results can occur. This is the reason that most of the work that employs deep learning often reports the average results or best result from the final output distribution. This makes the deep learning model become harder to evaluate or compare to other data

mining algorithms. In other words, we may consider this as one of the modeling “Uncertainties” mentioned in the previous section as well.

Currently, there is a very limited solution to improve algorithm stability. One approach is to perform a stability training approach [132]. However, this additional step of learning may not be favored in the PHM area. To solve some issues in the deep learning model, new steps of the modeling framework must be introduced. The uncertainty quantification of training attributes or modeling parameters might be the starting point of the study.

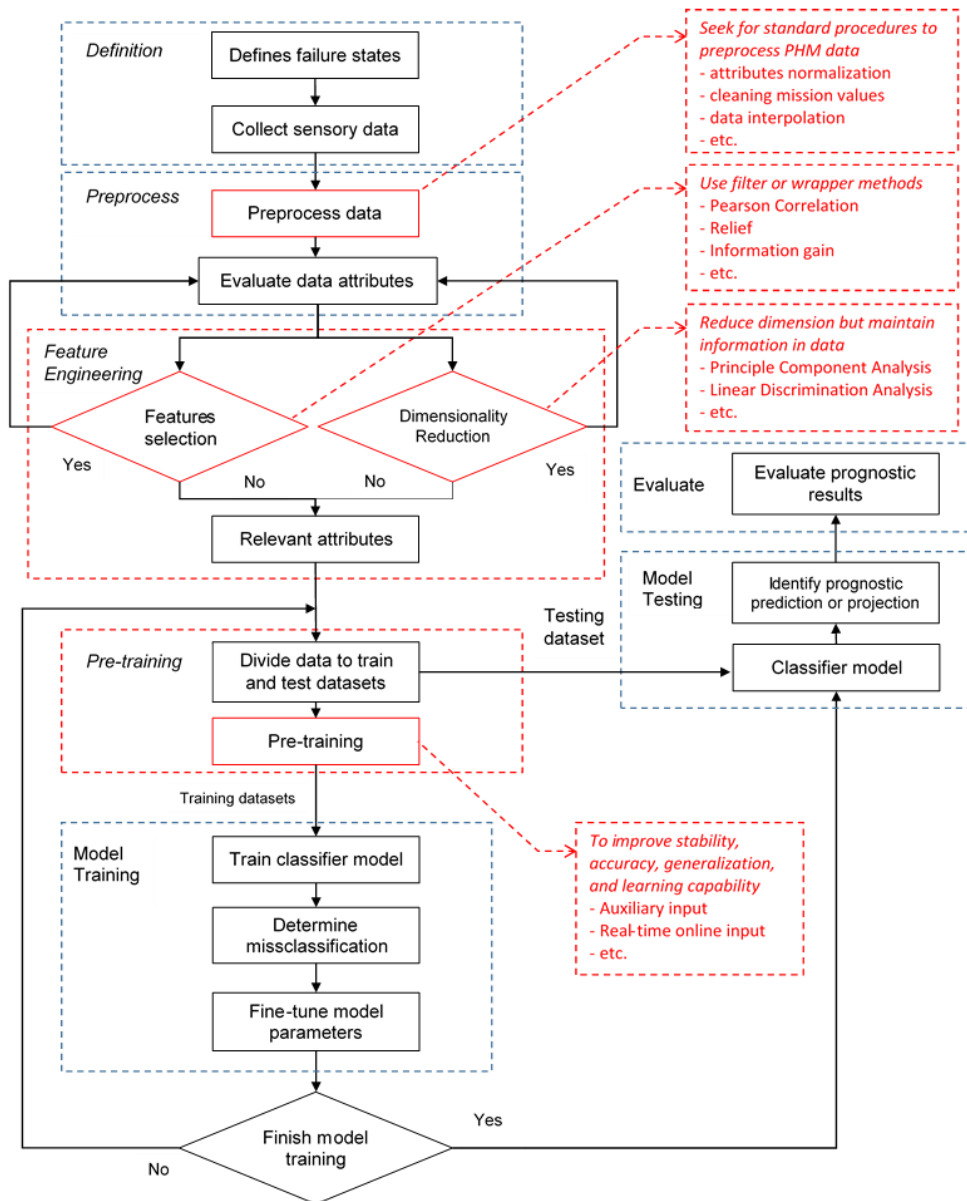


Figure 11. Proposed prognostics framework using a deep learning algorithm.

### **2.3.4. The Proposed Framework for Deploying Deep Learning in the PHM Domain**

Our ultimate goal is to introduce a general universal framework for using deep learning in the prognostics domain and also, to have an organized procedure to generate a deep learning model for prognostics data in the future. One of the possible new frameworks is as illustrated in Figure 11. Preprocessing, feature engineering, and pre-training steps have been emphasized in this framework. These three steps are commonly performed when training a deep learning network for prognostics applications. However, this is only one example of a deep learning framework. More studies must be made, and new steps are added to address all aforementioned issues.

Another minor challenge is the unity of PHM performance metrics. As mentioned in the previous chapter, there are so many performance metrics used in PHM works. However, there are no general metrics aimed to measure the effectiveness of deep learning for prognostics data that can be used for all and every prognostics application. It might be true that the error measurements have already been used to evaluate the accuracy of the deep learning model for PHM but there is still no direct measurement for generalization, robustness, and stability of the deep learning model. To precisely evaluate the model, new measurement metrics might also be introduced to the deep PHM model as well as a new framework in which all angles of performance measurements are considered.

## **2.4. Chapter Summary**

Degradation patterns prediction and recognition play a great deal in PHM application. Many frameworks and algorithms have been introduced over the years in PHM studies. There is still no one perfect algorithm that can guarantee to deliver the best result for every prognostic application. However, the deep learning approach is believed to be able to outperform many conventional prediction algorithms, which can be extended to PHM applications. Although deep learning seems to be a promising approach for prognostics applications, there are still some challenges and issues (such as difficulties in training the deep learning algorithms) that should be addressed and studied more. This chapter gathered useful initial information and highlights some key issues of deploying deep learning in PHM areas. We believe that addressing these issues will engage the PHM community to employ more deep learning in the PHM domain from this point onwards.

# 3. A DATA-DRIVEN PREDICTIVE PROGNOSTICS MODEL FOR LITHIUM-ION BATTERIES BASED ON A DEEP LEARNING ALGORITHM

## 3.1. Data-Driven Prognostics Approach for Lithium-ion Battery

Historically, nickel-cadmium batteries were generally the only common electrical power source for various portable equipment, until nickel-metal hybrid and lithium-ion batteries were developed in the 1990s [133]. Currently, lithium-ion battery technology is rapidly growing, and it is the most reliable portable electrical power source for numerous appliances. Lithium-ion batteries are extensively used in both high and low-power products, such as hybrid-motor engines, electric cars, smartphones, tablets, and laptops. To date, lithium-ion technology is considered to be a standard electrical storage system, and its performance continues to improve. The main focus of the ongoing technology remains to improving the lithium-ion system in terms of both its performance and reliability. The following are the main advantages of lithium-ion batteries: (1) high energy density (up to 23–70 Wh/kg), (2) high efficiency (~90%), and (3) long life cycle (provides 80% capacity at 3,000 cycles) [134].

To ensure that the lithium-ion battery system performing reliably, there must be a method that helps track and determine the state of health (SoH) of the battery system, along with its RUL (Remaining Useful Life). This method provide insight when the battery should be replaced. This type of evaluation falls into the PHM paradigm.

### 3.1.1. Overview of Data-Driven Prognostics

Similar to other applications within the PHM paradigm, the PHM of batteries must be included as part of the CBM (Condition-Based Maintenance) plan of the system. The CBM plan is considered a preventive strategy, which means that maintenance is be performed only when the need arises. This need can be determined by continuously evaluating the health status of a particular system's components, or the health state of the system as a whole [135]. CBM has included two major tasks: diagnostics and prognostics. Diagnostics is the process of the identification of faults and part of the current health status of the system, which is described as a SoH, whereas prognostics is the process of forecasting the time to failure. The time left before observing a failure is described as the RUL of such a

system [136]. To avoid catastrophic failure, the maintenances must be performed when the system is operational. These types of maintenance require early plans and preparation [22]. Thus, CBM can be included as part of the system's operation, especially for the critical systems. The prognostic of the system is a crucial factor in CBM.

The prognostic process additionally involves two phases. The first phase aims to assess the current SoH. Terms that are usually used to describe this phase in most of the literature are severity detection and degradation detection, which can also be considered under diagnostics. Classification or clustering techniques can be utilized to perform tasks such as pattern recognition in this phase. The second phase aims to predict the failure time by forecasting the degradation trend, and by identifying the RUL. Trend projection, tracking techniques, or time series analysis are included in this phase. Most of the articles regarding prognostics analysis only consider the first phase [137]. This paper aims to construct and analyze both SoH and RUL, in which focus is made on both the first and the second phase of prognostics for the battery system.

Table 3. Difference between data-driven and physics-based models for PHM.

	<b>Data-Driven Model [24]</b>	<b>Physics-Based Model [25, 26]</b>
<b>Based on</b>	The empirical lifetime data and the use of previous data of the operation of the system	Physical understanding of the physical rules of the system, the exact formulas that represent the system
<b>Advantages</b>	The real behavior of the complex physical system is not required.	Higher accuracy because the model is based on an actual (or near-actual) physical system
	Models are less complex, easier to employ in a real application	The model represents a real system, the model can be observed and judged in a more realistic manner
<b>Limitations</b>	Needs a large amount of empirical data to construct a high accuracy model	Highly complex requires extensive computational time/resources, which may not be very suitable for employment in real-world applications
	The models do not represent the actual system, it requires more effort to understand the real system behavior based on the collected data	Limitations in modeling, especially in cases of large and complex systems with non-measurable variables

Generally, there are two existing major approaches for prognostics evaluation; the data-driven model, and physics-based models. Data-driven methods require adequate data or samples from systems that were run until failure, while physics-based methods evaluate the system's failures through the physics of failure progression. Both the data-driven and physics-based models also require different input



and apply to different scenarios. Both have different advantages and limitations. Table 3 summarizes the information on the differences and advantages of each model.

One of the data-driven model approaches for prognostics and diagnostics mentioned earlier in the previous chapters are machine learning approaches, which will be used to generate the prognostics model for lithium-ion battery used in this chapter.

### **3.1.2. Prognostics of the Lithium-ion Battery**

The lithium-ion battery data used in the prognostics analysis of this work was from the NASA Ames Prognostics Center of Excellence (PCoE) data repository [138]. This dataset contains the test results of commercially available lithium-ion 1850-sized rechargeable batteries, and the experiment has been performed under controlled conditions in the NASA prognostics testbed [139].

Experimental data were obtained from three different lithium-ion battery-operational test conditions: charge, discharge, and impedance. All experiments were performed at room temperature. The charge was performed at a constant current of 1.5 A until the voltage reached 4.2 V, and then it continued charging at a constant voltage until the charge current dropped to 20  $\mu$ A. The discharge was also performed at a constant current of 2 A until the voltage dropped to 2.7 V, 2.5 V, 2.2 V, and 2 V. These same tests were performed for batteries No. 05, No. 06, No. 07, and No. 18. The impedance test was completed using EIS (Electrochemical Impedance Spectroscopy) frequency adjustment from 0.1 kHz to 5 kHz. Repeatedly performing charge and discharge tests in multiple cycles accelerated the aging characteristics of the batteries. This aging effect of the lithium-ion battery is related to physics-based model established in work by W. He, et al [140]. The tests were stopped when the batteries reached the end of life criteria, which was defined as a 30% loss of the rated charge capacity.

Figure 1 is the schematic diagram of the tested battery. The parameters of the schematic diagram included the Warburg impedance ( $R_W$ ) and the electrolyte resistance ( $R_E$ ), the charge transfer resistance ( $R_{CT}$ ), and the double-layer capacitance ( $C_{DL}$ ). The two parameters  $R_W$  and  $C_{DL}$  showed a negligible change over the aging process of the battery, and these were excluded from further analysis [141]. Figure 13 shows a typical response of the current and voltage behaviors during the charging and discharging cycles of battery No. 05.

In order to evaluate the prognostics of the battery, the SoH of the battery must be defined. Therefore, it is important to understand the clear definition of SoH, as the SoH will be the main prediction attribute of the proposed data-driven model, along with RUL. It is also important to note that in this work, all attributes from the test data were used as training attributes.

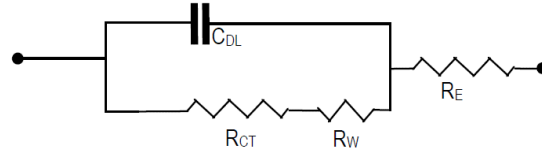


Figure 12. The schematic diagram of the tested battery.

The SoC of the battery indicates the reliability of the battery system. In the literature, the ratio between the available amount of charge and the maximum amount of charge is commonly referred to as the SoC [142]. In some cases, the available amount of charge can also be replaced by the rated capacity (or nominal capacity) provided by battery manufacturers. The SoC can be mathematically expressed as:

$$SoC = \frac{Q_{available}}{C_N} \quad (4)$$

where  $Q_{available}$  represents the available amount of charge and  $C_N$  represents the rated capacity from battery manufacturers.

However, there are problems using SoC as battery health measurement. First, the only way to derive the rated capacity of a battery is through experiments under a constant discharge rate within a controlled experimental environment. This reason explains the difficulty in using a rated capacity as a reference point in real-world applications [143]. As all conditions must be controlled in order to derive the capacity directly, is difficult to measure the capacity data from the battery sets. Second, SoC is not considered to have a strong correlation with battery capacity. This is important to note for making a long-term estimation of the battery's health because the capacity is the main indication of the battery's health, which fades over time.

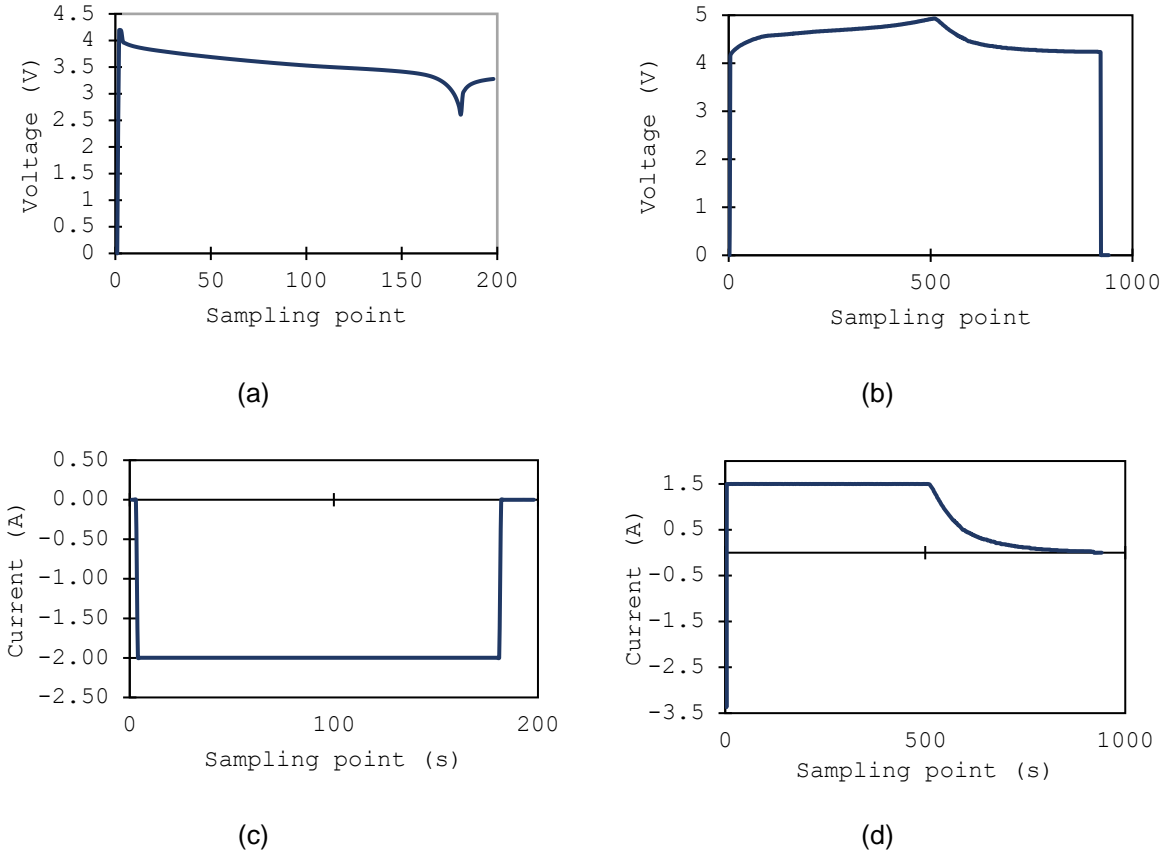


Figure 13. The current and voltage during the discharging and charging of battery No. 05. (a) Current of discharge, (b) Current of charging, (c) Voltage of discharging, and (d) Voltage of charging.

Many alternative SoC definitions have been studied to address the aforementioned issues. One definition is practical state-of-charge, or  $SoC_N$  [144]. This definition uses the maximum practical operational capacity, instead of the manufactured rated capacity, as the maximum amount of charge.  $SoC_N$  can be expressed as:

$$SoC_N = \frac{Q_{available}}{C_{max,p}} \quad (5)$$

where  $C_{max,p}$  represents the maximum practical capacity as measured from the operating battery at the current time.  $C_{max,p}$  may fade over time, because of the effect of battery aging. It is important to note that  $C_{max,p}$  can only be measured directly from the battery while it is operating.

For batteries, SoH can be generally defined as:

$$SoH = \frac{C_{max,p}}{C_N} \quad (6)$$

One of the most important tasks in prognostics health management of a battery is to accurately estimate the  $C_{max,p}$ , as  $C_{max,p}$  is required in both Equations (5) and (6) for SoC and SoH, respectively. The battery dataset used in this study, contained all the aging information of the battery, and the battery SoH was calculated from cycle 0 to cycle 168. As shown in Figure 14, the predicted SoH of battery No. 05 exponentially degraded as the cycle number increased. The acceptable predicted results must be within the 95% confidence bound [145]. The regression model for SoC and SoH estimation, which aimed to perform similar tasks, was also proposed by S. C. Huang, et al. [146]. This work introduced a new variable called, unit time voltage drop or  $V'$  to directly indicate the voltage drop of the battery cell as the prediction variable. This work delivered very interesting results. However, it is not within the scope of our deep learning approach. The work presented here aimed to use only existing test variables to train and generate the deep learning model for the SoH and RUL estimation of lithium-ion batteries based on equation (4).

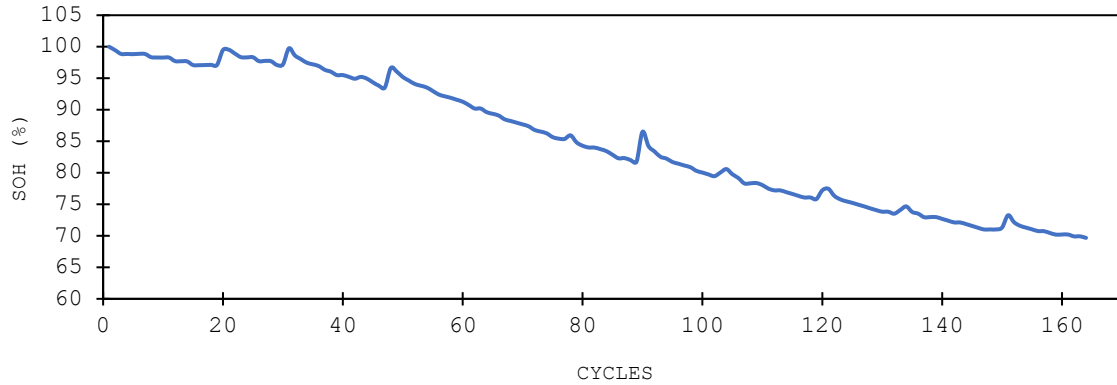


Figure 14. Predicted state of health of battery No. 05.

To evaluate the performance of the prediction model in this work, the root means square error (RMSE) was used for SoH, and the error of RUL cycle ( $E_{RUL}$ ) was employed for RUL. The following are the formulas of RMSE and  $E_{RUL}$ :

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n [x_i - \bar{x}_i]^2} \quad (7)$$

$$E_{RUL} = |RUL_{real} - RUL_{prediction}| \quad (8)$$

where  $n$  is the number of prediction datasets,  $x_i$  is the real value of testing and monitoring the battery capacity, and  $\bar{x}_i$  is the prediction value. RMSE and  $E_{RUL}$  are used as the key performance measures of the performance of all traditional machine learning approaches and the proposed deep learning algorithm. RMSE and  $E_{RUL}$  will be calculated within the testing phase of the modeling framework (as in Figure 9 from Chapter 1).

### **3.2. Data-Driven Predictive Prognostics Model for Lithium-ion Batteries Based on A Deep Learning Algorithm**

Data-driven PHM models, require large empirical data to create a high accuracy model of the systems. Traditional machine learning methods are called ‘shallow’ learning models. To compare deep learning or neural network-based algorithms to traditional ‘shallow’ learning algorithms, this work use shallow learning modeling algorithms with the lithium-ion battery data as well. The results will be illustrated later in this chapter. The algorithms used, include: Linear Regression (LR) [147] with the Akaike Information Criterion (AIC) [148, 149], k-Nearest Neighbors algorithm (k-NN) [150], SVM [151], and one layer ANN [7, 28-31].

#### **3.2.1. Related Works**

There have been many advancements from various disciplines to the PHM of lithium-ion batteries using various methods using both physic-based and data driven approach. Downey et al. proposed a physics-based prognostic approach that considered multiple concurrent degradation mechanisms [152]. Susilo et al. studied the estimation of the lithium-ion battery SoH with the combination of Gaussian distribution data and the least square support vector machines regression approach [153]. Mejdoubi et al. employed the Rao–Blackwellization particle filter to evaluate the aging condition of lithium-ion batteries, and to estimate the SoH and RUL of the battery system [154]. Bai et al. developed a generic model-free approach based on ANN and the Kalman filter, to help to improve the health management system of the lithium-ion battery [142]. Other filtering techniques, for example, particle filtering [155] or its variation of the unscented particle filtering technique [156] had been employed in the PHM aspect for lithium-ion batteries. Recently, Li et al. proposed the Gauss-Hermite particle filter (GHPF) technique for battery state-of-charge estimation, which is another extension of the particle filter technique, which not only improves the estimation accuracy but also reduces the number of sampling particles, which reduces the complexity

of the algorithm [157]. Another interesting work also aims to predict the health state of the lithium-ion battery, as proposed by Wang et al. This work employed the Brownian motion technique, which is the combination of the Kalman filter and the Gaussian distribution state-space technique, to determine battery prognostics based on the drift coefficient [158].

### **3.2.2. Deployment of Deep Learning for Prognostics Model of Lithium-ion Battery Data**

In the diagnostics and prognostics fields, the developing trend of employing the deep learning approach has evolved from fault detection and failure diagnosis to degradation pattern recognition and time series predictive analysis. The modeling methods have grown from using only a single algorithm such as DNN, CNN, and RNN, to the Hybrid model (or a combination of multiple layer types and traditional algorithms, as previously discussed in Chapter 2. The application range of using deep learning has also been expanding continuously over the years, from machinery, electrical, and electronics systems, to wind-power and high-end aerospace equipment.

For the experiment in this chapter, only DNN was employed to model the SoH and RUL of the battery data compared to traditional machine learning algorithms. Each of these deep learning algorithms has its own advantages and limitations. It has reported that ANN and DNN are more suitable for tackling one-dimensional data. CNN is able to make predictions relative well with multidimensional data, as it has adopted types of convolutional techniques. RNN is suitable for applications that deal with time series or time dependent data, and DNN is usually employed for extracting global features from fault data, which will be suitable for the lithium-ion battery data. Additionally, as aforementioned, the layers of CNN and RNN are more complex than those of DNN. Therefore, CNN and RNN require additional learning, which is their major drawback. These reasons make DNN more suitable for employment in real applications for most cases.

The prognostic model of the battery data using DNN was developed based on the basic framework from Chapter 1 (Figure 9). The experiment with the data was constructed by varying the number of dense layers in DNN. In this experiment, the number of hidden layers was varied to analyze the SoH of battery data until it delivered the minimum RMSE error. In addition, the dropout layer was also applied as the last layer before the output layer, to prevent the “overfitting” issue in the model. When the overfitting occurs, this means the model tend to “remember” instead of “learning” the pattern from the

actual data. Dropout is applied to the neural network so that some information is randomly removed and to prevent the network to copy the same information from the original data. The dropout layer is applied to the last layer of DNN, to randomly drop neurons during the model training, as shown in Figure 15. Each neuron is retained with a fixed probability,  $p$ , which is independent of other neurons. The neural network after being sampled, the so-called “thinned” network, will contain only the surviving neurons (Figure 15b).

By training a neural network with some dropouts, the whole network can be trained faster than training regular networks without dropout, because the network is thinned and requires less training time. The network then becomes less sensitive to some specific weights. This results in the network being better at generalization. In this work, a  $p$ -value of 0.25 is applied to the network, as suggested, to be the optimal dropout rate for the network to avoid overfitting but to still maintain the best prediction accuracy [159].

In this section, an analysis of battery No. 06, No. 07, and No. 18 degradation datasets obtained from the NASA Ames Prognostics Center of Excellence (PCoE) database [138] was studied to validate the effectiveness of using the DNN approach. The dataset of battery No. 05 was used as a training dataset for all algorithms.

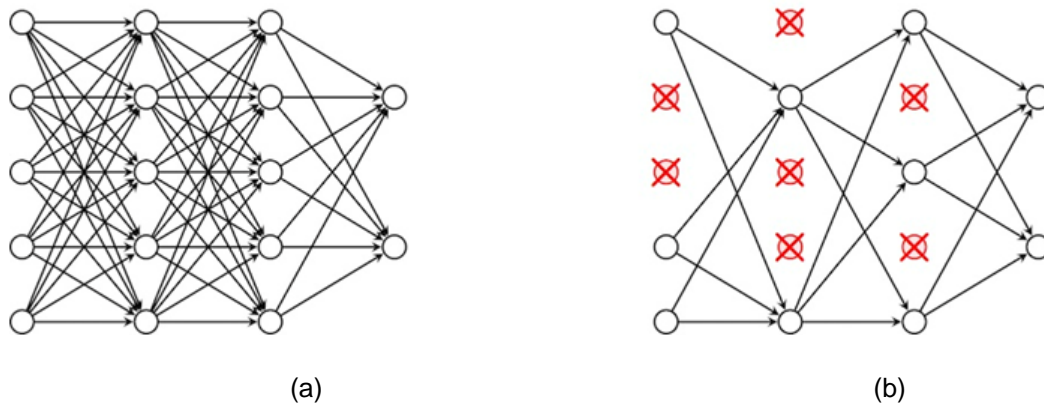


Figure 15. Dropout in deep neural network model.  
(a) A standard network with two hidden layers, and (b) the network after applying dropout.

### 3.2.3. The Proposed Deep Neural Network Prognostics Model for Lithium-ion Battery

By varying the number hidden layers of DNN from two layers to four layers, the RMSE results from Table 2 show that the best formation of DNN consists of three stacked dense or fully-connected hidden layers, with the Rectifier Linear Unit or ReLU activation function described as:

$$f(x) = x^+ = \max(0, x) \quad (9)$$

where  $x$  is the input to a neuron, and  $+$  represents the positive part of its arguments.

The proposed model architecture is illustrated in Figure 16. Additionally, it is important to note that there are some prediction fluctuations in the DNN model that are implemented by using the Keras library [160], which is the open-source neural network library that is employed in this experiment. To show the level of prediction fluctuations, each DNN experiment was performed with 10 trials then the reported RMSE was averaged. The RMSE results of the final model (three dense layers) are as shown in Table 4. The best result from the three layers trials was chosen to be the final model.

Table 4. RMSE results of each stacked hidden layer model.

No. of Hidden Layers	RMSE
2	3.815
3	3.247
4	3.275

Table 5. RMSE results of 10 trials for a model with three stacked hidden layers.

Trial	1	2	3	4	5	6	7	8	9	10
RMSE	3.917	3.877	3.667	3.507	3.487	3.321	3.296	3.253	3.249	3.247

The preliminary model of deep learning for PHM of the lithium-ion battery was developed, based on the deep neural network. The model architecture is illustrated in Figure 16. The objective of the experiment is to prove that the deep learning algorithm outperformed other traditional machine learning algorithms, and to provide a complete benchmark of SoH and RUL prediction for the lithium-ion battery.

### 3.2.4. Results for Model's SoH Estimation

In this experiment, the discharge data for all 164 cycles and 11,345 sample points from battery No. 05 were used for training. The SoH was calculated from the initial capacity at 1.9 AHr. Figure 17a–c shows the SoH estimation performance for batteries No. 06, 07, and 18, using k-NN, LR, SVM, ANN, and DNN respectively. The x-axis represents the cycles, and the y-axis represents the SoH. The triangle marked with a light blue line curve shows the true SoH, and the balance of the curves show the predicted SoH by the k-NN, LR, SVM, ANN, and the developed DNN.

It is important to note that the SVM formulation used in this work was based on the radial basis kernel function, with a regularization parameter of 200, and tolerance-of-loss function of 0.1. LR employed



the greedy algorithm with 0.1 minimum tolerance parameters. Additionally, k-NN employed the Euclidean distance measurement to evaluate distances among the neighbor data points.

Both the curve fitting of the trained DNN model and the RMSEs of the SoH estimated by the proposed model are better compared to the ones estimated by k-NN, LR, SVM, and ANN. In addition, after the batteries aged from the first cycle to the 164th cycle, it is seen that the proposed DNN approach could capture the degradation pattern better than the other algorithms. The input capacity in the developed DNN model could provide sufficient information for the stability of the SoH estimation when the batteries were aged. However, the lack of knowledge of other approaches resulted in an increasing error for the SoH estimation in the aged cycles. In addition, the performance of the capacity convergence by the proposed DNN approach was better, because the knowledge of capacity fade could be captured better by using the DNN model. Considering the result illustrated in Figure 17, it is also important to note that the results from batteries No. 06 performed slightly worse when compared to batteries No. 07 and 18. This could be because of the aging pattern of battery No. 06 being slightly different from the training dataset. Additionally, there was a greater distribution of the data of battery No. 06, compared to the other batteries.

The RMSE results between traditional machine learning, k-NN, LR, SVM, and ANN, along with the developed DNN, are shown in Figure 17 and Table 6. When comparing other algorithms to DNN, DNN was shown to perform the best among the four approaches in terms of predicting both data patterns and minimizing the RMSE. Additionally, the models constructed from each algorithm could also be observed in detail, from Table 7.

Other aspects of the DNN that should be considered further, are the optimizer of the network, and the loss function. DNN in this work was performed by employing “Adam” or Adaptive Moment Estimation, as an optimizer (see Appendix for more details). Additionally, based on the nature of the battery dataset in this work, the absolute error function was used as the loss function [161]. Absolute errors measured the mean absolute value of the difference between predicted and actual value. Absolute error loss is not the same as RMSE. While the RMSE measured the error from the whole SoH curve, the absolute error loss measured the difference between the actual and the prediction point only ‘at the end of life’ cycle. The absolute error formula used as the loss function can be expressed by equation (10):

$$\text{Absolute error loss} = \frac{1}{k} \sum_{i=1}^k |y_i - \bar{y}_i|^2 \quad (10)$$

where  $y_i$  and  $\bar{y}_i$  are, respectively, the predicted data and the input data of each iteration or epoch  $i$ , and  $k$  is the number of iterations. In this work, the total number of iterations was set to be equal to 1024, as suggested in reference [162].

Table 6. RMSE of the SoH estimation by using DNN and traditional machine learning algorithms.

