DEGRADATION MODELING AND REMAINING USEFUL LIFE ESTIMATION: FROM STATISTICAL SIGNAL PROCESSING TO DEEP LEARNING MODELS

Ali Al-Dulaimi

A PhD Thesis
in
The Department
of
Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements
For the Degree of Doctor of Philosophy
Concordia University
Montréal, Québec, Canada

DECEMBER 2020

Concordia University

School of Graduate Studies

This is to certify that the thesis prepared

By: **Mr. Ali Al-Dulaimi**

Entitled: **Degradation Modeling and Remaining Useful Life Estimation: From Statistical Signal Processing to Deep Learning Models**

and submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy (Electrical Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

_____ Chair

Dr. Youmin Zhang

_____ External Examiner

Dr. Zhaojun Li

_____ External to Program

Dr. Mingyuan Chen

_____ Examiner

Dr. M Omair Ahmed

_____ Examiner

Dr. Yousef R. Shayan

_____ Thesis Supervisor

Dr. Amir Asif

_____ Thesis Co-supervisor

Dr. Arash Mohammadi

Approved by _____

Dr. Wei-Ping Zhu, Graduate Program Director

December 18, 2020 _____

Dr. Mourad Debbabi, Dean

Gina Cody School of of Engineering and Computer Science

Abstract

Degradation Modeling and Remaining Useful Life Estimation: From Statistical Signal Processing to Deep Learning Models

Ali Al-Dulaimi, Ph.D.
Concordia University, 2020

Aging critical infrastructures and valuable machineries together with recent catastrophic incidents such as the collapse of Morandi bridge, or the Gulf of Mexico oil spill disaster, call for an urgent quest to design advanced and innovative prognostic solutions, and efficiently incorporate multi-sensor streaming data sources for industrial development. Prognostic health management (PHM) is among the most critical disciplines that employs the advancement of the great interdependency between signal processing and machine learning techniques to form a key enabling technology to cope with maintenance development tasks of complex industrial and safety-critical systems. Recent advancements in predictive analytics have empowered the PHM paradigm to move from the traditional condition-based monitoring solutions and preventive maintenance programs to predictive maintenance to provide an early warning of failure, in several domains ranging from manufacturing and industrial systems to transportation and aerospace. The focus of the PHM is centered on two core dimensions; the first is taking into account the behavior and the evolution over time of a fault once it occurs, while the second one aims at estimating/predicting the remaining useful life (RUL) during which a device can perform its intended function. The first dimension is the degradation that is usually determined by a degradation model derived from measurements of critical parameters of relevance to the system. Developing an accurate model for the degradation process is a primary objective in prognosis and health management. Extensive research has been conducted to develop new theories and methodologies for degradation modeling and to accurately capture the degradation dynamics of a system. However, a unified degradation framework has yet not been developed due to: (i) structural uncertainties in the state dynamics of the system and (ii) the complex nature of the degradation process that is often non-linear and difficult to model statistically. Thus even for a single system, there is no consensus on the best degradation model. In this regard, this thesis tries to

bridge this gap by proposing a general model that able to model the true degradation path without having any prior knowledge of the true degradation model of the system. Modeling and analysis of degradation behavior lead us to RUL estimation, which is the second dimension of the PHM and the second part of the thesis. The RUL is the main pillar of preventive maintenance, which is the time a machine is expected to work before requiring repair or replacement. Effective and accurate RUL estimation can avoid catastrophic failures, maximize operational availability, and consequently reduce maintenance costs. The RUL estimation is, therefore, of paramount importance and has gained significant attention for its importance to improve systems health management in complex fields including automotive, nuclear, chemical, and aerospace industries to name but a few. A vast number of researches related to different approaches to the concept of remaining useful life have been proposed, and they can be divided into three broad categories: (i) Physics-based; (ii) Data-driven, and; (iii) Hybrid approaches (multiple-model). Each category has its own limitations and issues, such as, hardly adapt to different prognostic applications, in the first one, and accuracy degradation issues, in the second one, because of the deviation of the learned models from the real behavior of the system. In addition to hardly sustain good generalization. Our thesis belongs to the third category, as it is the most promising category, in particular, the new hybrid models, on basis of two different architectures of deep neural networks, which have great potentials to tackle complex prognostic issues associated with systems with complex and unknown degradation processes.

Acknowledgments

Firstly, I would like to express my deepest appreciation and heartfelt gratitude to my supervisor Dr. Amir Asif for his support, guidance and encouragement. His overall insights in this field have inspired me to reach my goals.
Secondly, I would like to convey my special appreciation and thanks to my co-supervisor Dr. Arash Mohammadi for his supervision, friendly guidance, and expert advice which were of invaluable significance to my research. I've been always overwhelmed with his positive energy.
To my best friend Mohammed, and my dear friend Hussam, I wouldn't have made it this far without having both of you beside me.
To my dear friends Noha and Maisa'a, I can't thank you enough for your kind support.
I will forever be grateful to my friends – Suhail, Somayeh, Ibrahim, Soroosh, Yang, Mojtaba, Albert, and others, who have contributed in their own unique ways during my study journey.
Finally, to my family who live in my heart, you are my everything.

Contents

List of Figures **x**

List of Tables **xiii**

Abbreviations **xv**

1 Thesis Overview **1**

1.1 Introduction 1

1.1.1 Generalized Degradation Modeling 4

1.1.2 Accurate RUL Estimation 5

1.2 Thesis Contributions 6

1.3 Organization of the Report 8

1.4 Publications 9

2 Preliminaries and Literature Review **11**

2.1 The Degradation Modeling 11

2.1.1 Stochastic Process Models 12

2.1.1.1 Wiener Process (Brownian Motion) 13

2.1.1.2 Gamma Process 14

2.1.1.3 Inverse Gaussian Process (IG) 16

2.1.2 General Path Models 17

2.1.3 Existing Work on Degradation Modeling 18

2.1.4 The Basic Definitions and the Key Concepts for the Proposed Degradation Modeling 20

2.1.4.1 The Particle Filter (PF) 20

2.1.4.2 The Multiple Model (MM) 26

2.1.4.3 The Multiple Model Adaptive Estimation (MMAE) . 27

2.1.4.4 The Interactive Multiple Model (IMM) 28
2.2 Remaining Useful Life Estimation 31
2.2.1 Classification of RUL prediction approaches 33
2.2.1.1 Physics-based Category 33
2.2.1.2 Data-driven Category 34
2.2.1.3 Artificial Neural Networks (ANNs) 35
2.2.1.4 Multilayer Perceptron Networks (MLP) 37
2.2.1.5 Convolutional Neural Networks (CNN) 41
2.2.2 One Dimensional Convolution 44
2.2.2.1 Recurrent Neural Networks (RNN) 46
2.2.2.2 The Long Short-Term Memory (LSTM) 49
2.2.2.3 The Bidirectional Long Short-Term Memory (BLSTM) 52
2.2.2.4 The Gated Recurrent Unit (GRU) 55
2.2.2.5 The Bidirectional Gated Recurrent Unit (BGRU) . . 58
2.2.2.6 Hybrid (Multiple-Model) Approaches 60

3 General Degradation Modeling Frameworks 64
3.1 Multiple-Model Degradation Path (MMDP) Estimation Framework . 65
3.1.1 Wiener Process (Brownian Motion) 67
3.1.2 Gamma Process 68
3.1.3 The main steps of the proposed (MMDP) 68
3.2 Interactive Multiple Model Particle Filters (IMMPF) Framework . . . 70
3.2.1 Inverse-Gaussian Process 71
3.2.2 The IMMPF Algorithm 72
3.3 Simulations 74
3.3.1 Evaluation of the MMDP Framework 75
3.3.1.1 The Three Tank DTS200 Model 75
3.3.2 Evaluation of the IMMPF Framework 78
3.4 The Summary 80

4 Hybrid Parallel Deep Neural Network Models for RUL Estimation 82
4.1 NASA C-MAPSS Data Set 83
4.1.1 Operating Conditions and Fault Modes 84
4.1.2 Data Normalization 84
4.1.3 Performance Evaluation 86

4.1.4 RUL Target Function 86
4.2 The General Settings of the Proposed Frameworks 88
4.3 Frameworks of The Proposed Networks 89
4.3.1 The First Parallel Path 90
4.3.2 The CNN Path (The Second Parallel Path) 91
4.3.3 The Fusion Path (The Third Path) 93
4.3.4 The Overfitting Issue 93
4.3.5 The Dropout Technique 94
4.3.6 The Early Stopping Technique 94
4.3.7 Training Process 95
4.4 Simulations 96
4.4.1 Results 96
4.4.1.1 The RUL Estimation Results 96
4.4.1.2 The Results with Different Time Window Size 97
4.4.1.3 Comparison with Existing Methods 100
4.4.1.4 Monte Carlo Simulations with Additive Noise 102
4.5 The Summary 103

5 The Noisy and Hybrid Deep Neural networks for Remaining Useful Life Estimation 105

5.1 The Noise Injection Strategy 106
5.2 Frameworks of The Proposed Networks 108
5.2.1 The First Parallel Path 109
5.2.2 The Noisy CNN Path (The Second Parallel Path) 109
5.2.3 The Fusion Path (The Third Path) 110
5.3 Noisy Training 110
5.3.1 Grid Search Technique 111
5.3.2 Hyperparameters Optimization 113
5.4 Simulations 113
5.4.1 Results 114
5.4.1.1 The RUL Estimation Results 114
5.4.1.2 The Effects of Different Time Window Size 116
5.4.1.3 Effects of Operating Conditions and Fault Modes on RUL Estimation Results 116
5.4.1.4 Effects of Training based on Different Noise Levels . 117

5.4.1.5 Comparison with the Proposed Models of Chapter 4 119
5.4.1.6 Comparison with State-of-The-Art Methodologies . . 120
5.4.1.7 Performance Evaluation with Additive Noise 122
5.5 The Summary 123

6 Multipath Parallel Hybrid Deep Neural Networks 124
6.1 Frameworks of The Proposed Networks 125
6.1.1 The First Parallel Path 126
6.1.2 The Second Parallel Path 127
6.1.3 The Third Parallel Path 127
6.1.4 The Fusion Path (The Fourth Path) 128
6.2 Noisy Training 129
6.2.1 Batch Normalization 129
6.2.2 Global Average Pooling 130
6.3 Simulations 130
6.3.1 Results 131
6.3.1.1 The RUL Estimation Results 131
6.3.1.2 The Effects of Different Time Window Size 133
6.3.1.3 The Effect of Training based on Different Noise styles 133
6.3.1.4 The Effects of Multiple Parallel Paths 134
6.3.1.5 Performance Evaluation with Additive Noise 135
6.3.1.6 Comparison with the Proposed Models of Chapter 5 137
6.3.1.7 Comparison with Existing Methods 139
6.4 The Summary 139

7 Contributions and Future Research Directions 142
7.1 Summary of Contributions 143
7.1.1 Summary of Achieved Results 146
7.2 Future Research Directions 149

List of Figures

1.1 The Percentages of Maintenance Types [4]. 2
1.2 (a) An illustrative example of degradation-threshold failure. (b) The concept of Remaining Useful Lifetime (RUL). 4
2.1 The SIS diagram. 24
2.2 The PF diagram. 25
2.3 The Structure of MM estimator. 26
2.4 The Block Diagram of the MMAE Approach. 28
2.5 The Block Diagram of the IMM Approach. 31
2.6 RUL prediction more informative than Health state prediction. 33
2.7 The structure of an MLP with 3 layers. 39
2.8 The CNN Architecture. 41
2.9 The 5×5 Input image matrix, and the 3×3 Filter matrix 42
2.10 The 3×3 Convolution process. 42
2.11 Illustration of the 1D convolution operation utilized in the CNN. . . . 44
2.12 The Recurrent Neural Networks Concept. 46
2.13 The relationship structures of input and output data. 47
2.14 The Bidirectional Recurrent Neural Networks. 48
2.15 Block diagram of LSTM. 49
2.16 Block diagram of the BLSTM. 53
2.17 Block diagram of the GRU. 55
2.18 Block diagram of the BGRU. 58
2.19 The classification of hybrid category. 62
3.1 The complete structure of the three-tank DTS200 Model. 75
3.2 (a) Adaptive weights associated with the Brownian and Gamma models. (b) The RMSE results obtained based on the proposed MMDP framework, and stand-alone Gamma and Browning matched filters. 77

3.3 Descriptioin of the figure goes here.(a) The Probability for model 1 (Brownian). (b) The Probability for model 3 (Gamma). (c) The Root-Mean-Square Error for stand-alone filters and the proposed IMMPF. 80
4.1 A simplified diagram of the simulation engine in C-MAPSS [311]. . . 83
4.2 The illustration of the data from the 14 selected sensors of FD004 before and after normalization. (a) Raw sensor data (before). (b) Normalized sensor data (after). 85
4.3 (a) The comparison of scoring function and RMSE function. (b) Illustration of piece-wise linear RUL target function. 87
4.4 (a) Training sample of 15 time window (14 selected features). (b) Flowchart of the proposed models. 89
4.5 The proposed hybrid deep neural network (HDNN) framework. . . . 91
4.6 The proposed hybrid deep neural network (BiLSTM) framework. . . 92
4.7 The proposed hybrid deep neural network (GRU) framework. 92
4.8 The proposed hybrid deep neural network (BiGRU) framework. . . . 92
4.9 The detailed architecture of the CNN path. 93
4.10 The Early Stopping Technique. 95
4.11 The prediction for the last recorded data point of different testing engine units in FD001-FD004. (a) Prediction for the 100 testing engine units in FD001 for the BiGRU model. (b) Prediction for the 256 testing engine units in FD002 for BiLSTM model (c) Prediction for the 100 testing engine units in FD003 for the HDNN model. (d) Prediction for the 248 testing engine units in FD004 for the GRU model. 98
4.12 Different examples of lifetime RUL prediction for a sample engine unit of each dataset. (a) The testing engine Unit 77 in FD001 for the HDNN model. (b) The testing engine Unit 162 in FD002 for the BiGRU model. (c) The testing engine Unit 92 in FD003 for the GRU model. (d) The testing engine Unit 202 in FD004 for the BiLSTM model. 100
5.1 The NLSTM framework. 108
5.2 The NBLSTM framework. 108
5.3 The NGRU framework. 108
5.4 The NPBGRU framework. 109

5.5 The prediction for the last recorded data point of different testing engine units in FD001-FD004. (a) Prediction for the 100 testing engine units in FD001 for the NLSTM model. (b) Prediction for the 256 testing engine units in FD002 for NBLSTM model (c) Prediction for the 100 testing engine units in FD003 for NGRU model. (d) Prediction for the 248 testing engine units in FD004 for the NPBGRU model. 114
5.6 Different examples of lifetime RUL prediction for a sample engine unit of each dataset. (a) The testing engine Unit 35 in FD001 for the NLSTM model. (b) The testing engine Unit 164 in FD002 for the NBLSTM model. (c) The testing engine Unit 92 in FD003 for the NGRU model. (d) The testing engine Unit 151 in FD004 for the NPBGRU model. 115
5.7 Distribution histogram of prediction error for NPBGRU. (a)FD001 Dataset. (b) FD002 Dataset. (c) FD003 Dataset. (d) FD004 Dataset. 117
5.8 The Effects of Training Different Noise Levels. (a) On the Results of NBLSTM Model. (b) On the Results of NPBGRU Model. 119
6.1 The NMPM framework. 125
6.2 The MPHD framework. 125
6.3 The NPHM framework. 125
6.4 The TDHA framework. 125
6.5 The Global Average Pooling Technique. 130
6.6 The prediction for the last recorded data point of different testing engine units in FD001-FD004. (a) Prediction for the 100 testing engine units in FD001 for TDHA model. (b) Prediction for the 256 testing engine units in FD002 for MPHD model (c) Prediction for the 100 testing engine units in FD003 for the NPHM model. (d) Prediction for the 248 testing engine units in FD004 for NMPM model. 131
6.7 Different examples of lifetime RUL prediction for a sample engine unit of each dataset. (a) The testing engine Unit 32 in FD001 for TDHA model. (b) The testing engine Unit 162 in FD002 for the MPHD model. (c) The testing engine Unit 78 in FD003 for the NPHM model. (d) The testing engine Unit 151 in FD004 for NMPM model. 132
6.8 The Improvement percentages for adopting noisy training. 134

List of Tables

3.1 Specifications of the experimental setup based on the three-vessel water tank system, DTS200 [308]. 79
4.1 Data Set Details (Simulated From C-MAPSS) [312]. 84
4.2 The results of 30 time window size. 97
4.3 The results of 15 time window size. 98
4.4 Comparison of results obtained from the proposed models with those reported by 7 state-of-the-art methodologies. 99
4.5 Performance comparison with 4 methods that do not use the piece-wise linear degradation model with the proposed frameworks 100
4.6 Monte Carlo simulation results of (HDNN) model: (a) The results under SNR =30 dB. (b) The results under SNR =25dB. (c) The results under SNR =20 dB. 103
4.7 Monte Carlo simulation results of (BiGRU) model: (a) The results under SNR =30 dB. (b) The results under SNR =25dB. (c) The results under SNR =20 dB. 103
5.1 Details of the incorporated dataset from C-MAPSS [312]. 113
5.2 Performance results obtained based on window size of length 30. 116
5.3 Performance results obtained based on the window size of length 15. . . . 116
5.4 The Effects of Training based on Different Noise Levels. 118
5.5 The Effects of Increasing the Noise Level From (0.01 to 0.1). 118
5.6 Comparison with the Proposed Models of Chapter 4: (a) The results of BiLSTM and the NBLSTM. (b) The results of BiGRU and the NPBGRU. 120
5.7 Performance comparison of the proposed models with 7 state-of-the-art methodologies. 121
5.8 Comparing performance obtained from four approaches that exclude the need for utilization of the piece-wise linear degradation model with the proposed NBLSTM model based on the C-MAPSS datasets. 122