RMSE	k-NN	LR	SVM	ANN	DNN
	5.598	4.558	4.552	4.611	3.427

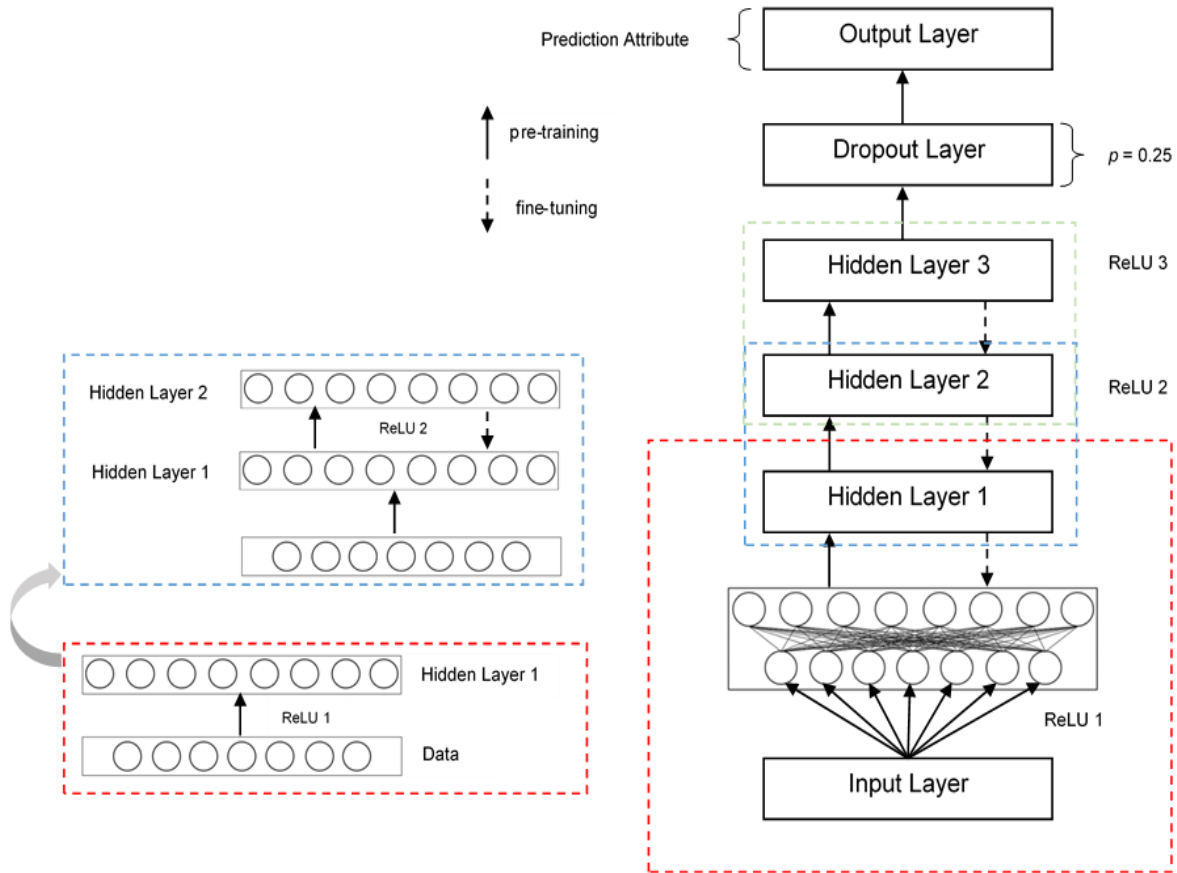
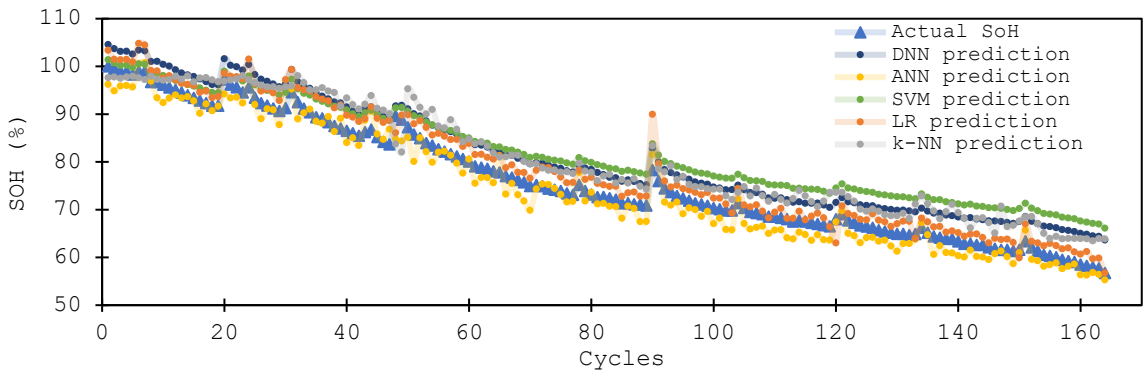
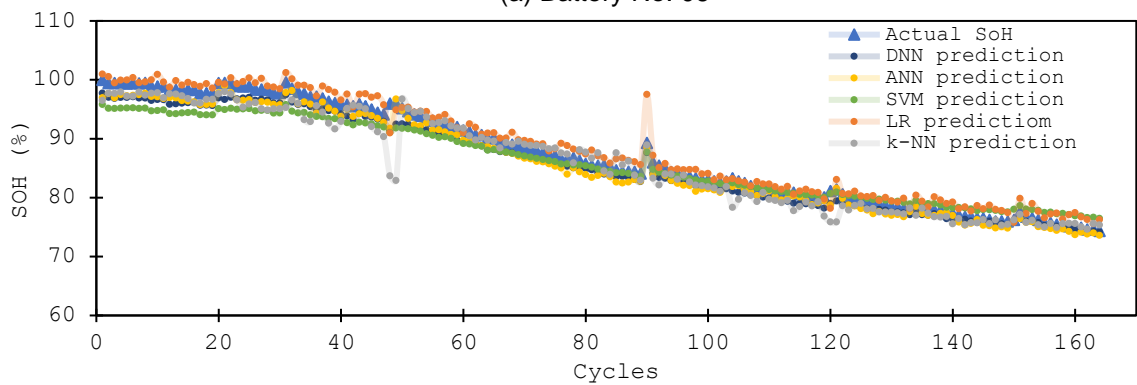


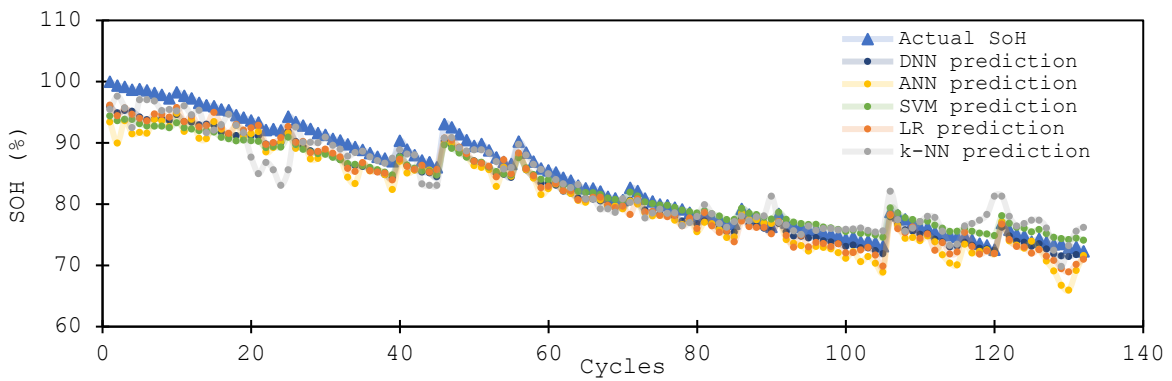
Figure 16. The proposed Deep Neural Network model for lithium-ion battery data.



(a) Battery No. 06



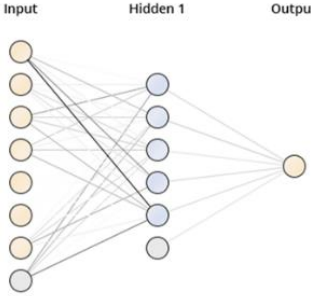
(b) Battery No. 07



(c) Battery No. 18

Figure 17. The SoH estimation with all algorithms for battery No. (a) 06, (b) 07, and (c) 18.

Table 7. Models created from the lithium-ion battery training dataset.

Algorithm	Model Description			
k-NN	22-Nearest Neighbor model for regression The model contains 624 examples with seven dimensions			
LR	$228.765 * \text{Voltage\_measured} + 237.439 * \text{Current\_measured} - 1.495 * \text{Temperature\_measured} - 1098.506 * \text{Current\_charge} + 50.156 * \text{Capacity} - 918.727$			
SVM	Total number of Support Vectors: 613 Bias (offset): -85.065 $w[\text{Voltage\_measured}] = 42686654.125$ $w[\text{Current\_measured}] = -17208.396$ $w[\text{Temperature\_measured}] = 243822393.316$ $w[\text{Current\_charge}] = 3952.097$ $w[\text{Voltage\_charge}] = 0.000$ $w[\text{Time}] = 0.000$ $w[\text{Capacity}] = 16430099.458$ number of classes: 2 number of support vectors: 613			
ANN	Node 1 (Sigmoid) Voltage_measured: -0.172 Current_measured: -0.448 Temperature_measured: 2.894 Current_charge: -1.458 Voltage_charge: 0.005 Time: 0.042 Capacity: -0.155 Bias: -2.726	Node 2 (Sigmoid) Voltage_measured: 1.954 Current_measured: 0.328 Temperature_measured: -1.124 Current_charge: -0.397 Voltage_charge: 0.036 Time: -0.014 Capacity: 0.943 Bias: -1.930	Node 3 (Sigmoid) Voltage_measured: 0.406 Current_measured: 1.254 Temperature_measured: 1.472 Current_charge: 1.391 Voltage_charge: -0.049 Time: -0.036 Capacity: 1.107 Bias: -1.055	
	Node 4 (Sigmoid) Voltage_measured: -3.468 Current_measured: -0.975 Temperature_measured: 0.080 Current_charge: -0.018 Voltage_charge: 0.044 Time: -0.020 Capacity: 2.457 Bias: -0.108	Node 5 (Sigmoid) Voltage_measured: -7.072 Current_measured: -0.455 Temperature_measured: 2.095 Current_charge: 2.091 Voltage_charge: -0.004 Time: 0.045 Capacity: -0.464 Bias: -4.078	Output Regression (Linear) Node 1: 1.278 Node 2: 1.460 Node 3: 0.865 Node 4: 1.214 Node 5: -1.134 Threshold: -0.819	
	Neural Network created: 			
DNN	Layer (Type)	No. of Hidden Nodes	No. of Parameters	Total parameters: 217 Trainable parameters: 217 Non-trainable parameters: 0
	dense_1 (Dense)	8	64	
	dense_2 (Dense)	8	72	
	dense_3 (Dense)	8	72	
	dropout_1 (Dropout)	8	0	
	dense_4 (Dense)	1	9	

### 3.2.5. Results for Model's RUL Estimation

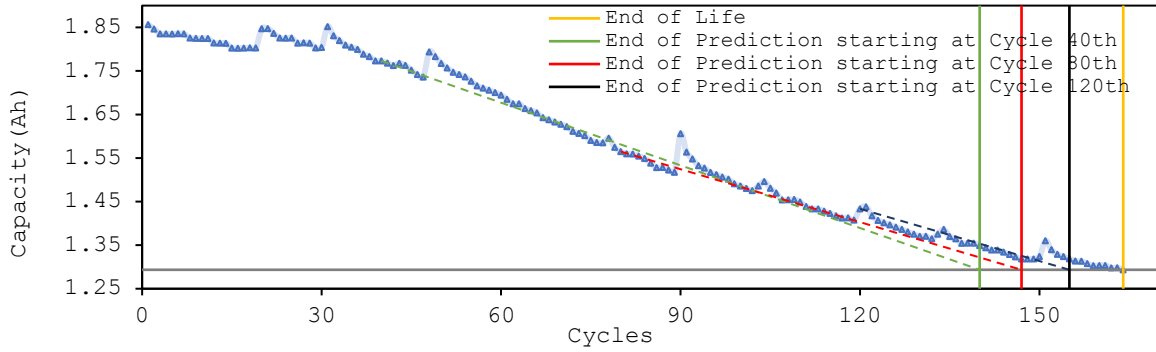
In addition to the SoH prediction of the batteries from the previous section, another aspect of the prognostic analysis of the battery data was to predict the RUL of the batteries. RUL prediction focuses on projecting the degradation results from a certain cycle until the EOL of the batteries, which is different from that of the SoH prediction, which focuses on detecting the pattern of degradation. In this experiment, the goal was to compare the RUL prediction result by using k-NN, LR, SVM, and ANN to the proposed DNN algorithm.

The RUL predictions experiments were performed from three different starting points, which were at the 40th cycle, 80th cycle, and the 120th cycle of battery No. 05. The threshold of the EOL of the battery data was set to be 30% remaining capacity or the 164th cycle. This was deemed to be best practices of the EOL threshold, for the battery to remain active. The data before the starting cycle was used as a training dataset to predict each starting cycle, and the error of RUL (Equation (8)) was calculated to compare the accuracy of each machine learning algorithm. The accumulated errors of the RUL results are as shown in Table 6, and the projection results of RUL are as shown in Figure 8. Note that the RUL results focus on projects, not to recognize the data's pattern

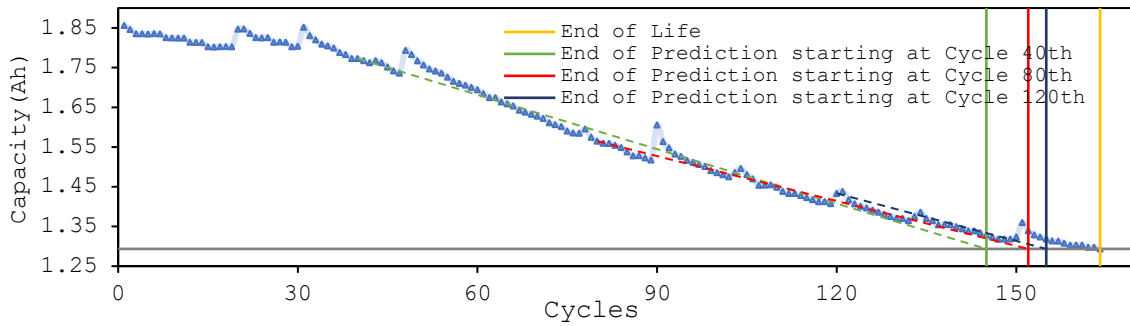
The results from Table 8 and Figure 18 show that, overall, the proposed DNN algorithm had a smaller error compared to the other machine learning algorithms. The prediction result at the 120th cycle of DNN and ANN are similar. However, DNN did performed better than ANN in terms of smaller error while having a smaller set of training data, i.e., when start the prediction at the later cycle mean more training data was used. As seen in the result, DNN provided a slightly better result when starting at the 40th cycle and the 80th cycle. Additionally, the trend of the results also showed that having more training data improves the prediction result for every algorithm in this experiment.

Table 8. The error of RUL estimation by using DNN and traditional machine learning algorithms.

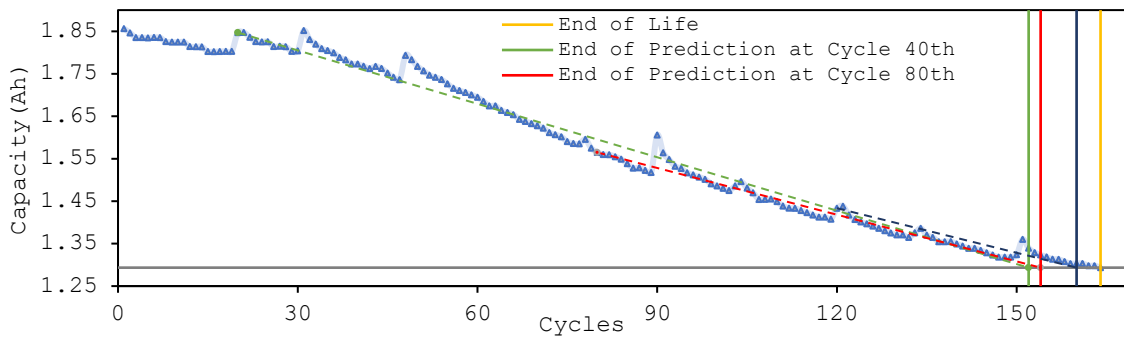
Error of RUL	Starting Points	k-NN	LR	SVM	ANN	DNN
	40th cycle	24	19	12	6	5
	80th cycle	17	12	10	3	2
	120th cycle	19	9	4	1	1



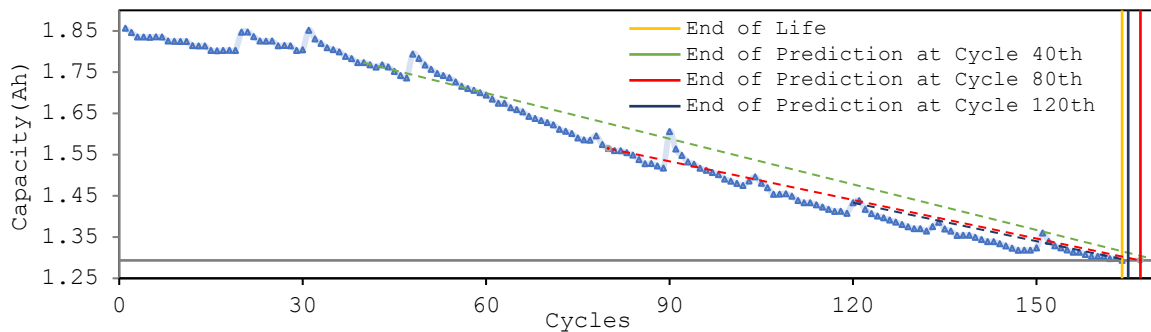
(a) The RUL estimation using k-NN



(b) The RUL estimation using LR

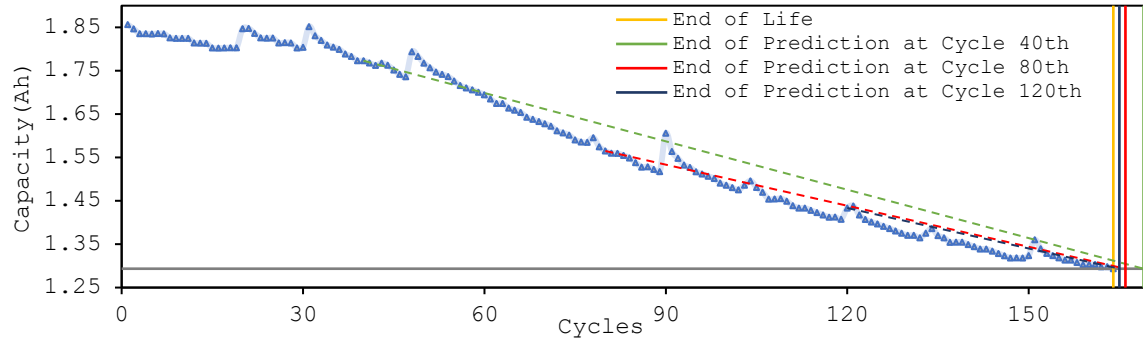


(c) The RUL estimation using SVM



(d) The RUL estimation using ANN

Figure 18. (a - d) The RUL estimation of battery No. 05 using different learning algorithms.



(e) The RUL estimation using DNN

Figure 18. (e) The RUL estimation of battery No. 05 using different learning algorithms (continued).

### 3.3. Result Discussion

From the experimental results in the previous sections, it is seen that the proposed DNN algorithm predict SoH and RUL with smaller error compared to k-NN, LR, SVM, and ANN in these specific lithium-ion battery datasets. However, two points should be addressed. First, the DNN proposed in this work can use (learn) the degradation pattern to predict the SoH. However, the DNN method was comparable to ANN for predicting the RUL. This is expected because of the fundamentals of DNN being based on ANN. It is also important to note the smaller training dataset for DNN, the DNN performed better overall. In more detail, from the RUL prediction results, the DNN provided better results when started from the smaller amount of training data at the 40th and 80th cycles, compared to the typical neural network.

The results obtained from this work show that the deep learning algorithm is effective and suitable for predicting prognostic and diagnostic data modeling, particularly in the prognostics of the battery data set. The prognostic results can eventually aid in condition-based monitoring of maintenance activities, to obtain the best time to replace the batteries without causing a long downtime in the main systems. Based on this experiment, the downsides of using a deep learning algorithm include: (1) a higher computational time and (2) more computational resources are required by DNN than for the other two algorithms. These drawbacks are also true for other deep learning algorithms as well. This conclusion is that deep learning is more suitable for studies that require higher accuracy, however, may not be suitable for applications that need real-time processing. In the battery PHM application, real-time processing is typically not critical, because the prediction should be before the end-of-life of the batteries. In addition, with the

advancement of the computational tools, the real-time processing issue could be minimized, and the computational time will be improved.

The deep learning model in this paper was only developed based on the Deep Neural Network algorithm (DNN). Other more complex deep learning algorithms have been developed over the years, such as the Convolutional Neural Network (CNN), the Recurrent Neural Network (RNN), and the Long Short-Term Memory network (LSTM). It is important to note that some researchers have used the LSTM network for similar battery prognostic data, to predict the remaining useful life (RUL) of the battery [98]. Although this has already been done, it is incomparable to the experiments in this work. Based on the fact that the experiment on LSTM in the literature implemented a different dataset. More importantly, the experiment only focused on testing the LSTM network in the model and did not provide a complete comparison to other models that use traditional machine learning algorithms. This leaves gaps to be explored in the future, particularly with benchmarking all deep learning algorithms.

In addition to benchmarking deep learning approaches with other machine learning algorithms, in the future, physical experiments can also additionally be explored to bridge the gap between data-driven models and physics-based models for PHM applications. Applying physical understanding to data-driven approach can help to identify the crucial parameters that effect the aging behavior of the batteries as well as help to better interpret the data-driven models that only directly derive from the data points or raw data. Then, data-driven models will be employed to help in easing the modeling complexity in physics-based models of lithium-ion batteries. Thus, accurate battery models that mimic physical operation of the battery in real-world applications could be obtained without the need for extensive expense of time and computational resources.

### **3.4. Chapter Summary**

This work aims to accomplish two tasks. First, a complete benchmarking of the data-driven model by using a machine learning algorithm with the battery prognostic data is made. Second, a preliminary data-driven model is developed by using a deep learning algorithm for the prognostic data. This paper presented as a benchmark, the prognostic data-driven model for battery data using machine learning algorithms and based on the results from the case studies. In more detail it has been shown that the deep learning algorithm provides a promising prediction tool and modeling of prognostic data, especially in the



battery prognostic. Based on the accuracy archived, it has been shown that the traditional physics-based model may be replaced by data-driven models in the near future, in various fields and applications. The reliable data-driven model has many advantages over a traditional physics-based model. The first major advantage is that it overcomes the complexity of the physics-based model. In the future, the predictive models may be able to predict the health of batteries without experts in the field. The second advantage is that data-driven models can be employed in real-time situations, because of their shorter computational time, when compared to physics-based models in general. The last point is that the data-driven model is more cost-effective to construct and to employ in real applications. As an example, a data-driven model can be generated and monitored by using only regular personal computing devices, without the need for exclusive and excessive expert resources. This future trend of data-driven models is in line with the recent achievement of deep learning algorithms and artificial intelligence. These methodologies are believed to be the main approaches in the further development of data-driven models. However, the accuracy of prediction and the higher performance of using deep learning algorithms also come with the drawback of higher computational time. With rapid advancements in technology, the computational time could be substantially reduced. The future direction of this work will focus on developing a hybrid-deep learning model that could be universally applicable to multiple types of prognostic data.

## **4. DEEP NEURAL NETWORK FEATURE SELECTION APPROACHES FOR DATA-DRIVEN PROGNOSTIC MODEL OF AIRCRAFT ENGINES**

### **4.1. Deep Neural Feature Selection Approach for Modeling RUL Prediction of Aircraft Engines Data**

Modern computational capability has become more powerful over the past decades. This has induced a new trend of using various data-driven models in many fields. Even though modern computers can complete complex tasks, researchers are still searching for solutions to reduce the computational time and complexity of the data-driven models to increase the likelihood that the models can be applied in real-time operations.

The same challenge has also been applied to a certain type of aerospace data, which in this case, is the estimation of Remaining Useful Life or RUL of the aircraft gas turbine engines. The main purpose of this work is to prove that a particular group or a set of prognostics features (attributes or variables) from the aircraft gas turbine engines data can be selected before the training phase of Artificial Neural Network (ANN) modeling in order to reduce the complexity of the model. The same assumption may be applicable to the Deep Neural Network (DNN) models. For example, it might also be applied to other complex deep learning models, i.e., Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), and their variations as well.

To validate this theory, the prognostics of aircraft gas turbine engines dataset or Commercial Modular Aero-Propulsion System Simulation (C-MAPSS) dataset derived from NASA Ames Prognostics Center of Excellence (PCoE) [163] was used to develop preliminary vanilla ANN models with selected features using different feature selection methods. In addition, to prove that similar assumptions can also be deployed to other deep learning algorithms, the Deep Neural Network or DNN models have also been developed based on selected features derived from the ANN validation models. The final goal was to determine which feature selection method was the most suitable for the deep learning model in general to predict prognostics state or remaining useful life for aircraft gas turbine engines data. Results from various feature selection methods were compared to the one that is using original features. The ANN and DNN models with selected features were studied and compared based on their performance.

Based on the aforementioned goal, the summary of the main contributions of this experiment are:

1. Extract meaningful features for neural network-based and deep learning data-driven models from the C-MAPSS dataset.
2. Suggest the novel neural network-based feature selection method for aircraft gas turbine engines RUL prediction.
3. Develop deep neural network models from selected features.
4. Show how the developed methodology can improve the RUL prediction model by comparing its performance/error and complexity to the model derived from original features.

#### **4.1.1. Feature Selection Methods for Neural Network Architectures**

In prognostic AI predictions with large data, feature extraction of raw data from the sensors can reduce enhance learning of the system. The feature extraction usually involves signal processing and analysis in the time or frequency domains. The purpose is to transform raw signals into more informative data [164]. In more detail, feature extraction is the process of sensor signals into data that results in proper training of AI systems. In contrast, the purpose of feature selection is to extract a particular set of features in the dataset that is believed to be more relevant for modeling. These feature selection processes are typically executed after the feature extraction and occur in between pre-processing and the training or pre-training phase of the data modeling framework.

Three common feature selection strategies have been discussed in the literature: (1) filter approach, (2) wrapper approach, and (3) embedded approach. This paper will only discuss the filter and wrapper approaches. Figure 3 shows the processes flow and role difference role of feature extraction and feature selection in the data modeling process.

Filter methods use statistical, correlation, and information theory to identify the importance of the features. The performance measurement metrics of filter methods usually use the local criteria (which is the different statistical matric from each algorithm) that do not directly relate to model performance [165].

There are currently multiple baseline filter methods commonly sued for feature selection processes. However, the result from the experiments showed in section 4.3 that only the correlation-based methods were suitable for the case study data. This is because correlation-based methods evaluate the feature with a direct relationship to the target variable. In more detail, the correlation-based

filter methods make selections based on the modeling objectives, which can lead to better filtering of suitable to the data with the target variable. The correlation-based filter method included in this work is Pearson correlation [166, 167]. Additionally, the result from other statistical-based methods, namely Relief algorithm, Deviation selection, SVM selection, and PCA selection [168], was also included to provide comparison.

Wrapper methods use a data-driven algorithm that performs the modeling for the dataset to select the set of features that yield the highest modeling performance [169]. Wrapper methods are typically more computationally intensive compared to filter methods. There are four main baseline wrapper methods [169]: (1) forward selection, (2) backward elimination, (3) brute force selection, and (4) evolutionary selection.

Forward selection and backward elimination are search algorithms with different starting and stopping conditions. The forward selection starts with an empty selection set of features, then adds an attribute in each searching round. Only the attribute that provides the highest increase in performance is retained. Afterward, another new searching cycle is started with the modified set of selected features. The searching of forward selection stops when the added attribute in the next round does not further improve the model performance. The best way to actually search or select the best set of features or attributes is to use the method called “Brute Force” selection. Unlike forward selection and backward elimination, brute force selection will not stop searching based on any criteria other than getting the best performance. However, it is unlikely to test on all combination of features or attributes in practical application as it is required computational time and resource. Therefore, it is uncommon to use brute force selection in a larger dataset or typical real-world dataset.

In contrast, the backward elimination method performs in the reverse process. Backward selection starts with a set of all attributes, and then the searching processes continue to eliminate attributes until the next set of eliminated attributes does not provide any further improvements in modeling performance. The brute force selection method uses search algorithms that try all combinations of attributes. evolutionary selection employs a genetic algorithm to select the best set of features based on the fittest function measurement [170]. Because of computational and time limitations, brute force

selection could not be included in this experiment. Only forward selection, backward elimination, and evolutionary selection were implemented [171].

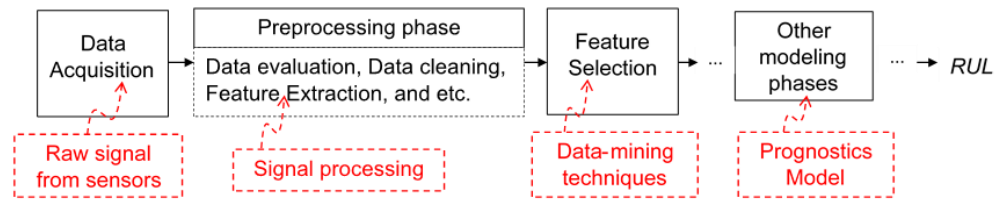


Figure 19. Role of feature extraction and feature selection in the prognostics modeling process.

#### 4.1.2. C-MAPSS Aircraft Engines Data

Commercial Modular Aero-Propulsion System Simulation (C-MAPSS) is a simulation tool that was used to generate the turbofan engine degradation run-to-failure test dataset. This test dataset was derived from the NASA Ames prognostics data repository [163]. The C-MAPSS dataset is one of the most popular benchmark datasets used in the prognostics and diagnostics research community. This dataset provides a set of editable input parameters to simulate various operational conditions for aircraft gas turbine engines [38]. Note that varying or adjusting the simulated input parameter is beyond the scope of this work. All the experiments performed in this work only used the existing simulation data provided by NASA Ames. The operational conditions include sea-level temperature, Mach number, and altitude. The C-MAPSS dataset includes four sub-datasets described in Table 9.

Table 9. C-MAPSS dataset description [38].

Description	C-MAPSS			
	FD001	FD002	FD003	FD004
Number of training engines	100	260	100	248
Number of testing engines	100	259	100	248
Operational conditions	1	6	1	6
Fault modes	1	1	2	2

Each sub-dataset FD001, FD002, FD003, and FD004 contains several training engines with run-to-failure information and several testing engines with information terminating before failure is observed. In reference to the operating conditions, each dataset can have one or six operational conditions based on altitude (0–42,000 feet), throttle resolver angle (20–100°), and Mach (0–0.84). As for fault mode, each dataset can have one mode or two modes, which are, High Pressure Compressor (HPC) degradation and Fan degradation.

Sub-dataset FD002 and FD004 are generated with six operational conditions, which are believed to be a better representation of general aircraft gas turbine engines operation compared to FD001 and FD003, which could be generated from only one operational condition. In this study, the data from the FD002 set was selected as a training dataset. In the model validation set-up (which will be described in Section 3.2), the wrapper methods required roughly 2 to 3 weeks to complete the run. In addition, the number of data points used in feature selection validations and model training—in both ANN feature selection validation and DNN model training was kept constant. The experiments have been designed to demonstrate the effectiveness of the feature selection methods used for neural network-based algorithms.

There are 21 features included in the C-MAPSS dataset for every sub-dataset. These attributes (note that the word ‘attribute’ and ‘feature’ might be used interchangeably throughout the dissertation) represent the sensor signals from the different parts of the aircraft gas turbine engines, as seen in Figure 5 [172]. Short descriptions of the features and the plots of all 21 sensor signals of sub-dataset FD002 are seen in Figure 21.

It has been suggested by multiple literature references to normalize the raw signal before performing modeling and analysis [90, 94, 173]. Figure 22 shows the data signals before and after applying z-normalization:

$$\tilde{x}_t^{ij} = \frac{x_t^{ij} - \min(x^j)}{\max(x^j) - \min(x^j)} \quad (11)$$

where,  $x_t^{ij}$  denotes the original  $i$ -th data point of  $j$ -th feature at time  $t$  and  $x_j$  is the vector of all inputs of the  $j$ -th feature. Each attribute value was normalized individually and scaled to the same range across all data points.

From the dataset, aircraft gas turbine engines start with various initial wear levels, but all are considered to be at a “healthy state” at the start of each record. The engines begin to degrade at a point in time at higher operation cycles until they can no longer function normally. This is considered as the time when the engine system is being in an “unhealthy state”. The training datasets have been collected over the time of run-to-failure information to cover the entire life until the engines fail.

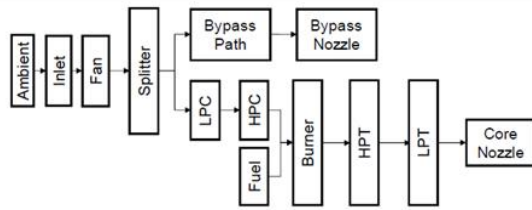
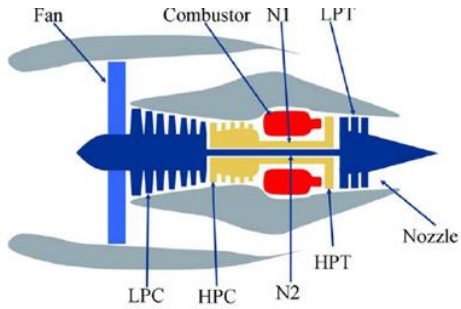
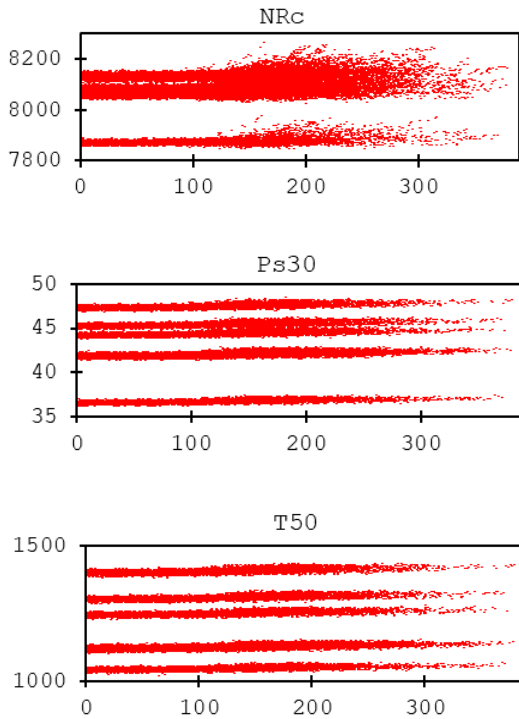


Figure 20. Engine and sensor points (left) and engine parts modules connections (right) [172].