5.9 Noisy testing results of (NPBGRU). 122
5.10 Monte Carlo simulation results of (NBLSTM). 123
5.11 Noisy testing results of (NPBGRU). 123
5.12 Monte Carlo simulation results of (NBLSTM). 123
6.1 Details of the incorporated dataset from C-MAPSS [312]. 130
6.2 Performance results obtained based on the window size of length 30. . . . 133
6.3 Performance results obtained based on the window size of length 15. . . . 133
6.4 The Effect of Noisy Training 134
6.5 The Effects of Multiple Parallel Paths. 135
6.6 Noisy testing results of (NPHM). 136
6.7 Noisy testing results of (TDHA). 136
6.8 Monte Carlo simulation results of (NPHM). 136
6.9 Monte Carlo simulation results of (TDHA). 136
6.10 Comparison with the Proposed Models of Chapter 5: (a) The results of MPHD, TDHA, NPBGRU, and NBLSTM at TW=30. (b) The results of MPHD, TDHA, NPBGRU, and NBLSTM at TW=15. 137
6.11 Comparison with the Proposed Models of Chapter 5: (a) The results of MPHD, TDHA, NPBGRU, and NBLSTM at SNR=30. (b) The results of MPHD, TDHA, NPBGRU, and NBLSTM at SNR=20. 138
6.12 Performance comparison of the proposed models with 7 state-of-the-art methodologies. 140
6.13 Comparing performance obtained from four approaches that exclude the need for utilization of the piece-wise linear degradation model with the proposed three parallel paths models based on the C-MAPSS datasets. 141
7.1 All the proposed models in Chapters 4, 5, and 6. 147

Abbreviations

Abbreviation	Description
RUL	Remaining Useful Life
PHM	Prognostic health management
CBM	Condition Based Maintenance
CPES	Cyber-physical energy systems
HI	Health Index
PDF	Probability Density Function
DP	Degradation Path
IG	Inverse Gaussian
KF	Kalman Filter
EKF	Extended Kalman Filter
UKF	Unscented Kalman Filter
SIS	Sequential Importance Sampling
PF	Particle Filter
MC	Monte Carlo
MM	Miltiple Models
IMM	Interactive Multiple Model
MMAE	Multiple Model Adaptive Estimation
MMDP	Multiple Model Degradation Path
IMMPF	Interactive Multiple Model Particle Filter
C-MAPSS	Commercial Modular Aero-Propulsion System Simulation
HPC	High Pressure Compressor
SNR	Signal to Noise Ratio
ML	Machine Learning
NN	Neural Network
ANN	Articial Neural Network

DNN	Deep Neural Network
FC	Fully connected
ReLU	Rectied Linear Unit
MLP	Multilayer Perceptron Networks
Adam	adaptive moment estimation
RMSE	Root Mean Square Error
MSE	Mean Square Error
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short Term Memory
NLSTM	Noisy Long Short Term Memory
BLSTM	Bidirectional Long Short Term Memory
NBLSTM	NoisyBidirectional Long Short Term Memory
GRU	Gated Recurrent Unit
NGRU	Noisy Model based on Gated Recurrent Unit
BGRU	Bidirectional Gated Recurrent Unit
NPBGRU	Noisy Parallel Model based on Bidirectional Gated Recurrent Unit
DBM	Deep Boltzmann Machine
DBN	Deep Belief Networks
RBM	Restricted Boltzmann Machine
HDNN	Hybrid Deep Neural Network
NBLSTM	Noisy Bidirectional Long Short Term Memory
NMPM	Noisy Multipath Parallel Hybrid Model
NPHM	Noisy Parallel Hybrid Model
MPHD	Multipath Parallel Hybrid Deep Neural network Model
TDHA	Ternary Deep Learning-based Hybrid Architecture

Chapter 1

Thesis Overview

1.1 Introduction

In the era of smart manufacturing and the Industrial Internet of Things (IIoT), there is an increased global demand and strategic urgency for the development of advance, smart, and reliable prognostic health management technologies to cope with the maintenance needs of industrial complex systems and safety-critical infrastructures. Prognostic Health Management (PHM) is an enabling discipline in several industrial and manufacturing applications where monitoring the reliability of the underlying complex engineering infrastructure is of paramount importance. Within the PHM context, one key objective is to improve maintenance effectiveness, safe operability, and provide enhanced performance. Maintenance refers to a combination of different technical, managerial, and administrative activities that meant to ensure the best functionality status of a system. It has been reported in the literature [1] that 15-40% of manufacturing expenses across different industrial sectors are attributable to maintenance management decisions. This is of particular importance in the capital and energy-intensive industries such as Cyber-Physical Energy Systems (CPES). Such systems are, typically, designed by the integration of control, communication, and computation technologies for reliable and real-time monitoring and management of the underlying infrastructure.

Recent advancements in communication and sensor technologies have paved the way for the deployment of a large number of sensor nodes for condition monitoring in CPESs, resulting in exceptional growth in practical implementations and opportunistic applications of such systems. The rapid growth of CPESs has introduced a surge

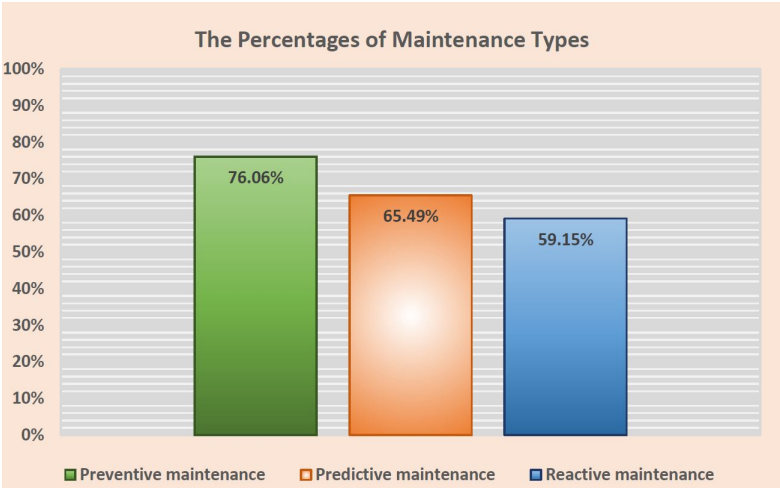


Figure 1.1: The Percentages of Maintenance Types [4].

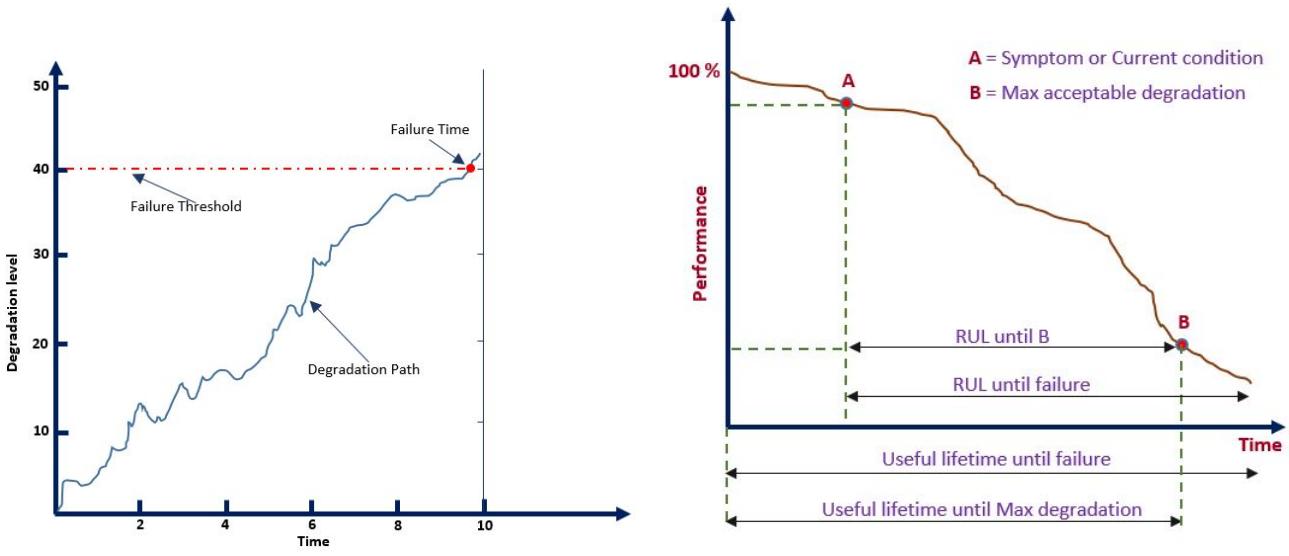
of interest in issues related to low-cost health monitoring systems and integrated predictive maintenance frameworks of such systems. Most of the maintenance decisions are taken either based on the traditional "run-to-failure" strategy or based on the degradation process trend during a specific period of time. However, these types of maintenance can not efficiently prevent tragic accidents that may cause partial or full system failure. There are many cases where unplanned downtime has led to incidents with significant losses. According to analyst firm Aberdeen Research [2] over the past three years, 82% of companies have encountered unplanned downtime, and the average cost across all businesses was $260,000 per hour. Furthermore, the cost is far greater in some sectors such as the auto industry, where the downtime cost can go up to $50,000 per minute, which equals to $3 million per hour [3]. On the other hand, recently, several catastrophic events occurred mainly due to a lack of proper PHM considerations resulting in considerable loss of human lives as well as serious/major environmental damages. Examples of such catastrophic events are the 2010 Macondo blowout and explosion in the Gulf of Mexico; the 2009 Caribbean petroleum refining tank explosion and fire, and; Collapse of the Morandi bridge in the Italian city of Genoa in 2018. Such accidents and many others could have been prevented if proper maintenance procedures were implemented.

Fig. 1.1 shows the commonly used maintenance strategies [4]. It is interesting to note that about 59% of the companies utilize the reactive maintenance approach, i.e., some type of corrective action will be carried out once the failure has occurred. This is the easiest strategy for maintaining equipment as it follows the "run-to-failure"

concept, i.e., the failed item/part is fixed or replaced. The reactive maintenance approach has the advantage of minimizing the involved costs, which is achieved by using equipment without maintenance interruption until failure. However, a key drawback of a reactive maintenance strategy is unpredictable failures resulting in increased equipment downtime. On the other hand, 76% of the companies follow a preventive maintenance approach, which is the most popular type of maintenance, where maintenance usually occurs over frequent periods of time, regardless of the actual equipment conditions. This strategy, however, leads to situations where companies conduct unnecessary maintenance actions resulting in higher maintenance costs. A key advantage of the preventive maintenance strategy is achieving improved overall safety and stability, which is significant especially for a company that runs heavy machinery.

Although the preventive maintenance approach is currently considered the most popular approach, there is a paradigm shift by the leading companies to adopt smart maintenance strategies based on accurate prediction of future failure occurrences. Consequently, there has been a recent surge of interest in "Predictive Maintenance" that anticipates maintenance needs and only perform service when it is really necessary. Future failure monitoring enables predictive maintenance techniques to be conducted at the right time it is needed, rather than too early when the equipment still has life, or too late, when the equipment has already failed. The predictive maintenance uses condition-monitoring by employing different measurement technologies such as vibration analysis, infrared thermography, oil analysis, and acoustic monitoring to monitor the equipment's real-time condition to identify imminent failures, and proactively schedule the required maintenance actions..

The main pillar of predictive maintenance is "Remaining Useful Life (RUL) Estimation", which is defined as the time a machine is expected to work before requiring repair or replacement. Accurate RUL estimation could have significant impacts on the decision making process in different application domains including automotive, nuclear, chemical, and aerospace industries to name but a few. In this context, the inter-relation between "Condition Based Maintenance (CBM), prognostics, and system health management", at one hand and "reliability, degradation process modeling, and RUL estimation", on the other hand, can provide vital information on the system's failure behaviour. In practice, the failure time of a system is, typically, determined from a degradation model derived from measurements made across the



(a) (b)

Figure 1.2: (a) An illustrative example of degradation-threshold failure. (b) The concept of Remaining Useful Lifetime (RUL).

system's lifetime. Due to the diverse nature of industrial and manufacturing systems, and the hidden behavior of the degradation process, developing an accurate model for the degradation process is a key challenge especially with the increase in volume, velocity, and variety of data collected in CPES. Hence, timely detection of faults and failures, through the implementation of an effective reliability framework, enables the decision makers to schedule appropriate maintenance actions to prevent catastrophic incidents.

This thesis' research work has been motivated based on the above discussion. In brief, capitalizing on the importance of implementing innovative, advanced, and smart predictive maintenance strategies, the following two interrelated research directions were pursued:

(i) Proposing a general degradation-modeling framework that can cover a wide range of potential degradation scenarios, and;

(ii) Proposing an accurate framework for RUL estimation.

Next, the aforementioned two research directions are briefly outlined.

1.1.1 Generalized Degradation Modeling

The degradation, which can be regarded as damage to a system, accumulates over time and eventually leads to a product failure when the accumulated damage reaches

a failure limit (failure threshold), either randomly or by industrial norms. Fig. 1.2(a) shows an example of a degradation-threshold failure. Thus, the degradation-threshold failure mechanism forms a natural link between the degradation process and product failures, which leads to assess product reliability by making use of degradation analysis. As a result, the failure time distribution and its parameters can be determined through the analysis of the degradation mechanism. At the core of degradation modeling is development of "accurate" probability models capable of describing the underlying degradation phenomenon. Generally speaking, there are two broad classes of degradation models, i.e., stochastic process models, such as (Wiener process, Gamma process, or Inverse Gaussian process), and general path models. These are the most popular degradation models considered in the existing literature. There are, however, other models available for degradation modeling in the literature that cannot be classified into these two categories. Such models include time-delay model, shock models, continuous-time Markov models, and data-driven approaches [5]. The focus of the thesis in this context is mainly on the degradation models developed based on stochastic processes, as will be discussed comprehensively in Chapter 2.

1.1.2 Accurate RUL Estimation

In this research direction, the thesis shifts focus from the diagnostic stage (degradation analysis) to the prognostic level (RUL prediction), where prognostic is used to handle risks resulting from unexpected failure of machinery. RUL prediction is an enabling discipline in several industrial applications where monitoring the reliability of the underlying complex engineering infrastructure is of paramount importance. RUL estimation has emerged as a key enabling technology to provide an early warning of failure.

Generally speaking, the RUL is defined as a time window that a machine is probable to function properly without requiring repair or replacement actions. In other words, the residual lifetime during which a device can perform its intended function [6]. Fig. 1.2(b) illustrates the RUL concept. RUL estimation is, therefore, essential in a variety of engineering industries, including aerospace, medical instrumentation, civil infrastructure, automobiles, and power plants. Many tools and methods have been introduced for failure prognostics and RUL estimation [6–11]. It seems that the prognostic techniques usually vary based on the type of the considered application, whereas the implemented tools are primarily determined based on the type

of available data and knowledge. Moreover, these methods and tools can be classified into three categories; (i) Physics-based; (ii) Data-driven, and; (iii) Hybrid approaches (multiple-model) [12]. The focus of the thesis in this context is mainly on the category (iii), which is hybrid RUL prediction models, as will be discussed comprehensively in Chapter 2.

1.2 Thesis Contributions

In what follows, the main contributions of the thesis are outlined:

- **Chapter** 3 : General Degradation-modeling Frameworks.

  (1) Introducing a new category for degradation modeling.

  (2) Introducing a new degradation framework referred to as the Multiple-Model Degradation Path (MMDP) estimation. The proposed MMDP framework takes into consideration a set of candidate models for the degradation path, performs degradation prediction based on each model in parallel, and then combines the outputs of localized filters adaptively and in an intelligent fashion to form the overall estimate of the degradation process over time.

  (3) The proposed MMDP is a generalized hybrid non-linear filtering framework that is capable of simultaneously handling different linear and non-linear degradation models.

  (4) The proposed MMDP framework provides near-optimal results without having any prior knowledge of the true degradation model of the system.

  (5) Introducing a new degradation modeling framework referred to as Interactive Multiple Model Particle Filter (IMMPF) estimation, which considers a set of candidate models for the degradation path, localized estimates of the degradation states are then collapsed, in an intelligent fashion, to form the overall estimate of the degradation states adaptively over time.

  (6) A Gaussian IMMPF is developed, where each mode-matched filter forms a Gaussian approximation of its local particles, which is then used in the collapsing and interaction steps.

  (7) Similar to the MMDP approach, the proposed IMMPF framework provides precise results without having any prior knowledge of the true degradation model of the system.