Symbol	Description	Units
T2	Total temperature at fan inlet	°R
T24	Total temperature at LPC outlet	°R
T30	Total temperature at HPC outlet	°R
T50	Total temperature at LPT outlet	°R
P2	Pressure at fan inlet	psia
P15	Total pressure in bypass-duct	psia
P30	Total pressure at HPC outlet	psia
Nf	Physical fan speed	rpm
Nc	Physical core speed	rpm
epr	Engine pressure ratio (P50/P2)	--
Ps30	Static pressure at HPC outlet	psia
phi	Ratio of fuel flow to Ps30	pps/psi
NRf	Corrected fan speed	rpm
NRc	Corrected core speed	rpm
BPR	Bypass Ratio	--
farB	Burner fuel-air ratio	--
htBleed	Bleed Enthalpy	--
Nf_dmd	Demanded fan speed	rpm
PCNfR_dmd	Demanded corrected fan speed	rpm
W31	HPT coolant bleed	lbm/s
W32	LPT coolant bleed	lbm/s

Figure 21. Example of sensor signals (NRc and Ps30) and all feature descriptions.

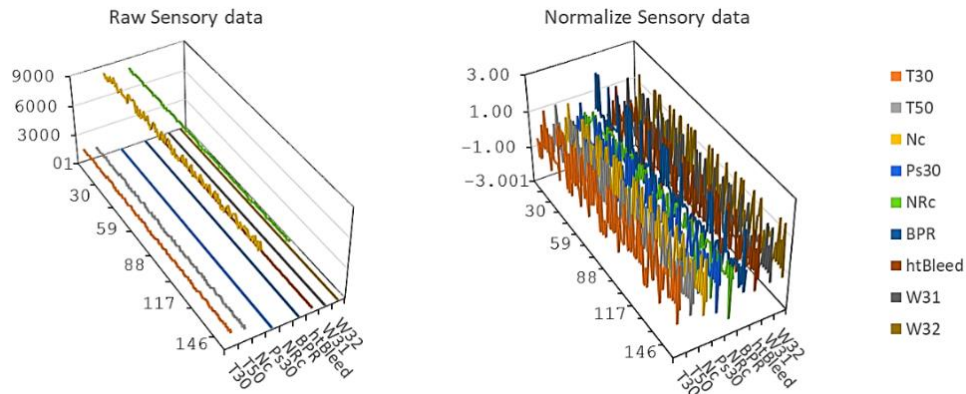


Figure 22. Example of before (left) and after (right) z-normalization.

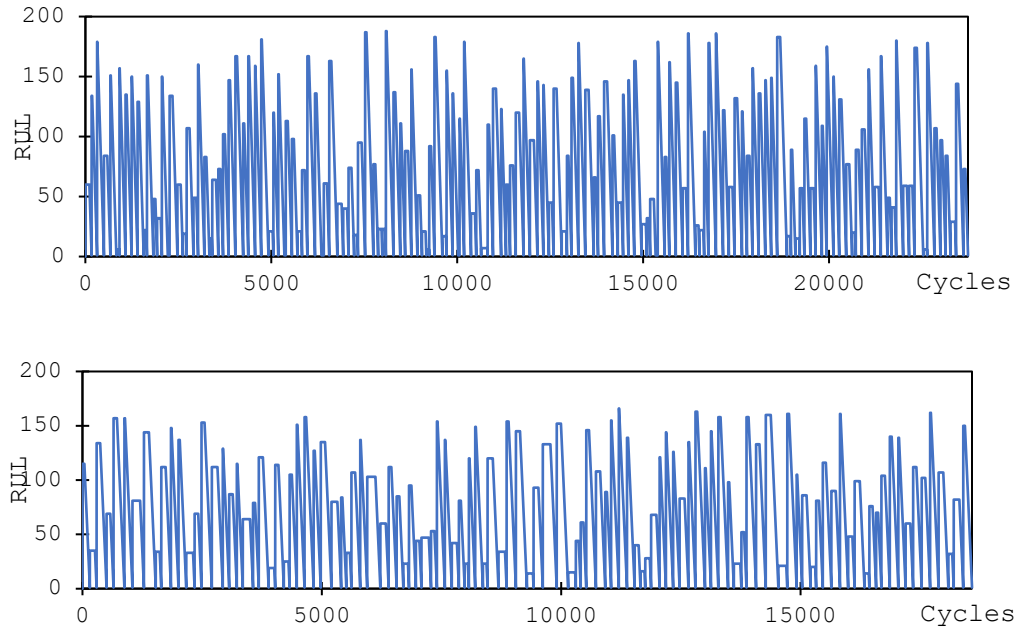


Figure 23. RUL curve of all testing engines FD002 (top) and FD004 (bottom).

The RUL curves of all unseen or test data sets containing testing engines from FD002 and FD004 datasets illustrates in Figure 23. The unseen data contained the new set of input data that can be used to validate the model. Figure 24 show the example of RUL curves from one degradation engine from the FD002 and FD004 dataset. The same degradation behavior is also applied to the training set. These RUL curves represent the health state or prognostic of the aircraft gas turbine engines over cycles until the end-of-life or the point that the aircraft gas turbine engines can no longer operate normally degradation behavior of the aircraft gas turbine engines can be seen clearer from Figure 24. It noted that the RUL is a constant cycle until it reaches a critical point,  $R_{th}$ , when the performance of the engine starts to degrade. In the degradation phase, the RUL is represented by a linear function. Thus, the entire RUL curve is identified as a piece-wise linear degradation function. The critical points of the aircraft gas turbine engines were predefined based on the condition described by the data source—NASA Ames prognostics data repository [163].



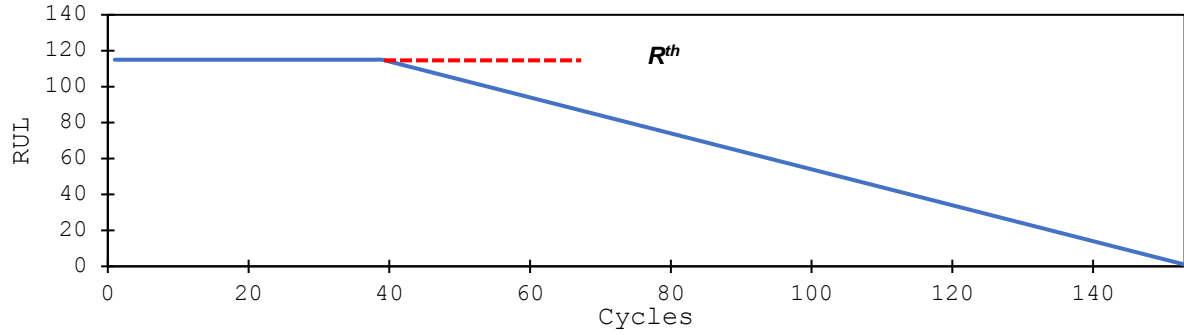
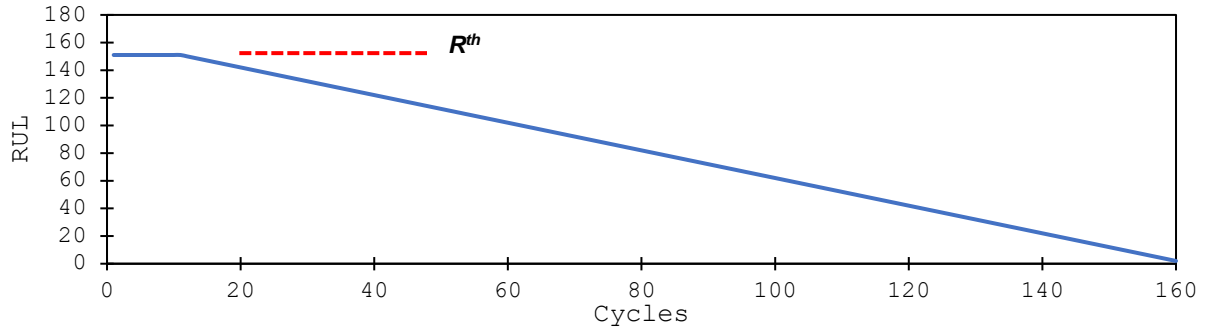


Figure 24. Example of RUL curve of one testing engine FD002 (top) and FD004 (bottom).

To measure and evaluate the performance of the models with selected features, root means square error (RMSE) and the scoring algorithm were used[172] .

RMSE is commonly used as a performance indicator for regression models. The following is the formula of RMSE:

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n [x_i - \bar{x}_i]^2} \quad (12)$$

where  $n$  is the number of prediction datasets,  $x_i$  is the real value, and  $\bar{x}_i$  is the prediction value. In this case, the  $x$  parameters refer to the data points in the RUL curve while  $x_i$  is the actual RUL value and  $\bar{x}_i$  is the RUL value predicted by our models.

The scoring algorithm is as described in the formula below:

$$s = \begin{cases} \sum_{i=1}^n e^{-\frac{d}{a_1}} - 1 & \text{for } d < 0 \\ \sum_{i=1}^n e^{-\frac{d}{a_2}} - 1 & \text{for } d \geq 0 \end{cases} \quad (13)$$

where  $s$  is the computed score,  $n$  is the number of units under test (UTT),  $d = \hat{t}_{RUL} - \hat{t}_{RUL}$  or Estimated RUL—True RUL, while  $a_1 = 10$  and  $a_2 = 13$ .

Note that the value of  $a_1$  and  $a_2$  here are standard value that have been used in all other works performing similar experiments. In summary,  $a_i$  is the difference between predicted and observed RUL values and  $s$  is summed over all examples. From the formula, the scoring matrix penalizes positive errors more than negative errors as these have a higher impact on maintenance policies. Also, note that the lower score reflect a better prediction performance of the model [172].

#### **4.1.3. Related Works**

Multiple deep learning algorithms have been used to generate data-driven models to predict RUL for C-MAPSS aircraft gas turbine engines data. It seen from the literature [90, 94, 111, 173, 175-179] that the most suitable deep learning algorithm for training the high accuracy C-MAPSS models is the Long-Short Term Memory Recurrent Neural Network (LSTM). The hybrid deep neural network layers with LSTM is also an ongoing investigation and experiment on the C-MAPSS dataset. This approach believes to achieve higher accuracy among other algorithms that have been employed. The most important drawback of the hybrid models is the high complexity of the model architectures. These models can also have vast variations and architectures. To reduce the complexity of the model it is possible to limit the number of input nodes. In addition, feature selection methods can be used to reduce the complexity. There are many publications on applying ANN-based or deep learning algorithms to C-MAPSS aircraft gas turbine engines data. However, all previous works have never introduced the feature selection approaches into their model architectures. Also, the usefulness of any particular feature selection methods has not been addressed in any prior works.

Chen Xiongzi, et al., (2011) conducted a comprehensive survey of the three main data-driven methods for aircraft gas turbine engines, namely particle filtering methods, neural network, and relevant vector machine methods. Their comprehensive study showed that the neural network perform best [175]. Mei Yuan, et al., (2016) applied RNN network methods for fault diagnosis and estimation of the remaining useful life of engines [94]. Faisal Khan, et al., (2018) used particle filter algorithms to generate the arbitrary input data points before training their models with neural networks. Unlike, the vanilla neural network algorithm, their models employed radial basic function (RBF) as activation function instead of the original sigmoid function the result from using RBF as activation function was better compared to sigmoid function [173]. Xiang Li, et al., (2018) applied the Convolutional Neural Network (CNN) as a time window

approach to generate a feature extraction model of engine data [90]. Ansi Zhang et al., (2018) proposed a supervised domain adaptation approach by exploiting labeled data from the target domain that aims to fine-tune a bi-directional Long-Short Term Memory Recurrent Neural Network (LSTM) previously trained on the source domain [176]. Zhengmin Kong et al., (2019) also developed the models based on CNN. They employed CNN as part of the network layers in their experiment and proposed the hybrid models by combining the CNN layers with LSTM layers. Their approaches achieved relatively high accuracy in SoH predictions over the other standard methods [177]. Other works previously published [111, 177, 179] mostly focused on adopting the LSTM network and proposing new models without addressing the complexity reduction in their approaches. While each work proposed the different network architectures and the performances of the models have been improved over time, there remains the need to reduce the complexity of ANN-based models. This work aims to address the issue of a selection approach to reduce learning times.

#### **4.2. Methodology**

In this section, the details of the auto-encoder deep neural network used in this work will be discussed. The problem will be defined, and all notations will also be defined, as well as the illustration of how the proposed deep neural network architecture can be applied to predict RUL of aircraft gas turbine engines with feature selection and neural network modeling framework mentioned in Figure 11 Chapter 2.

The experiment only used DNN with auto-encoder as a modeling algorithm. All encoded and decoded processes were designed to occur inside the hidden layers of the network through parameterized functions [10, 11]. The construction of DNN with auto-encoder is detailed in Figure 25. Unlike the ANN that uses the sigmoid function as an activation function, the proposed DNN layers used Rectified Linear Units (ReLU) as an activation function. The ReLU function has demonstrated to achieve better general regression training for deeper networks compared to other activation functions such as the logistic sigmoid and the hyperbolic tangent (tanh) [11]. Therefore, the ReLU function was selected for modeling Remaining Useful Life (RUL) prediction for our PHM data while the ANN with sigmoid function has been used as a validation algorithm for feature selection methods.

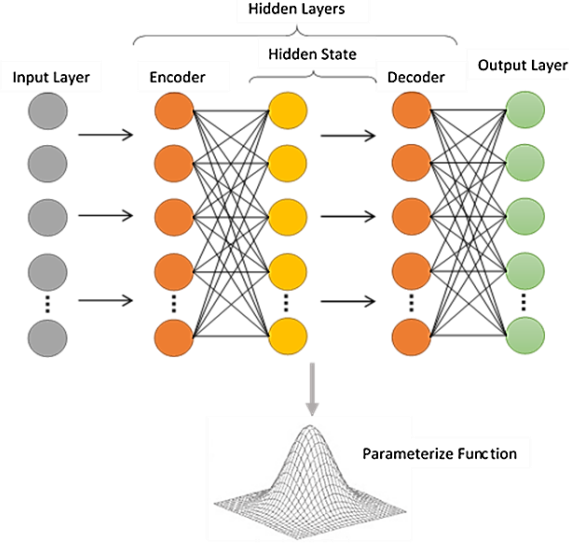


Figure 25. Auto-encoder Deep Neural Networks construction.

#### 4.2.1. Problem Statement

Starting with the raw data, which is denoted as,  $D_S = \{(x_S^i, y_S^i)\}_{i=1}^{N_S}$ , the data contains  $N_S$  training sample where  $x_S^i \in \mathcal{X}_S$  is a feature with a length of  $T_i$  and  $q_S$  is the number of features, in which,  $x_S^i = \{x_t^i\}_{t=1}^{T_i} \in \mathbb{R}^{q_S \times T_i}$ . In addition,  $y_S^i \in \mathcal{Y}_S$  is denoted as Remaining Useful Life (RUL) also with the length  $T_i$  (feature space and RUL space are within the same length) with  $y_S^i = \{y_t^i\}_{t=1}^{T_i} \in \mathbb{R}_{\geq 0}^{T_i}$ . where  $t \in \{1, 2, \dots, T_i\}$ ,  $x_t^i \in \mathbb{R}^{q_S}$ , and  $y_t^i \in \mathbb{R}_{\geq 0}$ , represent the  $t$ -th measurement of all variables and RUL label, respectively. Similarly, the estimated target domain,  $D_T = \{x_T^i\}_{i=1}^{N_T}$  where  $x_T^i \in \mathcal{X}_T$  and  $\mathcal{X}_T \in \mathbb{R}^{q_T \times T_i}$  with no labels. The source and target domain,  $D_S$  and  $D_T$ , are assumed to possibly have a different probability distribution,  $P(X_S) \neq P(X_T)$ . The primary goal is to define a function  $g$  that can derive or learn from the source data that can approximate the corresponding RUL for the target domain at the testing time, such,  $y_T^i \approx g(x_T^i)$ , with the preliminary assumption that mapping between input ( $x$ ) and output ( $y$ ) is somehow similar across all domains.

#### 4.2.2. Deep Neural Network Architecture

While there are existing deep learning algorithms that have been proposed to predict PHM of aircraft gas turbine engines data modeling [90, 94, 111, 173, 175-179], this work focuses on using a deep neural network with an auto-encoder.

The DNN used in this work focused on the feedforward architecture by the H2O package in Python API [21]. H2O is based on multi-layer feedforward neural networks for predictive modeling [180]. The following are some of the H2O DNN features used for this experiment.

- Supervised training protocol for regression tasks
- A multi-threaded and distributed parallel computation that can be run on a single or a multi-node cluster
- The automatic, per-neuron, adaptive learning rate for fast convergence
- Optional specification of the learning rate, annealing, and momentum options
- Regularization options to prevent model overfitting
- Elegant and intuitive web interface (Flow)
- Grid search for hyperparameter optimization and model selection
- Automatic early stopping based on the convergence of user-specified metric to a user-specified tolerance
- Model check-pointing for reduced run times and model tuning
- Automatic pre- and post-processing for categorical numerical data
- Additional expert parameters for model tuning
- Deep auto-encoders for unsupervised feature learning

In the proposed DNN model, deep neural network layers are used to extract the temporal features from the time length,  $T_i$ . The hidden state units of the neural consist of, the hidden state vector  $h_{t-1} \in \mathbb{R}^h$ , input vector,  $x_t^i \in \mathbb{R}^i$ , and the activation function,  $f$ , where  $h$  is the dimension of hidden state layers. All operations in DNN layers can be written as:

$$i_t = f(W_i x_t^i + W'_i h_{t-1} + b_i) \quad (14)$$

$$o_t = f(W_o x_t^i + W'_o h_{t-1} + b_o) \quad (15)$$

where  $i$  and  $o$  represent input and output states.  $W$  and  $W'$  are matrices of updated weights and weights from the hidden state, and  $b$  is the bias vector.

Unlike in vanilla ANN, in the proposed DNN, the activation function  $f$  is the Rectifier Linear Unit or ReLU function [23] instead of the sigmoid function. The DNN activation function can be represented as;

$$f(\alpha) = \max(0, \alpha) \in \mathbb{R}_+ \quad (16)$$

where, in this case,  $\alpha$  represents the state functions (Formulas (14) and (15)) that firing into the input neural.

Another important aspect of the DNN model architecture is the loss function, denoted by,  $\mathcal{L}$ . For this work, the Huber loss function was selected because it [181] has been shown to work best in terms of accurately projecting the RUL,  $y_s^i \in \mathcal{Y}_s$ , of the source domain,  $D_s$ . The Huber loss function can be described as;

$$\mathcal{L}_y^i(\theta_f, \theta_y) = \begin{cases} \frac{1}{2} \|\hat{\mathcal{Y}}_t^i - \mathcal{Y}_t^i\|_2^2 & , \text{for } \|\hat{\mathcal{Y}}_t^i - \mathcal{Y}_t^i\|_1 \leq 1 \\ \|\hat{\mathcal{Y}}_t^i - \mathcal{Y}_t^i\|_1 - \frac{1}{2} & , \text{Otherwise} \end{cases} \quad (17)$$

where,  $\theta_f$  is the space representation of the target input that mapped through the feature extraction layers into a new space. In addition,  $\theta_y$  is the domain regression space generated by logistic repressor [181], and,  $\hat{\mathcal{Y}}_t^i$  is RUL prediction from the source domain.

The objective in training DNN is to minimize the prediction loss,  $\mathcal{L}_y^i$ , which can be described by;

$$\min_{\theta_f, \theta_y} \left[ \frac{1}{N_s} \sum_{i=1}^{N_s} \mathcal{L}_y^i(\theta_f, \theta_y) \right] \quad (18)$$

The DNN model used in this work is depicted in Figure 26. This DNN model architecture is trained to predict for each input,  $x^i$ , real value  $y^i$  and its domain label  $d^i$  for the source domain and only the domain label for the target domain. The first part of the DNN architecture is the feature extractor,  $g_f$ , that decomposes the inputs and maps them through the hidden state to form the outputs,  $h_{t-1} \in \mathbb{R}^h$ . The model then embeds the output space as a feature space  $f$  of the deeper layers and repeats this process as needed. As previously detailed, this vector space parameter that is the result of feature mapping is,  $\theta_f$  i.e.,  $f = g_f(\theta_f)$ . This feature space  $f$  is first mapped to a real-value  $\mathcal{Y}_t^i$  variable by the function,  $g_y(f; \theta_y)$ , which is composed of fully-connected neural network layers with parameter,  $\theta_y$ . The dropout layer with a rate of 0.4 was applied to avoid the overfitting issue [159].

Another goal of this work was to find the feature space that is domain invariant, i.e., finding a feature space  $f$  in which  $P(X_S)$  and  $P(X_T)$  are similar. This is one of the challenges in training, which can be improved by applying the “feature selection” before training (detailed in the further section). Another objective was to minimize the weights of feature extractor in the direction of the regression loss,  $\mathcal{L}_y^i$ . In more detail, it is proposed that the model loss function can be used to derive the final learning function,  $g$ , through parameter  $\theta$ , which means the RUL prediction result (described in Equation (17)),  $\hat{y}_t^i = g_y(g_f(\theta_f); \theta_y)$ .

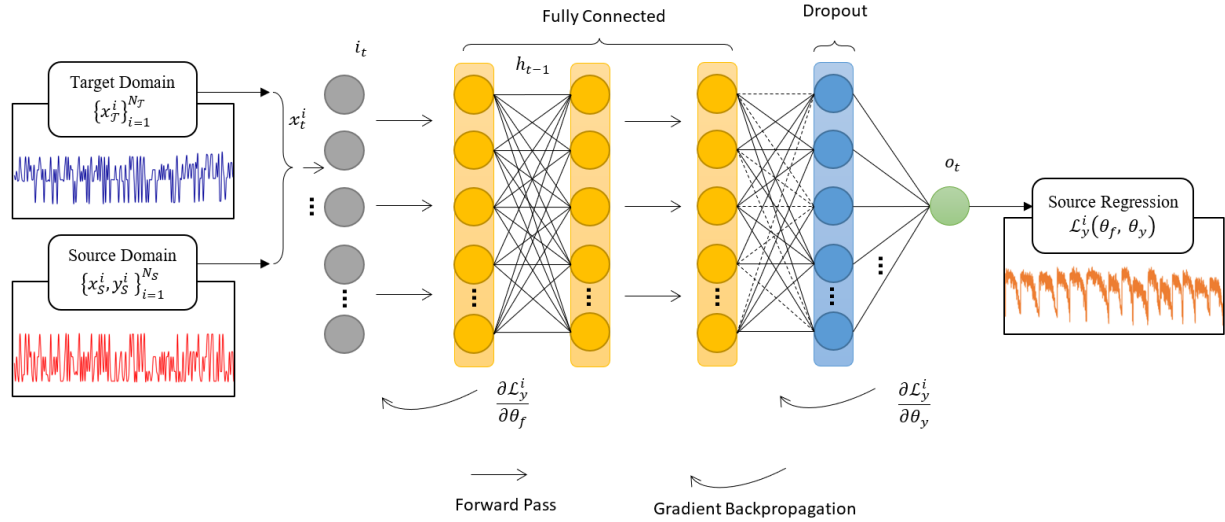


Figure 26. The proposed Deep Neural Networks model architecture for C-MAPSS data.

The way the DNN algorithm update its learning weights,  $\theta$ , is through the gradient descent update [26] in the form of;

$$\theta_f \leftarrow \theta_f - \lambda \left( \frac{\partial \mathcal{L}_y^i}{\partial \theta_f} \right) \quad (19)$$

$$\theta_y \leftarrow \theta_y - \lambda \left( \frac{\partial \mathcal{L}_y^i}{\partial \theta_y} \right) \quad (20)$$

Usually, the Stochastic Continuous Greedy (SCG) estimate is used to update the Equations (19) and (20). The learning rate,  $\lambda$ , represents the learning steps taken by the SCG as training processes.

### 4.3. Experiment and Result

The first part of the experiment was designed to compare the effectiveness of using different feature selection methods and filtering for ANN modeling of the prognostics dataset. The aircraft gas turbine engines dataset with 21 attributes was fed into different filter and wrapper feature selection methods to identify particular sets of features before the model training phase. The selected sets of features were then used as training features or training attributes for the ANN model. The second part was to test the feature selected using ANN modeling with the DNN architecture. The results from different sets of features were compared to determine the most suitable set of selected features. Finally, the final-best DNN model for predicting the RUL of aircraft gas turbine engines was determined.

#### 4.3.1. Training Procedure and Hyperparameters Selection

For training, the data from input sensors, operational setting, and labeled RUL value from the source data, and only sensors and settings from the target dataset were used. The raw data were normalized, and the feature selection was applied before the start of all models training. For the training process, the training dataset (as a source) from dataset FD002 was used. The FD002 and FD004 test dataset were used to validate (RMSE and Score). In reference to the wrapper methods, we used ANN as a validation algorithm for our wrapper methods. The cross-validation within the FD002 training data was employed for measuring the performance of the wrapper algorithms. The set-up parameters for ANN validation were fine-tuned based on the best model that was derived from complete attributes (21 features) modeling.

- 5 Folds Cross-Validation
- 1000 Training cycles
- 0.001 Learning rate
- 0.9 Momentum
- Linear sampling.

For the DNN hyperparameters selection, the model parameters in the H2O DNN algorithm varied as detailed in Table 10. The grid search to identify the range of the learning rate,  $\lambda$ , was performed after fine-tuning the remaining parameters manually. Additionally, the training sample per iteration was set to auto-tuning, and batch size was set to 1 for all variations.



The best-case scenario is the combination of following hyperparameters; Epoch = 5000, Learning rate =  $10^{-8}$ , Momentum = 0.99, L1 =  $10^{-5}$ , L2 = 0, and Max w2 set to infinity. These are all hyperparameters employed in the final DNN model proposed.

Table 10. Hyperparameters values evaluated in the proposed DNN model for C-MAPSS data.

Hyperparameters	Range
Epoch	{100, 1000, <b>5000</b> , 7000, 10000}
Training sample per iteration	AUTO
Batch size	1
Leaning rate annealing	{ $10^{-10}$ , <b><math>10^{-8}</math></b> , $10^{-5}$ , $10^{-1}$ }
Momentum	{0.1, 0.2, 0.3, 0.5, 0.6, 0.8, <b>0.99</b> }
L1: Regularization that constraint the absolute value	{ $10^{-20}$ , $10^{-15}$ , $10^{-10}$ , <b><math>10^{-5}</math></b> , $10^{-1}$ , 0}
L2: Regularization that constraint the sum of square weights	{ $10^{-20}$ , $10^{-15}$ , $10^{-10}$ , $10^{-5}$ , $10^{-1}$ , <b>0</b> }
Max w2: Maximum sum of square of incoming weight into the neuron	{0, 10, 100, 10000, $\infty$ }

### 4.3.2. Experiment Setup and Results

All experiments were implemented on an Intel® Core i7 10th generation i7-10510U 4 cores processor with 8 MB Cache, 1.8 GHz clock speed, and up to 4.9 GHz boost speed with 16 GB RAM and Intel® UHD integrated graphic. The DNN architecture was implemented using Python 3.6 with the H2O library/package [180]. The experimental results presented in this section will be separated into three parts: (1) Feature selected using feature selection methods, (2) Results and models from ANN with the selected feature, and (3) Proposed DNN model. All RMSE and all performance measurements of DNN models reported in this paper are the average results from 20 trials.

#### 4.3.2.1. Feature Selection for Aircraft Engine Dataset

All possible feature selection methods were used with the C-MAPSS dataset. Filter methods include, Deviation selection, PCA selection, Relief algorithm selection, selection, SVM selection, and Pearson correlation selection. Only three wrapper methods were implemented: forward selection, backward elimination, and evolutionary selection.

Table 11 detail the ranking of attributes based on coefficients and weights calculated from each filter feature selection method. It is important to note that the ranking of the attributes based on different methods is dependent upon the statistical measures or weights obtained from each method.

For the Pearson correlation, the attributes were not selected if the coefficient was less than -0.01 based on the work of others [166, 167]. For PCA, the features were selected based on weight

(selected if weight is more than 0.2) and the PCA matrix [168]. For the Relief algorithm, the attributes were not selected if the calculated weight was below zero [168]. For deviation selection, the feature was selected if the weights were higher than 1 [168]. It is important to note that the weights of the attributes calculated using the Relief algorithm were unacceptably low (less than  $10^{-12}$ ) suggesting little learning and in addition there were large gaps between calculated weights. Similar results were observed with other filter selection methods, including the SVM. It was found that by using the filter methods that provided statistically low weight for selecting features, the models trained from those features were unable to provide usable prediction results.

The following are the features selected based on these two filtering methods. In addition to the feature weights from Pearson correlation selection and PCA selection in Table 11, the Pearson correlation matrix and PCA matrix are also provided in Appendix.

- Pearson correlation; 8 attributes: T30, T50, Ne, Ps30, NRc, BPR, farB, and htBleed.
- Relief algorithm; 2 attributes: P15 and Nf\_dmd.
- SVM selection; 11 attributes: T2, T24, P30, Nf, epr, phi, NRF, Nf\_dmd, PCNfR\_dmd, W31, and W32.
- PCA selection; 17 attributes: T2, T24, T30, T50, P2, P15, P30, Nf, Ne, epr, Ps30, phi, farB, htBleed, Nf\_dmd, W31, and W32.
- Deviation selection; 11 attributes: T2, T24, T50, P2, P15, Ne, epr, Ps30, farB, PCNfR\_dmd, and W32.

In reference to the wrapper methods study, below are the sets of features selected from each method. It is important to note that for the wrapper methods, ANN validation with the modeling set-up, as mentioned in Section 3.2 was used. Figure 27 shows the validation process using ANN for evolutionary selection.

- Backward elimination; validate RMSE 46.429 from 19 attributes; T2, T30, P2, P15, P30, Nf, epr, Ps30, phi, NRF, NRc, BPR, farB, htBleed, Nf\_dmd, PNCfR\_dmd, W31, and W32.
- Evolutionary selection; validate RMSE 46.451 from 14 attributes; T2, T30, T50, P2, Nf, Ne, epr, Ps30, NRc, BPR, farB, htBleed, W31, and W32.

- Forward selection methods; validate RMSE 46.480 from 11 attributes; T2, T30, T50, P2, P15, Ps30, NRc, BPR, farB, htBleed, and Nf\_dmd.

Unlike forward selection and backward elimination methods, which are both based on search algorithms [169], selection is based on genetic algorithms [182]. However, instead of using fitness function from genetic theory, the evolutionary selection method used ANN validation as fitness measurement. The parameters set-up in this experiment were; population size = 10, maximum number of generation = 200, using tournament selection with 0.25 size, initial probability for attributes (features) to be switched = 0.5, crossover probability = 0.5 with uniform crossover, and mutation probability =  $\frac{1}{\text{number of attributes}}$ .

It is also important to note that, in this case, the brute force algorithm was not used. The brute force algorithm is the selection algorithm that can derive the best features set from the data. However, with limited computational capability, it cannot be used in real-time. Therefore, it was not included in this experiment.

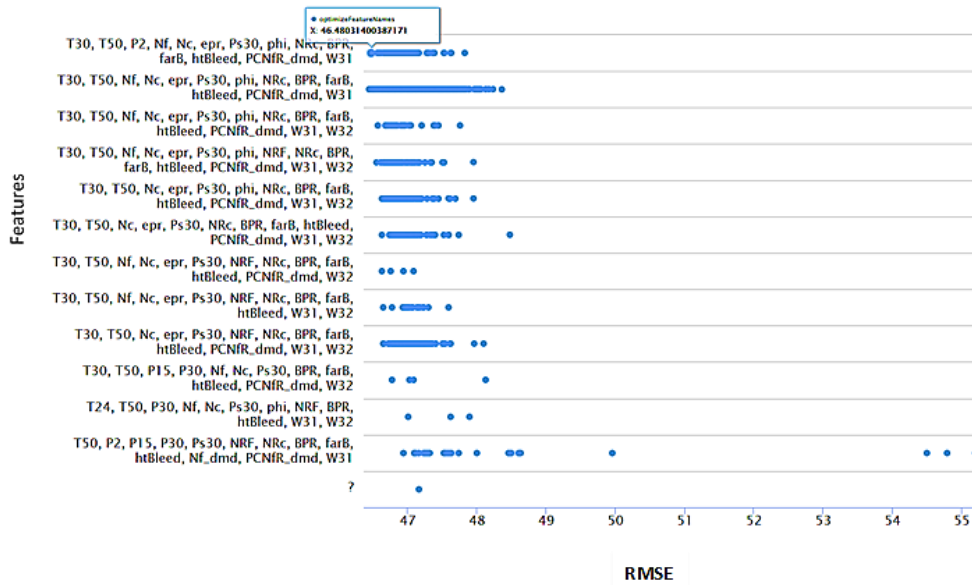


Figure 27. Validation result using evolutionary selection from C-MAPSS data.

Table 11. C-MAPSS attribute values from different filter methods.

Pearson Correlation		Relief Algorithm		SVM		PCA		Deviation	
Attributes	Weight	Attribute	Weight	Attribute	Weight	Attribute	Weight	Attribute	Weight
farB	-0.0648807	P15	2.55555E-05	epr	28.062965	htBleed	0.24226001	PCNfR_dmd	1.00002156
Ps30	-0.0426395	Nf_dmd	4.29878E-13	T2	24.031467	T30	0.24219398	farB	1.00000884
T50	-0.0377657	farB	-1.76803E-13	Nf_dmd	15.921074	Ne	0.24213648	P15	1.00000751
BPR	-0.0320325	T2	-3.5083E-13	Nf	15.293535	T50	0.24212279	epr	1.00000215
NRc	-0.0308729	P2	-1.41209E-12	T24	11.169562	T24	0.23799320	P2	1.00000079
htBleed	-0.0254014	PCNfR_dmd	-3.58802E-12	W31	9.070028	epr	0.23251894	T2	1.00000049
T30	-0.0253007	phi	-8.18383E-07	W32	8.806654	Ps30	0.23247642	Ne	1.00000022
Ne	-0.0133643	Nf	-1.94057E-06	PCNfR_dmd	6.597514	phi	0.22942893	T50	1.00000016
T24	-0.0063673	NRF	-2.22812E-06	NRF	5.849870	P30	0.22931342	Ps30	1.00000013
P2	-0.0031016	P30	-3.43389E-06	P30	5.529144	W31	0.22654883	W32	1.00000013
P15	-0.0028634	T24	-3.2525E-05	phi	5.262733	W32	0.22654313	T24	1.00000011
T2	-0.0023212	W31	-6.1066E-05	Ne	0.026252	P15	0.21870245	T30	1.00000000
phi	-0.0004811	W32	-6.76249E-05	P15	-0.151776	Nf	0.21427293	P30	0.99999998
P30	-0.0003329	epr	-9.125E-05	P2	-0.726430	Nf_dmd	0.21420247	NRF	0.99999993
epr	0.0013847	Ne	-0.00017538	farB	-16.274719	T2	0.21253812	BPR	0.99999985
Nf	0.0026742	BPR	-0.000324083	T30	-24.291950	P2	0.20884536	NRc	0.99999984
W32	0.0029798	NRc	-0.000344686	htBleed	-24.530502	farB	0.20473956	phi	0.99999978
Nf_dmd	0.0030117	Ps30	-0.000364589	NRc	-32.369914	NRc	0.18353047	Nf	0.99999973
W31	0.0030517	T50	-0.000397835	T50	-40.853420	NRF	0.14637480	W31	0.99999957
NRF	0.0044269	T30	-0.000422547	Ps30	-53.894591	PCNfR_dmd	0.14634719	htBleed	0.99999829
PCNfR_dmd	0.0048232	htBleed	-0.000613424	BPR	-65.865476	BPR	-0.21428742	Nf_dmd	0.99998466

#### 4.3.2.2. DNN Models and Results

Table 12 summarizes RMSE and prediction score results from all of the DNN models. The complete RUL best fit prediction curves for testing data of all feature selection methods are shown in Figure 28 for FD002 test data, and in Figure 29 for FD004 test data, respectively. The blue curves represent the actual RUL from the dataset, and the red lines/dots are the prediction points from our feature selection DNN models. For illustration purposes, Figures 30 and 31 include the prediction curve from one engine of each testing data FD002 and FD004 to demonstrate the prediction of the DNN model of one degradation cycle. Additionally, Table 14 includes all DNN models and all prediction error values measured from the DNN models using the FD002 test dataset, i.e., absolute error, relative error, relative error lenient, relative error strict, normalized absolute error, root relative squared error, squared error, correlation, squared correlation, prediction average, spearman rho, and Kendall tau. The number of hidden nodes in the DNN layers was identified based on the best models fine-tuned from one-layer ANN models for each feature selection method. The same number of hidden nodes from the best ANN models were used to construct the DNN model layers. Note that only the DNN models from feature selection

methods that provided usable prediction results are presented. Therefore, the results from Relief algorithms and SVM selection are not presented.

Table 12. Best RMSE and prediction score results of RUL prediction from all DNN models.

Methods	RMSE		Score		
	FD002	FD004	FD002	FD004	
Original data	45.439	45.302	645,121	427,968	
SVM					<b>Unusable</b>
Relief algorithm					
Backward elimination	45.121	45.436	645,132	211,129	
Deviation	45.374	45.630	740,936	256,776	
<b>Evolutionary Selection</b>	<b>44.717</b>	<b>44.953</b>	<b>518,025</b>	<b>355,458</b>	<b>Best Overall</b>
Forward selection	45.242	46.505	1,353,749	423,997	
PCA	45.368	45.108	1,450,397	406,872	
Pearson correlation	45.272	46.216	502,579	338,400	

Because of the fluctuations in the prediction results from the DNN algorithm, the experiments (training and testing) were ran 100 times for each model. The result in Table 12 is the best prediction result. The fluctuations across 100 iterations for FD002 and FD004 are presented in Figure 32. In addition to the best prediction, the mean RMSE and error distributions from the 100 times testing are seen in Table 13 and Figure 33. These fluctuations in prediction errors are commonly found in most deep learning algorithms because of the random initial training weights assignment and the amplification effect from the optimizer function in deeper networks. The fluctuations in the prediction result can be more obvious when models are more complex and take a large number of input attributes.

#### 4.4. Result Discussion

As mentioned in the related works (section 4.1.3), there have been several efforts in developing deep learning models for a C-MAPSS aircraft gas turbine engines dataset [90, 94, 111, 173, 175-179]. Currently, the deep learning model with the highest accuracy was proposed by Zhengmin Kong et al. [177]. Their deep learning architecture consists of CNN and LSTM-RNN combined layers and were able to achieve a RMSE of 16.13, while the best previously reported evolutionary DNN model was a RMSE value of 44.71. This indicates that the performance of our DNN models is less accurate than the modern hybrid deep learning models developed in recent years.

However, no work has addressed the complexity of the models and the computational requirement for model training. All hybrid deep neural network layers are generally overly complex and require exponentially more computational time and resources compared to the proposed evolutionary

DNN. All proposed models in recent years also took all features from the C-MAPSS dataset and disregard the features performance benchmark. Different from those models, the proposed approach applies the feature selection prior to the model training phase to help reduce the number of input attributes, and to reduce the model complexity as a result. The reduction in the complexity when using fewer input features is more evident for the high complexity hybrid deep neural network layers.

Additionally, as seen in Figures 32 and 33, prediction error fluctuations can be noticed when training deep learning models. This effect has occurred not only in DNN but also in other types of network layers, such as LSTM-RNN, CNN, and other modern hybrid layers. Based on the results as seen in Table 13 and Figures 28 to 33, the key observations of such an effect include:

First, utilizing fewer features to train the model has been shown to lower the error distribution range, compared to using more features. This is because the initial random weights assigned to the hidden nodes are smaller when using fewer features in model training. The reduction in the error may also be result of eliminating irrelevant (or less important) training the data and allow those features that are actually relevant to increase model learning. In more detail, the models are more robust and reliable when using fewer features. The same observation is also applied for the fluctuation of the prediction errors, in that the prediction results are more stable when using fewer features in model training.

Second, in reference to model performance and accuracy, although using selected features from some feature selection methods does not promote better results, the feature selection methods can reduce the computational burden while offering better prediction performance. In this experiment, the evolutionary selection can achieve both better performance and complexity reduction.

It is important to note that the current goal is not to improve model performance compared against other existing works; rather, the aim is to provide baseline results and demonstrate the feature selections on deep learning models, can reduce learning times. It is believed that the results can be further improved when applying the proposed feature selection results in the modern hybrid deep neural network architectures.

The experimental results in general, the best accuracy based on the RMSE results in Table 4 were generated from the evolutionary method. The complexity of the model was improved using a reduced set of features, from 21 attributes to only 14 attributes.

When considering the complexity and computational time, the filter methods were less complex and learning time was reduced because they do not require train-and-test multiples of ANN model validation in the process. In this study, when performing the selection process, most of the filter methods required only 5–10 min while wrapper methods required 10 h to 10 days to complete.

It is also important to note that the curve fitting and pattern recognition were improved, as can be seen when comparing the RUL prediction curves in Figures 30 to 31. In greater detail, the DNN model from most of the selected features can reasonably capture the trend of both before and after aircraft gas turbine engines' degradation intervals as illustrated in Figure 30 and 31, the curve fitting is better for both interval before  $R_{th}$  and after.

In summary, the evolutionary DNN model architecture performs best as a simplified deep neural network data-driven model for C-MAPSS aircraft gas turbine engines data. The feature selection phase (as described in the modeling framework in Figure 11 Chapter 2) should be included as a standard in the modeling framework for such a PHM dataset. This presents a potential improvement to the overall performance for RUL prediction for the prognostics of aircraft gas turbine engines data as well as other prognostic datasets.

Table 13. Mean RMSE from all DNN models.

Average RMSE													
Original Data		BW Elimination		Deviation		Evo Selection		FW Selection		PCA		Pearson	
FD002	FD004	FD002	FD004	FD002	FD004	FD002	FD004	FD002	FD004	FD002	FD004	FD002	FD004
48.398	50.541	47.907	50.331	48.160	50.081	47.452	49.650	48.434	50.708	48.072	49.737	49.203	52.111

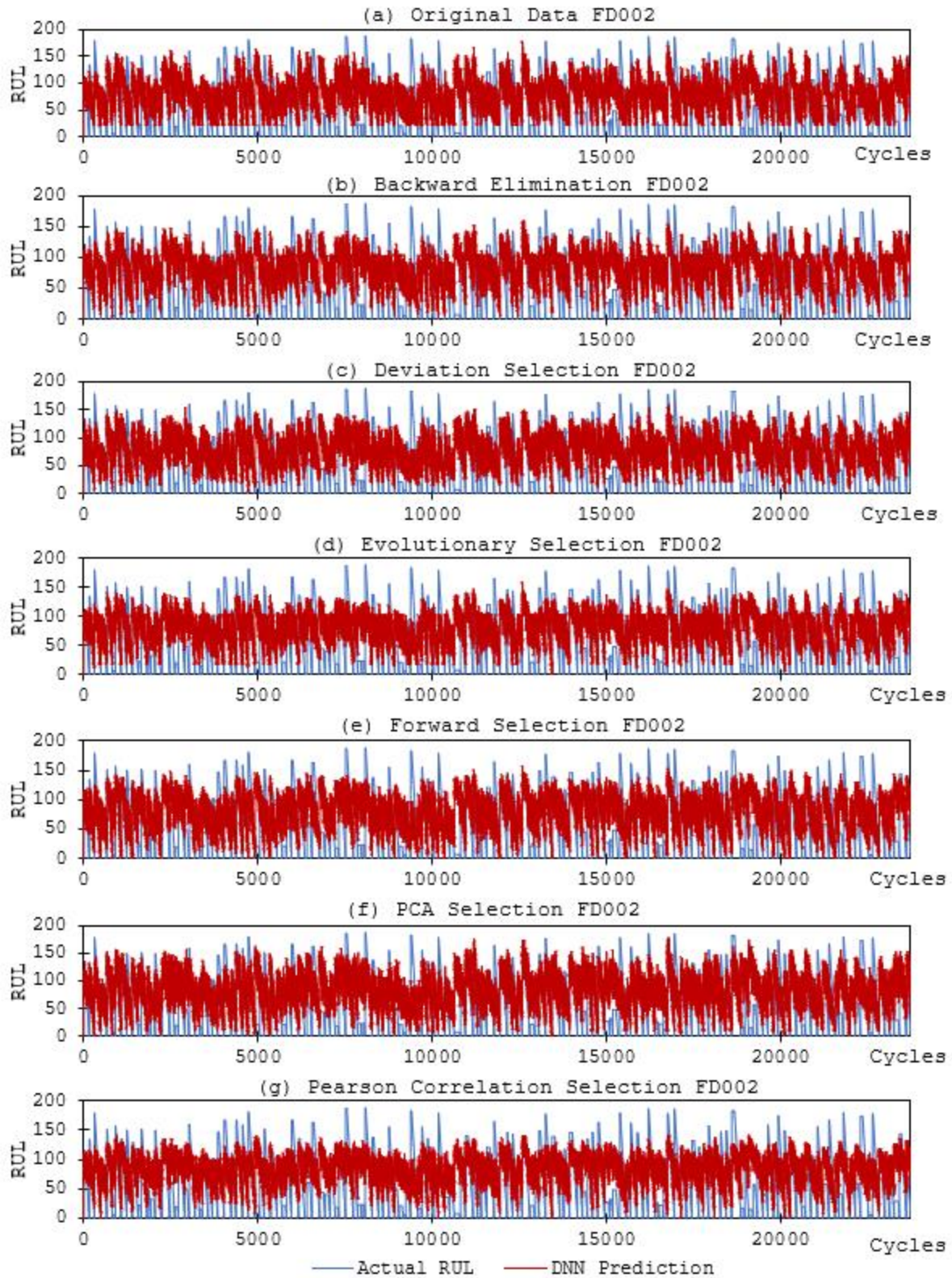


Figure 28. (a - g) All RUL prediction curves for FD002.



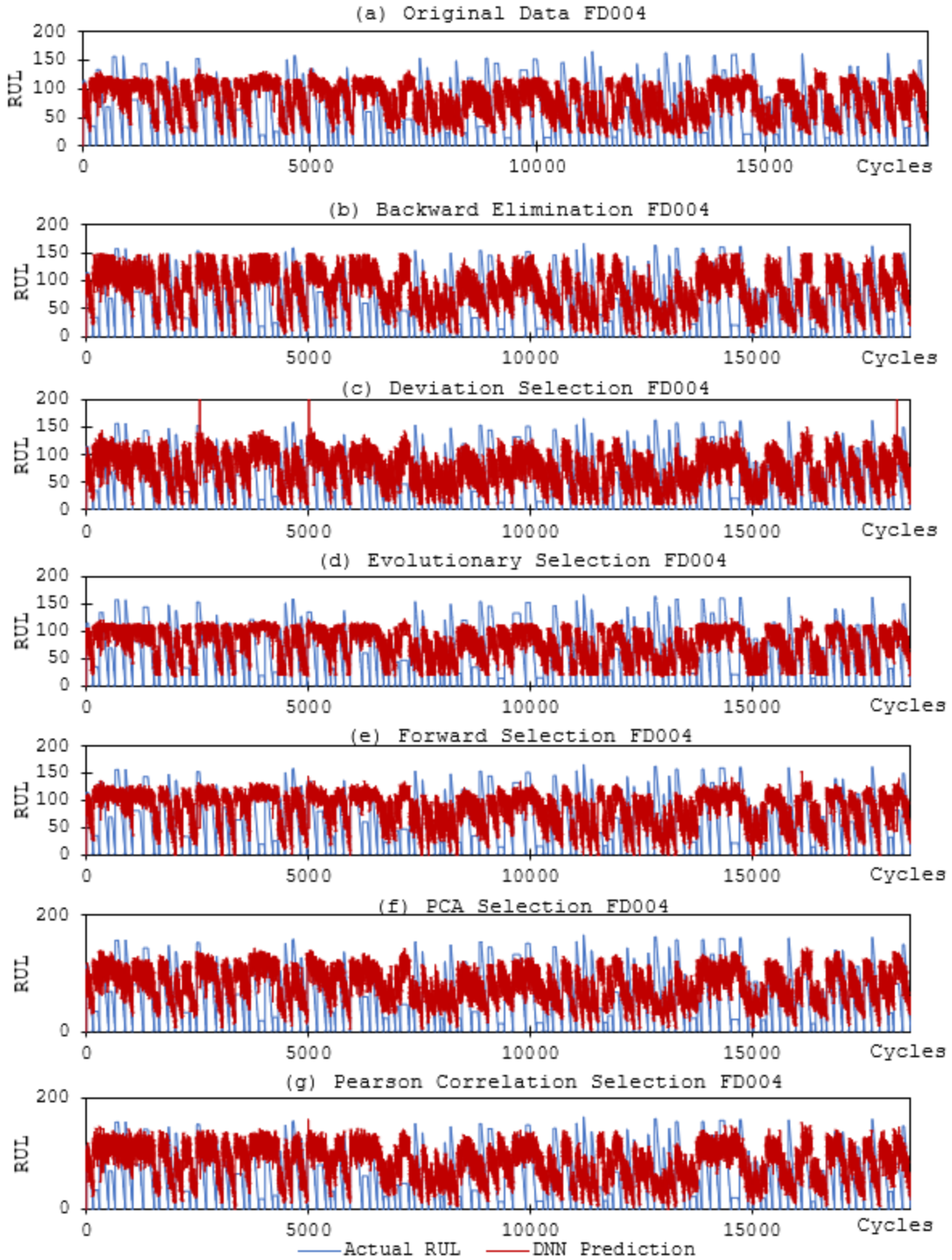


Figure 29. (a - g) All RUL prediction curves for FD004.

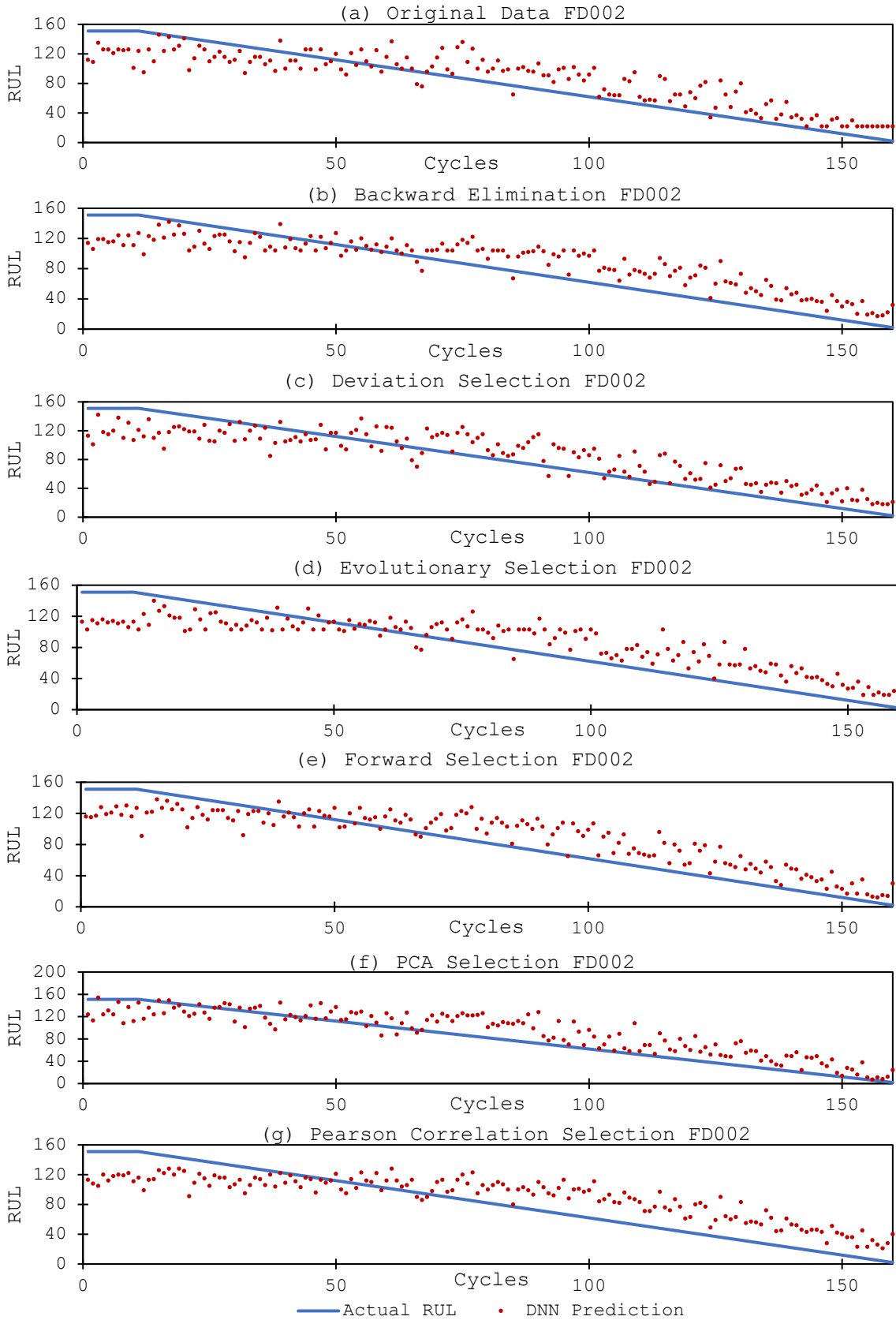


Figure 30. (a - g) RUL prediction points for one engine of FD002 test data.

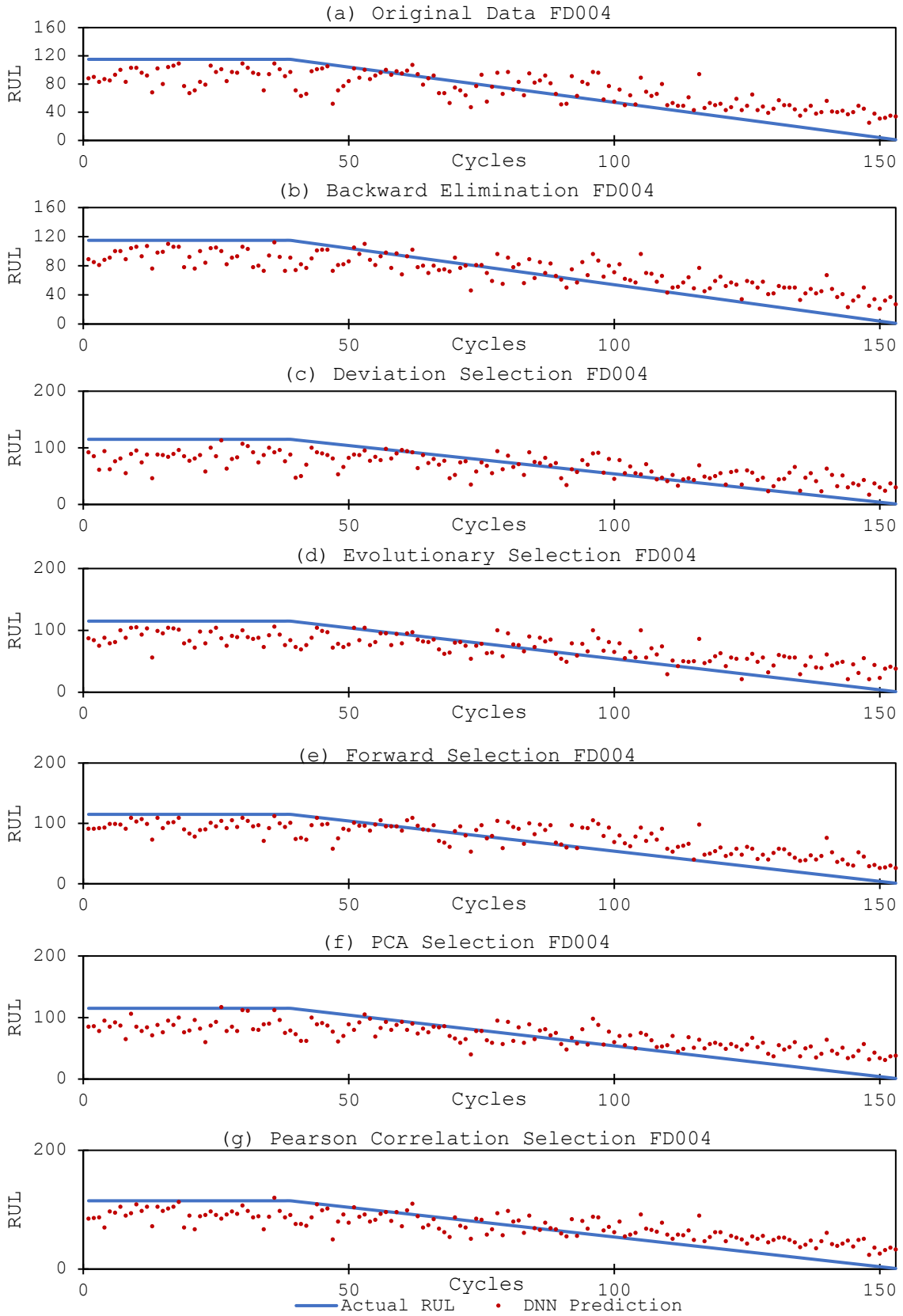


Figure 31. (a - g) RUL prediction points for one engine of FD004 test data.

Table 14. The best DNN models for FD002 test data.

Feature Selection Method	Model			Output Weights	Errors
	Layer	Unit	Type		
Original (All 21 Attributes)	----	----	-----	Layer 2: -0.389707	root_mean_squared_error: 45.439 +/- 0.000 absolute_error: 37.062 +/- 26.289 relative_error: 285.29% +/- 1071.56% relative_error_lenient: 40.87% +/- 26.92% relative_error_strict: 290.30% +/- 1070.51% normalized_absolute_error: 0.933 root_relative_squared_error: 0.963 squared_error: 2064.669 +/- 2549.829 correlation: 0.426 squared_correlation: 0.182 prediction_average: 68.095 +/- 47.177 spearman_rho: 0.406 kendall_tau: 0.28
	1	21	Input	Layer 3: -0.954436	
	2	12	Rectifier	Layer 4: -0.798112	
	3	12	Rectifier	Layer 5: 1.135641	
	4	12	Rectifier		
	5	1	0.4 Dropout Linear		
Backward Elimination	Layer	Unit	Type	Layer 2: -0.383010	root_mean_squared_error: 45.121 +/- 0.000 absolute_error: 36.707 +/- 26.240 relative_error: 275.51% +/- 1043.67% relative_error_lenient: 40.75% +/- 26.64% relative_error_strict: 281.59% +/- 1042.46% normalized_absolute_error: 0.924 root_relative_squared_error: 0.956 squared_error: 2035.929 +/- 2509.247 correlation: 0.417 squared_correlation: 0.174 prediction_average: 68.095 +/- 47.177 spearman_rho: 0.399 kendall_tau: 0.278
	1	19	Input	Layer 3: -0.791862	
	2	11	Rectifier	Layer 4: -0.706631	
	3	11	Rectifier	Layer 5: 1.064392	
	4	11	Rectifier		
	5	1	0.4 Dropout Linear		
Deviation Selection	Layer	Unit	Type	Layer 2: -0.274669	root_mean_squared_error: 45.374 +/- 0.000 absolute_error: 37.420 +/- 25.662 relative_error: 283.25% +/- 1026.67% relative_error_lenient: 41.82% +/- 26.69% relative_error_strict: 290.19% +/- 1025.24% normalized_absolute_error: 0.942 root_relative_squared_error: 0.962 squared_error: 2058.794 +/- 2489.328 correlation: 0.383 squared_correlation: 0.147 prediction_average: 68.095 +/- 47.177 spearman_rho: 0.375 kendall_tau: 0.261
	1	11	Input	Layer 3: -0.962801	
	2	7	Rectifier	Layer 4: -0.156934	
	3	7	Rectifier	Layer 5: 0.528834	
	4	7	Rectifier		
	5	1	0.4 Dropout Linear		
Evolutionary Selection*	Layer	Unit	Type	Layer 2: -0.820539	root_mean_squared_error: 44.717 +/- 0.000 absolute_error: 36.402 +/- 25.971 relative_error: 271.60% +/- 1022.51% relative_error_lenient: 40.89% +/- 26.50% relative_error_strict: 278.38% +/- 1021.18% normalized_absolute_error: 0.917 root_relative_squared_error: 0.948 squared_error: 1999.604 +/- 2499.212 correlation: 0.415 squared_correlation: 0.172 prediction_average: 68.095 +/- 47.177 spearman_rho: 0.401 kendall_tau: 0.280
	1	14	Input	Layer 3: -0.729643	
	2	9	Rectifier	Layer 4: -1.375567	
	3	9	Rectifier	Layer 5: 1.658891	
	4	9	Rectifier		
	5	1	0.4 Dropout Linear		
Forward Selection	Layer	Unit	Type	Layer 2: -0.598193	root_mean_squared_error: 45.242 +/- 0.000 absolute_error: 36.817 +/- 26.294 relative_error: 275.71% +/- 1038.01% relative_error_lenient: 41.12% +/- 26.56% relative_error_strict: 282.81% +/- 1036.64% normalized_absolute_error: 0.927 root_relative_squared_error: 0.959 squared_error: 2046.830 +/- 2564.139 correlation: 0.403 squared_correlation: 0.163 prediction_average: 68.095 +/- 47.177 spearman_rho: 0.390 kendall_tau: 0.272
	1	11	Input	Layer 3: -1.333539	
	2	7	Rectifier	Layer 4: -1.583420	
	3	7	Rectifier	Layer 5: 0.341112	
	4	7	Rectifier		
	5	1	0.4 Dropout Linear		

Table 14. The best DNN models for FD002 test data (continued).

Feature Selection Method	Model			Output Weights	Errors
	Layer	Unit	Type		
PCA Selection	-----	-----	-----	Layer 2: -0.022651	root_mean_squared_error: 45.368 +/- 0.000 absolute_error: 36.694 +/- 26.680 relative_error: 264.95% +/- 1016.32% relative_error_lenient: 41.31% +/- 26.37% relative_error_strict: 275.07% +/- 1014.64% normalized_absolute_error: 0.924 root_relative_squared_error: 0.962 squared_error: 2058.300 +/- 2623.562 correlation: 0.390 squared_correlation: 0.152 prediction_average: 68.095 +/- 47.177 spearman_rho: 0.382 kendall_tau: 0.266
	1	17	Input	Layer 3: -1.327223	
	2	10	Rectifier	Layer 4: -1.541491	
	3	10	Rectifier	Layer 5: 1.298059	
	4	10	0.4 Dropout		
Backward Elimination	-----	-----	-----	Layer 2: -0.853966	root_mean_squared_error: 45.272 +/- 0.000 absolute_error: 37.002 +/- 26.084 relative_error: 269.63% +/- 1010.61% relative_error_lenient: 41.23% +/- 26.36% relative_error_strict: 277.67% +/- 1009.11% normalized_absolute_error: 0.932 root_relative_squared_error: 0.960 squared_error: 2049.533 +/- 2474.691 correlation: 0.382 squared_correlation: 0.146 prediction_average: 68.095 +/- 47.177 spearman_rho: 0.364 kendall_tau: 0.253
	1	8	Input	Layer 3: -1.340343	
	2	6	Rectifier	Layer 4: -0.972141	
	3	6	Rectifier	Layer 5: 0.786599	
	4	6	0.4 Dropout		
5	1	Linear			

\*Note: Complete model layers for the proposed evolutionary selection DNN model will be described in detail in Appendix.