- **Chapter** 4 : HDNN: A Multiple-Model and Hybrid Deep Neural Network Model for Remaining Useful Life Estimation.

  (1) Proposing the first parallel hybrid deep neural network framework for RUL estimation, referred to as the Hybrid Deep Neural Network Model (HDNN). The proposed HDNN framework consists of two parallel paths (one Long Short Term Memory (LSTM) and one Convolutional Neural Network (CNN)) followed by a fully connected multilayer neural network, which combines (fuses) the output of each path to form the target RUL.
  (2) Proposing three other hybrid deep neural network frameworks for RUL estimation based on different deep neural network architectures (BLSTM, GRU, BGRU and CNN).
  (3) Utilizing Monte Carlo simulations to evaluate the effectiveness and robustness of the proposed methods over all the datasets FD001 to FD004 based on different levels (30, 25, and 20)dB of Signal to Noise Ratio (SNR). The results show remarkably stable performance of the proposed models.

- **Chapter** 5 : NBLSTM: Noisy and Hybrid CNN and BLSTM-based Deep Architecture for Remaining Useful Life Estimation

  (1) Proposing a hybrid deep learning framework developed for RUL estimation that, for the first time, integrates noisy Bidirectional Long Short Term Memory (NBLSTM) and noisy Convolutional Neural Network (NCNN) in a parallel fashion.
  (2) Proposing the first noisy hybrid deep learning model for RUL estimation, based on noisy training and at the same time tested on noisy datasets.
  (3) Three other structures, noisy LSTM (NLSTM), noisy GRU (NGRU) and noisy BGRU (NPBGRU), are also incorporated within the proposed framework providing comparable results. To the best of our knowledge, such noisy structures are used for the first time within the domain of RUL estimation.
  (4) Utilizing different styles and values of noisy training and evaluating their effects on the proposed models.
  (5) Utilizing Monte Carlo simulations to evaluate the effectiveness and robustness of the proposed methods. The results show remarkably stable performance of the proposed models.

- **Chapter** 6: Multipath Parallel Hybrid Deep Neural Networks.

  (1) Proposing, for the first time, a multi-path parallel noisy hybrid framework that integrates different noisy deep neural network techniques for RUL estimation.
  (2) Employing the concept of collecting more informative features to achieve better results.
  (3) Utilizing different styles of noisy training and evaluating the effects of them on the proposed models.
  (4) Employing the batch normalization technique with BGRU for the first time to improve the RUL estimation.
  (5) Utilizing Monte Carlo simulations to evaluate the effectiveness and robustness of proposed methods. The results show remarkably stable performance of the proposed models.

1.3 Organization of the Report

Chapter 1 (this chapter) provided an overview and a summary of important contributions made in the thesis. The rest of thesis is organized as follows:

- **Chapter** 2 presents an introduction to the problem at hand, and thoroughly reviews the relevant literature to each topic. This chapter also encapsulate the required technical background for following developments of the thesis.
- **Chapter** 3 develops state-space model of different degradation paths and presents the proposed Multiple-Model Degradation Path (MMDP) estimation and the Interactive Multiple Model Particle Filters (IMMPF) estimation frameworks.
- **Chapter** 4 Proposing hybrid deep neural network frameworks for RUL estimation (HDNN, BiLSTM, GRU and BiGRU), describing the constituent components of the proposed models, developing the different deep learning paths, also describing the used datasets (C-MAPSS), evaluation metrics, operating conditions and fault modes, showing the effects of different time window size, and presents the performance evaluation with additive noise.
- **Chapter** 5 Presenting Noisy Hybrid Deep Neural Network Models for Remaining Useful Life Estimation (NBLSTM, NLSTM, NGRU, and NPBGRU), presenting the network structures and developing the two parallel noisy paths for each model, developing the noisy training and at the same time noisy testing,

evaluation the effects of the operating conditions and fault modes, showing the effects of different time window size, presenting the performance evaluation with additive noise, and displaying the performance comparison of 7 methods with the proposed methods.

- **Chapter** 6 Proposing Multipath Parallel Hybrid Deep Neural Network models (NMPM, TDHA, MPHD, NPHM) based on the integration of three different parallel paths, introducing the network structures and developing the different parallel noisy paths, in addition to the noisy fusion center, developing the different styles of noisy training, introducing the use of batch normalization technique to improve the RUL estimation task.
- **Chapter** 7 Finally, Chapter 7 concludes the thesis and provides some directions for future work.

1.4 Publications

[J-1] **A. Al-dulaimi**, S. Zabihi, A. Asif, A. Mohammadi, "NBLSTM: Noisy and Hybrid CNN and BLSTM-based Deep Architecture for Remaining Useful Life Estimation," *ASME Journal of Computing and Information Science in Engineering,* no. 1182, 2019.

[J-2] **A. Al-Dulaimi**, A. Asif, and A. Mohammadi, "Noisy parallel hybrid model of NBGRU and NCNN architectures for remaining useful life estimation," *Quality Engineering, Special Issue on Reliability Engineering*, vol. 32, issue 3, pp. 371-387 , 2020.

[J-3] **A. Al-Dulaimi**, S. Zabihia, A. Asif, and A. Mohammadi, "A multimodal and hybrid deep neural network model for Remaining Useful Life estimation," *Computers in Industry*, vol. 108, pp. 186-196, 2019. ar, R. V. Patel, and A. Mohammadi, "HMFP-DBRNN: Real-time Hand Motion Filtering and Prediction via Deep Bidirectional RNN," *IEEE Robotics and Automation Letters*, 2019.

[C-1] **A. Al-Dulaimi**, A. Asif, and A. Mohammadi, "Multipath Parallel Hybrid Deep Neural Networks Framework for Remaining Useful Life Estimation," *2020 IEEE International Conference on Prognostics and Health Management (ICPHM), USA* pp. 1-7, 2020.

[C-2] **A. Al-Dulaimi**, A. Asif, and A. Mohammadi, "Noisy Parallel, Hybrid Model for Remaining Useful Life Estimation (NPHM)," *Proceedings of the 2020 IISE Annual Conference*, 2020.

[C-3] **A. Al-Dulaimi**, A. Asif, and A. Mohammadi, "The Noisy Multipath Parallel Hybrid Model for Remaining Useful Life Estimation (NMPM)," *12th Annual Conference of the Prognostics and Health Management Society*, 2020.

[C-4] **A. Al-Dulaimi**, A. Asif, and A. Mohammadi, "Remaining Useful Life Estimation via Hybrid of NBGRU and CNN," *Proceedings of the 2020 IISE Annual Conference*, 2020.

[C-5] **A. Al-Dulaimi**, S. Zabihia, A. Asif, and A. Mohammadi, "Hybrid Deep Neural Network Model for Remaining Useful Life Estimation," *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP),* pp. 3872-3876, 2019.

[C-6] **A. Al-Dulaimi**, S. Zabihia, A. Asif, and A. Mohammadi, "A Noisy Parallel Hybrid Model of CNN and BLSTM for Remaining Useful Life Estimation," *IEEE Global Conference on Signal and Information Processing (GlobalSIP),* 2019.

[C-7] **A. Al-Dulaimi**, A. Mohammadi, and A. Asif, "Modeling Degradation Paths based on Interactive Multiple Model Particle Filters for Prognostic Health Management," *IISE Annual Conference*, QCR-S8: Reliability Analysis - II, 2018.

[C-8] **A. Al-Dulaimi**, A. Mohammadi, and A. Asif, "Generalized Degradation Model for Health Management of Mission Critical Systems," *IISE Annual Conference* pp.1496-1501, 2017.

Chapter 2

Preliminaries and Literature Review

As stated previously, the bases of the research work reported in this thesis are built based on the following two interrelated dimensions:

(i) Proposing a general degradation-modeling framework that can cover a wide range of potential degradation scenarios, and;
(ii) Proposing accurate frameworks for RUL estimation.

The main objective of this chapter is to provide the basic definitions and the key concepts required for the development of this thesis. In what follows, first, fundamentals of modeling the degradation process is presented in Section 2.1. Different aspects of the RUL estimation problem is then described comprehensively in Section 2.2.

2.1 The Degradation Modeling

Degradation refers to the process of lowering the rank, status, or grade of an engineering system leading to a less successful performance level. Developing an accurate model for the degradation process is a primary objective in the prognosis and health management. Degradation models are, typically, derived from measurements related to the critical parameters of relevance to the system. The degradation modeling plays a defining role to improve the decision-making process due to its ability to track the underlying conditions of the system over time [13]. The main objective of degradation modeling is to define the asset's future condition and conduct the maintenance activities in an optimal fashion before the actual system failure occurs. Hence, degradation

modeling is a key task of the diagnosis process, as it provides essential information about the health status of the system.

Diagnostics, is the set of activities performed to recognize a specific fault and its cause in an operating component/system [14]. In other words, diagnostic techniques are designed to determine when equipment is in a deficient condition, thus maintenance activities must be conducted. Therefore, the decision-making of the diagnostic is sensitive to the condition of the asset, because replacement or repair occurs when the asset has reached some pre-determined degradation stage. No prediction or estimation is made in most diagnostic techniques. In other words, the diagnosis is an evaluation stage based on observed symptoms to assess the system's current and past health state [15].

The existing degradation modeling methods can be classified into two broad categories, i.e., stochastic process models such as the Wiener process, Gamma process, Inverse Gaussian process, and Inverse Gamma process, and; general path models [5]. These models are the most common and the main models used in the existing literature.

2.1.1 Stochastic Process Models

In this approach, degradation is assumed to follow a stochastic process. Developing a statistical model for degradation data is to identify a probability distribution model (e.g., Wiener process, Gamma process, Inverse Gaussian process, etc.) to represent the measurements at each observation time [16]. In other words, physical or mathematical models are needed to formulate the degradation process using a specific shape of the degradation path, as a function of the variable measuring the lifetime of a unit (i.e., time in service, cycles, or other mounts of use) [17].

In stochastic process modeling, the degradation signal $\{X(k), k \in T\}$ is assumed to have stationary independent increments, which means for any time k and $\Delta k > 0$, the increment $\Delta X(k) = X(k + \Delta k) - X(k)$ only depends on Δk and some other parameters. Generally speaking, $\Delta X(k)$ follows a distribution, which has additivity property. As far as stochastic process models are concerned, three types of common and well-exploited degradation models are Wiener process, Gamma process, and inverse Gaussian process. Next, these models are described in more detail.

2.1.1.1 Wiener Process (Brownian Motion)

Wiener process, also called Brownian motion is a continuous-time stochastic process with independent, real-valued increments and decrements that randomly projects the degradation based on the drift and shift parameter of the wiener process over time. It is basically a sequence of normally distributed random variables, and for later times, the variances of these normally distributed random variables increase to reflect the fact that estimating the value of the method over a longer period of time becomes more uncertain. Wiener processes are appropriate for non-monotonous degradation processes resulting from minor repair, self-healing, or reduced use intensity, which are frequently found in practice. The process can provide a satisfactory and robust description of degradation signals for the unit/system including bearings and rotating machinery, lumen degradation data, bridge beam degradation, light emitting diode (LED) lamps, and batteries [18], to name but a few. The Wiener process is widely used for modeling degradation processes due to its useful mathematical properties and physical interpretations, in addition to its ability to capture the inherent uncertainty associated with the progression of degradation over time [19].

Pan *et al.* [20] introduced a degradation modeling and reliability estimation approach by modeling the degradation process of the deteriorating system using a wiener process with truncated normal distribution to characterize the unit-to-unit variability. Pan *et al.* [21] proposed an approach based on a time-transformed Wiener process with jointly considering temporal variability, measurement errors, and unit-to-unit heterogeneity. Tsai *et al.* [22] proposed a model for the lumen degradation of LED via a wiener diffusion process. Hao and Su [23] proposed a general random effect wiener process to characterize the population degradation path. The proposed model can capture the sources of uncertainty including unit-to-unit variation and time correlated structures for some laser devices. Jin *et al.* [24] utilized the Wiener process with random drift, measurement error, and diffusion coefficient to identify the off-line population degradation of secondary battery capacity. Such a model is developed to capture several sources of uncertainty including unit-to-unit variation, stochastic correlation, and time uncertainty.

In addition to these electronic devices, wiener processes can be used to model the mechanical structural degradation processes and some electromechanical operations. For example, Li *et al.* [25] used failure modes, mechanisms, and effects analysis, to establish a degradation model. This reliability modeling and life estimation approach

is based on a Wiener process with the random effects for the momentum wheel used in satellites. Mishra and Vanli [26] combined Wiener process degradation modeling and principal component regression to introduce a new approach that predicts the RUL of a structure from Lamb wave sensor data. Wang *et al.* [27] used the Wiener process to model the degradation process of an axial piston pump. Recently, Cheng *et al.* [28] proposed a new model by integrating a double-Wiener process model with Monte Carlo algorithms to provide a new solution to the degradation modeling and reliability prediction of machinery with multiple degradation characteristics. More recently, Dong *et al.* [29] proposed a two-stage degradation model to deal with the rail track geometry degradation issue, where a correlated bivariate Wiener process and a univariate Wiener process are adopted to model the degradation levels of the system in the first stage and the second stage, respectively.

Although Wiener processes have been employed to model the degradation behavior in many fields and applications, they are not appropriate for modeling monotonic degradation processes such as wear or cumulative damage processes. The degradation under the Wiener process for the non-monotonous degradation system is given by

$$\phi_\tau = \tau_0 + \eta\tau + \sigma\mathcal{B}_\tau, \tag{2.1.1}$$

where $\tau_0 = \phi(0) \in R$ is the initial degradation value, $\eta \in R$ is the drift parameter, σ denotes the variance parameter, and $\mathcal{B}_\tau$ with $\mathcal{B}_0 = 0$ is the standard Brownian motion.

2.1.1.2 Gamma Process

The Gamma process is a stochastic model with independent, non-negative increments having a gamma distribution with an identical scale parameter. The Gamma process is appropriate to model gradual damage monotonically accumulating over time in a sequence of tiny increments [30]. The Gamma process has been proven to be an effective tool for modeling such degradation behavior because the required mathematical calculations and the physical interpretation are fairly straightforward. Moreover, it is also capable of modeling the temporal variability of the degradation process [31]. On the other hand, as it is strictly applicable to monotonic processes, that may limit its application for certain degradation processes, for example, wear, corrosion, fatigue, erosion, crack growth, creep of materials, consumption, and degrading health index,

to name but a few. The common use of the gamma process to model uncertain degradation in a wide variety of applications encourages the researchers to introduce new algorithms based on the gamma process.

Pan *et al.* [32] proposed an approach by using the Gamma process to model degradation processes, which is a product fatigue crack in some engineering systems. Qiu *et al.* [33] utilized a two-stage Gamma process that is one is normal and the other is a defective stage, to investigate the optimal mission abort policy on an unmanned aerial vehicle (UAV). Sun *et al.* [34] introduced a method to find the reliability and the storage lifetime for O-rings of gas steering engine at different temperature by integrating the Gamma stochastic process with the traditional accelerated model. Zhang *et al.* [35] proposed a reliability demonstration method by applying the Gamma process to describe the monotonic degradation process of alloy products. Duan *et al.* [36] examined the optimal design problems for constant-stress accelerated degradation test for carbon film resistor based on gamma processes with fixed effect and random effect. Cholette *et al.* [37] presented an approach to handle the degradation of boiler heat exchangers due to erosion by combining a physical erosion model and a Gamma process to account for the uncertainties in the thickness degradation process. In addition to these applications that belong to many fields, the Gamma process has been applied to model various types of degradation processes in the management of civil infrastructure assets. such as, Edirisinghe *et al.* [38] developed a model that considered the Gamma process to be used for predicting building element deterioration because of the associated temporal variability of degradation. Also, Mahmoodian *et al.* [39] presented a stochastic gamma process model to account for temporal variability and corrosion-related uncertainties in concrete sewer pipes that usually increase the risk of pipe aging failure. Zhang *et al.* [40] adopted the gamma process model to investigate the time-dependent reliability of carbonation behavior in recycled aggregate concrete (RAC).

A Gamma distribution with shape parameters $\alpha > 0$ and scale parameter $\beta > 0$ has a probability density function given by

$$Gamma(x|\alpha,\beta) = \frac{\beta^{\alpha} x^{\alpha-1} e^{-\beta x}}{\Gamma(\alpha)}, \tag{2.1.2}$$

where the Gamma function for $\alpha > 0$ is $\Gamma(\alpha) = \int_{z=0}^{\infty} z^{\alpha-1} e^{-z} dz$.

2.1.1.3 Inverse Gaussian Process (IG)

Is another important stochastic model for degradation modeling aside from the Wiener and Gamma process models. Which is similar to the Gamma process in terms of the monotone degradation path with independent, non-negative increments, however, it has an important practical advantage over the gamma process, which is the closed form of its first-time passage distribution, moreover, the flexibility in dealing with random effects and covariates [41]. IG has received more attention in degradation modeling due to its clear physical interpretation and nice mathematical properties. Noteworthy, there is an inverse relationship between Wiener and IG processes that makes it possible to apply many of the Wiener process properties to the IG process. These advantages of IG made it more suitable and capable in a range of degradation analysis applications where the two processes (Gamma and Wiener) mentioned earlier have failed, such as the GaAs laser degradation analysis [42]. It has also been demonstrated that the IG process is applicable to a number of different applications such as energy pipelines, crack growth, corrosion, fatigue, and contamination [43], to name but a few examples. It is important to notice that the path of an inverse Gaussian process is strictly monotone, thus, the IG is no longer valid when the degradation path is not monotonous [42].