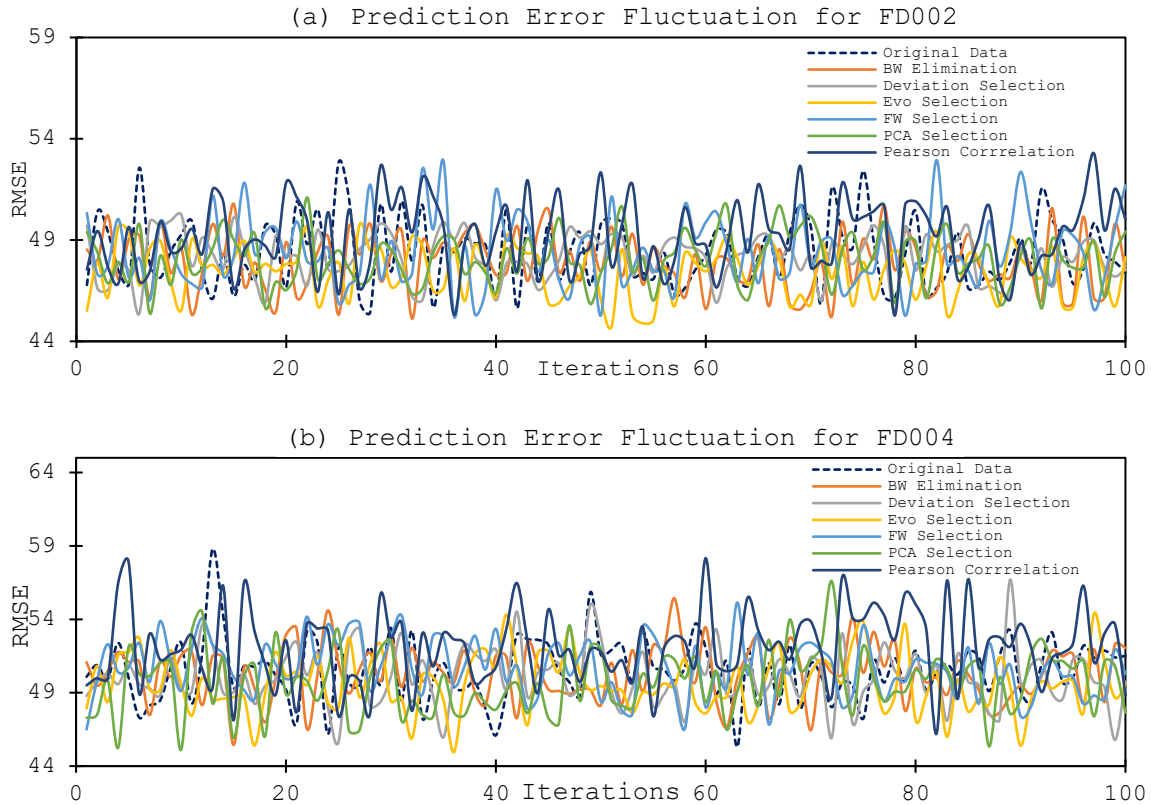
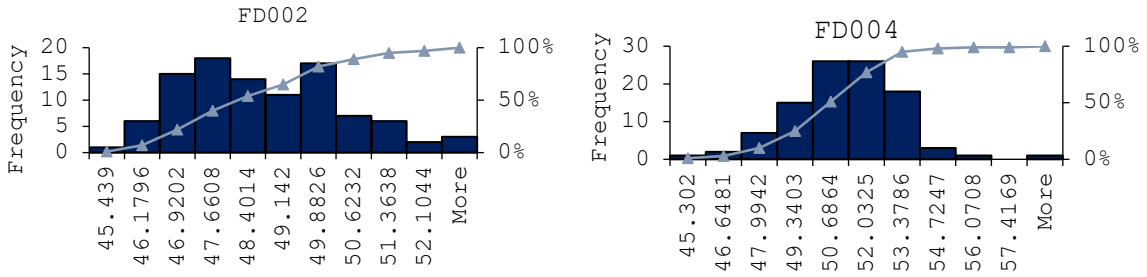
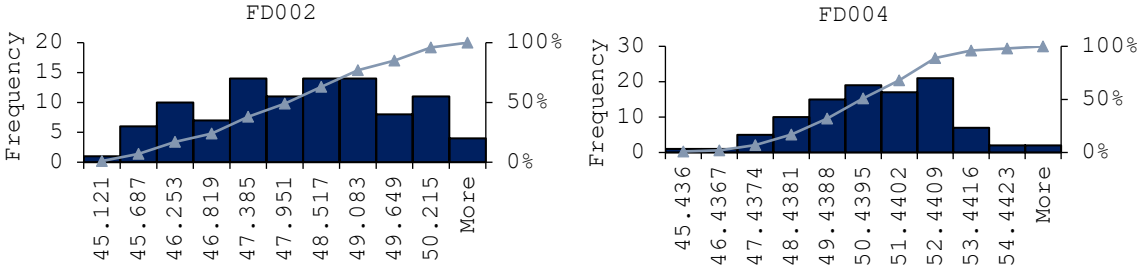


Figure 32. (a, b) RMSE fluctuation for FD002 and FD004 test data.

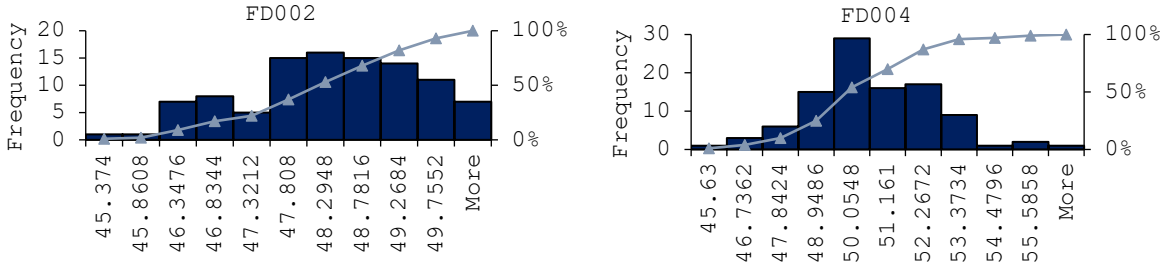
(a) RMSE Distribution for Original Data



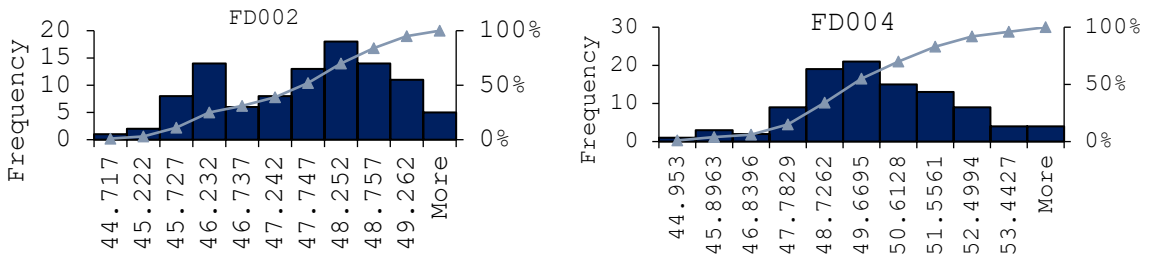
(b) RMSE Distribution for Backward Elimination



(c) RMSE Distribution for Deviation Selection



(d) RMSE Distribution for Evolutionary Selection



(e) RMSE Distribution for Forward Selection

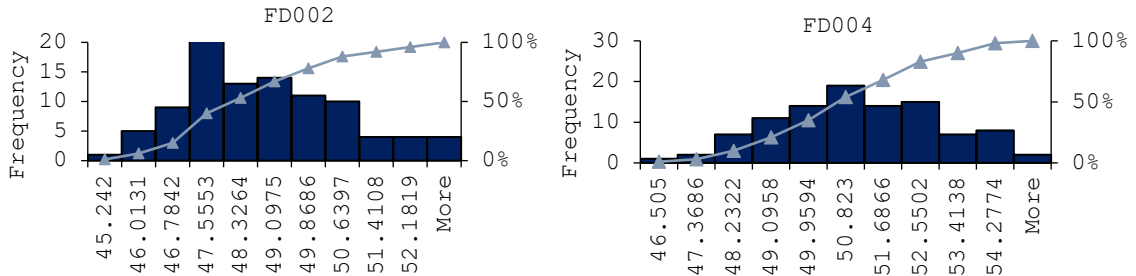


Figure 33. (a - e) Prediction error distributions from feature selection methods.

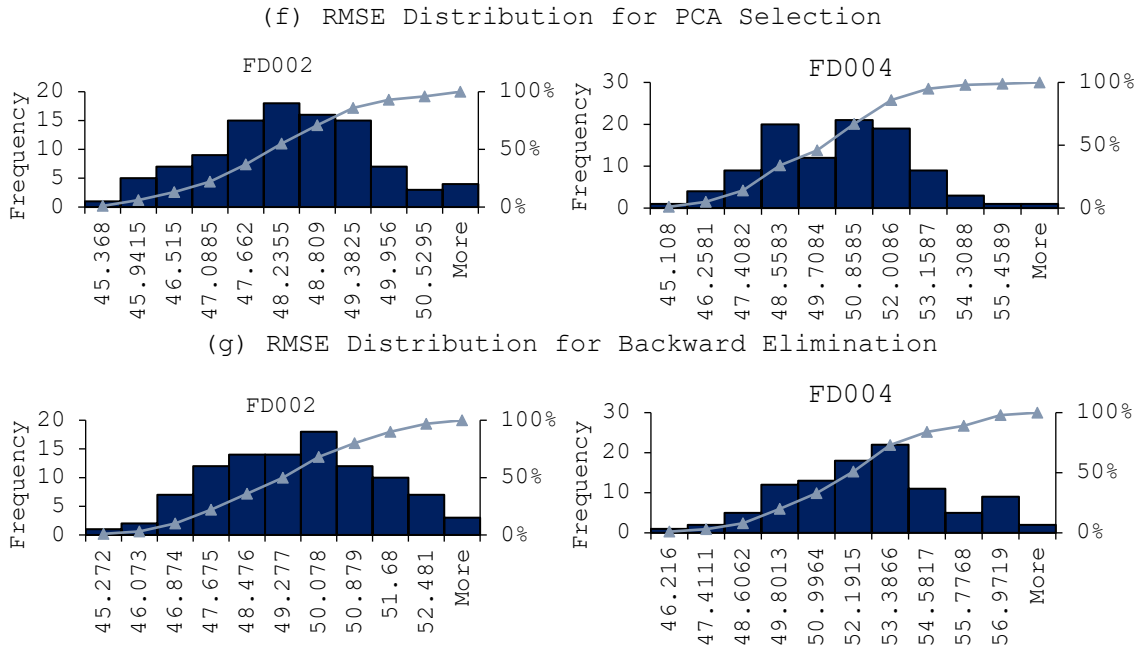


Figure 33. (f - g) Prediction error distributions from feature selection methods (continued).

#### 4.5. Chapter Summary

While there has been a review of the work on deep neural network algorithms and proposed new DNN model architecture, in this chapter, the features selected, and other new deep learning algorithms and methods were reviewed. As demonstrated in the related works [90, 94, 111, 173, 175-179], their RNN, LSTM, and CNN have been shown to result in more accurate RUL predictions when compared to shallow DNN models. However, as seen in this work, further improvements can be achieved by applying new algorithms to the selected features. One of the aspects that can improve such selected features that can accelerate machine learning through less complex models. The result in this chapter detailed a preliminary study on selecting features to generate a data-driven neural network model for the prognostic of aircraft gas turbine engines data. More complex deep learning algorithms; however, still need to be performed and tested for the effectiveness of such a feature selection technique. These are the key aspects that should be tested and experimented with in the future.

# **5. AN EVOLUTIONARY CONVOLUTIONAL LONG SHORT-TERM MEMORY DEEP NEURAL NETWORK DATA-DRIVEN MODEL FOR PROGNOSTICS OF AIRCRAFT GAS TURBINE**

## **5.1. Hybrid Deep Neural Network Layers Approach for Modeling RUL Prediction of Aircraft Engines Data**

In the previous chapter, a benchmark was prepared using the C-MAPSS aircraft engines data [163] with selected features using a deep neural network approach. However, the results from the vanilla Deep Neural Network (DNN) model can be further improved. Developing a hybrid deep neural network model and using the selected features from the previous experiments is a potential approach of improving the predictions. An emerging scheme combining Convolutional Neural Network (CNN) and Long Short-Term Memory recurrent neural network (LSTM) or CNN-LSTM has been suggested in several recent articles [109, 183-186], most of which focuses on natural language processing, speech processing, video processing task, and as a few examples. There is a good example of prognostics application using CNN-LSTM ensemble on predicting RUL [187]. Also, another interesting article introduced a hybrid scheme to predict residential energy consumption [188]. Both works relied on raw sensor data for the predictions. However, in most of the published works in the past years in RUL data-driven prognostics, more features or all features often used to describe equipment degradation [177]. This causes the training time or running time of the modeling phase to increase exponentially. As the preliminary results from the previous chapter showed, using selected features can help significantly reduce the complexity of the model and reduce training time. In addition, the accuracy of the prediction model is slightly better or similar compared to when using all features from raw data, less complexity and less training time can be realized.

In this chapter, a novel scheme based on a hybrid deep neural network, namely CNN-LSTM model is proposed to predict the RUL of aircraft engines. First, redundant features are removed using the result of DNN evolutionary selection from the previous experiments. Secondly, the spatial-temporal features are sequentially extracted by using CNN and multilayer LSTM from the preselected data. In addition, RUL is predicted by a multilayer fully connected neural network at the end layer. Finally, the effectiveness and



accuracy of the proposed scheme are validated using RMSE and a scoring algorithm as suggested by F. Khan, et al [172].

The main contributions of the experiments in this chapter are summarized as follows:

1. Design and execute a preselection of data with the evolutionary selection method which best depicts aircraft engines' degradation model using DNN.
2. Propose a novel scheme based on hybrid deep neural CNN-LSTM layers to efficaciously predict aircraft engines RUL collected in using C-MAPSS turbofan engine degradation simulation data.
3. Design a less complex with higher accuracy AI system compared with the non-hybrid models.

## 5.2. Methodology

Taking a similar approach detailed in Chapter 4 (section 4.2), the neural network modeling framework mentioned in Figure 11 Chapter 2 was again used. The problem statement is again defined including additional notations are also stated with the CNN and LSTM scheme described in Chapter 4 is detailed.

Previously the experiments used a hybrid CNN-LSTM scheme. This chapter will further investigate a hybrid an architecture of CNN and LSTM layers coupled with a sliding time window processing approach, which is an essential data preprocessing method that can expand the dimensions of the input data.

### 5.2.1. Problem Statement

The majority of the problem statement described in section 4.2.1 Chapter 4 can be used again for the CNN-LSTM modeling scheme. This is because the DNN algorithm that has been used in the previous experiments also falls into the same nature, in which, they are all based on neural network algorithm.

- The raw data is denoted as,  $D_S = \{(x_S^i, y_S^i)\}_{i=1}^{N_S}$ , the data contains  $N_S$  training sample, where  $x_S^i \in \mathcal{X}_S$  is a feature with a length of  $T_i$  and  $q_S$  is the number of features.
- $x_S^i = \{x_t^i\}_{t=1}^{T_i} \in \mathbb{R}^{q_S \times T_i}$ . and  $y_S^i \in \mathcal{Y}_S$  is denoted as Remaining Useful Life (RUL) also with the length  $T_i$  (feature space and RUL space are within the same length)
- $y_S^i = \{y_t^i\}_{t=1}^{T_i} \in \mathbb{R}_{\geq 0}^{T_i}$ . where  $t \in \{1, 2, \dots, T_i\}$ ,  $x_t^i \in \mathbb{R}^{q_S}$ , and  $y_t^i \in \mathbb{R}_{\geq 0}$ , represent the  $t$ -th measurement of all variables and RUL label.

- The estimated target domain is denoted as,  $D_T = \{x_T^i\}_{i=1}^{N_T}$  where  $x_T^i \in \mathcal{X}_T$  and  $\mathcal{X}_T \in \mathbb{R}^{q_T \times T_i}$  with no labels.
- The source and target domain,  $D_S$  and  $D_T$ , are assumed to a different probability distribution,  $P(X_S) \neq P(X_T)$ .
- Function  $g$  is assumed to be able to derive or learn from the source data that can approximate the corresponding RUL for the target domain at the testing time, such,  $y_T^i \approx g(x_T^i)$ .

### 5.2.2. Sliding Time Window Processing

Sliding time window processing is essential data preprocessing approach to transform the raw input data into a dimension that can recognize by CNN and LSTM layers. The dimension of the input data is important because the CNN and LSTM layer generally operate in higher dimension or 3D. The layer cannot read (or recognize) the input data if the data dimension does not fit with the layer scheme. The window processing data is a technique that converts the time series data into the sequence data and retains the local dependence of the series data. It is important to note that the time dimension is not loss, instead the time dimension is transformed or reshaped. The data processed by the sliding time window can be encapsulated in a time window with a fixed length. The window continues to be “sliding” through the data continuously.

In the case of using this technique to predict the RUL, adapting different sequence lengths allow information from the past multivariate temporal sequences to influence the RUL prediction at a point in time. As mentioned earlier, the sequential input is assumed to be  $x_S^i = \{x_t^i\}_{t=1}^{T_i}$  where  $T_i$  denotes the size of each sequence length. The hidden state function,  $h$  is used to divide each sequence of size  $T_i$  in the sequential time window of size  $T_w$  i.e.  $h_t(x^i) = \{(x_{t-T_w}^i, \dots, x_{t-1}^i)\}_{t=T_w-1}^{T_i}$ . At time  $t$  all previous sensor data within the time window  $(x_{t-T_w}^i, \dots, x_{t-1}^i)$  are transformed to a high-dimensional input vector  $i$  used to predict the target variable as previously defined as  $y_t^i$ . If  $T_i < T_w$  the zero-padding will be applied to the left size of  $x^i$  until  $T_i$  has size of  $T_w + 1$  to make each original time series have a training samples size,  $n_i = T_i + T_w$ . The updated number of examples is also defined as  $\tilde{N}_S = \sum_{i=1}^{N_S} n_i$  and  $\tilde{N}_T = \sum_{i=1}^{N_T} n_i$  for the source and target domain respectively. In the experiment,  $T_w$  was a fixed value at 30 for all domains to

allow consistency on the number of time steps seen by the CNN and LSTM neural network. Also, in this case, the transformed input has a dimension of  $p \times q$ .

### 5.2.3. Defined Convolutional Neural Network

The Convolutional Neural Network or CNN, as briefly mentioned in Chapter 1, was introduced in 1982 by Fukushima and Miyake [189]. CNN was first introduced to perform pattern recognition and image processing tasks which involved extracting high-level spatial patterns. This poses a potential using CNN for sensor data which involved a similar task as well. In addition to the aspects of CNN that have already been described in the previous chapter, there are mainly two types of CNN layer employed in this work: convolutional layer, and pooling layer.

The convolutional layer convolves the feature map from the previous layer with convolution kernels. The kernels are used for feature extraction and feature mapping, then the feature map of the next layer is computed through a type of non-linear function or in this case, the ReLU function. These functions can also call an activation function as in the vanilla DNN layer.

The pooling layer reduces the feature map resolution. The feature maps of the convolutional layer are subsampling by a predefined factor. Maximum pooling and average pooling are the most commonly used sampling approaches for the pooling layer. The pooling layer may also refer to as the subsampling layer in some literature [189].

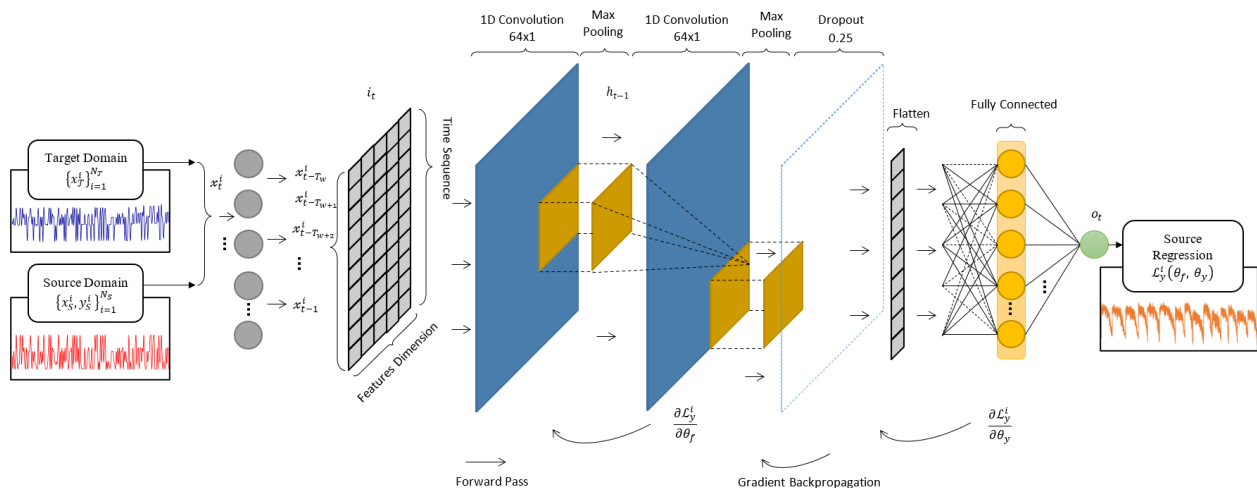


Figure 34. The defined Convolutional Neural Network architecture.

The defined CNN architecture is seen in Figure 34, which is constructed by using two one-dimensional convolution layers stacked with the max-pooling layer. The flatten layer at the end is used to reduce the dimension from the CNN layer and connected to the fully connected layer at the end to form a proper regression output value. The input of CNN obtained from the transformed actual input data using the sliding time window processing technique (previously described) to expand the input dimension into a shape that the CNN layer can recognize. Similar to the formula used for DNN layers in the previous chapter, the CNN layers are used to extract the spatial features. However, the features are contained in the predefined time windows of size,  $T_w$ . Similar to DNN layers, the hidden state units of CNN consist of the hidden state vector  $h_{t-1} \in \mathbb{R}^h$ , input vector,  $x_t^i \in \mathbb{R}^q$ , and the activation function,  $f$ . Where  $h$  is the dimension of the CNN layer. The CNN layers' operation can be formulated as:

$$i_t = \zeta(W_i x_t^i + \widehat{W}_i h_{t-1} + b_i) \quad (21)$$

$$o_t = \zeta(W_o x_t^i + \widehat{W}_o h_{t-1} + b_o) \quad (22)$$

where  $i$  and  $o$  represent input and output states.  $W$  and  $\widehat{W}$  are matrices of updated weights and weights from the convolutional state, and  $b$  is the bias vector.

The activation function  $\zeta$  for CNN also used the Rectifier Linear function [23] which can be represented as;

$$\zeta(\alpha) = \max(0, \alpha) \in \mathbb{R}_+ \quad (23)$$

The loss function ( $\mathcal{L}$ ), was defined as a simple error (or regression loss) between the actual and predicted output values from the RUL. The loss function can be expressed as:

$$\mathcal{L}_y^i(\theta_f, \theta_y) = |\hat{y}_t^i - y_t^i|^p \quad (24)$$

The  $\theta_f$  is the space representation of the target input that mapped into a new space (the denote,  $f$  represents the feature space). The  $\theta_y$  is the domain regression space and,  $\hat{y}_t^i$  is RUL prediction from the source domain. The objective in training DNN is to minimize the prediction loss,  $\mathcal{L}_y^i$ , which can be described as:

$$\min_{\theta_f, \theta_y} \left[ \frac{1}{N_s} \sum_{i=1}^{N_s} \mathcal{L}_y^i(\theta_f, \theta_y) \right] \quad (25)$$

Another objective is to minimize the weights in the direction of the regression loss,  $\mathcal{L}_y^i$ . The final learning function,  $g$ , can also derive from the loss function through parameter  $\theta$ . The RUL prediction result is  $\hat{y}_t^i = g_y(g_f(\theta_f); \theta_y)$ . The CNN update layers' learning weights,  $\theta$ , through the gradient descent update [26] in the form of;

$$\theta_f \leftarrow \theta_f - \lambda \left( \frac{\partial \mathcal{L}_y^i}{\partial \theta_f} \right) \quad (26)$$

$$\theta_y \leftarrow \theta_y - \lambda \left( \frac{\partial \mathcal{L}_y^i}{\partial \theta_y} \right) \quad (27)$$

However, in the case of CNN, Adaptive Moment Estimation (Adam) is used to optimize the weight update (see Appendix for more details of Adam). The learning rate,  $\lambda$ , also represents the learning steps.

#### 5.2.4. Defined Long Short-Term Memory Recurrent Neural Network

The concept of Recurrent Neural Network (RNN) was briefly introduced in Chapter 1. Long Short-Term Memory neural network (LSTM) is an extension or an improvement version of the RNN. One of the obvious issues using RNN is that the original RNN can accumulate the derivatives of the activation function during the backpropagation process which can cause of divergence from an accurate model. More importantly, RNN is usually faced with the “fading memory” issue. This makes the “future” time-steps of RNN can no longer contain the virtual memory of the first inputs. All of these issues occur when the ‘number of time-steps’ in the RNN network becomes too large for the number of connections and layers. Therefore, LSTM has been developed to improve on the issues that original or vanilla RNN has.

LSTM was introduced in 1997 by Hochreiter and Schmidhuber [33]. Instead of containing all data in long time-steps, LSTM adopts the gating structure to control the flow of the information. The corresponding structure diagram of the LSTM memory cell is seen in Figure 35. The memory cell of LSTM consists of the forget gate, input gate, and output gate. Although these three gates of the memory cell of LSTM can take the current input layer data and the recurrent state before, the corresponding functions are different from each other. The forget gate identifies how many states of the memory cells at the last moment that can be fed to the current state. This helps to shorten the virtual time-steps number and makes the LSTM network capable of controlling how information flows within the memory cells by updating the series of gates capable of learning long-term relationships in the input data.

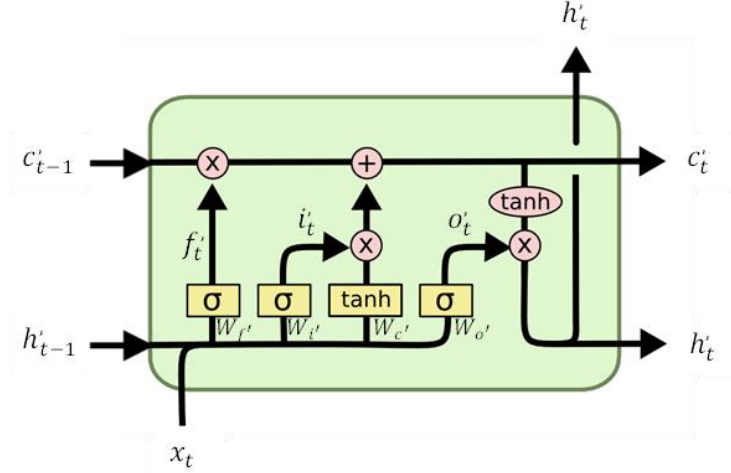


Figure 35. LSTM memory cell [190].

Table 15. The notation of the symbol in the LSTM memory cell.

Symbols	$\otimes$	$\oplus$	$\sigma$	$\tanh$
Notation	Element-wise Multiplication	Sum	Sigmoid Function	Hyperbolic Tangent

In the proposed LSTM architecture seen in Figure 36, the LSTM layers take the input from the reshaped-input data using sliding time window processing. The features were contained in the predefined time windows of size,  $T_w$ . However, the input dimension was considered as an imaginary time dimension. This allows the LSTM layers to separate the time dimension from the input data for the recurrent training process. In the LSTM layer, a set memory cell consists of three non-linear gating units (Figure 35 illustrated a unit of LSTM memory cell) that update a cell state,  $C^t \in \mathbb{R}^{h'}$  in each iteration, using a hidden state vector,  $h_{t-1}^i \in \mathbb{R}^{h'}$  with input  $x_t^i \in \mathbb{R}^q$ , where  $h'$  is the recurrent time dimension of the LSTM layer. Operation formulas of the LSTM layers are defined as:

$$f_t' = \sigma(W_{f'}x_t^i + R_{f'}h_{t-1}^i + b_{f'}) \quad (28)$$

$$i_t' = \sigma(W_{i'}x_t^i + R_{i'}h_{t-1}^i + b_{i'}) \quad (29)$$

$$o_t' = \sigma(W_{o'}x_t^i + R_{o'}h_{t-1}^i + b_{o'}) \quad (30)$$

where  $f'_t$ ,  $i'_t$ , and  $o'_t$  are forget gate, input gate, and output gate respectively,  $\sigma$  is a sigmoid activation function inside the LSTM cell (as described in Figure 35 and Table 15) used to scale the output to the 0-1 range,  $W \in \mathbb{R}^{h \times q}$  are input weight matrices,  $R \in \mathbb{R}^{h \times h}$  are the recurrent weight matrices, and  $b \in \mathbb{R}^h$  are bias vectors.

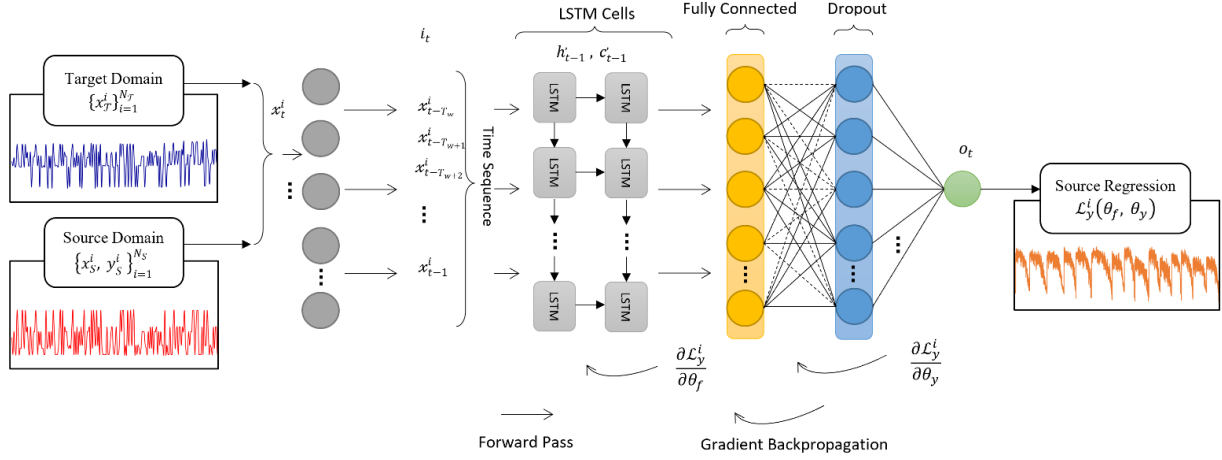


Figure 36. The defined Long Short-Term Memory network architecture.

All  $f'_t$ ,  $i'_t$ , and  $o'_t$  are first calculated and fed through the gate to compute the new cell state  $\tilde{C}_t$  using hyperbolic tangent (tanh) activation function as follow:

$$\tilde{C}_t = \tanh(W_C x_t^i + R_C h_{t-1} + b_C) \quad (31)$$

Then, the previous cell state  $C_{t-1}$  were updated into the new cell state  $C_t$  as seen in Figure 35 and Table 15, or as the formula:

$$C_t = f'_t \otimes C_{t-1} + i'_t \otimes \tilde{C}_t \quad (32)$$

As previously detailed (and described in the previous equation), the forget gate,  $f'_t$  decide on which part of information vectors to keep or ignore from the cell state. Then, the input gate,  $i'_t$  decides part of the state vectors that can be updated based on the candidate cell states. Then, the forget gates and input gates are used to update the information for the new state in the next iteration steps. The last part is the output gate,  $o'_t$  which decides on which information is output data. Then the new hidden state,  $h_t$  can be calculated by employing hyperbolic tangent activation function to the current cell state times the result from the output gate (also as described in Figure 35 and Table 15):

$$h_t = o_t \otimes \tanh(C_t) \quad (33)$$

The rest of the formulation architecture is similar to the definition for CNN layers in the previous section (equation 25 to 37). It is important to note that the  $f$  represents the feature space while  $f'$  represent the forget gate and there are not the same.

The loss function of LSTM can be expressed as:

$$\mathcal{L}_y^i(\theta_f, \theta_y) = |\hat{\psi}_t^i - \psi_t^i|^p \quad (34)$$

The objective in training LSTM is to minimize the prediction loss which can be expressed as:

$$\min_{\theta_f, \theta_y} \left[ \frac{1}{N_s} \sum_{i=1}^{N_s} \mathcal{L}_y^i(\theta_f, \theta_y) \right] \quad (35)$$

Also, the weights in the direction of the regression loss must be minimized and the final learning function,  $g$ , derives through the loss function through parameter  $\theta$ . And final the RUL prediction result is  $\hat{\psi}_t^i = g_y(g_f(\theta_f); \theta_y)$ . The gradient descent update [26] of the learning rate,  $\lambda$  (also using Adaptive Moment Estimation) can be described as;

$$\theta_f \leftarrow \theta_f - \lambda \left( \frac{\partial \mathcal{L}_y^i}{\partial \theta_f} \right) \quad (36)$$

$$\theta_y \leftarrow \theta_y - \lambda \left( \frac{\partial \mathcal{L}_y^i}{\partial \theta_y} \right) \quad (37)$$

### 5.2.5. CNN-LSTM Hybrid Architecture

The hybrid scheme combines the CNN and LSTM architectures. In the proposed hybrid scheme, there are two layers of CNN with two pooling layers and one flatten layer stacked with two LSTM layers and two fully connected at the end to form regressive RUL prediction output. The hybrid architecture takes advantage of CNN's spatiality and LSTM's long-term temporal memory characteristics. If there are high-frequency sensor measurements involved or there is an increase of variation in the sensor's signals, the CNN-LSTM hybrid architecture is able to process the tasks better than the traditional deep neural layers. Additionally, the predicted results can have smaller errors compared to the homogenous neural network scheme. CNN layers are interspersing with the pooling layer to reduce the computation time. While LSTM layers can expose the shorter long-term temporal dependency features. It is important to note that the fully connected layers are required at the output to better form the output mapping and shape the regression RUL prediction result. One minor change is that the LSTM layers take the



information from the last layer of CNN layers as their input instead of the reshaped raw input data as described previously. The proposed CNN-LSTM hybrid architecture is illustrated in Figure 37.