The researchers have shown great interest in modeling the degradation behavior through the inverse Gaussian process. For example, Ye *et al.* [41] investigated the use of IG processes in modeling laser device degradation. Also, Peng *et al.* [43] proposed a general Bayesian framework for degradation analysis using Gaussian process models. Qin *et al.* [44] used the IG process to model the degradation process of energy pipelines. Ye*et al.* [45] proposed a model to examine the optimal constant-stress accelerated degradation tests plan based on the IG process model in an electrical connector. Chen *et al.* [46] introduced the IG process with skew-normal distribution as a random effect to represent unit-to-unit variability of the degradation rate in aluminum alloy specimens. Xu*et al.* [47] presented random effects model using the inverse Gaussian process for Integrated circuit device degradation, where the mixture normal distribution has been used to account for both unit-specific and subpopulation-specific heterogeneities. *et al.* [48] proposed an improved Bayesian framework by considering the IG process degradation models with constant, monotonic, and S-shaped degradation rates, for analyzing the degradation of heavy machining tool's spindle system.

The probability density function (PDF) with shape parameter $\mu\boldsymbol{\Lambda(k)}$ and scale parameter $\lambda\boldsymbol{\Lambda^2(k)}$ when $\mu > 0$, $\lambda > 0$ and monotonic increasing function of time k, where the mean of $IG(k)$ is $\mu\boldsymbol{\Lambda(k)}$, and its variance is $\mu^3\boldsymbol{\Lambda(k)}$ / $\lambda\boldsymbol{\Lambda(k)}$. Then the probability density function (PDF) is given by

$$\boldsymbol{f}(x|\mu,\lambda) = \sqrt{\frac{\lambda\Lambda^2(k)}{2\pi x^3}} e\left[-\frac{\lambda}{2x}(\frac{x}{\mu} - \Lambda(k))^2\right] \tag{2.1.3}$$

As our research is mainly focused on stochastic process models, we will not consider the other models with details.

2.1.2 General Path Models

Statistical models for continuous degradation data, where the degradation process is defined as a function of time, possibly with two sets of parameters fixed-effects and random-effects parameters [5]. Many extensions of the general path model were introduced by examining various types of statistical modeling methods for different applications [17]. The simplicity and ability to model continuous processes have increased the popularity of the general path model however, sometimes these models may not well represent the actual process of deterioration due to the oversimplification of the nature of that process. Moreover, general path models assume that the underlying deterioration to be deterministic and therefore capturing a product's time-varying behavior is an issue [49]. Based on reference [50] the model can be represented as

$$D_{ij} = D(k_{ij}; \varphi\ \theta_i) + \varepsilon_{ij}, \tag{2.1.4}$$

where $D(\cdot)$ denotes the actual degradation of the i^{th} unit at time k_{ij}. ε is the vector of fixed-effects parameters (common for all units), θ_i is the vector of random-effects parameters representing the characteristics of thei^{th} unit, and ε_{ij} is associated random error of the i^{th} unit at time k_{ij} which is assumed to be $\varepsilon \sim \mathcal{N}(0, \sigma^2)$ with zero mean and variance σ^2. General path models have been considered with more detail by many references [5, 17, 49, 50].

2.1.3 Existing Work on Degradation Modeling

The extensive research on degradation modeling has led to many diagnostic/prognostic tools and techniques aid in developing an accurate and generalized model for the degradation process. The degradation modeling designs have been built based on different directions and resulting in many classes. For example references [19, 20, 22, 25–27] have considered a single bivariate degradation model based on Wiener process, while the references [35–38, 51, 52] have followed the same direction but using the Gamma process to model the degradation behavior. In this line of research, another group of methods [53–57] has been proposed to follow the Inverse Gamma processes. On the other side, [21, 23, 28, 29, 58], [33, 59, 60], and [43, 47, 61] have introduced another category of techniques which considered multiple degradation measures by assuming that all degradation paths statistically follow one specific type of distribution (Wiener, Gamma, and IG, respectively). Whilst [46, 62, 63] extended the degradation modeling approaches to incorporate two different statistical distributions to model system degradation.

Similarly [16, 64–69], have utilized multiple-path degradation models where the underlying statistics for each of the degradation model is either the same or different in each approach, however, these approaches represented another class of degradation models (referred to as the copula-based approaches) and that because of utilizing the copula technique that has an important role in identifying the complex interdependence structure among the degradation modes [70]. Another class of algorithms [70–77], based on Kalman Filter (KF) and Particle Filter (PF) and their extensions, have proven to be robust in this field.

It can be observed that all the aforementioned degradation models have mainly two issues, (i) first; most of these works deal with one specific degradation path, which is developed based on the assumption that a particular type of statistical distribution governs the degradation process (e.g., Wiener, or Gamma distribution,... etc.). In practice, a system may consist of multiple components or a component may have multiple degradation measures that require simultaneous consideration of multiple degradation paths. While few attempts have made toward developing multi-path degradation models, incorporation of several degradation path models simultaneously have rarely been considered in the literature. (ii) The second issue is that the models are built based on prior knowledge of the true degradation model of the system, which is often unable to comply with the time-varying degradation characteristics of the real

system. In other words, all the aforementioned algorithms whether they are based on single or multiple degradation paths, they built their algorithms based on an implicit assumption of the degradation model.

Another important class of algorithms [78–83], that based on data-driven methods, has made a big change in degradation modeling as it does not follow the traditional mechanisms of molding the degradation behavior, but it has its own way by using the field data to construct empirical models of degradation. Technically, these approaches use data mining and machine learning techniques to learn the system behaviors directly from the collected condition monitoring data. The learned knowledge is then employed to determine the health state, acquire the degradation trend [84]. These methods can be integrated with other algorithms such as KF or PF [72, 85–87] to achieve the same task. Although most of the recent studies have considered the data-driven class and its extensions for degradation modeling applications, these approaches have their limitations. For example, there exist two main deficiencies in using data-driven methods. First, they are highly dependent on the quantity and quality of the collected degradation data. Second, some of them can hardly maintain good generalization performance across various prognostic scenarios, especially when this model is well configured for a certain scenario.

The Integration of PF with powerful approaches such as multiple-model (MM) or multiple model adaptive estimation (MMAE) (as a special case of the MM) and interacting multiple models (IMM) (as an improved version of the MM) has introduced a new class of algorithms. As mentioned earlier, the PF based approaches have been widely used for prognostic applications [74–77, 85–87], On the other hand, multiple models (MM) or multiple model adaptive estimation (MMAE) filtering have been studied as powerful approaches ranges from target tracking to fault detection and isolation [88–95]. Despite the fact that these approaches are powerful, they still suffering from the previously mentioned limitations of the other classes of algorithms.

The PF integrated with IMM, is widely used in the target tracking literature, however, few studies such as [96] has proposed interacting multiple particle filters for fault diagnosis, but depended on linear system model assumptions and Gaussian noise and disturbances. And [97] has proposed interacting multiple particle filters for fault diagnosis but considered only sharp degradations. While [98] has introduced a new approach using a state augmented particle filtering integrated with IMM considering different degradation mechanisms, however, the degradation mechanism was very

limited and particularly designed for the "Crack growth degradation problem". This class of algorithms which was based on the integration of PF and MM or IMM was the primary motive behind the development of the proposed degradation modeling in this thesis as we will see in Chapter 3.

2.1.4 The Basic Definitions and the Key Concepts for the Proposed Degradation Modeling

2.1.4.1 The Particle Filter (PF)

Is an effective and powerful technique for sequential signal processing with a broad variety of science and engineering applications. It is a Monte Carlo based method particularly useful in dealing with nonlinear and non-Gaussian problems. The particle filter is also known as the bootstrap filter, condensation algorithm, and survival of the fittest [99]. The underlying principle of particle filters is that any probability density function (pdf) can be represented as a set of samples (particles) based on the concept of sequential importance sampling and the use of Bayesian theory [100]. The main advantage of the particle filter is that It does not depend on any local linearization technique but rather approximations in the representation of the desired distributions by discrete random measures [100]. The main part of the particle filter is the sequential importance sampling (SIS) that is built based on the importance sampling.

Importance Sampling (IS) Importance sampling is a form of approximation, rather than sampling, and it used to generate random variables from complicated densities. Let us use the Importance Sampling (IS) to evaluate the following integration

$$\mathbb{E}_{p(x|z)}\big\{h(x)\big\} = \int h(x)p(x|z)dx, \tag{2.1.5}$$

where $\mathbb{E}\big\{\ \cdot\ \big\}$ represents the expectation. To avoid this type of integration in the Bayesian statistics, N_s random samples $\mathbb{X}^i$, for $(1 \leq i \leq N_s)$, drawn from the probability distribution $p(x|z)$, then by evaluation the function $h(x)$ on these samples, we can estimate their mean as follows

$$\mathbb{E}\big\{h(x)\big\} \approx \sum_{i=1}^{N_s} h(\mathbb{X}^i)p(\mathbb{X}^i|z). \tag{2.1.6}$$

Since sampling from the true posterior $p(x|z)$ is generally unavailable, or difficult, it is common to derive the particles from a proposal distribution denoted by $q(x|z)$, hence, the integration form in Eq. (2.1.5) can be written in terms of the proposal distribution as follows

$$\mathbb{E}\big\{h(x)\big\} = \int h(x)\frac{p(x|z)}{q(x|z)}q(x|z)dx, \tag{2.1.7}$$

as a result the statistical mean in Eq. (2.1.8) will be as follows

$$\mathbb{E}\big\{h(x)\big\} \approx \sum_{i=1}^{N_s} h(\mathbb{X}^i)W^i p(\mathbb{X}^i|z), \tag{2.1.8}$$

where $W^i = \frac{p(\mathbb{X}^i|z)}{q(\mathbb{X}^i|z)}$, for $(1 \leq i \leq N_s)$, represents the weights related to the vector particles $\mathbb{X}^i$.

Sequential importance sampling (SIS) Now, let us consider the following dynamic state-space form, where the discrete time state equation and the observation equation of the nonlinear systems are given as follows

$$x_k = \boldsymbol{f}(x_{k-1}) + \boldsymbol{w}_k, \tag{2.1.9}$$

$$\text{and} \qquad z_k = \boldsymbol{h}(x_k) + \boldsymbol{v}_k, \tag{2.1.10}$$

where functions f$(\cdot)$ and h$(\cdot)$ in Eqs. (2.1.9) and (2.1.10) represent the state and observation models, respectively. $x_k \in \mathbb{R}^{n_x}$ is the state vector for the system, k denotes the time instant, $z_k \in \mathbb{R}^{n_z}$ is the observation vector, $\boldsymbol{w}_k$ and $\boldsymbol{v}_k$ are the system noise and the observation noise, respectively. The posterior distribution of initial state can be represented as $p(x_0)$ based on the Bayes theorem. Then x_k can be speculated as $p(x_k|x_{k-1})$ and z_k can be speculated as $p(z_k|x_k)$. In general, the assumptions are also made for the system that x_k subjects to the first-order Markov process and z_k is conditionally independent of previous observation $(z_1, z_2, ...z_{k-1})$ given x_k.

Using $\boldsymbol{x}_{0:k} = \{x_0, \dots x_k\}$ and $\boldsymbol{z}_{1:k} = \{z_1, \dots z_k\}$ to denote the sequences of states and observations, respectively. Thus, using the framework of Bayes theorem, the posterior distribution of the hidden states x_k can be written as

$$\boldsymbol{P}(x_{0:k}|z_{1:k}) = p(x_{0:k-1}|z_{1:k-1})\frac{p(z_k|x_k)p(x_k|x_{k-1})}{p(z_k|z_{1:k-1})}, \tag{2.1.11}$$

since $p(z_k|z_{1:k-1})$ is a normalizing constant then Eq. (2.1.11) can be simplified as

$$\boldsymbol{P}(x_{0:k}|z_{1:k}) \propto p(x_{0:k-1}|z_{1:k-1})p(z_k|x_k)p(x_k|x_{k-1}), \tag{2.1.12}$$

where $\propto$ means being proportional to. For the nonlinear systems Eq. (2.1.9) and Eq. (2.1.10), the posterior distribution $p(x_{0:k}|z_{1:k})$ is difficult to obtain because of the complicated integral calculation [101].Instead of that, PF approximates it with a mass of particles $\boldsymbol{x}_{0:k}^i$ $i \in (1, \dots N_s)$, in which i represents the serial number of particles, and N_s is the sum total of them. The initial particles i.e., $\mathbb{X}_0^i$ are drawn from $p(x_0)$. Then the importance distribution is chosen as

$$q(x_{0:k}|z_{1:k}) = q(x_{0:k-1}|z_{1:k-1})q(x_k|x_{0:k-1}, z_{1:k}). \tag{2.1.13}$$

And every particle is given a weight W_k^i, then according to Eq. (2.1.11) to Eq. (2.1.13), the weight can be formulated as

$$W_k^i = \frac{p(z_{0:k}|z_{1:k})}{p(x_{0:k}|z_{1:k})} \propto W_{k-1}^i \frac{p(z_k|\mathbb{X}_k^i)p(\mathbb{X}_k^i|\mathbb{X}_{k-1}^i)}{q(\mathbb{X}_k^i|\mathbb{X}_{0:k}^i, z_{1:k})}, \tag{2.1.14}$$

where W_k^i is the importance weight. Denote the normalized W_k^i as $\bar{W}_k^i$ that can be found as

$$\bar{W}_k^i = \frac{W_k^i}{\sum_{i=1}^N W_k^i}, \tag{2.1.15}$$

then, the posterior distribution $p(x_{0:k}|z_{1:k})$ can be approximated by particles as

$$\boldsymbol{P}(x_{0:k}|z_{1:k}) \approx \sum_{i=1}^{N} \bar{W}_k^i \delta(x_{0:k} - \mathbb{X}_{0:k}^i), \tag{2.1.16}$$

where $\delta(\cdot)$ is the Dirac delta measure. In the case that the importance distribution satisfies

$$q(x_k|x_{0:k-1}, z_{1:k}) = q(x_k|x_{k-1}, z_{1:k}), \tag{2.1.17}$$

then

$$W_k^i \propto W_{k-1}^i \frac{p(z_k|\mathbb{X}_k^i)p(\mathbb{X}_k^i|\mathbb{X}_{k-1}^i)}{q(\mathbb{X}_k^i|\mathbb{X}_{k-1}^i, z_k)} \tag{2.1.18}$$

Commonly, the PF chooses the transitional distribution probability as importance distribution, that can be written as

$$q(\mathbb{X}_k^i|\mathbb{X}_{k-1}^i, z_k) = p(\mathbb{X}_k^i|\mathbb{X}_{k-1}^i), \tag{2.1.19}$$

then

$$W_k^i \propto W_{k-1}^i p(z_k|\mathbb{X}_k^i). \tag{2.1.20}$$

then the posterior distribution $p(x_k|z_{1:k})$ can be written as

$$\boldsymbol{P}(x_k|z_{1:k}) \approx \sum_{i=1}^{N} \bar{W}_k^i \delta(x_k - \mathbb{X}_k^i). \tag{2.1.21}$$

With the approximated $p(x_k|z_{1:k})$, the states x_k can be estimated by the corresponding methods, such as MMSE (Minimum Mean Square Error) [102].

$$\hat{x}_k = \sum_{i=1}^{N} \bar{W}_k^i \mathbb{X}_k^i. \tag{2.1.22}$$

Up to this point, the above procedures are known as sequential importance sampling (SIS), which is a basic form of PF. The SIS algorithm thus consists of recursive propagation of the weights and support points as each measurement is received sequentially. Fig. 2.1 presents the main idea of the SIS algorithm.