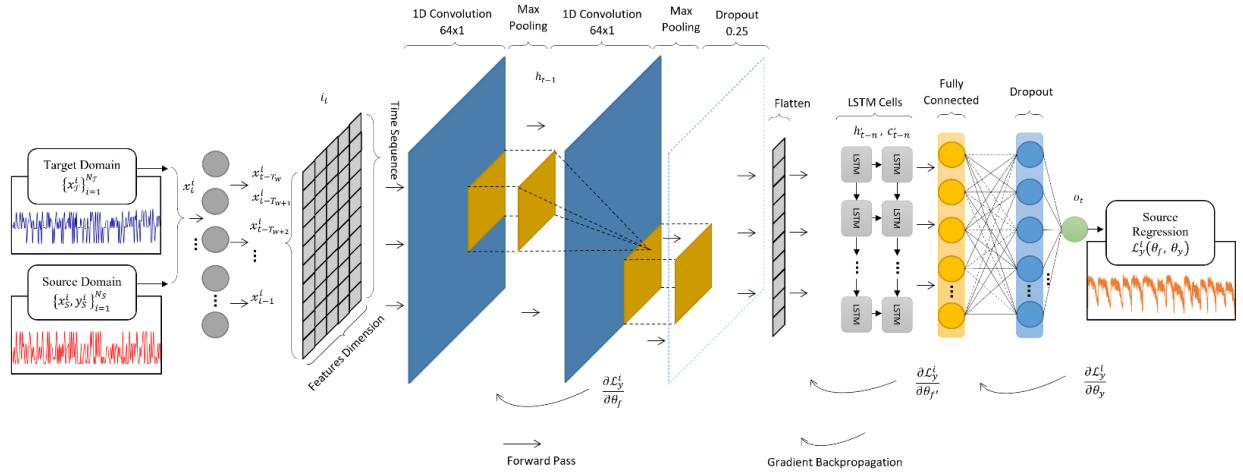


Figure 37. The proposed evolutionary hybrid CNN-LSTM deep neural network model architecture.

### 5.3. Experiment and Result

The processor used for CNN-LSTM experiments was an Intel® Core i7 10th generation i7–10510U 4 cores processor with 8 MB Cache, 1.8 GHz clock speed, and up to 4.9 GHz boost speed with 16 GB RAM and Intel® UHD integrated graphic. Two model configurations were studied: one using only CNN layers, and the newly proposed CNN-LSTM hybrid layers architectures. Both configurations were implemented using Python 3.6 with Keras library/package [160]. The Commercial Modular Aero-P propulsion System Simulation (C-MAPSS) data [163] with selected features using the evolutionary selection (from Chapter 4) is used across all experiments. These are the 14 attributes from C-MAPSS data we obtained using evolutionary selection that will also be used for the experiment in this section: T2, T30, T50, P2, Nf, Ne, epr, Ps30, NRc, BPR, farB, htBleed, W31, and W32.

For training, the data from input sensors, operational setting, and labeled RUL value from the source data, and only sensors and settings from the target dataset were used. The raw data were normalized using equation (8) described in Chapter 4. For the training process, the training dataset (as a source) from dataset FD002 was used. The FD002 and FD004 test datasets were used to validate the models and calculate prediction errors, which are, RMSE (equation (9)) and Score (equation (10)). Note that the “training” datasets of FD002 and FD004 are not the same dataset as “testing” dataset of FD002 and FD004 (as previously describe in Table 9–Chapter 4, section 4.1.2.)

The two models being presented here are Evolutionary CNN and Hybrid Evolutionary CNN-LSTM as employed evolutionary Selection and Genetic Algorithm prior to the neural network model training were used

### **5.3.1. Evolutionary CNN Model**

Initially only with CNN layers were studied. Fourteen features obtained from the evolutionary selection methods (from Chapter 4.) to develop the model were included in the model.

#### **5.3.1.1. Evolution and Selection of CNN Hyperparameters**

The list of CNN hyperparameters used for the experiments is detailed in Table 16. The grid search was used to select the best model configuration. However, because CNN layers on Keras requires more processing time to train compared to the basic DNN using H2O that has previously been implemented in Chapter 4, the range and number of hyperparameters trained was limited due to only vanilla deep neural network algorithm was employed. It is important to note that the H2O library [180] has a very limited set of hyperparameters that we can adjust, unlike Keras library [160]. In addition, in Keras library, the input shape that the CNN layers take must be predefined. The input shape of (30, 14) which is the sliding time window dimension, and the number of features or attributes of the dataset respectively were studied. Two CNN layers stacked with pooling layers for each CNN layer and flatten layer and fully connected layer to form the RUL output shape at the end was used. The type of CNN layer was 1 dimensional CNN with 64 filter size and a kernel size of 2, and 1 stride. The ReLU function was used as an activation function, with a dilation rate of 1, and the padding technique was applied for both layers. For the pooling layer, max pooling, with a pool size of 2 and 2 strides was used. Padding was applied to both pooling layers. The 0.25 dropout rate was applied for the dropout layer. The first fully connected layer (or dense layer) took the reshaped output from the flatten layer using the ReLU activation function with 250 output space then feed information into one node dense layer at the end to form an RUL prediction output for each vector space.

The CNN configuration based on the Keras library were; Epoch = 256, Batch size = 8, Callback function = TensorBoard, Loss function = Mean Square Error, Optimizer function: Adam, Learning rate = 0.001, Beta 1 and Beta 2 parameter = 0.999 and 0.999, Epsilon parameter =  $10^{-8}$ , and Decay rate = 0.

Table 16. Hyperparameters values evaluated in the proposed CNN model.

Hyperparameters	Range
Epoch	{64, 100, <b>256</b> }
Batch size	{ <b>8</b> , 16, 32}
Callback function	TensorBoard
Optimizer function	Adam
Learning rate	{ <b>0.001</b> , 0.0001, 0.00001}
Beta 1	1
Beta 1	1
Epsilon	$10^{-8}$
Decay rate	0

### 5.3.1.2. Evolutionary CNN Results

The result of RMSE and prediction score from the various CNN model configurations are detailed in Table 17. The best hyperparameters of 1 dimension 2 CNN layers as detailed in Table 16 and with 2 dimensions 2 CNN Layers, 1 dimension 3 CNN Layers, and 1 dimension 3 CNN layers respectively, were used. It is seen that the lowest RMSE values were obtained from the base configuration-1 dimension 2 CNN layers. It is important to note that the variations of model configurations were limited because of the training time for the CNN algorithm. A better result may have been achieved with additional CNN model configurations. In these experiments only the basic configuration was studied to test the premise that the hybrid scheme can have smaller error compared to the homogeneous modeling scheme. The RUL prediction curves from the best CNN model configuration (1 dimension 2 CNN layers) are shown in Figures 37 and 38 for FD002 and FD004 test data respectively. The blue curves represent the actual RUL from the dataset, and the red lines/dots are the prediction points from the CNN model.

Table 17. RMSE and prediction score for RUL prediction from different CNN configurations.

Model Configurations	RMSE		Score		
	FD002	FD004	FD002	FD004	
1 Dimension 2 CNN Layers	38.235	39.536	469,997	386,584	<b>Best Overall</b>
2 Dimension 2 CNN Layers	42.239	45.505	790,875	538,979	
1 Dimension 3 CNN Layers	44.687	43.581	557,397	468,792	
2 Dimension 3 CNN Layers	46.492	47.612	689,086	315,601	

### **5.3.2. Hybrid Evolutionary CNN-LSTM Model**

The hybrid evolutionary CNN-LSTM model includes the 14 features selected using the evolutionary selection method as input data for CNN-LSTM stacked layers. The final model architecture is seen in Figure 36. There are two CNN layers with two pooling layers stacked together and the output of CNN is connected to the two LSTM layers and the fully connected layer is applied at the end to form RUL output. The dropout layers were used at the end of both CNN layers and LSTM layers.

#### **5.3.2.1. Hybrid Evolutionary CNN-LSTM Hyperparameters Selection**

In addition to the hyperparameters selected for CNN from section 5.3.1.1, a sub-set of hyperparameters sets for the FD002 were also used as detailed by P. R. d. O. da Costa, et al. [101]. The work by da Costa [101] used only LSTM layers using the domain adaptation technique. In this case, the feature of CNN layer using in section 5.3.1.1: 64 filter size, 2 kernel size, and 1 stride, with Max Pooling, with a pool size of 2 and 2 strides for pooling layer, and same padding, were used. The adjustment completed through the LSTM layer, and the number of units in the LSTM layers was varied between 32 and 64 based on work by P. R. d. O. da Costa, et al. [101]. Various optimizers were tested, including Stochastic Continuous Greedy (SCG) and Adaptive Moment Estimation (Adam). The number of the batch size experimented on has also been added. In this case, the batch size was varied from 8, 16, 32, 64, 256, 512, and 1,024 as additionally suggested by P. R. d. O. da Costa, et al. [101] respectively.

However, the best performance is still primarily based on the best model parameters obtained from section 5.3.1.1 for CNN. The best hyperparameters were; Epoch = 256, Batch size = 8, Callback function = TensorBoard, Loss function = Mean Square Error, Optimizer function: Adam, Learning rate = 0.001, Beta 1 and Beta 2 parameter = 0.999 and 0.999, Epsilon parameter =  $10^{-8}$ , Decay rate = 0, and Number of LSTM units (for both LSTM layers) = 32. The sets of hyperparameters used in the Hybrid CNN-LSTM experiment are detailed in Table 18.

Table 18. Hyperparameters values evaluated in the proposed hybrid CNN-LSTM model.

Hyperparameters	Range
Epoch	{64, 100, <b>256</b> }
Batch size	{ <b>8</b> , 16, 32, 64, 256, 512, 1,024}
Callback function	TensorBoard
Optimizer function	<b>Adam</b> , SGD
Learning rate	{ <b>0.001</b> , 0.0001, 0.00001}
Beta 1	1
Beta 1	1
Epsilon	10 <sup>-8</sup>
Decay rate	0
Number of LSTM units (for both LSTM layers)	{ <b>32</b> , 64}

### 5.3.2.2. Hybrid Evolutionary CNN-LSTM Results

Only 4 alterations from LSTM layers are reported here. Based on the results of the previous experiment, the CNN layers configurations were held constant with; 1 dimension 2 CNN layers. The results of RMSE and prediction score from LSTM alterations are detailed in Table 19. The number of LSTM units was 32 and 64, and the number of LSTM layers were varied from 1, 2, and 3 layers. The best prediction result came from a configuration with 2 layers of LSTM with 32 units. The results from the hybrid scheme are proven to be superior compared to what has been achieved from the homogenous scheme or in our case CNN layers as RMSE and prediction score illustrated in Table 17 and 19, as well as prediction curve in Figure 38, 39, 40 and 41.

The RUL prediction curves from the best Evolutionary Hybrid CNN-LSTM model (1 dimension 2 CNN layers, and 2 LSTM layers) are seen in Figures 39 and 40 for FD002 and FD004 test data. Same as the previous curves, the blue curves represent the actual RUL from the dataset, and the red lines/dots are the prediction points from the CNN model.

Table 19. RMSE and prediction score for RUL prediction from different CNN-LSTM configurations.

Model Configurations	RMSE		Score		
	FD002	FD004	FD002	FD004	
32 Units 1 LSTM Layer	30.938	32.856	500,139	436,584	
64 Units 1 LSTM Layer	34.579	33.976	458,890	420,895	
<b>32 Units 2 LSTM Layers</b>	<b>15.376</b>	<b>16.743</b>	<b>306,768</b>	<b>331,852</b>	<b>Best Overall</b>
64 Units 2 LSTM Layers	18.975	20.233	302,896	324,142	
32 Units 3 LSTM Layers	22.491	25.154	422,679	464,280	
64 Units 3 LSTM Layers	24.320	23.437	405,960	521,214	

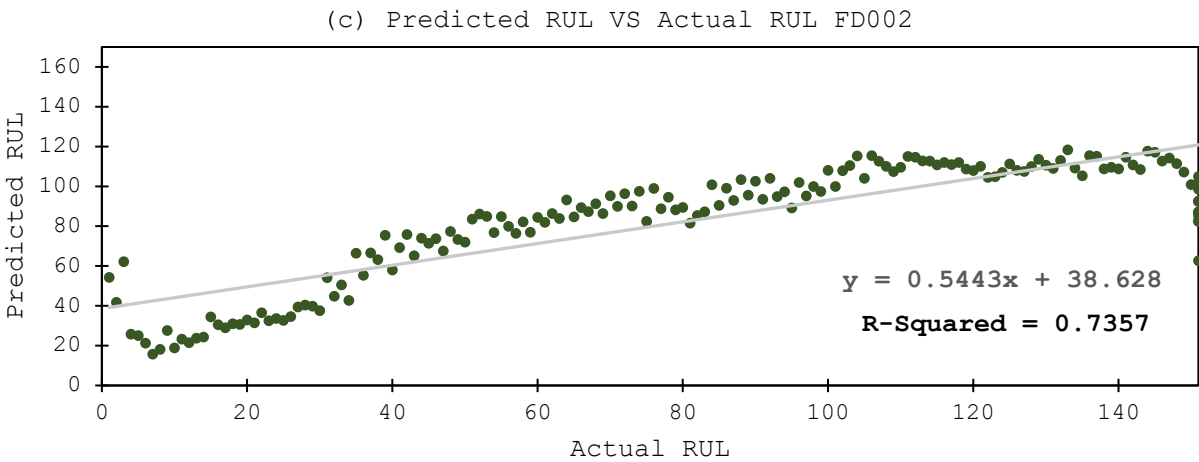
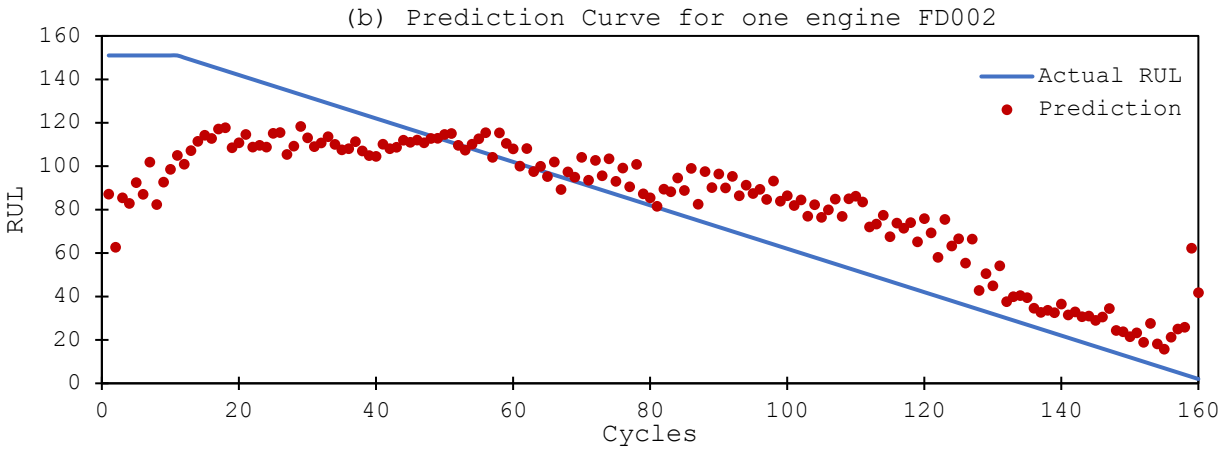
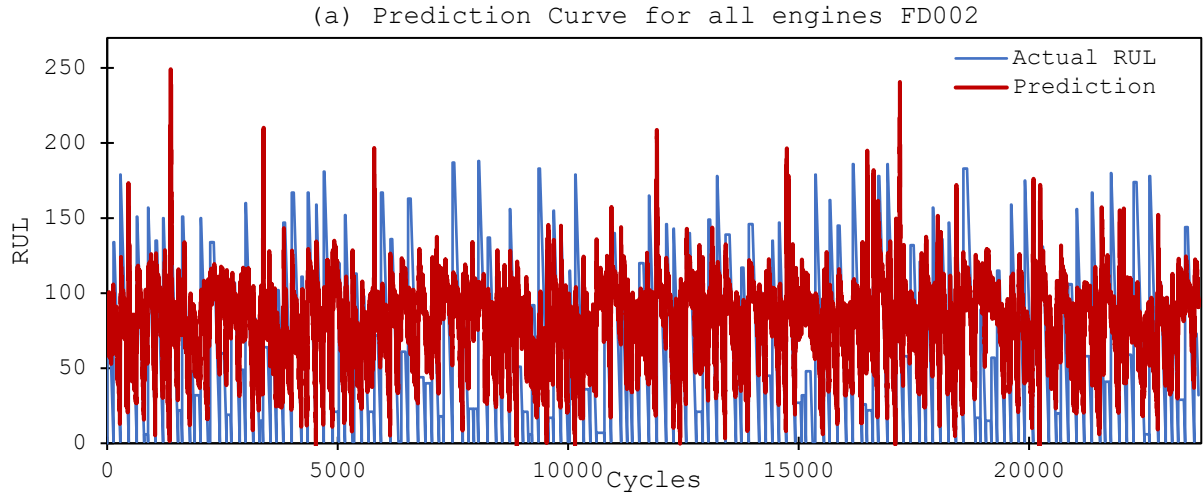


Figure 38. (a, b, and c) Evolutionary CNN RUL prediction curves for FD002 test data.

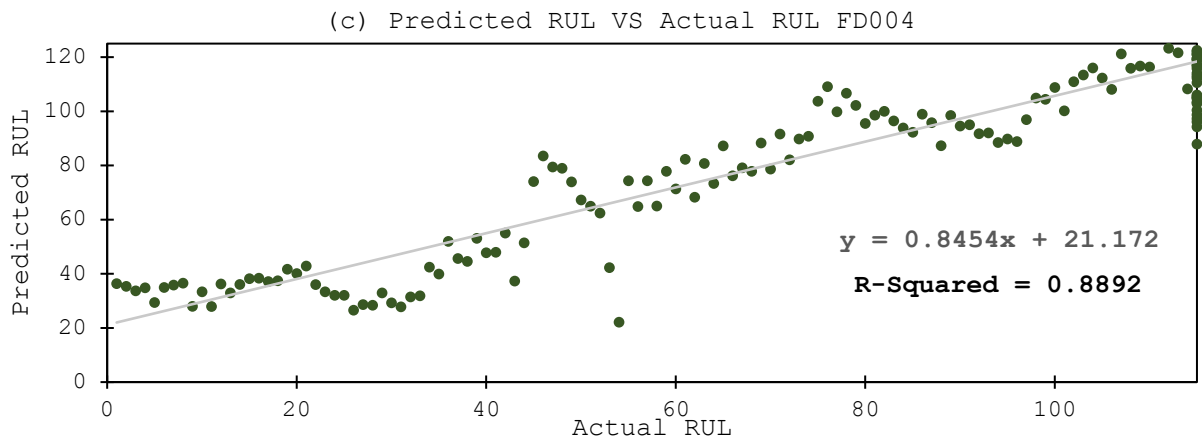
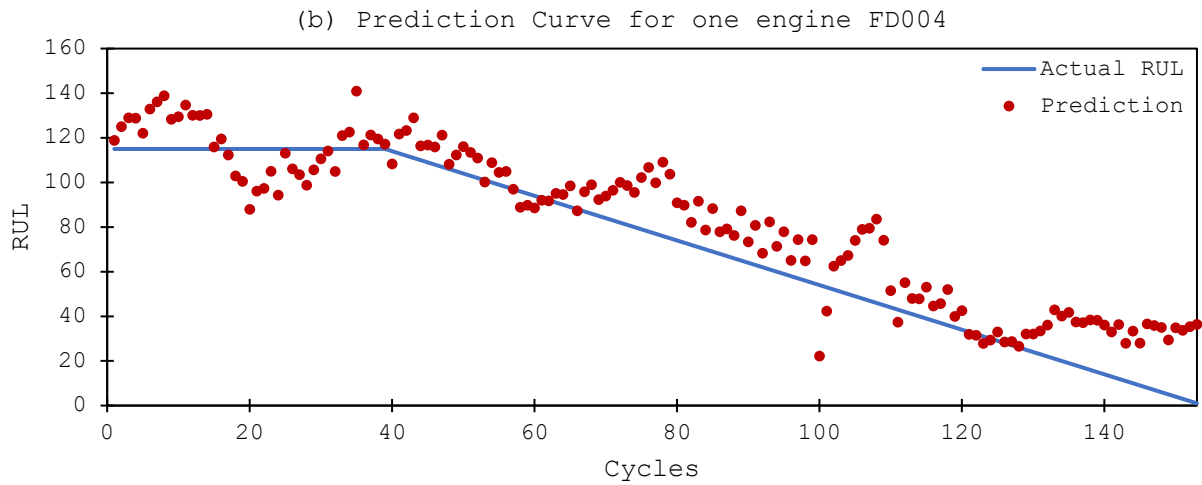
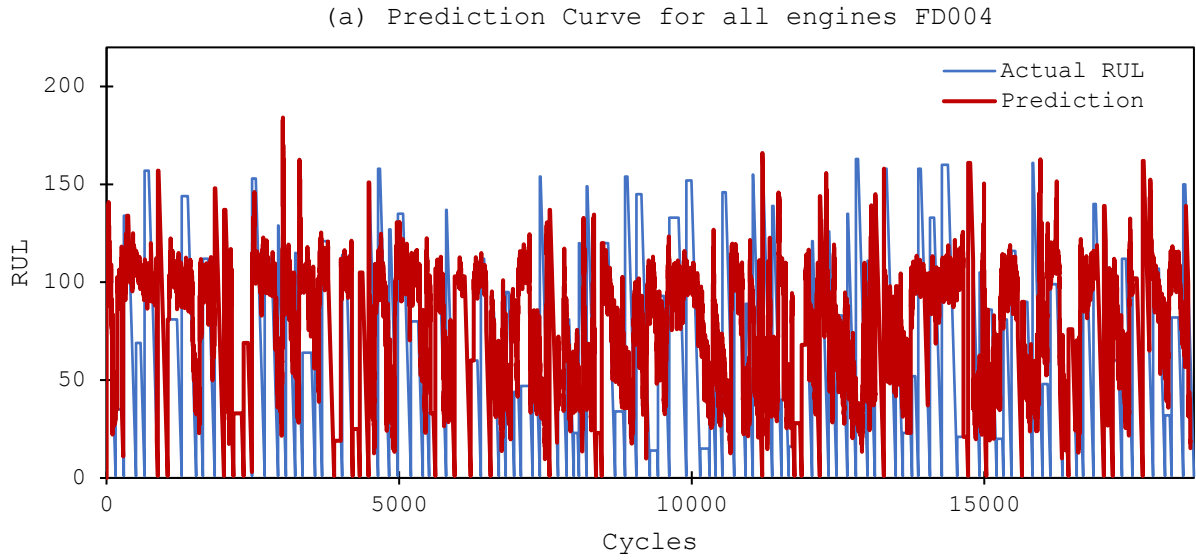


Figure 39. (a, b, and c) Evolutionary CNN RUL prediction curves for FD004 test data.

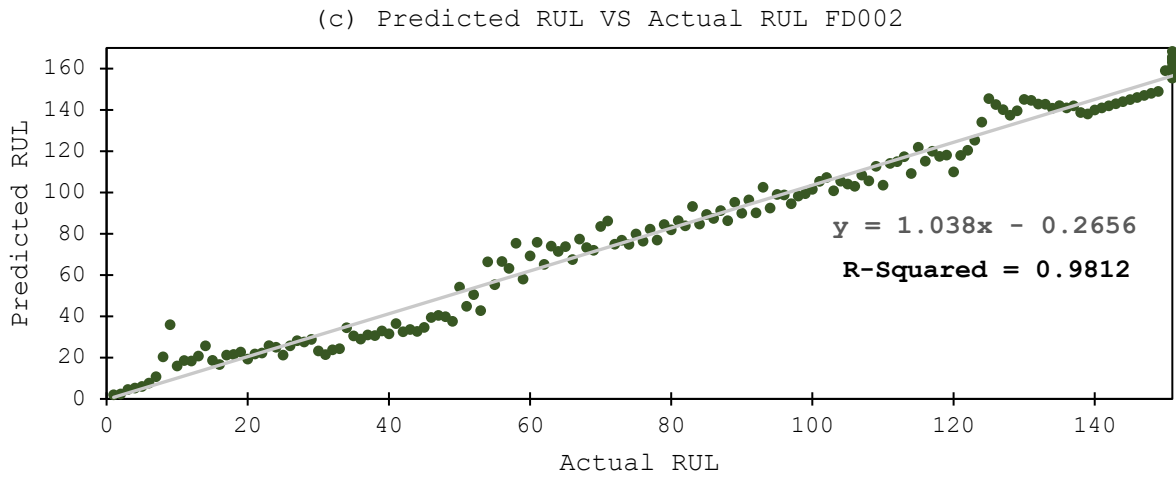
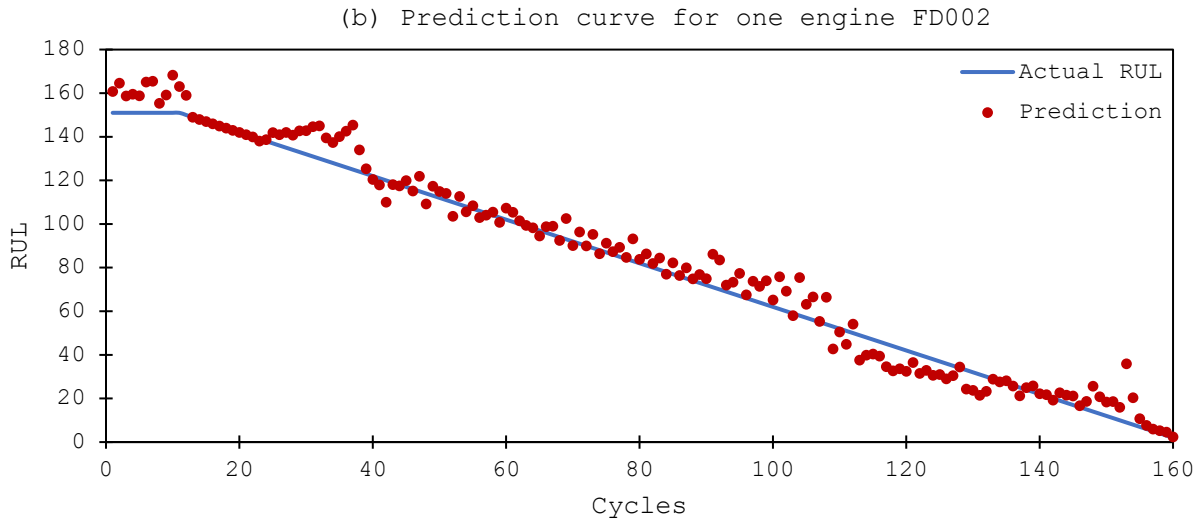
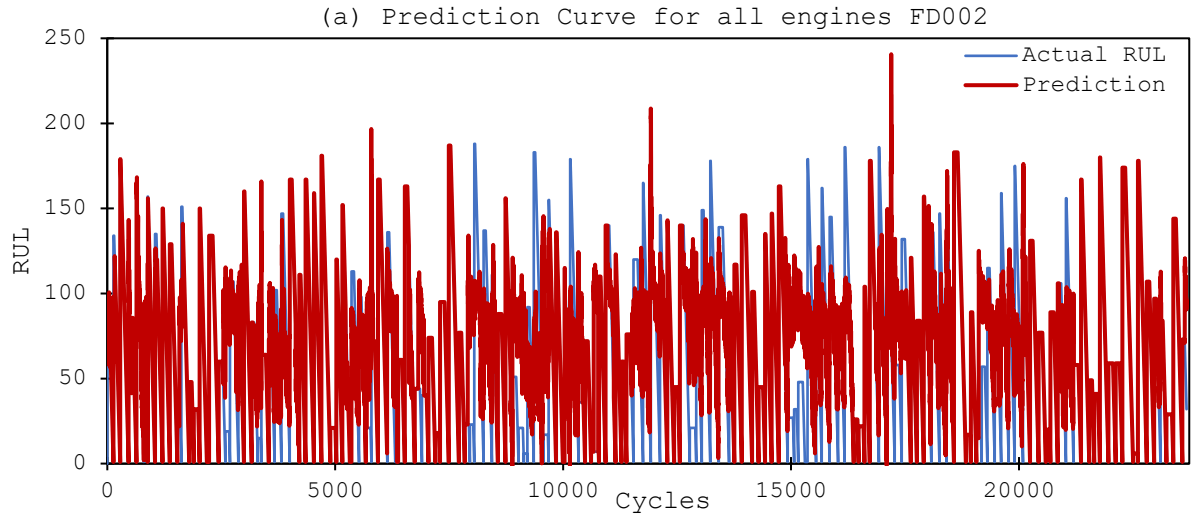


Figure 40. (a, b, and c) Hybrid evolutionary CNN-LSTM RUL prediction curves for FD002 test data.



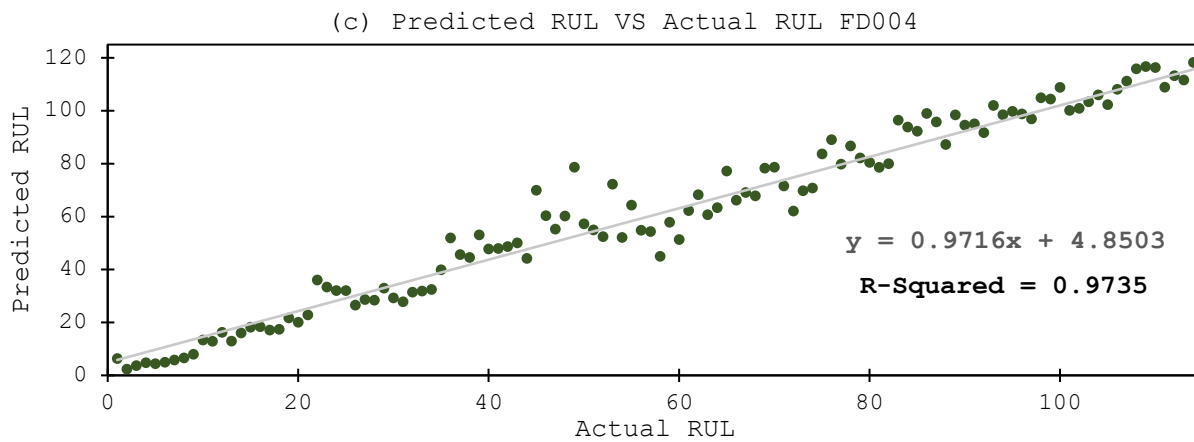
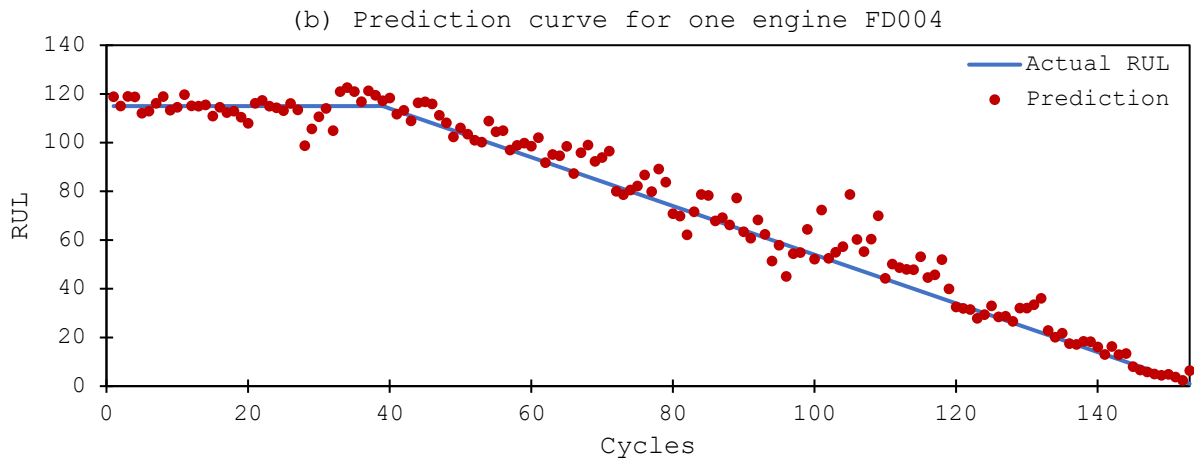
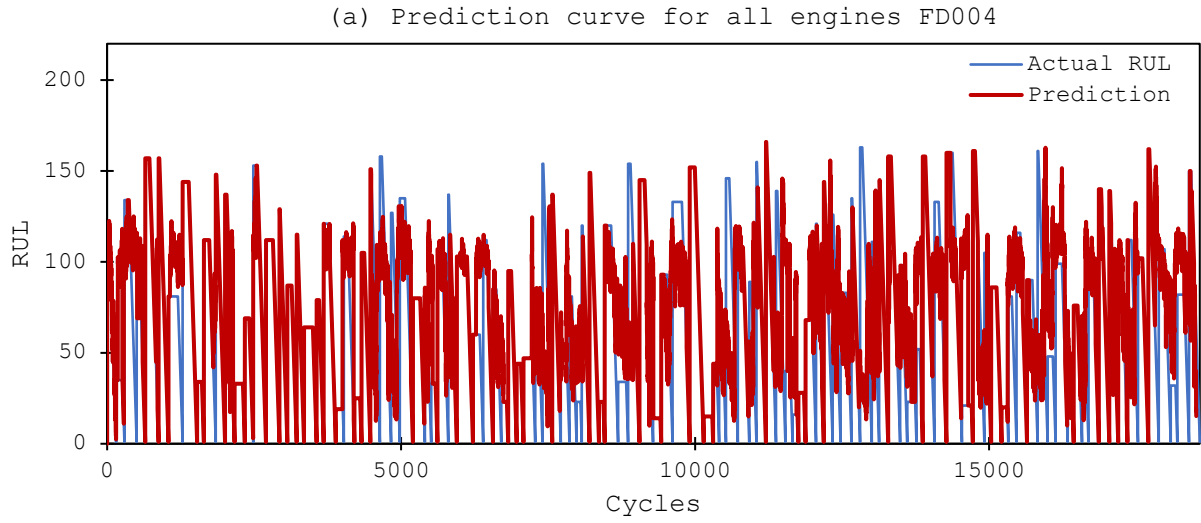


Figure 41. (a, b, and c) Hybrid evolutionary CNN-LSTM RUL prediction curves for FD004 test data.