A common problem with the SIS particle filter is the degeneracy phenomenon, where after a few iterations, the weights of particles concentrate on the minority in the PF algorithm and the majority of particles will have negligible weight. It has been shown that the variance of the importance weights can only increase over time, and thus, it is impossible to avoid the degeneracy phenomenon [103]. This degeneracy indicates a significant computational effort to update particles whose contribution to the approximation to $p(x_k|z_{1:k})$ is almost zero. In this case, the collection of particles are not able to express the actual posterior distribution accurately. The particle degeneracy degree can be evaluated by the effective sample size N_{eff} [104], which is given by

$$N_{\text{eff}} = \frac{1}{\sum_{i=1}^{N} (\bar{W}_k^i)^2}. \tag{2.1.23}$$

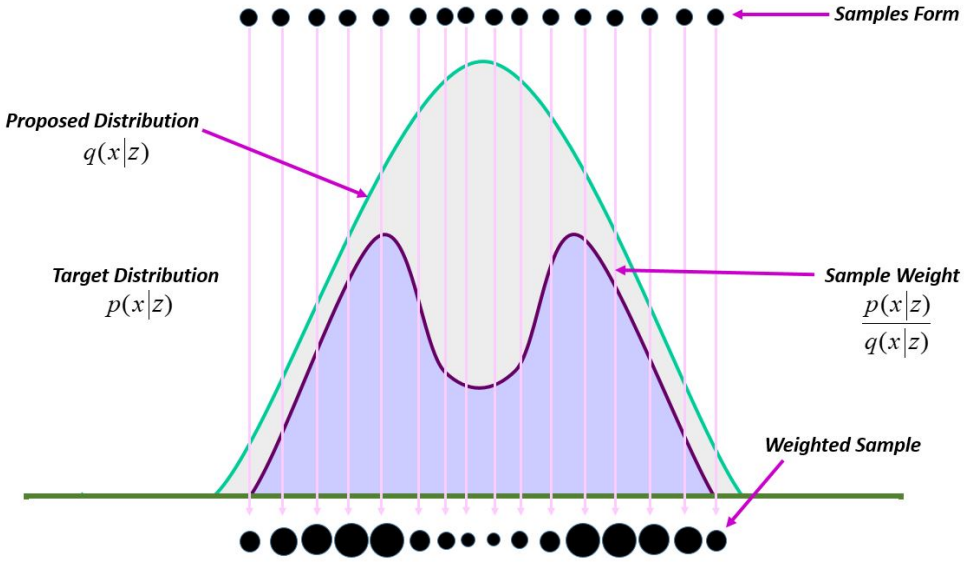


Figure 2.1: The SIS diagram.

The smaller the effective sample size is, the more degeneracy the particles will present. In order to solve the problem of particle degeneration, the resampling technique is introduced to improve the general PF. Resampling is used to sample the particles some times by means of the posterior probability density function, and thus obtain a new particles collection, so the main idea of resampling is a scheme that eliminates particles with small weights and replicates particles with large weights. During the past decades, many resampling techniques have been constructed such as, multinomial resampling (which is the most popular one), stratified/systematic resampling, and residual Resampling [105]. The following is the most popular and straightforward resampling technique, which is the multinomial resampling, and the procedures of that as follows

Perform the following three steps for $i = 1, \dots N$

(i) Generate a random number u_i from the uniformly distribution over$(0, 1]$

(ii) Search the variable $j \in \{1, \dots N\}$ which satisfies

$$\sum_{m=1}^{j-1} \bar{W}_k^i < u_i \leq \sum_{m=1}^{j} \bar{W}_k^i. \tag{2.1.24}$$

(iii) Store the $\mathbb{X}_k^i$ as a offspring particle

After the resampling, the particles with small weights are eliminated, and many offsprings are created for the particles with large weights. The posterior distribution

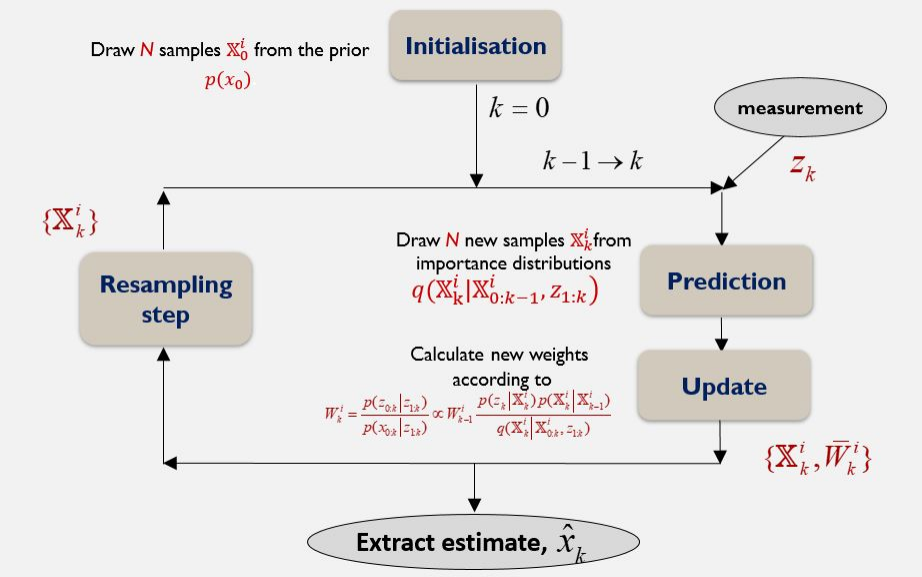


Figure 2.2: The PF diagram.

$p(x_k|z_{1:k})$ can be approximated by these offspring particles as

$$\boldsymbol{P}(x_k|z_{1:k}) \approx \frac{1}{N}\sum_{i=1}^{N} N_k^i \delta(x_k - \mathbb{X}_k^i), \tag{2.1.25}$$

where N_k^i is the number of offsprings for the parent particle $\mathbb{X}_k^i$. The resampling technique can reduce the degradation of particles effectively. However, it also brings some negative effects.The large-weight particles will become the major choice in the samples, and the sampling results contain a host of repeated points with the increase of the number of iterations, leading to the loss in the diversity of the particles. This then generates another problem which is the particle impoverishment phenomenon. In the procedure of resampling, the small-weight particles are eliminated and the large-weight particles are selected as the parent particles. This leads to the loss of the diversity of the offspring particles. Offspring particles are not included in the region of the posterior distribution and they have no contribution to the approximation of posterior distribution. In general, we can increase the number of particles to solve the problem of particle impoverishment, but the computation load will also be increased [106]. Fig. 2.2 presents the main steps of the PF.

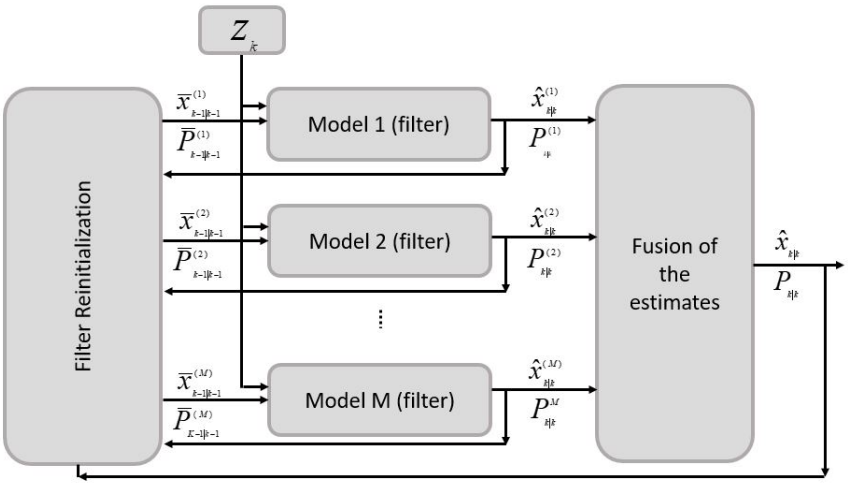


Figure 2.3: The Structure of MM estimator.

2.1.4.2 The Multiple Model (MM)

Multiple model estimation is one of the most reliable and efficient approaches that is a more likely way to hybrid estimation. This approach has been designed based on a collection of multiple models used to cover the possible patterns of system behavior (called modes), These models represented by a bank of filters, that run in parallel at every time, each based on a unique model in the collection [107]. Then, the algorithm fuses the output from the running filters to generate the overall estimates. The literature includes many applications of the MM method ranging from target tracking to fault detection and isolation [88–95]. Although MM approaches have played a major role in many fields, they had many limitations and challenges that need to be improved as we mentioned earlier in section. 2.1.3.

The efficiency of an MM algorithm largely depends on several factors, where the major one is the set of models used and their associated filters, more specifically, the number, types, parameters, and designs of the filters and then the models. Another important factor is the fusion of the estimates from the elemental filters. Fig. 2.3 demonstrates the main idea of the MM estimator, where $\hat{\boldsymbol{x}}^{(i)}_{k|k}$ denotes the estimate of x_k that obtained from the filter of model i at time k, $\bar{x}^{(i)}_{k-1|k-1}$ is the equivalent estimate at $k-1$ and the input to filter i for k_{th} time cycle, and $\hat{\boldsymbol{x}}_{k|k}$ is the overall estimate. While $\boldsymbol{P}^{(i)}_{k|k}$, $\bar{\boldsymbol{P}}^{(i)}_{k-1|k-1}$, and $\boldsymbol{P}_{k|k}$ are the associated covariances.
Since this approach was first introduced more than 50 years ago [108] has been improved and developed by many researchers and scientists. Today there are different

improved versions of the MM among the most successful once of them are (i) The Multiple Model Adaptive Estimation (MMAE) and (ii) The Interactive Multiple Model (IMM).

2.1.4.3 The Multiple Model Adaptive Estimation (MMAE)

An attractive class of adaptive estimators that represents one of the successful versions of the multiple model approach. It has received wide attention and achieved significant progress in handling problems with model uncertainty [108]. The key idea underlying the multiple model adaptive estimation is combining dynamic hypotheses evaluating concepts with linear or nonlinear estimators leading to an algorithm for system identification. Where this approach is composed of a parallel bank of filters, each one of them corresponds to a model in the model set that constructed to identify a particular fault status or the uncertainty model of the system. When the set of models utilized by the MMAE approach doesn't change, it is called a fixed set of models, which represents an impractical estimator when handling partial failures or simultaneous failures, as the number of models needed to cover all expected failures can be large [109]. To address this problem, different methods have been proposed, for example [110] suggested using a hierarchical structure, while [111–113] utilized a moving-bank MMAE algorithm, in order to reduce the required number of filters. However, all these methods can not accommodate more than two faults simultaneously. Recently, [114] proposed the selective reinitialization algorithm based on the Unscented Kalman Filter (UKF) for reducing the size of the model set.

The estimation of the MMAE is based on collecting the residuals (the difference between the predicted measurements and sensor measurements) from the bank of filters to determine the respective model weights (probabilities of the models being the correct one) and the final state estimation is provided by the weighted sum of each filter's estimate. The explanation for this is that the most accurate filter is the one that provides the state estimation that has the highest probability [115]. Almasri *et al.* [116] proposed a model to accurately identify and isolate four wheel block faults in a robot model, by integrating Multiple model Adaptive Estimation (MMAE) and the Extended Kalman Filter (EKF). While Lu *et al.* [117] proposed the MMAE approach based on (UKF) for tracking and compensating sensor and actuator failures in aircraft flight control systems. Moreover, Vaezi and Izadian [118] utilized MMAE based on Kalman Filters that representing specific operating conditions, to

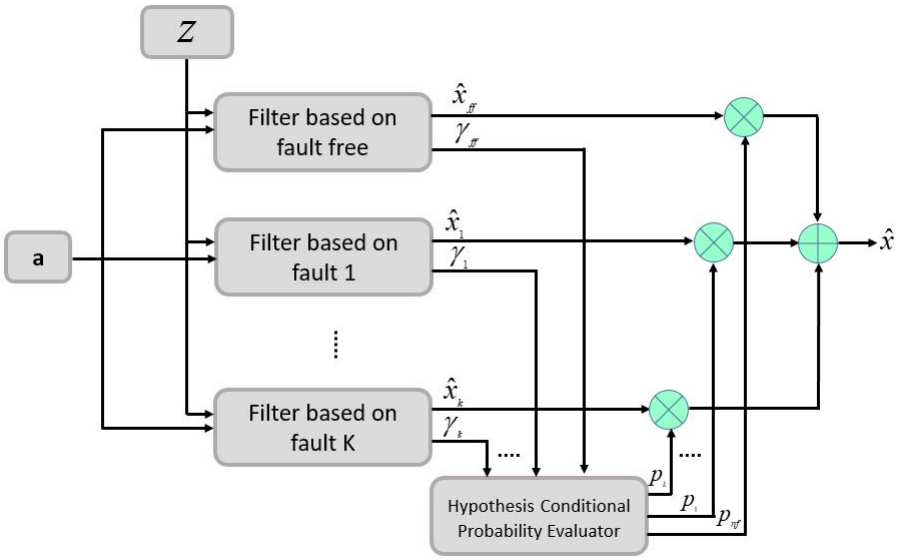


Figure 2.4: The Block Diagram of the MMAE Approach.

estimate the state of a nonlinear hydraulic wind power transfer system subject to different operating regimes that are caused by external factors such as variations in wind speed. Also, Lu *et al.* [119] proposed MMAE based on a bank of Kalman Filters for fault detection in nuclear power plants. In addition, Renwick *et al.* [120] presented a reinforcement learning based MMAE supervisor to perform fault detection for either accelerometer failure or pitot tube in the simulation of the flying fish autonomous unmanned seaplane. Fig. 2.4 presents the block diagram of the MMAE approach. Where $\boldsymbol{z}$ denotes the vector of measurement, a represents the control input, $\hat{\boldsymbol{x}}_k$ is the state estimate at sample step k, γ_k is the vector of innovations, p_k represents the conditional probability, and finally "ff" means fault free.

All the aforementioned approaches have shown various outcomes in terms of accuracy, generality, robustness, and efficiency, however, all of them have used either the KF or one of its extensions (EKF and UKF), also, the approaches that aimed to model the degradation behavior, were designed based on prior knowledge about the real degradation models.

2.1.4.4 The Interactive Multiple Model (IMM)

The IMM is the algorithm that firstly proposed by (Bar-Shalom and Blom) [121] and has been the most successful, powerful, and cost-effective multiple model method for addressing multi-behavior issues in several areas such as target tracking, fault

detection, and isolation and many other problems. What distinguishes IMM from MMAE is the interaction of the associated filters with each other resulting in better performance in the state estimation. The IMM is a version of the MM, and then it comprises of a bank of parallel filters each of them representing a separate model of the system under consideration. For each filter, at the beginning of each cycle, the initial estimate is a mixture of all recent estimates from a specific model. This mixing allows the IMM to take full account of the history of the modes, leading to faster and more reliable estimation for the changed system states. The switching probabilities and the likelihood of each of the models are controlling the interaction between the models, then the IMM result is a combined state vector which is the sum of the state vectors for each of the modes weighted by their model probabilities [122].

Although the IMM is an improved and powerful approach, it's sharing the same problem of missing the model representing the true behavior of a system. Few methods have been proposed to overcome this issue. Li and Jilkov [123]. proposed the expected-mode augmentation to manage the model set, however, this method needs an extra feature to be added to the IMM design. Ru and Li [109] suggested utilizing a maximum likelihood estimator for estimating the extent of the faults after determining a fault using the IMM, which again ends with the same problem that is a larger model set.

The flexibility, cost-effective, and exceptional tracking ability, have made the IMM to be widely adopted approach in many fields, such as target tracking, traffic control, human tracking [124]., In addition to the PHM applications [98]. For the PHM field, Zhang and Li *et al.* [125] Introduced an integrated framework that used the IMM with a bank of KF.s, to detect single and double faults of both sensors and actuators of an aircraft, in addition to the superior ability of diagnosis, and state estimation. Kim *et al.* [126] introduced a new fault detection and diagnosis algorithm by integrating the fuzzy logic and the IMM (based on KF) to handle the failure of the aircraft actuator. Zhao *et al.* [95] proposed recursive fault detection and diagnosis algorithm for the ball-and-tube system, by utilizing a KF based IMM to overcome the problem of inaccurate transition probabilities. Yan *et al.* [124] proposed a general Prognostic framework using a KF based IMM to determine the system health by a health index, and then estimating the RUL from the division of the current health value on the degradation rate of the health index at that moment. Ru *et al.* [109] proposed an integrated framework for fault detection, identification, and state estimation, for the

Actuator Failures of B747 aircraft. This approach has been built based on using KF based IMM for fault detection and identification in addition to the maximum likelihood estimator for estimating the extent of failure. Vianna and Yoneyama *et al.* [122] proposed an integration of IMM with a filter bank of extended Kalman filters that contain augmented state-space models in order to model both the dynamics of the valve and the dynamics of the degradation. Zhao *et al.* [127] Introduced a new fault-detection and diagnosis approach for the stochastic hybrid system taking into account the uncertainty of the model parameter, where all the possible behaviors of the quadruple water tank system (normality, single fault, and multiple faults) have been considered by using a suitable set of modes, and at each time step, the most likely mode is selected. Judalet *et al.* [128] utilized an adapted IMM algorithm based on different banks of filters (EKF, UKF, and the first-order divided differences filter (DD1)), for detecting and isolating the failures of sensor and actuator in a drive-by-wire road vehicle.

All the aforementioned approaches were employed the IMM algorithm integrated with the bank of either the KF or one of its extensions (EKF and UKF). Additionally, the methods intended at modeling the degradation behavior were built on the basis of prior knowledge of the actual degradation models, and many other assumptions. Furthermore, a few approaches have utilized the IMM based on particle filters bank, for PHM applications. The details will be in chapter 3. The IMM estimator is a recursive algorithm, where four steps are carried out in each cycle.

(i) Model Mixing/ Interaction.
(ii) Filtering.
(iii) Model Probability Update.
(iv) Overall Estimation.

Fig. 2.5 presents the block diagram of the MMAE approach. Where $\hat{\boldsymbol{x}}^{(j)}_{k-1|k-1}$ is the estimated state at time $k-1$, $\boldsymbol{P}^{(j)}_{k-1|k-1}$ is the associated covariance matrices, for j (degradation modes) $(1 \leq j \leq N)$, $\mu^{(i)}_{k-1}$ is the probability of a model $m^{(i)}$ being in effect at the time step $(k-1)$ for $(1 \leq i \leq N)$, Λ_k denotes the likelihood function, and $\hat{\boldsymbol{x}}_{k|k}$ represents the overall state estimate.