#### 5.4. Result Discussion

As seen in Table 19 in the previous section, the hybrid evolutionary CNN-LSTM model proposed in this chapter has better prediction accuracy compared to the vanilla CNN model. The best CNN model can only archive 38.24 RMSE, while the hybrid evolutionary CNN-LSTM can produce a prediction error at 15.38 RSME. In addition, when comparing the best result among all the deep neural models previously proposed; Hybrid, CNN, and DNN (previously proposed Chapter 4), the vanilla DNN can produce the best performance with a 44.71 RMSE, which is the largest error among the other two models.

While it has been seen from our baseline that the evolutionary hybrid approach can produce predictions with smaller error compared to both DNN and CNN, it has also been seen to perform slightly better than the hybrid architectures proposed in other literature as well. As mentioned in Chapter 4, there are several efforts over the years to improve the deep learning models for predicting the RUL of the C-MAPSS aircraft gas turbine engines dataset [90, 94, 111, 173, 175-179]. Based on this study, the best result of the hybrid deep layers model was from Zhengmin Kong, et al. [177]. Their hybrid architecture was constructed with CNN and LSTM-RNN combined and achieved error of 16.13 RMSE, while the proposed evolutionary hybrid scheme achieved an error of 15.38 RMSE. Given that genetic algorithm [182] or evolutionary selection has been applied to the input data to reduce the complexity before the model training phase of or model, the evolutionary hybrid approach not only perform better but also provide less complexity, which can consider being a significant improvement among other models previously proposed. Additionally, when comparing R-Squared value between standard CNN and evolutionary CNN-LSTM hybrid approach, the hybrid CNN-LSTM also perform better as illustrated in Figure 38 through 41. For FD002 the R-Squared of CNN is 0.736 and hybrid CNN-LSTM is 0.981. While for FD004 the R-Squared of CNN and hybrid CNN-LSTM are 0.889 and 0.974 respectively.

In summary the experiments and models introduced in this chapter are extensions from the previous chapter. In Chapter 4, provided a baseline using a neural network-based feature selection approach to improve on reducing the complexity of the model. The main goal of the experiments in Chapter 4 was to demonstrate that feature selection to improve machine learning, In this chapter, both feature selection and hybrid modeling was studied. It was seen that these models are able to predict outcomes reasonably well with accelerated learning.

One aspect that has not been tested is the fluctuation in the prediction results or the “uncertainty” in the model. The reported results were obtained from training and testing the model three times. While we reported the RMSE average of one hundred trials and observed the fluctuation for DNN models in Chapter 4, the reported RMSE of the CNN models and the hybrid CNN-LSTM models are from only three trials. This is because of the limitation of running CNN and the CNN-LSTM algorithm. With the proposed configurations of the model using the specified computing machine described in section 5.3, it took 4 to 5 hours on average per training and testing of a model. However, the RMSE results from the three times training and testing were very close with only  $\pm 5\%$  difference on average. The same effect occurred across all experiments using both vanilla CNN and hybrid CNN-LSTM algorithms.

### **5.5. Chapter Summary**

The work in this chapter was an extension of the experiment from the previous chapter. The set of features using the evolutionary selection method has been used to develop models with better prediction accuracy using deep Convolutional Neural Network (CNN.) and applied to a hybrid Convolutional Long Short-Term Memory Neural Network (CNN-LSTM.) The results from both CNN and CNN-LSTM are better compared to ones obtained from the previous chapter based on only DNN. The result from CNN-LSTM has larger prediction error compared to existing models recently proposed by Z. Kong, et al. [177] as well as being more complex. Despite the approaches used to implement the “Evolutionary Convolutional Long Short-Term Memory Deep Neural Network Data-Driven Model for Prognostics of Aircraft Gas Turbine” in this work might have already been outperformed other existing works in the same nature in terms of prediction accuracy, the future tasks to further improve the work within the same scope remains. Better fine-tuning methods to achieve better accuracy from the adjacent or similar model architectures might be needed. Additionally, some aspects of dimensionality reduction such as PCA might be included to reduce the model complexity. In this case, a model was tested using PCA but the prediction results were unusable, which might need to investigate more in the future. Moreover, more prognostics datasets might need to be tested and developed in order to have a definite conclusion on what might be a universally best algorithm, or model, or architecture, ultimately suitable for prognostics data.

## 6. GENERAL SUMMARY

This dissertation works as a part of ongoing studies on deep learning for prognostics and health management applications and developed new deep learning models and approaches for prognostics and health management applications. In the early chapters, the groundworks of both Prognostics and Health Management (PHM) and Deep Learning (DL) were introduced and investigated. All the PHM matrices, ideologies terms, and units were included in this chapter as well as the general concepts of DL. In Chapter 1, the PHM and DL were introduced as non-related entities. In Chapter 2, the PHM and DL were linked. The aspects of PHM that DL were applied to further investigate, the development of models to predict the Remaining Useful Life (RUL). The challenges and research gaps in this have also been addressed in this chapter. Additionally, an in-depth survey of the PHM approaches that applied DL algorithms over the years were conducted in Chapter 2. Because DL is still relatively new to PHM applications, there are areas that can be further investigated and improved for multiple PHM applications. Two of which, PHM for Li-ion battery and aircraft gas turbine engine are possibilities. These two applications are the most popular among many PHM applications and can be used to develop the approach, that believes to possibly be, a universal schema in PHM research. In Chapter 3, the Li-ion battery data was used to develop a basic PHM deep learning model. The work in this chapter was dedicated to proving some examples that DL algorithms can most likely perform better compared to other traditional Machine Learning (ML) algorithms in predictive regression tasks. The result from this chapter showed that the model using DL is better in terms of predicting the RUL of Li-ion battery compared to other ML algorithms. In Chapter 4, part of the proposed new framework (as in Figure 7) was tested as the framework that potentially overcome some of the challenges of using DL in PHM addressed in Chapter 2. The feature selection methods were applied to the development of PHM model for the C-MAPSS aircraft gas turbine engine. The experiments in this chapter were limited to investigating the effect of feature reduction approaches for neural network-based or deep learning-based algorithms. Therefore, only neural network-based algorithms have been employed. The results showed that for PHM of aircraft gas turbine engine, the genetic algorithm or evolutionary selection approach can perform best and able to select only meaningful features or input data for DL-based modeling. However, the results were unsatisfactory when compared to other works prior published using the same dataset in terms of accuracy prediction.

However, this part aimed to be a baseline for choosing an appropriate set of features to implement DL for PHM of aircraft gas turbine engine data and provide concepts how to further improve the DL models using similar approaches for PHM applications only. In Chapter 5, the assumption from Chapter 4 were further improved for the DL model for aircraft gas turbine engines RUL predictions. The hybrid scheme was applied to the selected data from Chapter 4 to develop hybrid DL models. The result using both feature selection approach and hybrid scheme improved the prediction results in terms of both complexity and accuracy. In this chapter, a proposed “An Evolutionary Convolutional Long Short-Term Memory Deep Neural Network Data-Driven Model for Prognostics of Aircraft Gas Turbine” as the model was implemented based on all combinations of aforementioned methods and approaches. This final model outperformed the best existing models that fall within similar nature (based on the study between 2019 and 2020.) Lastly, as mentioned in Chapter 5, even though the final evolutionary CNN-LSTM proposed here was the best one among others. Better fine-tuning or optimizing methods may be needed to further improve the model. Additionally, more diverse PHM applications must be tested using similar approaches. All things considered, the approaches and schema proposed in this dissertation believe to be able to universally be used for most of the PHM datasets that employed DL as a modeling algorithm. The study in this dissertation reduces dilemmas on using DL in PHM among prognostics and diagnostics research communities.

## REFERENCES

- [1] R. Agrawal and G. Psaila, "Active Data Mining," in *KDD*, 1995, pp. 3-8.
- [2] S.-H. Liao, P.-H. Chu, and P.-Y. Hsiao, "Data mining techniques and applications—A decade review from 2000 to 2011," *Expert systems with applications*, vol. 39, no. 12, pp. 11303-11311, 2012.
- [3] M. Karegar, A. Isazadeh, F. Fartash, T. Sadari, and A. H. Navin, "Data-mining by probability-based patterns," in *ITI 2008-30th International Conference on Information Technology Interfaces*, 2008: IEEE, pp. 353-360.
- [4] T. Hill, P. Lewicki, and P. Lewicki, *Statistics: methods and applications: a comprehensive reference for science, industry, and data mining*. StatSoft, Inc., 2006.
- [5] I. H. Witten and E. Frank, "Data mining: practical machine learning tools and techniques with Java implementations," *Acm Sigmod Record*, vol. 31, no. 1, pp. 76-77, 2002.
- [6] D. K. Bhattacharyya and S. M. Hazarika, *Networks, data mining, and artificial intelligence: trends and future directions*. Narosa Pub House, 2006.
- [7] B. Samanta, "Artificial neural networks and genetic algorithms for gear fault detection," *Mechanical Systems and Signal Processing*, vol. 5, no. 18, pp. 1273-1282, 2004.
- [8] J. Yang, Y. Zhang, and Y. Zhu, "Intelligent fault diagnosis of rolling element bearing based on SVMs and fractal dimension," *Mechanical Systems and Signal Processing*, vol. 21, no. 5, pp. 2012-2024, 2007.
- [9] D. Liu, Y. Luo, Y. Peng, X. Peng, and M. Pecht, "Lithium-ion battery remaining useful life estimation based on nonlinear AR model combined with degradation feature," in *Annual Conference of the PHM Society*, 2012, vol. 4, no. 1.
- [10] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504-507, 2006.
- [11] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85-117, 2015.

- [12] D. S. Chaplot, C. MacLellan, R. Salakhutdinov, and K. Koedinger, "Learning cognitive models using neural networks," in *International Conference on Artificial Intelligence in Education*, 2018: Springer, pp. 43-56.
- [13] C. Chen, B. Zhang, G. Vachtsevanos, and M. Orchard, "Machine condition prediction based on adaptive neuro-fuzzy and high-order particle filtering," *IEEE Transactions on Industrial Electronics*, vol. 58, no. 9, pp. 4353-4364, 2010.
- [14] K. Medjaher, D. A. Tobon-Mejia, and N. Zerhouni, "Remaining useful life estimation of critical components with application to bearings," *IEEE Transactions on Reliability*, vol. 61, no. 2, pp. 292-302, 2012.
- [15] E. Ramasso and R. Gouriveau, "Prognostics in switching systems: Evidential Markovian classification of real-time neuro-fuzzy predictions," in *2010 Prognostics and System Health Management Conference*, 2010: IEEE, pp. 1-10.
- [16] L. Peel, "Data driven prognostics using a Kalman filter ensemble of neural network models," in *2008 international conference on prognostics and health management*, 2008: IEEE, pp. 1-6.
- [17] C. Hu, B. D. Youn, P. Wang, and J. T. Yoon, "Ensemble of data-driven prognostic algorithms for robust prediction of remaining useful life," *Reliability Engineering & System Safety*, vol. 103, pp. 120-135, 2012.
- [18] P. Wang, B. D. Youn, and C. Hu, "Concurrent design of functional reliability and failure prognosis for engineered resilience," in *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, 2013, vol. 55881: American Society of Mechanical Engineers, p. V03AT03A036.
- [19] E. Scanff, K. Feldman, S. Ghelam, P. Sandborn, M. Glade, and B. Foucher, "Life cycle cost impact of using prognostic health management (PHM) for helicopter avionics," *Microelectronics reliability*, vol. 47, no. 12, pp. 1857-1864, 2007.
- [20] G. J. Vachtsevanos, *Intelligent fault diagnosis and prognosis for engineering systems*. Wiley Online Library, 2006.
- [21] I. T. Jolliffe and D. B. Stephenson, *Forecast verification: a practitioner's guide in atmospheric science*. John Wiley & Sons, 2012.

- [22] O. Eker, F. Camci, and I. K. Jennions, "Major challenges in prognostics: Study on benchmarking prognostics datasets," in *PHM Society European Conference*, 2012, vol. 1, no. 1.
- [23] I. Jennions, *Integrated Vehicle Health Management: Essential Reading*. SAE, 2013.
- [24] A. Heng, S. Zhang, A. C. Tan, and J. Mathew, "Rotating machinery prognostics: State of the art, challenges and opportunities," *Mechanical systems and signal processing*, vol. 23, no. 3, pp. 724-739, 2009.
- [25] J. Luo, M. Namburu, K. Pattipati, L. Qiao, M. Kawamoto, and S. Chigusa, "Model-based prognostic techniques [maintenance applications]," in *Proceedings AUTOTESTCON 2003. IEEE Systems Readiness Technology Conference.*, 2003: IEEE, pp. 330-340.
- [26] H. Zhang, R. Kang, and M. Pecht, "A hybrid prognostics and health management approach for condition-based maintenance," in *2009 IEEE International Conference on Industrial Engineering and Engineering Management*, 2009: IEEE, pp. 1165-1169.
- [27] A. Saxena *et al.*, "Metrics for evaluating performance of prognostic techniques," in *2008 international conference on prognostics and health management*, 2008: IEEE, pp. 1-17.
- [28] T. Hegazy, P. Fazio, and O. Moselhi, "Developing practical neural network applications using back-propagation," *Computer-Aided Civil and Infrastructure Engineering*, vol. 9, no. 2, pp. 145-159, 1994.
- [29] J. Jeon, "Fuzzy and neural network models for analyses of piles." (accessed 2020). Available: <https://repository.lib.ncsu.edu/bitstream/handle/1840.16/5156/etd.pdf>
- [30] M. W. Emsley, D. J. Lowe, A. R. Duff, A. Harding, and A. Hickson, "Data modelling and the application of a neural network approach to the prediction of total construction costs," *Construction Management & Economics*, vol. 20, no. 6, pp. 465-472, 2002.
- [31] D. P. Alex, M. Al Hussein, A. Bouferguene, and S. Fernando, "Artificial neural network model for cost estimation: City of Edmonton's water and sewer installation services," *Journal of construction engineering and management*, vol. 136, no. 7, pp. 745-756, 2010.
- [32] G. Zhao, G. Zhang, Q. Ge, and X. Liu, "Research advances in fault diagnosis and prognostic based on deep learning," in *2016 Prognostics and system health management conference (PHM-Chengdu)*, 2016: IEEE, pp. 1-6.



- [33] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735-1780, 1997.
- [34] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE international conference on acoustics, speech and signal processing*, 2013: Ieee, pp. 6645-6649.
- [35] P. Wang and G. Vachtsevanos, "Fault prognostics using dynamic wavelet neural networks," *AI EDAM*, vol. 15, no. 4, pp. 349-365, 2001.
- [36] C. Byington, M. Roemer, P. Kalgren, and G. Vachtsevanos, "Verification and validation of diagnostic/prognostic algorithms," in *Machinery Failure Prevention Technology Conference (MFPT 59)*, 2005.
- [37] G. Vachtsevanos, "Performance metrics for fault prognosis of complex systems," in *Proceedings AUTOTESTCON 2003. IEEE Systems Readiness Technology Conference.*, 2003: IEEE, pp. 341-345.
- [38] V. Aho Alfred *et al.*, *Data structures and algorithms*. USA: Addison-Wesley, 1983.
- [39] D. L. Goodman, S. Wood, and A. Turner, "Return-on-investment (ROI) for electronic prognostics in mil/aero systems," in *IEEE Autotestcon, 2005.*, 2005: IEEE, pp. 73-75.
- [40] S. Vohnout, D. Goodman, J. Judkins, M. Kozak, and K. Harris, "Electronic prognostics system implementation on power actuator components," in *2008 IEEE aerospace conference*, 2008: IEEE, pp. 1-11.
- [41] S. M. Wood and D. L. Goodman, "Return-on-investment (ROI) for electronic prognostics in high reliability telecom applications," in *INTELEC 06-Twenty-Eighth International Telecommunications Energy Conference*, 2006: IEEE, pp. 1-3.
- [42] R. J. Hyndman and A. B. Koehler, "Another look at measures of forecast accuracy," *International journal of forecasting*, vol. 22, no. 4, pp. 679-688, 2006.
- [43] K. Goebel and P. Bonissone, "Prognostic information fusion for constant load systems," in *2005 7th International Conference on Information Fusion*, 2005, vol. 2: IEEE, p. 9 pp.
- [44] S. Makridakis *et al.*, "The accuracy of extrapolation (time series) methods: Results of a forecasting competition," *Journal of forecasting*, vol. 1, no. 2, pp. 111-153, 1982.

- [45] B. Ebert, "Forecast Verification-Issues, Methods and FAQ," *World Weather Research Programme Joint Working Group on Verification*, 2006.
- [46] M. Stuart, "Understanding Robust and Exploratory Data Analysis," ed: Wiley Online Library, 1984.
- [47] T. N. Palmer *et al.*, "Development of a European multimodel ensemble system for seasonal-to-interannual prediction (DEMETER)," *Bulletin of the American Meteorological Society*, vol. 85, no. 6, pp. 853-872, 2004.
- [48] R. Wirth and J. Hipp, "CRISP-DM: Towards a standard process model for data mining," in *Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining*, 2000, vol. 1: Springer-Verlag London, UK.
- [49] D. Erhan, P.-A. Manzagol, Y. Bengio, S. Bengio, and P. Vincent, "The difficulty of training deep architectures and the effect of unsupervised pre-training," in *Artificial Intelligence and Statistics*, 2009: PMLR, pp. 153-160.
- [50] E. Zio, "Some challenges and opportunities in reliability engineering," *IEEE Transactions on Reliability*, vol. 65, no. 4, pp. 1769-1782, 2016.
- [51] S. T. Kandukuri, A. Klausen, H. R. Karimi, and K. G. Robbersmyr, "A review of diagnostics and prognostics of low-speed machinery towards wind turbine farm-level health management," *Renewable and Sustainable Energy Reviews*, vol. 53, pp. 697-708, 2016.
- [52] Y. Lei, N. Li, L. Guo, N. Li, T. Yan, and J. Lin, "Machinery health prognostics: A systematic review from data acquisition to RUL prediction," *Mechanical systems and signal processing*, vol. 104, pp. 799-834, 2018.
- [53] M. Compare, P. Baraldi, and E. Zio, "Challenges to IoT-enabled predictive maintenance for industry 4.0," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4585-4597, 2019.
- [54] C. L. Gan, "Prognostics and health management of electronics: Fundamentals, machine learning, and the internet of things," ed: Springer, 2020.
- [55] I. K. Jennions, O. Niculita, and M. Esperon-Miguez, "Integrating IVHM and asset design," *International Journal of Prognostics and Health Management*, vol. 7, no. 2, 2016.

- [56] L. Yang, Q. Sun, and Z.-S. Ye, "Designing mission abort strategies based on early-warning information: Application to UAV," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 1, pp. 277-287, 2019.
- [57] D. Han, J. Yu, Y. Song, D. Tang, and J. Dai, "A distributed autonomic logistics system with parallel-computing diagnostic algorithm for aircrafts," in *2019 IEEE AUTOTESTCON*, 2019: IEEE, pp. 1-8.
- [58] D. Kwon, M. R. Hodkiewicz, J. Fan, T. Shibutani, and M. G. Pecht, "IoT-based prognostics and systems health management for industrial applications," *IEEE Access*, vol. 4, pp. 3659-3670, 2016.
- [59] K. Goebel, M. J. Daigle, A. Saxena, I. Roychoudhury, S. Sankararaman, and J. R. Celaya, *Prognostics: The science of making predictions*. 2017.
- [60] B. Samanta and K. Al-Balushi, "Artificial neural network based fault diagnostics of rolling element bearings using time-domain features," *Mechanical systems and signal processing*, vol. 17, no. 2, pp. 317-328, 2003.
- [61] Y. Ma *et al.*, "Deep learning for fault diagnosis based on multi-sourced heterogeneous data," in *2014 International Conference on Power System Technology*, 2014: IEEE, pp. 740-745.
- [62] O. Fink, E. Zio, and U. Weidmann, "Predicting component reliability and level of degradation with complex-valued neural networks," *Reliability Engineering & System Safety*, vol. 121, pp. 198-206, 2014.
- [63] W. Lu, X. Wang, C. Yang, and T. Zhang, "A novel feature extraction method using deep neural network for rolling bearing fault diagnosis," in *The 27th Chinese Control and Decision Conference (2015 CCDC)*, 2015: IEEE, pp. 2427-2431.
- [64] J. Qiu, W. Liang, L. Zhang, X. Yu, and M. Zhang, "The early-warning model of equipment chain in gas pipeline based on DNN-HMM," *Journal of Natural Gas Science and Engineering*, vol. 27, pp. 1710-1722, 2015.
- [65] K. Li and Q. Wang, "Study on signal recognition and diagnosis for spacecraft based on deep learning method," in *2015 Prognostics and System Health Management Conference (PHM)*, 2015: IEEE, pp. 1-5.

- [66] Y. Lei, F. Jia, X. Zhou, and J. Lin, "A deep learning-based method for machinery health monitoring with big data," *Journal of Mechanical Engineering*, vol. 51, no. 21, pp. 49-56, 2015.
- [67] S. Sarkar *et al.*, "Early detection of combustion instability from hi-speed flame images via deep learning and symbolic time series analysis," in *Annual Conference of the PHM Society*, 2015, vol. 7, no. 1.
- [68] F. Jia, Y. Lei, J. Lin, X. Zhou, and N. Lu, "Deep neural networks: A promising tool for fault characteristic mining and intelligent diagnosis of rotating machinery with massive data," *Mechanical Systems and Signal Processing*, vol. 72, pp. 303-315, 2016.
- [69] H. Liu, L. Li, and J. Ma, "Rolling bearing fault diagnosis based on STFT-deep learning and sound signals," *Shock and Vibration*, vol. 2016, 2016.
- [70] W. Sun, S. Shao, R. Zhao, R. Yan, X. Zhang, and X. Chen, "A sparse auto-encoder-based deep neural network approach for induction motor faults classification," *Measurement*, vol. 89, pp. 171-178, 2016.
- [71] Y. Lei, F. Jia, J. Lin, S. Xing, and S. X. Ding, "An intelligent fault diagnosis method using unsupervised feature learning towards mechanical big data," *IEEE Transactions on Industrial Electronics*, vol. 63, no. 5, pp. 3137-3147, 2016.
- [72] F. Zhou, Y. Gao, and C. Wen, "A novel multimode fault classification method based on deep learning," *Journal of Control Science and Engineering*, vol. 2017, 2017.
- [73] K. Ma, H. Leung, E. Jalilian, and D. Huang, "Deep learning on temporal-spectral data for anomaly detection," in *Ground/Air Multisensor Interoperability, Integration, and Networking for Persistent ISR VIII*, 2017, vol. 10190: International Society for Optics and Photonics, p. 101900D.
- [74] G. Jiang, P. Xie, H. He, and J. Yan, "Wind turbine fault detection using a denoising autoencoder with temporal information," *IEEE/Asme transactions on mechatronics*, vol. 23, no. 1, pp. 89-100, 2017.
- [75] P. Bangalore, S. Letzgus, D. Karlsson, and M. Patriksson, "An artificial neural network-based condition monitoring method for wind turbines, with application to the monitoring of the gearbox," *Wind Energy*, vol. 20, no. 8, pp. 1421-1438, 2017.

- [76] Z. Zhao, B. Liang, X. Wang, and W. Lu, "Remaining useful life prediction of aircraft engine based on degradation pattern learning," *Reliability Engineering & System Safety*, vol. 164, pp. 74-83, 2017.
- [77] H. Xiao, D. Huang, Y. Pan, Y. Liu, and K. Song, "Fault diagnosis and prognosis of wastewater processes with incomplete data by the auto-associative neural networks and ARMA model," *Chemometrics and Intelligent Laboratory Systems*, vol. 161, pp. 96-107, 2017.
- [78] E. Chemali, P. J. Kollmeyer, M. Preindl, and A. Emadi, "State-of-charge estimation of Li-ion batteries using deep neural networks: A machine learning approach," *Journal of Power Sources*, vol. 400, pp. 242-255, 2018.
- [79] S. Tolo *et al.*, "Robust on-line diagnosis tool for the early accident detection in nuclear power plants," *Reliability Engineering & System Safety*, vol. 186, pp. 110-119, 2019.
- [80] A. P. Marugán, A. M. P. Chacón, and F. P. G. Márquez, "Reliability analysis of detecting false alarms that employ neural networks: A real case study on wind turbines," *Reliability Engineering & System Safety*, vol. 191, p. 106574, 2019.
- [81] Z. Chen, C. Li, and R.-V. Sanchez, "Gearbox fault identification and classification with convolutional neural networks," *Shock and Vibration*, vol. 2015, 2015.
- [82] G. S. Babu, P. Zhao, and X.-L. Li, "Deep convolutional neural network based regression approach for estimation of remaining useful life," in *International conference on database systems for advanced applications*, 2016: Springer, pp. 214-228.
- [83] O. Janssens *et al.*, "Convolutional neural network based fault detection for rotating machinery," *Journal of Sound and Vibration*, vol. 377, pp. 331-345, 2016.
- [84] T. Ince, S. Kiranyaz, L. Eren, M. Askar, and M. Gabbouj, "Real-time motor fault detection by 1-D convolutional neural networks," *IEEE Transactions on Industrial Electronics*, vol. 63, no. 11, pp. 7067-7075, 2016.
- [85] H. Dong, L. Yang, and H. Li, "Small fault diagnosis of front-end speed controlled wind generator based on deep learning," *WSEAS Trans. Circuits Syst*, vol. 15, no. 9, pp. 64-72, 2016.
- [86] X. Gibert, V. M. Patel, and R. Chellappa, "Deep multitask learning for railway track inspection," *IEEE transactions on intelligent transportation systems*, vol. 18, no. 1, pp. 153-164, 2016.

- [87] C. Lu, Z. Wang, and B. Zhou, "Intelligent fault diagnosis of rolling bearing using hierarchical convolutional network based health state classification," *Advanced Engineering Informatics*, vol. 32, pp. 139-151, 2017.
- [88] M. Xia, T. Li, L. Xu, L. Liu, and C. W. De Silva, "Fault diagnosis for rotating machinery using multiple sensors and convolutional neural networks," *IEEE/ASME transactions on mechatronics*, vol. 23, no. 1, pp. 101-110, 2017.
- [89] O. Janssens, R. Van de Walle, M. Loccupier, and S. Van Hoecke, "Deep learning for infrared thermal image based machine health monitoring," *IEEE/ASME Transactions on Mechatronics*, vol. 23, no. 1, pp. 151-159, 2017.
- [90] X. Li, Q. Ding, and J.-Q. Sun, "Remaining useful life estimation in prognostics using deep convolution neural networks," *Reliability Engineering & System Safety*, vol. 172, pp. 1-11, 2018.
- [91] W. Xuhong and H. Yigang, "Diagonal recurrent neural network based on-line stator winding turn fault detection for induction motors," in *2005 international conference on electrical machines and systems*, 2005, vol. 3: IEEE, pp. 2266-2269.
- [92] Q. Hu, M. Xie, S. H. Ng, and G. Levitin, "Robust recurrent neural network modeling for software fault detection and correction prediction," *Reliability Engineering & System Safety*, vol. 92, no. 3, pp. 332-340, 2007.
- [93] O. Obst, "Distributed fault detection in sensor networks using a recurrent neural network," *Neural processing letters*, vol. 40, no. 3, pp. 261-273, 2014.
- [94] M. Yuan, Y. Wu, and L. Lin, "Fault diagnosis and remaining useful life estimation of aero engine using LSTM neural network," in *2016 IEEE international conference on aircraft utility systems (AUS)*, 2016: IEEE, pp. 135-140.
- [95] T. De Bruin, K. Verbert, and R. Babuška, "Railway track circuit fault diagnosis using recurrent neural networks," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 3, pp. 523-533, 2016.
- [96] L. Guo, N. Li, F. Jia, Y. Lei, and J. Lin, "A recurrent neural network based health indicator for remaining useful life prediction of bearings," *Neurocomputing*, vol. 240, pp. 98-109, 2017.

- [97] S. Zhang, Y. Wang, M. Liu, and Z. Bao, "Data-based line trip fault prediction in power systems using LSTM networks and SVM," *Ieee Access*, vol. 6, pp. 7675-7686, 2017.
- [98] Y. Zhang, R. Xiong, H. He, and Z. Liu, "A LSTM-RNN method for the lithium-ion battery remaining useful life prediction," in *2017 Prognostics and System Health Management Conference (PHM-Harbin)*, 2017: IEEE, pp. 1-4.
- [99] Y. Zhang, R. Xiong, H. He, and M. G. Pecht, "Long short-term memory recurrent neural network for remaining useful life prediction of lithium-ion batteries," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 7, pp. 5695-5705, 2018.
- [100] K. T. Nguyen and K. Medjaher, "A new dynamic predictive maintenance framework using deep learning for failure prognostics," *Reliability Engineering & System Safety*, vol. 188, pp. 251-262, 2019.
- [101] P. R. d. O. da Costa, A. Akçay, Y. Zhang, and U. Kaymak, "Remaining useful lifetime prediction via deep domain adaptation," *Reliability Engineering & System Safety*, vol. 195, p. 106682, 2020.
- [102] Z. Shi and A. Chehade, "A dual-LSTM framework combining change point detection and remaining useful life prediction," *Reliability Engineering & System Safety*, vol. 205, p. 107257, 2021.
- [103] P. Wang and C. Xia, "Fault detection and self-learning identification based on PCA-PDBNs," *Chinese Journal of Scientific Instrument*, vol. 36, no. 5, pp. 1147-1154, 2015.
- [104] H. Shao, H. Jiang, X. Zhang, and M. Niu, "Rolling bearing fault diagnosis using an optimization deep belief network," *Measurement Science and Technology*, vol. 26, no. 11, p. 115002, 2015.
- [105] Z. Chen, C. Li, and R.-V. Sánchez, "Multi-layer neural network with deep belief network for gearbox fault diagnosis," *Journal of Vibroengineering*, vol. 17, no. 5, pp. 2379-2392, 2015.
- [106] C. Li, R.-V. Sanchez, G. Zurita, M. Cerrada, D. Cabrera, and R. E. Vásquez, "Multimodal deep support vector classification with homologous features and its application to gearbox fault diagnosis," *Neurocomputing*, vol. 168, pp. 119-127, 2015.
- [107] C. Li, R.-V. Sanchez, G. Zurita, M. Cerrada, D. Cabrera, and R. E. Vásquez, "Gearbox fault diagnosis based on deep random forest fusion of acoustic and vibratory signals," *Mechanical Systems and Signal Processing*, vol. 76, pp. 283-293, 2016.

- [108] D. K. Jha, A. Srivastav, and A. Ray, "Temporal learning in video data using deep learning and Gaussian processes," *International Journal of Prognostics and Health Management*, vol. 7, no. 4, 2016.
- [109] R. Zhao, R. Yan, J. Wang, and K. Mao, "Learning to monitor machine health with convolutional bi-directional LSTM networks," *Sensors*, vol. 17, no. 2, p. 273, 2017.
- [110] S. Wang, J. Xiang, Y. Zhong, and Y. Zhou, "Convolutional neural network-based hidden Markov models for rolling element bearing fault identification," *Knowledge-Based Systems*, vol. 144, pp. 65-76, 2018.
- [111] A. L. Ellefsen, E. Bjørlykhaug, V. Æsøy, S. Ushakov, and H. Zhang, "Remaining useful life predictions for turbofan engine degradation using semi-supervised deep architecture," *Reliability Engineering & System Safety*, vol. 183, pp. 240-251, 2019.
- [112] X. Li, W. Zhang, and Q. Ding, "Deep learning-based remaining useful life estimation of bearings using multi-scale feature extraction," *Reliability engineering & system safety*, vol. 182, pp. 208-218, 2019.
- [113] S. Sankararaman and K. Goebel, "Uncertainty in prognostics and systems health management," *International Journal of Prognostics and Health Management*, vol. 6, no. 4, 2015.
- [114] J. L. Schafer and J. W. Graham, "Missing data: our view of the state of the art," *Psychological methods*, vol. 7, no. 2, p. 147, 2002.
- [115] I. Eekhout, R. M. de Boer, J. W. Twisk, H. C. de Vet, and M. W. Heymans, "Missing data: a systematic review of how they are reported and handled," *Epidemiology*, vol. 23, no. 5, pp. 729-732, 2012.
- [116] M. Ranjbar, P. Moradi, M. Azami, and M. Jalili, "An imputation-based matrix factorization method for improving accuracy of collaborative filtering systems," *Engineering Applications of Artificial Intelligence*, vol. 46, pp. 58-66, 2015.
- [117] R. Razavi-Far, S. Chakrabarti, M. Saif, and E. Zio, "An integrated imputation-prediction scheme for prognostics of battery data with missing observations," *Expert Systems with Applications*, vol. 115, pp. 709-723, 2019.



- [118] A. N. Baraldi and C. K. Enders, "An introduction to modern missing data analyses," *Journal of school psychology*, vol. 48, no. 1, pp. 5-37, 2010.
- [119] A. R. T. Donders, G. J. Van Der Heijden, T. Stijnen, and K. G. Moons, "A gentle introduction to imputation of missing values," *Journal of clinical epidemiology*, vol. 59, no. 10, pp. 1087-1091, 2006.
- [120] J. L. Schafer, "Multiple imputation: a primer," *Statistical methods in medical research*, vol. 8, no. 1, pp. 3-15, 1999.
- [121] D. Vergouw *et al.*, "The search for stable prognostic models in multiple imputed data sets," *BMC Medical Research Methodology*, vol. 10, no. 1, pp. 1-9, 2010.
- [122] J. Honaker and G. King, "What to do about missing values in time-series cross-section data," *American journal of political science*, vol. 54, no. 2, pp. 561-581, 2010.
- [123] U. Leturiondo, O. Salgado, L. Ciani, D. Galar, and M. Catelani, "Architecture for hybrid modelling and its application to diagnosis and prognosis with missing data," *Measurement*, vol. 108, pp. 152-162, 2017.
- [124] A. Marshall, D. G. Altman, P. Royston, and R. L. Holder, "Comparison of techniques for handling missing covariate data within prognostic modelling studies: a simulation study," *BMC medical research methodology*, vol. 10, no. 1, pp. 1-16, 2010.
- [125] T. G. Clark and D. G. Altman, "Developing a prognostic model in the presence of missing data: an ovarian cancer case study," *Journal of clinical epidemiology*, vol. 56, no. 1, pp. 28-37, 2003.
- [126] M. Xu, P. Baraldi, S. Al-Dahidi, and E. Zio, "Fault prognostics by an ensemble of Echo State Networks in presence of event based measurements," *Engineering Applications of Artificial Intelligence*, vol. 87, p. 103346, 2020.
- [127] Z. Yang, K. Tang, and X. Yao, "Self-adaptive differential evolution with neighborhood search," in *2008 IEEE congress on evolutionary computation (IEEE World Congress on Computational Intelligence)*, 2008: IEEE, pp. 1110-1116.
- [128] K. P. Yoon and C.-L. Hwang, *Multiple attribute decision making: an introduction*. Sage publications, 1995.

- [129] L. Tang, G. J. Kacprzynski, K. Goebel, and G. Vachtsevanos, "Methodologies for uncertainty management in prognostics," in *2009 IEEE Aerospace conference*, 2009: IEEE, pp. 1-12.
- [130] H. H. Dewey, D. R. DeVries, and S. R. Hyde, "Uncertainty quantification in prognostic health management systems," in *2019 IEEE Aerospace Conference*, 2019: IEEE, pp. 1-13.
- [131] R. Flage, T. Aven, E. Zio, and P. Baraldi, "Concerns, challenges, and directions of development for the issue of representing uncertainty in risk assessment," *Risk analysis*, vol. 34, no. 7, pp. 1196-1207, 2014.
- [132] S. Zheng, Y. Song, T. Leung, and I. Goodfellow, "Improving the robustness of deep neural networks via stability training," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4480-4488.
- [133] M. Bercibar, I. Gandiaga, I. Villarreal, N. Omar, J. Van Mierlo, and P. Van den Bossche, "Critical review of state of health estimation methods of Li-ion batteries for real applications," *Renewable and Sustainable Energy Reviews*, vol. 56, pp. 572-587, 2016.
- [134] G. P. Corey, "Batteries for stationary standby and for stationary cycling applications part 6: alternative electricity storage technologies," in *2003 IEEE Power Engineering Society General Meeting (IEEE Cat. No. 03CH37491)*, 2003, vol. 1: IEEE, pp. 164-169.
- [135] F. Camci and R. B. Chinnam, "Health-state estimation and prognostics in machining processes," *IEEE Transactions on automation science and engineering*, vol. 7, no. 3, pp. 581-597, 2010.
- [136] I. K. Jennions, *Integrated vehicle health management: perspectives on an emerging field*. SAE International Warrendale, PA, 2011.
- [137] H. Qiu, J. Lee, J. Lin, and G. Yu, "Robust performance degradation assessment methods for enhanced rolling element bearing prognostics," *Advanced Engineering Informatics*, vol. 17, no. 3-4, pp. 127-140, 2003.
- [138] B. Saha and K. Goebel, "Battery Data Set, NASA ames prognostics data repository," *NASA Ames Research Center*, 2007.
- [139] B. Saha and K. Goebel, "Uncertainty management for diagnostics and prognostics of batteries using Bayesian techniques," in *2008 IEEE aerospace conference*, 2008: IEEE, pp. 1-8.

- [140] W. He, M. Pecht, D. Flynn, and F. Dinmohammadi, "A physics-based electrochemical model for lithium-ion battery state-of-charge estimation solved by an optimised projection-based method and moving-window filtering," *Energies*, vol. 11, no. 8, p. 2120, 2018.
- [141] B. Saha, K. Goebel, and J. Christophersen, "Comparison of prognostic algorithms for estimating remaining useful life of batteries," *Transactions of the Institute of Measurement and Control*, vol. 31, no. 3-4, pp. 293-308, 2009.
- [142] G. Bai, P. Wang, C. Hu, and M. Pecht, "A generic model-free approach for lithium-ion battery health management," *Applied Energy*, vol. 135, pp. 247-260, 2014.
- [143] E. Meissner and G. Richter, "Battery monitoring and electrical energy management: Precondition for future vehicle electric power systems," *Journal of power sources*, vol. 116, no. 1-2, pp. 79-98, 2003.
- [144] S. Santhanagopalan and R. E. White, "Online estimation of the state of charge of a lithium ion cell," *Journal of power sources*, vol. 161, no. 2, pp. 1346-1355, 2006.
- [145] D. Liu, J. Pang, J. Zhou, and Y. Peng, "Data-driven prognostics for lithium-ion battery based on Gaussian process regression," in *Proceedings of the IEEE 2012 Prognostics and System Health Management Conference (PHM-2012 Beijing)*, 2012: IEEE, pp. 1-5.
- [146] S.-C. Huang, K.-H. Tseng, J.-W. Liang, C.-L. Chang, and M. G. Pecht, "An online SOC and SOH estimation model for lithium-ion batteries," *Energies*, vol. 10, no. 4, p. 512, 2017.
- [147] A. M. Bianco and E. Martínez, "Robust testing in the logistic regression model," *Computational statistics & data analysis*, vol. 53, no. 12, pp. 4095-4105, 2009.
- [148] H. Akaike, "A new look at the statistical model identification," *IEEE transactions on automatic control*, vol. 19, no. 6, pp. 716-723, 1974.
- [149] D. Anderson and K. Burnham, "Model selection and multi-model inference," *Second. NY: Springer-Verlag*, vol. 63, no. 2020, p. 10, 2004.
- [150] L. E. Peterson, "K-nearest neighbor," *Scholarpedia*, vol. 4, no. 2, p. 1883, 2009.
- [151] Z. Zhang, L. Gu, and Y. Zhu, "Intelligent fault diagnosis of rotating machine based on SVMs and EMD method," *The open automation and control systems journal*, vol. 5, no. 1, 2013.

- [152] A. Downey, Y.-H. Lui, C. Hu, S. Laflamme, and S. Hu, "Physics-based prognostics of lithium-ion battery using non-linear least squares with dynamic bounds," *Reliability Engineering & System Safety*, vol. 182, pp. 1-12, 2019.
- [153] D. D. Susilo, A. Widodo, T. Prahasto, and M. Nizam, "State of health estimation of lithium-ion batteries based on combination of gaussian distribution data and least squares support vector machines regression," in *Materials Science Forum*, 2018, vol. 929: Trans Tech Publ, pp. 93-102.
- [154] A. El Mejdoubi, H. Chaoui, H. Gualous, P. Van Den Bossche, N. Omar, and J. Van Mierlo, "Lithium-ion batteries health prognosis considering aging conditions," *IEEE Transactions on Power Electronics*, vol. 34, no. 7, pp. 6834-6844, 2018.
- [155] B. Saha and K. Goebel, "Modeling Li-ion battery capacity depletion in a particle filtering framework," in *Annual Conference of the PHM Society*, 2009, vol. 1, no. 1.
- [156] Q. Miao, L. Xie, H. Cui, W. Liang, and M. Pecht, "Remaining useful life prediction of lithium-ion battery with unscented particle filter technique," *Microelectronics Reliability*, vol. 53, no. 6, pp. 805-810, 2013.
- [157] B. Li, K. Peng, and G. Li, "State-of-charge estimation for lithium-ion battery using the gauss-hermite particle filter technique," *Journal of Renewable and Sustainable Energy*, vol. 10, no. 1, p. 014105, 2018.
- [158] D. Wang and K.-L. Tsui, "Brownian motion with adaptive drift for remaining useful life prediction: Revisited," *Mechanical Systems and Signal Processing*, vol. 99, pp. 691-701, 2018.
- [159] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929-1958, 2014.
- [160] F. Chollet, "Keras documentation," *keras.io*, vol. 33, 2015.
- [161] F. Schorfheide, "Loss function-based evaluation of DSGE models," *Journal of Applied Econometrics*, vol. 15, no. 6, pp. 645-670, 2000.
- [162] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, "Deep learning with limited numerical precision," in *International conference on machine learning*, 2015: PMLR, pp. 1737-1746.

- [163] A. Saxena and K. Goebel, "Turbofan engine degradation simulation data set. NASA Ames Prognostics Data repository, NASA Ames Research Center, Moffett Field," ed, 2008.
- [164] I. Guyon, S. Gunn, M. Nikravesh, and L. A. Zadeh, *Feature extraction: foundations and applications*. Springer, 2008.
- [165] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of machine learning research*, vol. 3, no. Mar, pp. 1157-1182, 2003.
- [166] J. Benesty, J. Chen, Y. Huang, and I. Cohen, "Pearson correlation coefficient," in *Noise reduction in speech processing*: Springer, 2009, pp. 1-4.
- [167] D. Sarwate, "Mean-square correlation of shift-register sequences," in *IEE Proceedings F (Communications, Radar and Signal Processing)*, 1984, vol. 131, no. 2: IET, pp. 101-106.
- [168] Y. Sun, "Iterative RELIEF for feature weighting: algorithms, theories, and applications," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 6, pp. 1035-1051, 2007.
- [169] S. Derksen and H. J. Keselman, "Backward, forward and stepwise automated subset selection algorithms: Frequency of obtaining authentic and noise variables," *British Journal of Mathematical and Statistical Psychology*, vol. 45, no. 2, pp. 265-282, 1992.
- [170] H. Vafaie and I. F. Imam, "Feature selection methods: genetic algorithms vs. greedy-like search," in *Proceedings of the international conference on fuzzy and intelligent control systems*, 1994, vol. 51, p. 28.
- [171] K. Javed, R. Gouriveau, R. Zemouri, and N. Zerhouni, "Features selection procedure for prognostics: An approach based on predictability," *IFAC Proceedings Volumes*, vol. 45, no. 20, pp. 25-30, 2012.
- [172] D. K. Frederick, J. A. DeCastro, and J. S. Litt, "User's guide for the commercial modular aero-propulsion system simulation (C-MAPSS)," 2007.
- [173] F. Khan, O. F. Eker, A. Khan, and W. Orfali, "Adaptive degradation prognostic reasoning by particle filter with a neural network degradation model for turbofan jet engine," *Data*, vol. 3, no. 4, p. 49, 2018.

- [174] 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*, 2008: IEEE, pp. 1-9.
- [175] C. Xiongzi, Y. Jinsong, T. Diyin, and W. Yingxun, "Remaining useful life prognostic estimation for aircraft subsystems or components: A review," in *IEEE 2011 10th International Conference on Electronic Measurement & Instruments*, 2011, vol. 2: IEEE, pp. 94-98.
- [176] A. Zhang *et al.*, "Transfer learning with deep recurrent neural networks for remaining useful life estimation," *Applied Sciences*, vol. 8, no. 12, p. 2416, 2018.
- [177] Z. Kong, Y. Cui, Z. Xia, and H. Lv, "Convolution and long short-term memory hybrid deep neural networks for remaining useful life prognostics," *Applied Sciences*, vol. 9, no. 19, p. 4156, 2019.
- [178] S. Zheng, K. Ristovski, A. Farahat, and C. Gupta, "Long short-term memory network for remaining useful life estimation," in *2017 IEEE international conference on prognostics and health management (ICPHM)*, 2017: IEEE, pp. 88-95.
- [179] Y. Wu, M. Yuan, S. Dong, L. Lin, and Y. Liu, "Remaining useful life estimation of engineered systems using vanilla LSTM neural networks," *Neurocomputing*, vol. 275, pp. 167-179, 2018.
- [180] A. Candel, V. Parmar, E. LeDell, and A. Arora, "Deep learning with H2O," *H2O. ai Inc*, pp. 1-21, 2016.
- [181] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan, "A theory of learning from different domains," *Machine learning*, vol. 79, no. 1, pp. 151-175, 2010.
- [182] A. Van der Drift, "Evolutionary selection, a principle governing growth orientation in vapour-deposited layers," *Philips Res. Rep*, vol. 22, no. 3, p. 267, 1967.
- [183] J. Wang, L.-C. Yu, K. R. Lai, and X. Zhang, "Dimensional sentiment analysis using a regional CNN-LSTM model," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 2016, pp. 225-230.
- [184] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak, "Convolutional, long short-term memory, fully connected deep neural networks," in *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, 2015: IEEE, pp. 4580-4584.

- [185] A. Ullah, J. Ahmad, K. Muhammad, M. Sajjad, and S. W. Baik, "Action recognition in video sequences using deep bi-directional LSTM with CNN features," *IEEE access*, vol. 6, pp. 1155-1166, 2017.
- [186] S. L. Oh, E. Y. Ng, R. San Tan, and U. R. Acharya, "Automated diagnosis of arrhythmia using combination of CNN and LSTM techniques with variable length heart beats," *Computers in biology and medicine*, vol. 102, pp. 278-287, 2018.
- [187] G. Yue, G. Ping, and L. Lanxin, "An end-to-end model based on cnn-lstm for industrial fault diagnosis and prognosis," in *2018 international conference on network infrastructure and digital content (IC-NIDC)*, 2018: IEEE, pp. 274-278.
- [188] T.-Y. Kim and S.-B. Cho, "Predicting residential energy consumption using CNN-LSTM neural networks," *Energy*, vol. 182, pp. 72-81, 2019.
- [189] K. Fukushima and S. Miyake, "Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition," in *Competition and cooperation in neural nets*: Springer, 1982, pp. 267-285.
- [190] C. Olah. "Understanding LSTM Networks." (accessed 2021). Available: <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>

# APPENDIX

## A1. Adaptive Moment Estimation

The Adaptive Moment Estimation (Adam) optimizer keeps an exponentially decaying average of past gradients  $M(t)$ , similar to momentum.  $M(t)$  and  $V(t)$  are values of the first moment, which is the Mean, and the second moment, which is the Un-centered variance of the gradients, respectively. The following is the formulas for the First Moment (Mean), and the Second Moment (Variance):

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \quad (A2)$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \quad (A3)$$

The following is the final formula for the Parameter update:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \quad (A4)$$

The value for  $\beta_1$  is 0.9, and 0.999 for  $\beta_2$  and  $10 \cdot \exp(-8)$  for  $\epsilon$



## A2. Pearson Correlation Matrix

Table A1. The Pearson correlation matrix (for C-MAPSS dataset)

Attributes	T2	T24	T30	T50	P2	P15	P30	Nf	Nc	epr	Ps30	phi	NRF	NRc	BPR	farB	htBleed	Nf_dmd	PCNIR_dmd	W31	W32	RUL
T2	1.0000	0.9441	0.8709	0.8979	0.9864	0.9864	0.9731	0.5725	0.8618	0.8266	0.7060	0.9729	0.1643	0.3528	-0.5426	0.7936	0.8732	0.5720	0.1642	0.9777	0.9777	-0.0023
T24	0.9441	1.0000	0.9822	0.9810	0.9158	0.9441	0.9686	0.8106	0.9785	0.9051	0.8957	0.9688	0.4801	0.6241	-0.7779	0.8050	0.9830	0.8103	0.4800	0.9624	0.9624	-0.0064
T30	0.8709	0.9822	1.0000	0.9896	0.8429	0.8848	0.9290	0.8957	0.9978	0.9290	0.9607	0.9295	0.6209	0.7520	-0.8759	0.8047	0.9987	0.8954	0.6208	0.9171	0.9171	-0.0253
T50	0.8979	0.9810	0.9896	1.0000	0.8841	0.9196	0.9567	0.8439	0.9873	0.9616	0.9368	0.9571	0.5447	0.7156	-0.8467	0.8591	0.9902	0.8436	0.5446	0.9464	0.9464	-0.0378
P2	0.9864	0.9158	0.8429	0.8841	1.0000	0.9963	0.9798	0.5242	0.8329	0.8438	0.6736	0.9795	0.1136	0.3305	-0.5253	0.8241	0.8455	0.5237	0.1135	0.9857	0.9857	-0.0031
P15	0.9864	0.9441	0.8848	0.9196	0.9963	1.0000	0.9933	0.5944	0.8762	0.8782	0.7339	0.9931	0.1981	0.4075	-0.5955	0.8403	0.8871	0.5940	0.1980	0.9964	0.9964	-0.0029
P30	0.9731	0.9686	0.9290	0.9567	0.9798	0.9933	1.0000	0.6791	0.9226	0.9187	0.8054	1.0000	0.3070	0.5081	-0.6842	0.8577	0.9309	0.6787	0.3069	0.9991	0.9991	-0.0003
Nf	0.5725	0.8106	0.8957	0.8439	0.5242	0.5944	0.6791	1.0000	0.9033	0.7829	0.9726	0.6801	0.9028	0.9245	-0.9712	0.5913	0.8937	1.0000	0.9028	0.6559	0.6558	0.0027
Nc	0.8618	0.9785	0.9978	0.9873	0.8329	0.8762	0.9226	0.9033	1.0000	0.9291	0.9643	0.9231	0.6349	0.7711	-0.8855	0.7996	0.9979	0.9030	0.6347	0.9100	0.9100	-0.0134
epr	0.8266	0.9051	0.9290	0.9616	0.8438	0.8782	0.9187	0.7829	0.9291	1.0000	0.8924	0.9192	0.5087	0.7271	-0.8475	0.9141	0.9297	0.7827	0.5086	0.9092	0.9091	0.0014
Ps30	0.7060	0.8957	0.9607	0.9368	0.6736	0.7339	0.8054	0.9726	0.9643	0.8924	1.0000	0.8062	0.8001	0.8931	-0.9654	0.7326	0.9597	0.9724	0.8000	0.7848	0.7847	-0.0426
phi	0.9729	0.9688	0.9295	0.9571	0.9795	0.9931	1.0000	0.6801	0.9231	0.9192	0.8062	1.0000	0.3084	0.5094	-0.6853	0.8579	0.9314	0.6797	0.3083	0.9991	0.9991	-0.0005
NRF	0.1643	0.4801	0.6209	0.5447	0.1136	0.1981	0.3070	0.9028	0.6349	0.5087	0.8001	0.3084	1.0000	0.9277	-0.8842	0.2952	0.6173	0.9031	1.0000	0.2766	0.2765	0.0044
NRc	0.3528	0.6241	0.7520	0.7156	0.3305	0.4075	0.5081	0.9245	0.7711	0.7271	0.8931	0.5094	0.9277	1.0000	-0.9574	0.5425	0.7496	0.9245	0.9275	0.4792	0.4792	-0.0309
BPR	-0.5426	-0.7779	-0.8759	-0.8467	-0.5253	-0.5955	-0.6842	-0.9712	-0.8855	-0.8475	-0.9654	-0.6853	-0.8842	-0.9574	1.0000	-0.6644	-0.8742	-0.9712	-0.8842	-0.6601	-0.6601	-0.0320
farB	0.7936	0.8050	0.8047	0.8591	0.8241	0.8403	0.8577	0.5913	0.7996	0.9141	0.7326	0.8579	0.2952	0.5425	-0.6644	1.0000	0.8060	0.5910	0.2950	0.8554	0.8553	-0.0649
htBleed	0.8732	0.9830	0.9987	0.9902	0.8455	0.8871	0.9309	0.8937	0.9979	0.9297	0.9597	0.9314	0.6173	0.7496	-0.8742	0.8060	1.0000	0.8934	0.6172	0.9191	0.9190	-0.0254
Nf_dmd	0.5720	0.8103	0.8954	0.8436	0.5237	0.5940	0.6787	1.0000	0.9030	0.7827	0.9724	0.6797	0.9031	0.9245	-0.9712	0.5910	0.8934	1.0000	0.9030	0.6554	0.6554	0.0030
PCNIR_dmd	0.1642	0.4800	0.6208	0.5446	0.1135	0.1980	0.3069	0.9028	0.6347	0.5086	0.8000	0.3083	1.0000	0.9275	-0.8842	0.2950	0.6172	0.9030	1.0000	0.2765	0.2764	0.0048
W31	0.9777	0.9624	0.9171	0.9464	0.9857	0.9964	0.9991	0.6559	0.9100	0.9092	0.7848	0.9991	0.2766	0.4792	-0.6601	0.8554	0.9191	0.6554	0.2765	1.0000	0.9999	0.0031
W32	0.9777	0.9624	0.9171	0.9464	0.9857	0.9964	0.9991	0.6558	0.9100	0.9091	0.7847	0.9991	0.2765	0.4792	-0.6601	0.8553	0.9190	0.6554	0.2764	0.9999	1.0000	0.0030

### A3. Principle Components Matrix

Table A2. The Principle Components (PC) matrix (for C-MAPSS dataset)

Component	Standard Deviation	Proportion Of Variance	Cumulative Variance	Eigenvector																				
				T2	T24	T30	T50	P2	P15	P30	Nf	Nc	epr	Ps30	phi	NRF	NRc	BPR	farB	htBleed	Nf_dmd	PCNfR_dmd	W31	W32
PC 1	4.1098	0.8043	0.8043	0.2125	0.2380	0.2422	0.2421	0.2088	0.2187	0.2293	0.2143	0.2421	0.2325	0.2325	0.2294	0.1464	0.1835	-0.2143	0.2047	0.2423	0.2142	0.1463	0.2265	0.2265
PC 2	1.8911	0.1703	0.9746	0.2432	0.0759	-0.0129	0.0338	0.2694	0.2294	0.1739	-0.2437	-0.0234	0.0343	-0.1505	0.1731	-0.4207	-0.3244	0.2381	0.1294	-0.0105	-0.2440	-0.4208	0.1900	0.1900
PC 3	0.6210	0.0184	0.9930	-0.2203	-0.2194	-0.1148	0.0276	-0.0668	-0.0637	-0.0351	-0.1583	-0.1055	0.4085	-0.0113	-0.0345	-0.0745	0.2675	-0.1636	0.7255	-0.1142	-0.1582	-0.0747	-0.0391	-0.0391
PC 4	0.2765	0.0036	0.9966	-0.1973	-0.1956	-0.1354	-0.0125	0.1133	0.1197	0.1552	-0.1492	-0.0729	0.3738	-0.0685	0.1558	-0.0679	0.3512	-0.2851	-0.6037	-0.1303	-0.1498	-0.0681	0.1400	0.1399
PC 5	0.1898	0.0017	0.9983	0.2407	0.1279	0.1338	0.2365	-0.1105	-0.1263	-0.1444	-0.1089	0.2504	0.0171	0.1765	-0.1436	-0.2556	0.5771	0.3024	-0.1262	0.1371	-0.1126	-0.2598	-0.1894	-0.1890
PC 6	0.1429	0.0010	0.9993	-0.1080	-0.0508	-0.1018	-0.2787	0.1666	0.1661	0.1402	-0.0006	0.0850	-0.5969	-0.2325	0.1398	0.0629	0.5242	0.1348	0.1816	-0.0949	-0.0067	0.0564	0.1473	0.1473
PC 7	0.0912	0.0004	0.9997	0.3962	0.2324	-0.0820	-0.3642	-0.0725	-0.1049	-0.1372	0.1574	0.2384	0.2671	-0.5964	-0.1412	-0.0471	0.0686	-0.2038	0.0105	-0.0991	0.1585	-0.0485	0.0028	0.0024
PC 8	0.0467	0.0001	0.9998	0.2674	0.1477	-0.3715	0.0978	-0.0681	-0.0160	0.1226	-0.0583	0.0358	-0.3458	0.2701	0.1267	-0.1464	-0.0277	-0.6182	0.0391	-0.1879	-0.0548	-0.1362	-0.1777	-0.1775
PC 9	0.0363	0.0001	0.9999	0.0132	-0.0133	0.7894	-0.0969	-0.0175	-0.0065	0.0249	-0.0527	-0.0336	-0.0869	0.0295	0.0252	-0.0552	0.0009	-0.1860	0.0047	-0.5529	-0.0523	-0.0533	-0.0472	-0.0457
PC 10	0.0328	0.0001	0.9999	0.0008	-0.0389	0.1788	-0.7296	-0.0113	-0.0056	0.0201	-0.0967	-0.0508	0.0283	0.3284	0.0208	-0.1045	0.0016	-0.1610	0.0012	0.5025	-0.0962	-0.1026	-0.0400	-0.0441
PC 11	0.0311	0.0000	1.0000	0.1051	0.1076	-0.2848	-0.3377	0.0413	0.0245	-0.0107	0.1014	0.1245	0.2238	0.5163	-0.0125	0.0465	0.0011	0.3575	-0.0016	-0.5287	0.1028	0.0463	0.0864	0.0873
PC 12	0.0138	0.0000	1.0000	-0.2578	-0.2810	-0.0020	0.0108	0.0167	-0.0099	-0.0864	-0.1056	0.8618	-0.0568	0.0304	-0.0848	-0.0164	-0.2225	-0.1099	-0.0024	-0.0025	-0.1110	-0.0253	0.0644	0.0496
PC 13	0.0118	0.0000	1.0000	0.0477	-0.0185	0.0025	0.0839	-0.0086	-0.1363	-0.4283	0.0260	-0.1718	-0.1710	0.1638	-0.4312	-0.1047	0.0539	-0.2109	-0.0039	0.0099	0.0176	-0.1249	0.4655	0.4744
PC 14	0.0101	0.0000	1.0000	-0.0021	-0.0067	-0.0005	-0.0023	0.0098	0.0070	-0.0020	0.0012	0.0090	0.0009	0.0001	-0.0028	0.0001	-0.0025	-0.0016	0.0000	0.0020	0.0007	-0.0008	-0.7132	0.7008
PC 15	0.0071	0.0000	1.0000	0.5000	-0.7791	0.0160	0.0235	-0.1718	-0.1149	0.1418	0.1483	-0.0349	-0.0279	0.0250	0.1270	-0.0272	0.0099	0.0680	0.0015	0.0178	0.1666	-0.0003	0.0504	0.0501
PC 16	0.0058	0.0000	1.0000	0.0583	-0.2380	-0.0142	0.0054	0.6356	0.4221	-0.2979	0.1703	-0.0600	0.0165	0.0288	-0.2753	-0.0146	0.0194	-0.0802	-0.0018	-0.0146	0.1399	-0.0673	-0.2421	-0.2634
PC 17	0.0025	0.0000	1.0000	0.0239	-0.0042	0.0006	-0.0014	-0.0321	0.0275	-0.7104	-0.0225	0.0000	0.0017	-0.0024	0.7009	0.0127	-0.0002	0.0038	0.0001	-0.0002	-0.0181	0.0232	0.0068	0.0081
PC 18	0.0011	0.0000	1.0000	-0.0506	0.0059	0.0006	0.0011	0.0861	-0.0696	-0.0029	-0.4001	0.0033	-0.0050	0.0004	-0.0007	-0.5777	0.0057	-0.0026	0.0000	0.0008	0.4897	0.5014	-0.0056	-0.0055
PC 19	0.0008	0.0000	1.0000	-0.2540	0.0089	0.0007	0.0033	0.4119	-0.6623	0.1074	0.3333	-0.0010	-0.0155	0.0051	0.1922	-0.1498	-0.0006	-0.0017	-0.0002	0.0011	0.2199	-0.3095	-0.0178	-0.0180
PC 20	0.0004	0.0000	1.0000	0.3279	-0.0012	0.0002	0.0002	0.4451	-0.4099	-0.0213	-0.4508	0.0004	0.0104	0.0000	-0.0349	0.3864	0.0000	-0.0001	0.0002	0.0002	-0.3040	0.2821	0.0029	0.0029
PC 21	0.0002	0.0000	1.0000	0.0385	-0.0005	0.0000	0.0000	0.0661	-0.0567	-0.0051	0.4972	0.0001	0.0022	-0.0001	-0.0091	-0.4057	0.0000	0.0000	0.0000	0.0000	-0.5855	0.4860	0.0009	0.0010

#### A4. Evolutionary DNN Model Description

```

Model Metrics Type: Regression
Description: Metrics reported on temporary training frame with 10098 samples
model id: h2o-model-deep_learning-558436
frame id: h2o-frame-deep_learning-202436.temporary.sample.19.37%
MSE: 2403.0352
RMSE: 49.020763
R^2: 0.34962007
mean residual deviance: 1546.6044
mean absolute error: 35.47905
root mean squared log error: 0.6627855
Variable Importances:
Variable Relative Importance Scaled Importance Percentage
Ps30 1.000000 1.000000 0.152379
BFR 0.986309 0.986309 0.150292
T2 0.850706 0.850706 0.129629
T50 0.677203 0.677203 0.103191
farB 0.556187 0.556187 0.084751
htBleed 0.498097 0.498097 0.075899
Nf_dmd 0.488625 0.488625 0.074456
P15 0.435972 0.435972 0.066433
P2 0.435133 0.435133 0.066305
T30 0.366792 0.366792 0.055891
NRC 0.267579 0.267579 0.040773
Status of Neuron Layers (predicting RUL, regression, huber distribution, Huber loss, 204 weights/biases, 7.5 KB, 55,402,031 training samples, mini-batch size 1):
Layer Units Type Dropout L1 L2 Mean Rate Rate RMS Momentum Mean Weight Weight RMS Mean Bias Bias RMS
1 11 Input 0.00 %
2 7 RectifierDropout 0.00 % 0.000010 0.000000 0.000295 0.000559 0.000000 -0.598193 3.983957 -3.185914 2.046618
3 7 RectifierDropout 0.00 % 0.000010 0.000000 0.000813 0.000763 0.000000 -1.333539 1.914081 1.391002 6.177969
4 7 RectifierDropout 40.00 % 0.000010 0.000000 0.006495 0.012685 0.000000 -1.583420 4.002537 12.428126 8.831219
5 1 Linear 0.000010 0.000000 0.001167 0.001137 0.000000 0.341112 1.382185 54.060656 0.000000
Scoring History:
Timestamp Duration Training Speed Epochs Iterations Samples Training RMSE Training Deviance Training MAE Training r2
2020-07-02 12:41:46 0.000 sec 0.00000 0 0.000000 NaN NaN NaN NaN
2020-07-02 12:41:47 0.139 sec 917100 obs/sec 1 1.93601 1 99964.000000 57.92967 2169.20537 41.41472 0.09174
2020-07-02 12:41:52 5.170 sec 1074425 obs/sec 106 51863 55 5499983.000000 50.66080 1665.27693 37.62549 0.30537
2020-07-02 12:41:57 10.170 sec 1039654 obs/sec 203 36424 105 10500509.000000 54.94959 1940.48408 40.22174 0.18279
2020-07-02 12:42:02 15.238 sec 1056334 obs/sec 309 87903 160 16000294.000000 51.23576 1685.89755 36.92628 0.28952
2020-07-02 12:42:07 20.319 sec 1073730 obs/sec 420 26758 217 21700096.000000 49.02076 1546.60437 35.47905 0.34962
2020-07-02 12:42:12 25.403 sec 1080248 obs/sec 528 74284 273 27301108.000000 58.10724 2153.83839 41.62051 0.08617
2020-07-02 12:42:17 30.464 sec 1072149 obs/sec 629 45201 325 32501125.000000 64.10426 2656.18999 46.66454 -0.11219
2020-07-02 12:42:22 35.549 sec 1082538 obs/sec 741 78412 383 38301281.000000 51.05237 1674.29623 36.81673 0.29459
2020-07-02 12:42:27 40.616 sec 1088379 obs/sec 852 19191 440 44002077.000000 49.25254 1558.64916 35.89076 0.34346
2020-07-02 12:42:32 45.656 sec 1091354 obs/sec 960 64756 496 49602076.000000 55.91947 2013.51190 40.12205 0.15368
2020-07-02 12:42:37 50.737 sec 1096787 obs/sec 1072 97577 554 55402031.000000 49.27854 1633.52927 38.24937 0.34276
2020-07-02 12:42:37 50.757 sec 1096744 obs/sec 1072 97577 554 55402031.000000 49.02076 1546.60437 35.47905 0.34962
H2O version: 3.30.0.1

```

Figure A1. The proposed evolutionary DNN model description (for C-MAPSS dataset)