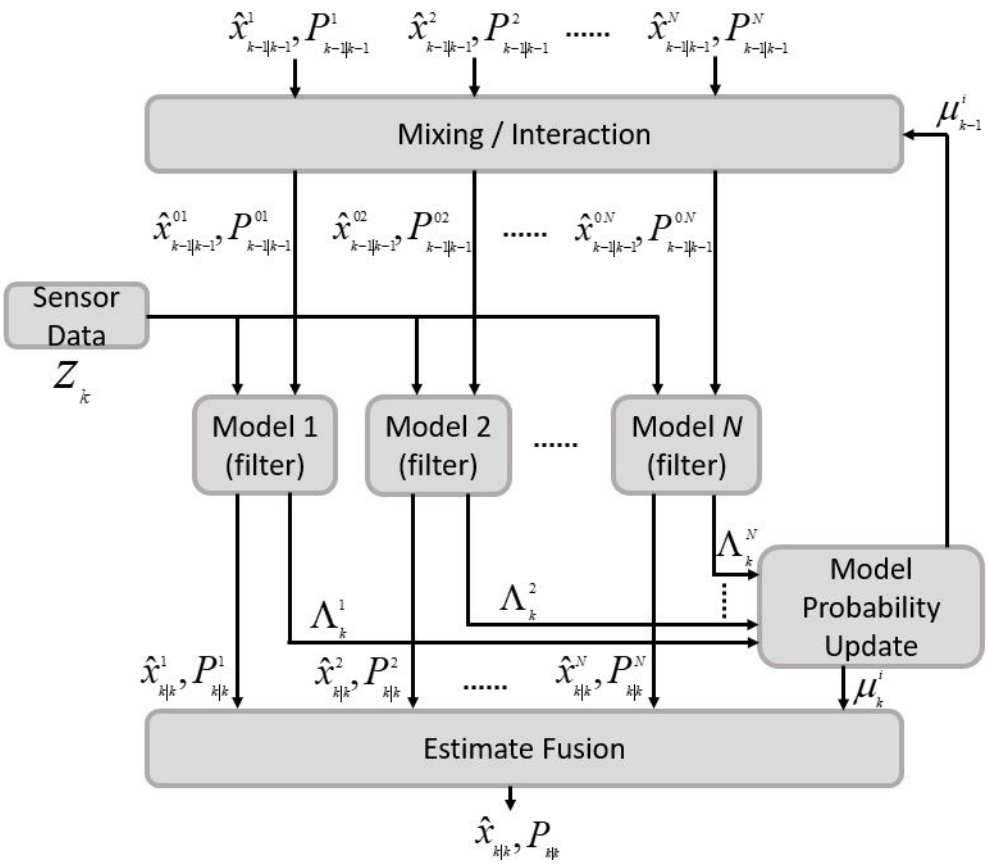


Figure 2.5: The Block Diagram of the IMM Approach.

2.2 Remaining Useful Life Estimation

The second direction of our research is based on finding an accurate framework to estimate the RUL, which means we are moving from a stage that deals with the past and current states of the system that is the diagnostic stage, to a different stage that able to predict the future health states of the system. This shift represents the new stage, which is the prognosis that fosters transition from a strict reliance on the scheduled activities to accurate prediction of future failure occurrences. Though diagnostic exhibits a retrospective nature, a predictive methodology is Prognostics.

Prognostics is used to predict the likelihood of future failures and provide early warnings by determining the failure patterns and factors that could affect industrial operations [129]. In other words, it is used to estimate the future health states of the system, often with a temporal estimation of the time of when the system will no longer be operational. This implies that the field of prognostic is not only interested in predicting the effects of known failure modes on asset life but also how these may initiate

other failure modes. Prognostics has the potential to deliver real enhancements over more traditional maintenance approaches by adopting intelligent predictive maintenance, that can offer an invaluable competitive advantage by enabling companies to discern problems in early stages instead of following "run-until-failure" ideology. Consequently, this type of maintenance is called (predictive maintenance). Predictive maintenance is considered the new face of the PHM that provides result-oriented guidance for analytical operations and maintenance actions. Remaining useful life (RUL) is the key metric for predictive maintenance solutions. In order to build an effective maintenance strategy, maximize machine uptime, and minimize maintenance costs an accurate RUL prediction is considered a substantial task. Therefore, existing RUL prediction solutions need to be continually developed and strengthened.

Remaining useful life estimation (RUL) is the length of time a machine is probable to function before repair or replacement is required. in other words, the residual lifetime during which a device can perform its intended function [6]. It has many synonyms from different fields of research such as residual life, remanent life, time to failure, etc.

Here, one important aspect to be highlighted, the prognostic techniques, in general, are in two directions, either based on predicting the RUL or predicting the health state of the system/component. The emphasis in our thesis will be on predictive RUL techniques, however, the importance of the RUL prediction over the health state evaluation will be discussed as follows. RUL prediction is a more Informative type of prediction, which is sometimes related to but still different from health state prediction, where in some cases RUL prediction needs to predict the health state. The key concept of the health state prediction is based on assigning a specific value as an indicator (Health Index HI, or threshold value) to distinguish if the system is healthy or not [130]. The HI is extracted from the raw condition monitoring data, which can be a unique system/component feature or a combination of features [131]. There is no doubt that RUL prediction provides intensive knowledge of the future condition of a particular asset for predictive maintenance purposes than the predicted health state that conveys ambiguous details about the severity of the condition.

As shown in Fig. 2.6, that system (a) is closer to the threshold value than system (b), so by adopting the health state prediction mechanism, we can tell that system (a) is going to be in an unhealthy state sooner than system (b), and as a result, will fail faster than system (b), but what if the degradation rate in the system (b)

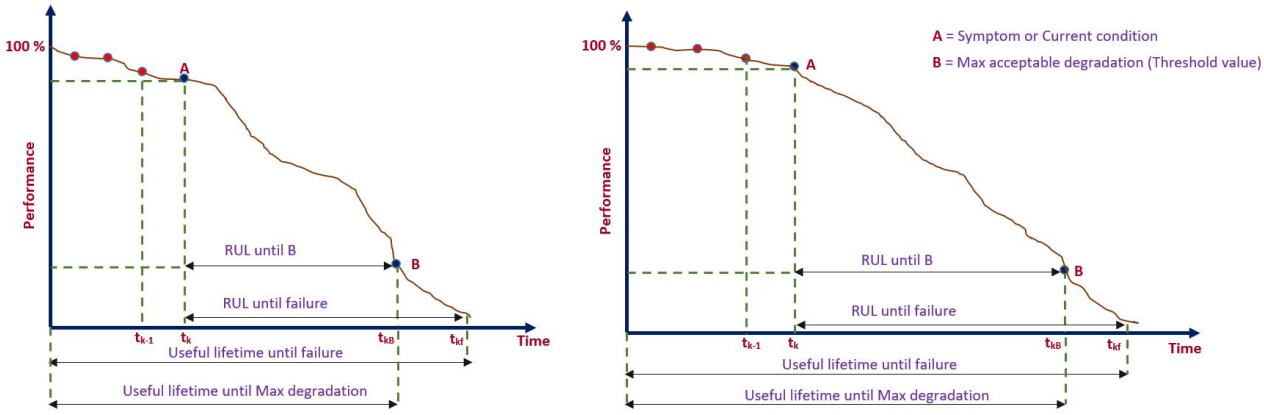


Figure 2.6: RUL prediction more informative than Health state prediction.

is faster than the one in (a), then the obtained conclusion from the HI method was misleading. On the opposite, the RUL prediction always considering the progression of the degradation pattern, offers a more direct estimator, which can be used more effectively for decision-making. Succinctly, the health state prediction method does not have the potential to transform the maintenance management concept from the conventional perspective of being reactive to being predictive. Therefore, adopting prognostic methods based on the RUL prediction is essential to distinguish the PHM from the traditional maintenance approaches.

2.2.1 Classification of RUL prediction approaches

RUL estimation is very essential in a variety of engineering industries, including aerospace, medical instrumentation, civil infrastructure, automobiles, and power plants.

Many tools and methods have been introduced for failure prognostic and RUL estimation [6–11], and It seems that the prognostic techniques usually vary based on the type of the considered application, whereas the implemented tools count primarily on the type of the of available data and knowledge. Moreover, these methods and tools can be classified into three categories; (i) Physics-based approaches; (ii) Data-driven approaches, and; (iii) Hybrid approaches (multiple-model) [12].

2.2.1.1 Physics-based Category

It involves the development of a dynamic model representing the behavior of the system and incorporating the degradation mechanism of the monitored system to

identify model parameters and to predict the RUL [12]. Many methods have been introduced based on this category [132–136]. The solutions in this category are specific to a particular industrial system under consideration and not easily extendible to other systems due to their dependence on the behavior of the specific considered system. They also require a solid understanding of the physical mechanism of failure, comprehensive experimentation, specialist knowledge, and model verification, which may be challenging and sometimes impossible for complex systems [137]. However, model based methods are the most common methods as they are very reliable once the model is built [138].

2.2.1.2 Data-driven Category

Data-driven approaches model the degradation characteristics based on historical measurements obtained from sensors embedded in the manufacturing systems and make predictions based on the learned models. So, these data-driven methods tend to extract machinery degradation processes from measuring signals rather than constructing physical models based on that need human expertise. Such approaches use models of artificial intelligence and machine learning to characterize the degradation conduct of the monitored systems or components. These models can access a broad range of data types and exploit variations in the data that cannot be detected by models based approaches. The key assumption of data-driven methods is the availability of run-to-failure data [139], for this reason, the prediction accuracy of these methods depends on the quality and the quantity of the used dataset. Data-driven prediction methods usually require two steps: training and predicting [140], during the first step the predictor is trained based on a common training strategy considering the recurrence relationship between the input variables and the target value. After the training step, the predictor is assumed to have learned the degradation behavior, in order to estimate the RUL.

Many data-driven algorithms have been proposed in recent years and good prognostic results have been achieved such as the artificial neural network (ANN), hidden Markov models (HMMs), support vector machines (SVM), relevance vector machines (RVM), and neuro-fuzzy systems (NFs), to name but a few [139, 141–143].

2.2.1.3 Artificial Neural Networks (ANNs)

Artificial Neural Networks (ANNs) are nonlinear mapping mechanisms constructed based on human brain functions, with a certain number of central processing units known as neurons, interconnected through unidirectional signal channels known as connecting weights [144]. The neural network is a machine learning approach (ML) that presents an important and useful alternative to traditional methods as they can handle the most complex situations that are not well specified for executing deterministic algorithms. Artificial neural networks provide an impressive mathematical mechanism to tackle nonlinear issues [145], and that due to its essential property that allows the approximation of any continuous non-linear relationship using a neural network with appropriate architecture and weight parameters. Furthermore, the artificial neural networks have many attractive properties such as, the ability of learning functional dependencies of data, the self-learning ability to internally mapping the functional relationships that represent the process, having high computation rates, and large input error tolerance, in addition to the ability of filtering out the noise, and handling correlations as well [144–146]. Neural networks have been implemented in a wide variety of fields including aerospace, manufacturing, engineering, defense, medical, oil, and gas industry, finance, securities, transportation, telecommunications, environment, and more importantly in fault diagnosis and prognosis fields to overcome modeling and classification problems [140, 145, 146].

One of the most remarkable trends in the world of machine learning is the precipitous growth of what has been called "Deep Learning". Deep learning has triggered a revolution in the study and applications of neural networks. As deep learning is a cluster based on different architectures of ANNs, then any multilayer artificial neural networks could be an example of deep learning and we call it deep neural networks (DNNs), where each layer can handle complex operations such as representation and abstraction that make sense of sound, images, and text. Then, the depth indicates the number of used layers in that network, that is why we used the term 'deep learning' because the neural networks have multiple (deep) layers which allow learning [147]. The deep learning approaches have many advantages over the traditional ML in terms of feature learning, model construction, and model training [148]. Deep learning allows for the automated processing of data through extremely nonlinear and complex abstraction of features across a cascade of multiple layers to find the complicated inherent structures, rather than handcrafting the optimal representation of data with

domain knowledge. Deep learning allows machines to solve complex problems even though they use very diverse, interconnected, and unstructured data, so, the more the model is learning, the better is achieving. It is important to note, that the ML has many different types of learning, but they can generally be divided into four groups according to their intended purpose:

(i) Supervised learning: This technique is used when the actual output with information is available, where the learning algorithm receives a set of inputs besides the desired outputs. This output can be discrete/categorical (a specific color, an animal picture, a car model,..etc.), or even real value. The learning process comes from comparing the actual output with the current output to find errors and then modifies the model accordingly by implementing different methods such as classification and regression [149]. All the proposed models in our thesis belong to this category.

(ii) Unsupervised Learning: It is a way of extracting all the valuable information that a specific dataset has for further processing and analysis, without having any supervision from domain experts or ground truth information (labels). Such a method has two directions; either by discovering interesting patterns using cluster analysis or by finding out some very useful relationships between parameters of a large dataset using association analysis [150].

(iii) Semi-supervised Learning: This method used whenever we have a mix of labeled and unlabeled data for training, usually a limited amount of labeled data with a large volume of unlabeled data, as the unlabeled data is inexpensive compared with the labeled one [149]. This type of learning can be used for the same applications of the supervised type, also it is possible to utilize the unsupervised technique for predicting the labels, in order to use it with the supervised method. This method is particularly appropriate for image datasets where not all images are usually labeled [151].

(iv) Reinforcement Learning: The reinforcement learning structure is based on an agent that learns by interacting with its environment, where the agent needs to choose actions to increase the expected reward over a specified period. By following a good policy the agent can hit the target much faster, therefore, in this type of learning the goal is to identify the optimal policy. The main components in this type of learning are: The agent or the learner, the environment, and the actions of the agent [149]. The reinforcement learning is specific to particular

problems such as gaming, robotics, and navigation. In addition, it follows either one of the following strategies, value function methods, or policy search methods, or both to resolve the reinforcement tasks [152].

A broad range of structures was built based on the ANN notion, which differs in architecture nature, data processing (input-output), and learning process. Among many, one can distinguish the most popular techniques, which were the main pillars of deep learning that is the new cluster of ANNs. These techniques are:

(i) Multilayer Perceptron Networks (MLP).
(ii) Convolutional Neural Networks (CNN).
(iii) Recurrent Neural Networks (RNN).

Deep learning techniques have shown superior performance to tackle complex prognostic issues with many systems whose degradation processes are tough to be interrelated by means of other methodologies. These architectures are designed to model high level representations of data and predict/classify patterns stacking multiple layers of information processing modules in hierarchical structures [153]. In addition, it was applied to hundreds of problems, within manufacturing and industrial systems and has exceeded expectations in terms of performance and the distinguished results in the application of prognostic health management [154]. In particular fault detection, diagnosis, and RUL prediction for many systems such as high speed CNC machine, induction motor, gearboxes, air compressor, and aircraft engine to name a few. The following subsections will provide an overview of the aforementioned architectures.

2.2.1.4 Multilayer Perceptron Networks (MLP)

MLP is among the most widely used neural networks, as it forms the basis for all neural networks [155]. Typically, it used either as a part of more advanced and complex neural architectures, to be the last layers of the CNN or RNN, or it used as a stand-alone model [156], as we will see. MLP is a feed forward artificial neural network architecture, as it does not contain any cycles and the performance of the network (Output) depends only on the present input instance [157]. It is composed of three main parts: an input layer, an intermediate layer (one or more), and an output layer, where each layer is fully connected to the following layer of nodes, in other words, this multi-layered perceptron consist of interconnected neurons that transmit information among themselves, similar to the human brain [155], and each layer is

connected with the adjacent layer by a set of connections, each connection is equipped by a weight. Fig. 2.7 shows the structure of an MLP with three layers, i.e., input layer, one hidden layer, and the output layer, where, $\boldsymbol{x} \in \mathbb{R}^k$ is the input vector that is directly moved to the next layer which is the first hidden layer $\boldsymbol{h} \in \mathbb{R}^n$, and then producing the output where $\boldsymbol{y}_o \in \mathbb{R}^o$. Then the output from each layer can be calculated as follows:

$$\boldsymbol{h} = \varphi^{(1)}(\boldsymbol{W}_1\boldsymbol{x} + \boldsymbol{b}_1) \tag{2.2.1}$$

$$\boldsymbol{y}_o = \varphi^{(o)}(\boldsymbol{W}_o\boldsymbol{h} + \boldsymbol{b}_o), \tag{2.2.2}$$

where $\boldsymbol{W}_1 \in \mathbb{R}^{n\times k}$, $\boldsymbol{W}_o \in \mathbb{R}^{o\times n}$, $\boldsymbol{b}_1 \in \mathbb{R}^n$, and $\boldsymbol{b}_o \in \mathbb{R}^o$ are the weights and corresponding bias of each layer, respectively. $\varphi(\cdot)$ denotes the activation function.

Activation functions are mathematical equations that nonlinearly describe the relations between input and output. The activation function has a role in each neuron, by identifying whether to consider this neuron as activated "fired" or not, and this is dependent on whether the input of each neuron is significant for the model's prediction or not, by determining the weighted sum and adding bias to it [158], which will add non-linearity to the neuron's output. This is an important property as most of the real world data have nonlinear nature. The activation functions can be as basic as a step function that controls the neuron output to be on and off based on specific rules or limits. On the other hand, they can be non-linear activation functions, which can enable the network to handle data that are more complex, learn, and compute almost any data-related feature and deliver precise predictions [158].

There are some common and famous activation functions such as:

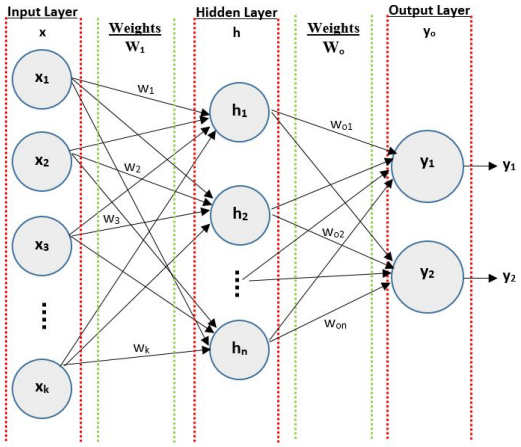
(i) Sigmoid or Logistic function: It is one of the most popular activation functions that has the following form

$$sigmoid(x) = \frac{1}{1+e^{-x}} \tag{2.2.3}$$

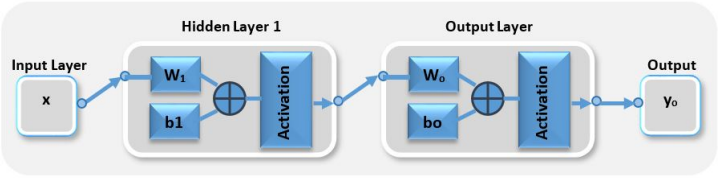
It takes real input values that are in the domain of $\mathbb{R}$ and transforms (normalizes) them to outputs in a range (0,1).

(ii) Hyperbolic Tangent function (tanh): Another activation function that is quite similar to the previous one, however it has the following form

$$tanh(x) = \frac{1-e^{-2x}}{1+e^{-2x}} \tag{2.2.4}$$



(a)



(b)

Figure 2.7: The structure of an MLP with 3 layers.

and its output between -1 and 1 which means it is a zero centered, thats why it is often preferred over Sigmoid function.

(iii) Rectified Linear units (ReLu): It is the most important and popular activation function that takes real input values $\mathbb{R}$ and then transforms the negative values to zero, while the positive values rise linearly, where the function has the following form

$$ReLu(x) = max(0, x) \tag{2.2.5}$$

MLP often applied to supervised learning models, which employ the back-propagation method to train the network. During the training, there is always an error that represents the difference between the outputs and the expected values that are already known. The function that quantifies this error known as the cost function. Minimizing this function is the target of supervised learning in order to optimize the correlation of the model to the system that it is seeking to represent [159]. The backpropagation algorithm is based on using a technique called gradient descent (or other techniques that have the same objective) to determine the minimum value of the cost function

within weight space [160]. The weights that achieve this target will be considered as the optimal solution to the current learning problem. To be more specific the idea of the gradient descent is computing the derivative of the cost function with respect to the weights in the network as follows

$$\frac{\partial Error}{\partial \boldsymbol{W}_1} = \frac{\partial Error}{\partial \boldsymbol{y}_o} \frac{\partial \boldsymbol{y}_o}{\partial \boldsymbol{W}_o} \frac{\partial \boldsymbol{W}_o}{\partial \boldsymbol{h}} \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{W}_1}, \tag{2.2.6}$$

where the term *Error* represents the error measured by the cost function. In other words, the concept is simple: change the weights and biases across the network to get the necessary output in the output layer. For more details about the backpropagation algorithm see [159, 160].

MLP has been used in a broad range of fields, like Handwritten Recognition [161], classification of healthcare data [162], Index of Industrial Production [163], Image Classifications [164], and Stock market analysis [165], to name just a few. In PHM applications, MLP has shown an impressive success, and a wide range of approaches have been proposed such as Huang *et al.* [166] proposed an integrated approach for investigating the whole life cycle of ball bearing. The minimum quantization error has been used as a degradation indicator and then monitoring the degradation period which is defined by fluctuating signal that rises from the beginning of the defect till the failure of the component, and then based on the NN and the weight application to failure times, the remaining useful life has been predicted. Kim *et al.* [167] presented an approach to estimate the state of health of Lithium-ion battery using the MLP network. Jedlinski *et al.* [168] utilized the MLP network to evaluate the technical condition of a gearbox and to detect the fault as early as possible. Hu*et al.* [169] employed the MLP approach to diagnosing the amount as well as the location of mass imbalance on aircraft engines, and the results were superior compared with other solutions. Almeida*et al.* [170] proposed an architecture based on MLP network to handle the basic classification job besides the fault identification, by utilizing the features of vibration signals in time-domain for bearings in both normal and defective cases. Geramifard *et al.* [171] used the MLP network for monitoring and predicting the health condition of a cutter, more specifically the wearing status in terms of the features or measured data. Loboda *et al.* [172] utilized the MLP to propose a diagnostic technique for fault identification and classification in a real gas turbine. Zolfaghari *et al.* [173] Integrated the wavelet analysis and the MLP network to propose an intelligent approach that combines the strength of both frequency domain analysis

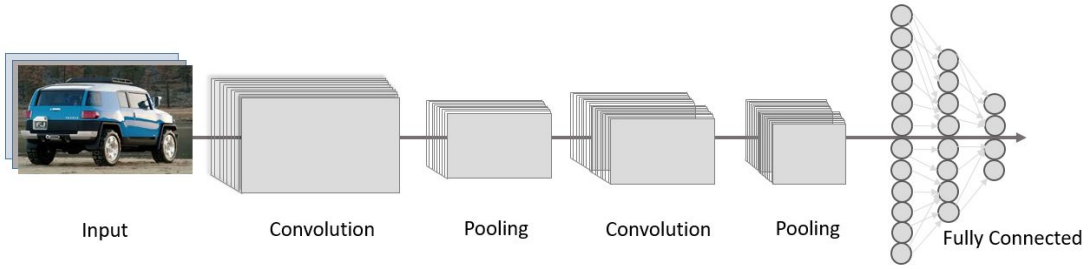


Figure 2.8: The CNN Architecture.

and time scale, for automatic diagnosis of fault severity during the life of a motor. Heidari *et al.* [174] proposed an approach by combining the MLP, wavelet support vector machine, and continuous wavelet for diagnosing different types of fault in the gearbox.

2.2.1.5 Convolutional Neural Networks (CNN)

The basic concepts of CNNs were firstly proposed by LeCun [175]. It is a multi-stage neural network, which is composed of two stages: (i) a feature extractor module that is composed of input and convolutional layers, in addition to the pooling layers, and; (ii) fully connected layers to perform the classification task [176] as in Fig. 2.8. It is a powerful feature extraction mechanism widely applied in many industrial and research fields, as it has an exceptional ability to capture the spatial and temporal dependencies. CNNs were originally designed to be used in image processing and computer vision, where the input usually organized as a two-dimensional (2D) array. Every image is represented by a matrix of pixel values at the coordinate indices x and y (horizontal and vertical), while for color images an additional 'depth' field to be added to the input data, changing the input to be in (3D) [177]. The raw pixel data of an image are the input to the CNN, which then learns how to identify useful observation or a specific pattern (The features) from the input and finally, conclude what object they represent. As earlier stated that CNN has two stages, consequently, the input has to pass through different steps to reach the output stage.

Convolution layer: It is a core component of the features extraction stage in the CNN architecture. Which usually consisting of an integration of linear and nonlinear processes, that is the convolution and activation function [178].

Convolution: It is a special type of linear mathematical operation used to combine two information sets, which are the input image matrix (input feature map) and the

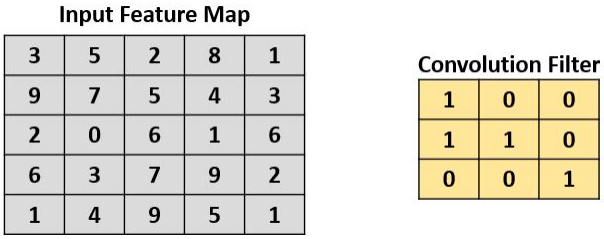


Figure 2.9: The 5×5 Input image matrix, and the 3×3 Filter matrix .

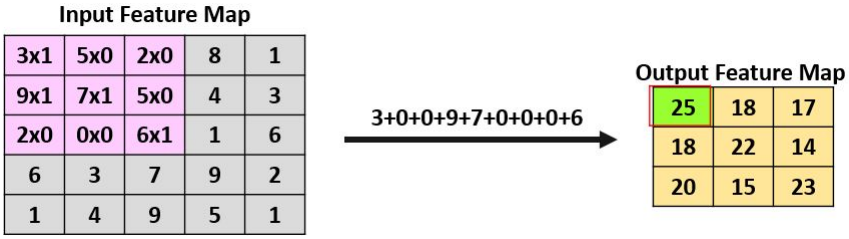


Figure 2.10: The 3×3 Convolution process.

convolution filter (also called a kernel). To show the convolution process, the following example is given. Consider (5×5) as an input image matrix, and filter matrix of (3×3), as shown in Fig. 2.9. The convolution process is achieved by sliding the filter over the input matrix, and the element-wise product is calculated at each position and then sum the result to get a single value that will represent an element in the new matrix that is the output feature map as shown in Fig. 2.10. It can be clearly seen that the most important hyperparameters that identify the convolution operation are two; The size and the number of the used kernels.

From this example, we can see that every convolution layer implicitly includes several filters trained to recognize a particular form of features. Each filter is applied sequentially to all suitable locations where it overlaps entirely with the input image resulting in a smaller feature map that defines the input regions where a particular feature has been observed. After implementing multiple filters, we end with a number of feature maps that are then passed to the next layer. Stacked convolutional layers are used (also pooling layers can be used, as we will see) for size reduction of the extracted feature maps. It is important to highlight that the first convolutional layers recognize the simple or the basic features, whereas the layers that are deeper have the ability to handle more complex patterns and structures [179]. Each layer can be seen as a new interpretation of the input image, represented in terms of patterns that are broader and even more abstract than those in the previous layer. Passing the Input image

through a sequence of layers, in this regard, it is similar to the MLP network, but the layer's structure is quite different, as the MLP uses fully connected layers where the output vector's elements are dependent on the input vector's elements. While CNNs use convolutional layers that benefit from the spatial locality, where every output element belongs to a small image region and depends only on the input values from this region. As a result, the number of parameters that describe each layer will be greatly reduced. Furthermore, the convolutional layers presume that the parameters for each local area of the image are the same, which renders the number of layer parameters independent from the image size. What it really needs to learn is a single kernel that determines how output features are measured from any local area of the image [179]. This local area is usually very small, maybe around 5 by 5 pixels [180], then the number of parameters to be learned is 25 times the output features' number for each area. This is a small number comparing to the fully connected Layer which makes CNNs significantly easier and smoother to train than MLPs.

Nonlinear activation function: The CNN applies a nonlinear activation function (sigmoid, tanh, and ReLu) after each convolution operation for introducing a nonlinearity into the model.

Pooling layer: The primary goal of the pooling layer is to progressively reduce the size of the feature maps by selecting certain features through a summarized version of the detected features. In a word, the CNN in this stage is down-sampling the convolved features. As a result, the pooling leads to reducing the number of model parameters, simplifying the computational complexity of the network, and avoiding the over-fitting issue [154]. The most common types of pooling operations are max pooling, average pooling, and sum Pooling. Just like the process of convolution conducted above, the pooling layer takes a sliding window through the data, which turns the values into representative values, either by taking the maximum value, the average of the values, or the summation of the values, from the observed values in the window. It is worth noting that in any of the pooling layers there is no learnable parameter [154].

Fully connected layer (FC): The last stage of the CNN network is the FC layers (dense layers) that received the flattened (converted from matrix to vector) output feature maps from the last convolution or pooling layer. The FC layers then use an activation function to construct the desired outputs either through classification or regression task.

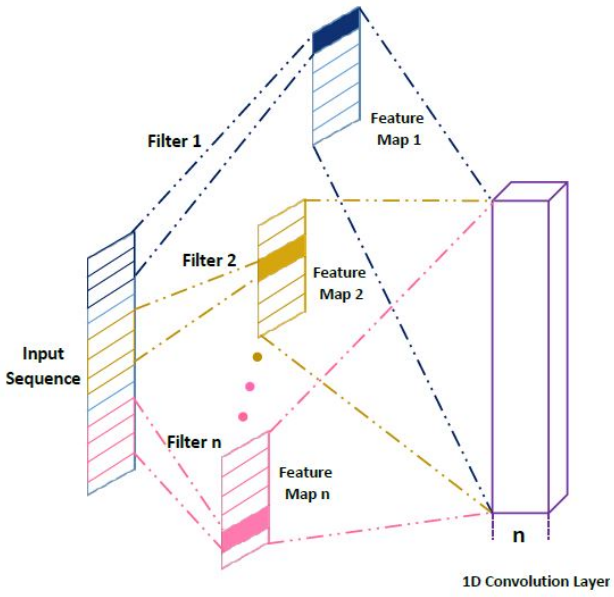


Figure 2.11: Illustration of the 1D convolution operation utilized in the CNN.

2.2.2 One Dimensional Convolution

For the proposed RUL approaches in our research, the time sequence and the selected number of features represent the 2D structure of the input data for the CNN. However, as the extracted features are collected from many different sensors in the prognostic problem of our research, there exists an unnoticeable relationship among spatially distributed features obtained from the samples [181]. Thus, although the input and the associated feature maps have 2D, the designed convolution filters in our proposed models are in the 1D format as shown in Fig. 2.11. Let the vector x^t represent the 1D input sequence at time t, the $x^t_{l:l+D-1}$ represents the concatenated vector, and $\boldsymbol{w} \in \mathbb{R}^{D\times 1}$ denotes a kernel with D size, then the convolution operation is given by

$$x^t_{l:l+D-1} = x^t_l \oplus x^t_{l+1} \oplus \ldots \oplus x^t_{l+D-1}, \tag{2.2.7}$$

where $x^t_{l:l+D-1}$ represents a window of length D that starts from the l^{th} point and $\oplus$ concatenates each data sample into a vector. The convolution operation is defined by

$$z_l = \varphi(\mathbf{w}^\top x^t_{l:l+D-1} + b), \tag{2.2.8}$$

where superscript $\top$ denotes the transpose operator and $\{\boldsymbol{b}, \varphi\}$ are, respectively, the bias and non-linear activation function. Consider z_l representing the learned feature

of the kernel $\boldsymbol{w}$ on the sub-sequence $x^t_{l:l+D-1}$, then by sliding the filtering window from the first point to the last point in the sample data, the feature map of the j^{th} kernel can be captured and is expressed as follows

$$z^t_j = [z^1_j, z^2_j, \ldots, z^{L-D+1}_j]^\top. \tag{2.2.9}$$

For the pooling layer, it operates independently on each feature map z^t_j and resizes it spatially using a specific operation such as (max, average, and summation). As an example, for max pooling, if k is the filter size then the output feature max pooling vector is given by

$$p = [p_1, p_2, \ldots, p_{\frac{L-D+1}{k}}], \tag{2.2.10}$$

where $p_i = \max(z_{ki-(k-1)}, \ldots, z_{(ki-2)}, z_{ki-1}, z_{ki})$. The Outstanding capability of the CNN in identifying spatial and temporal dependencies, made the CNN as one of the most powerful feature extraction tool, and it has been effectively utilized in a broad range of applications such as, computer vision [182], natural language processing [183], biomedical applications [184], speech recognition [185], face recognition [186], visual tracking [187] and health informatics [188], to mention just a few.

CNN has been impressively effective in PHM applications and a wide variety of solutions have been suggested. For instance Chen *et al.* [189] has introduced a deep learning technique based on CNN for fault pattern diagnosis of gearboxes, where the feature representations are selected as the input parameters of CNN. Similarly, Lee *et al.* [190] used CNN to achieve high classification accuracy in bearing fault detection on a signal dataset consisting of univariate and bivariate time series. Wang *et al.* [191] proposed a wavelet-based CNN for machinery fault diagnosis. Babu *et al.* [192] proposed a deep CNN based regression approach for estimating the RUL, where time window technique is employed for sample preparation in order to provide better feature extraction by CNN, and then the associated RUL value is estimated based on the learned representations. Sun *et al.* [193] developed a Convolutional discriminative feature learning method (CDFL) for induction motor fault diagnosis by using an unsupervised CNN to extract features directly from raw data to characterize different working conditions, followed by a support vector machine (SVM) classifier to classify the learned features for induction motor fault diagnosis. Li *et al.* [156] introduced a prognostic approach based on CNN, where time window strategy is

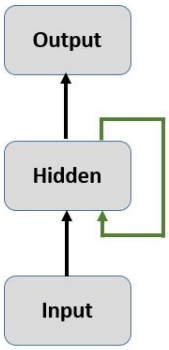


Figure 2.12: The Recurrent Neural Networks Concept.

adopted for the data preparation to ensure better feature extraction by deep CNN. In addition, the Dropout technique is used to avoid the overfitting problem. Ren *et al.* [194] proposed a prediction technique for bearing RUL based on deep CNN. A new method of feature extraction named spectrum-principal-energy-vector has been used to obtain the eigenvectors, to be provided as input to the deep CNN in order to get a series of eigenvectors, then deep neural network model is used for regression prediction to obtain the RUL of the bearing. Similarly, Pham*et al.* [195] proposed a fully automatic diagnosis approach for a bearing system by employing the wavelet packet spectral subtraction for converting the vibration signals to high resolution images, and then utilize the CNN to automatically extracts informative features in order to identify the current health status of the system. Maior *et al.* [196] employed the CNN for predicting the RUL of bearings with accelerated degradation by utilizing a real vibration time-series.

2.2.2.1 Recurrent Neural Networks (RNN)

RNN is an important and prominent class of artificial neural network architectures that among the most popular techniques in sequential modeling [197]. RNNs are intended to recognize sequential data characteristics and use trends and patterns in order to predict the next probable scenario. RNNs are different from the existing feedforward neural networks approaches, an RNN has an internal memory, which makes it possible to recall historical information and interpret current events accordingly [198]. When it comes to the traditional neural networks, all inputs and outputs are considered to be independent of one another, which is not the case for RNNs,

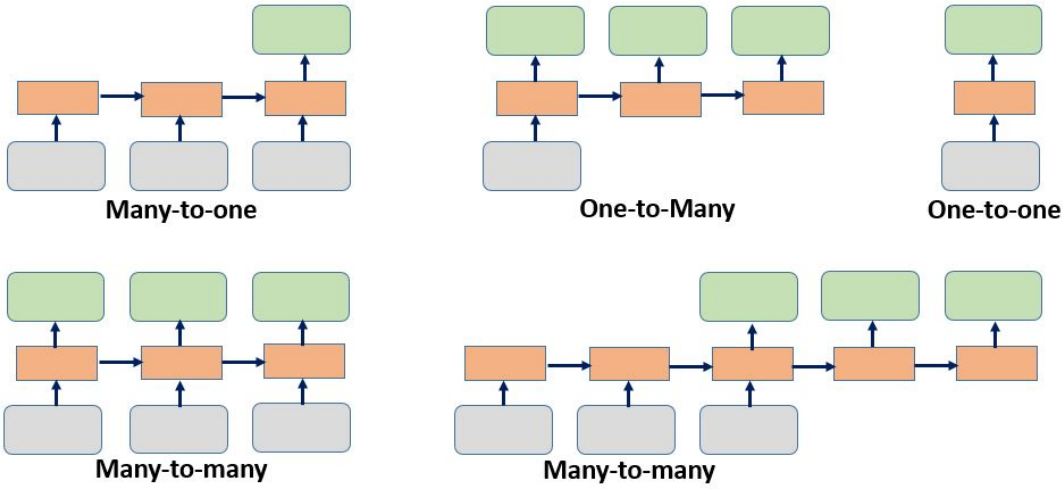


Figure 2.13: The relationship structures of input and output data.

where the term "recurrent" is used, since they conduct the same process for each sequence element, with the performance depending on the previous calculations. RNN functions on the idea of saving a specific layer's output and feeding it back into the input to predict the layer's output as in Fig. 2.12. RNNs have many structures that show the various input and output relationships. Fig. 2.13 illustrates five different styles for input and output relationships. The followings are the main three different categories [198]:

- Many-to-one: When the input data is a sequence, while the output is not a sequence but a vector of a fixed dimension. The sentiment classification is the typical example for many-to-one, where the input is text based sequence and the output is a specific label, such as positive or negative, etc.
- One-to-many: When the output is a sequence but the input is not. Image captioning is an example of this category, by inputting an image and receiving a sentence of words at the output.
- Many-to-many: This category can also be divided according to whether or not the input and output are synchronized, where both the input and output are sequences. The video classification task is an example of a synchronized many-to-many. While translating a language to another one is an example of the unsynchronized many-to-many.

Although the RNNs have demonstrated their impressive performance in many fields, these networks have their limitations such as, not considering any future contribution to the current state. That was the reason behind the improved version of the RNN

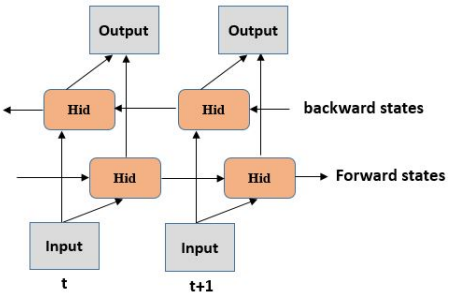


Figure 2.14: The Bidirectional Recurrent Neural Networks.

which is the bidirectional recurrent neural networks, that utilizes two time directions at the same time, input data from the past and future of the current timeline to calculate the same output [199], where the first direction is the forward states and the second one is the backward states as shown in Fig. 2.14.

In addition to that, the RNNs suffer from other main problems which are the vanishing and exploding gradients. As the RNNs are also used backpropagation during the training (learning), where the weight matrices are adjusted using the gradient. Throughout the backpropagation process, gradients are determined via continuous multiplications of derivatives (as stated earlier) and there is a high possibility that these derivative values become very small as we are going through the network, as a result, the gradient becomes smaller and smaller till the stage when it is almost zero when we call it the "vanish" stage, or "vanishing gradient problem" [197]. The gradients hold information that is used in updating the RNN parameters (weights and biases) and when the gradient is very small, updating the parameter is negligible which means that there is no real learning is performed. In particular, the initial layers' parameters (weights and biases) will not be effectively updated in every training session (as the gradient value is vanishing), and as these initial layers are essential for the identification of the core elements of the input data, this may contribute to the complete inaccuracy of the entire network [197, 198]. On the contrary, the exploding gradients happened when the large error gradients accumulate and resulting in really large updates to the weights of the neural network model during the training. The training process is perfectly working when those updates are small but controlled, Otherwise, it may lead to poor predictive results or even a model that doesn't reveal anything useful what ever [197, 198]. Researchers have built more sophisticated types of RNNs over the years to resolve some of the mentioned limitations of the RNN model, these types are:

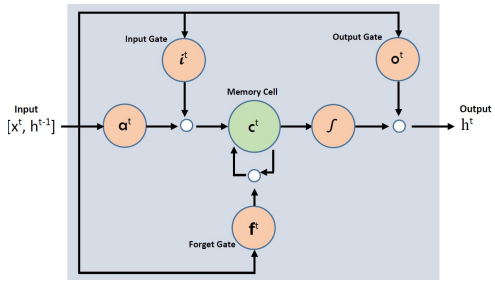


Figure 2.15: Block diagram of LSTM.

(i) The Long Short-Term Memory (LSTM).
(ii) The Bidirectional Long Short-Term Memory (BLSTM).
(iii) The Gated Recurrent Unit (GRU).
(iv) The Bidirectional Gated Recurrent Unit (BGRU).

2.2.2.2 The Long Short-Term Memory (LSTM)

The Long Short-Term Memory Networks (LSTMs) are considered as the state-of-the-art technique for sequence learning. The LSTM was first introduced by Hochreiter and Schmidhuber and were further refined and popularized in the following years by different works such as References [200] and [201]. The LSTMs as a special form of RNN, it has been designed to overcome the long-term dependency, in addition to the vanishing and exploding gradient problems of the traditional RNNs. This was achieved through a mechanism known as cell states, that is built on a gating system to provide a memory-based structure. The latter is used to control reading, writing, and removing (forget) the written information from the memory state [202].

The central idea of LSTM is based on employing the cell state and the gating system. In principle, the cell state will hold relevant information during the sequence processing. Thus even details from earlier time steps will make it possible to take later steps, minimizing the impact of short-term memory. Adding or removing the information from the cell state is controlled by the gating system. This gating system includes 3 types of gates that during training will learn what information is important to keep or forget. These gates are, a forget gate, input gate, and output gate [203].

Fig. 2.15 presents the LSTM architecture with $\boldsymbol{h}^t \in \mathbb{R}^M$ and $\boldsymbol{c}^t \in \mathbb{R}^M$ representing its hidden state vector also known as output vector of the LSTM unit, and the cell state vector at time t, while M denotes the number of nodes (hidden units).

The input to LSTM will be the sensor data x^t, in addition to $\boldsymbol{h}^{t-1}$ and $\boldsymbol{c}^{t-1}$ coming

from time $(t-1)$. The information flow of the internal cell unit is controlled by the gating system which is an internal mechanism working based on the following gates:

(i) Forget gate: A forget gate denoted by $\boldsymbol{f}^t \in \mathbb{R}^M$, that decides on the contents to be maintained or forgotten [204]. The current input and the previous hidden state are the input to this gate then multiplied by the weight matrices and a bias is added, after this the result passed through the sigmoid function that generates a vector with values between 0 and 1, corresponding to every number in the cell state. Basically, it is the responsibility of the sigmoid function to decide which values to hold and which ones to throw away. So, The closest to 0 means forgetting, and the closest to 1 means holding.

(ii) Input Gate: An input gate $\boldsymbol{i}^t \in \mathbb{R}^M$, that controls the cell state updating procedure based on $\boldsymbol{h}^{t-1}$ and x^t [204]. Similar to the forget gate, the current input, and the previous hidden state are input to the sigmoid function for deciding the important (1) and the unimportant once (0). In addition, the current input and the previously hidden state pass into *tanh* function (output values from -1 to +1) for creating a vector with all possible addable values to the cell state. Then by multiplying both sigmoid and *tanh* outputs, the sigmoid will finally decide which information to keep from the output of the *tanh*, and finally, these useful updates will be added to the cell state.

(iii) Output Gate: An output gate denoted by $\boldsymbol{o}^t \in \mathbb{R}^M$, that computes the next value of the hidden state [204]. Using *tanh* function on the cell state for creating a vector with values between -1 and +1, then the current input and the previous hidden state are input to the sigmoid function for regulating the output values of the vector (from the *tanh*). Then by multiplying the previous output to the vector (resulted from *tanh*), will get the output and the hidden state for the next cell.

At each step t, the LSTM cells are implemented based on the following set of equations [205]:

$$\boldsymbol{i}^t = \sigma(W_i x^t + U_i \boldsymbol{h}^{t-1} + \boldsymbol{b}_i), \tag{2.2.11}$$

$$\boldsymbol{o}^t = \sigma(W_o x^t + U_o \boldsymbol{h}^{t-1} + \boldsymbol{b}_o), \tag{2.2.12}$$

$$\boldsymbol{f}^t = \sigma(W_f x^t + U_f \boldsymbol{h}^{t-1} + \boldsymbol{b}_f), \tag{2.2.13}$$

$$\boldsymbol{a}^t = \tanh(W_c x^t + U_c \boldsymbol{h}^{t-1} + \boldsymbol{b}_c), \tag{2.2.14}$$

$$\boldsymbol{c}^t = \boldsymbol{f}^t \circ \boldsymbol{c}^{t-1} + \boldsymbol{i}^t \circ \boldsymbol{a}^t, \tag{2.2.15}$$

$$\boldsymbol{h}^t = \boldsymbol{o}^t \circ \tanh(\boldsymbol{c}^t), \tag{2.2.16}$$

where terms W_i, W_o, W_f, and $W_c \in \mathbb{R}^{M \times L}$ together with terms U_i, U_o, U_f, $U_c \in \mathbb{R}^{M \times M}$ constitute the weight matrices; Terms $\boldsymbol{b}_i$, $\boldsymbol{b}_o$, $\boldsymbol{b}_f$, $\boldsymbol{b}_c \in \mathbb{R}^M$ represent biases; L represents the number of input features; Term $\sigma(\cdot)$ denotes the sigmoid non-linear function; Operator "$\circ$" represents an entry-wise product operation, which is performed by element-wise multiplication of two vectors, and; finally, $\tanh(\cdot)$ denotes the activation function. The LSTMs have been successfully applied in wide range of applications, including speech recognition [206,207], natural language processing [208–210], human action recognition [211], handwriting recognition [212], and image captioning [213] to name a few.

The LSTM was remarkably successful in PHM applications and its fields, thus, a wide range of approaches have been proposed. For example: Zheng *et al.* [214] have introduced LSTM based model for RUL estimation, which utilizes multiple layers of LSTM cells along with standard feed forward layers to detect hidden patterns and learn complex features within the sensor and operational data with multiple operating conditions, fault, and degradation models. Wu *et al.* [215] have used vanilla LSTM neural networks, which is an effective technique in the field of natural language processing, to build a model for RUL estimation. In addition to vanilla LSTM neural networks, a dynamic difference methodology is proposed to extract new features from raw health monitoring data. Malhotra *et al.* [216] introduced the Long Short Term Memory based Encoder-Decoder (LSTM-ED) technique for estimating the unsupervised health index (HI) of a system (from which the RUL can be estimated) based on multi-sensor time-series data. The LSTM-ED is trained to reconstruct the multivariate time-series corresponding to the healthy state of a system, then the reconstruction error is used to compute the HI which is then used for RUL estimation.

Lei *et al.* [205] presented a fault diagnosis for wind turbines by employing the Long Short-term Memory (LSTM) technique to effectively learn features from multivariate time-series data then capture long-term dependencies via the recurrent actions and gate mechanism. Zhou *et al.* [217] proposed an approach for supercapacitor life prediction by utilizing the LSTM to capture long-term dependencies of a degraded supercapacitor. Liu *et al.* [218] used the LSTM to propose a method for proton exchange membrane fuel cell RUL prediction. This method is also used the locally weighted scatterplot smoothing and regular interval sampling for reconstruction and smoothing of the data. Wang *et al.* [219] extracted feature parameters from three different domains; time domain, frequency domain, and time–frequency domain, and then by selecting the parameters that could best describe the bearings degradation behavior and creating a feature set based on time factor. The LSTM employed this feature set for training and then predicting the RUL of the rolling bearing. Wu *et al.* [220] proposed a deep long short-term memory for predicting turbofan engine's RUL. The proposed approach combines time series signals from multiple sensors and identifies the hidden long-term dependencies among the sensors readings for RUL prediction. In addition, the grid search strategy has been adopted to find the best model parameters.

2.2.2.3 The Bidirectional Long Short-Term Memory (BLSTM)

To capture the temporal dependencies between extracted features and fully take advantage of the input information in the past and future of a specific time frame [199], the BLSTM is developed as a modified version of the conventional LSTM. It has been demonstrated that the bidirectional networks are significantly better than unidirectional ones in many fields [201,221]. The BLSTM comprises of two LSTM layers to be applied in both directions of the hidden sequences, i.e., forward $\overrightarrow{\boldsymbol{h}}^t$ and backward $\overleftarrow{\boldsymbol{h}}^t$, which are then joined to calculate the output sequence. Fig. 2.16 presents the block diagram of the BLSTM network. At each time step t, the BLSTM model calculates both directions ($\overrightarrow{\boldsymbol{h}}^t$, & $\overleftarrow{\boldsymbol{h}}^t$) separately, and then concatenates the outputs to form the BLSTM output denoted by $\boldsymbol{h}_{bi}^t$.

The corresponding hidden layer functions of the BLSTM architecture, which are exactly the same as in Eqs. (2.2.11)-(2.2.16) but now implemented in two different

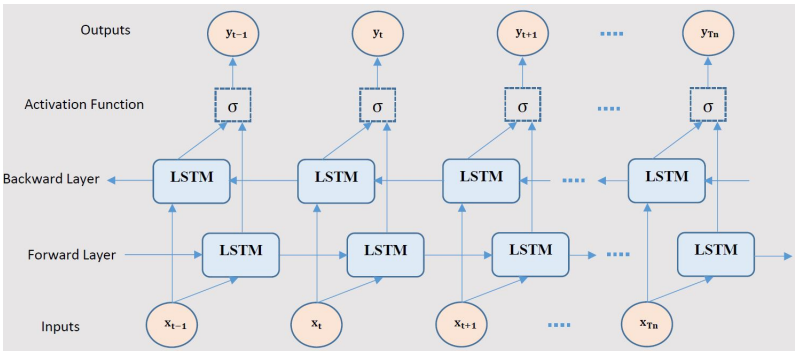


Figure 2.16: Block diagram of the BLSTM.

directions, are defined as

$$\overrightarrow{\boldsymbol{i}}^t = \sigma(\overrightarrow{W_i}\overrightarrow{x}^t + \overrightarrow{U_i}\overrightarrow{\boldsymbol{h}}^{t-1} + \overrightarrow{\boldsymbol{b}_i}), \tag{2.2.17}$$

$$\overrightarrow{\boldsymbol{o}}^t = \sigma(\overrightarrow{W_o}\overrightarrow{x}^t + \overrightarrow{U_o}\overrightarrow{\boldsymbol{h}}^{t-1} + \overrightarrow{\boldsymbol{b}_o}), \tag{2.2.18}$$

$$\overrightarrow{\boldsymbol{f}}^t = \sigma(\overrightarrow{W_f}\overrightarrow{x}^t + \overrightarrow{U_f}\overrightarrow{\boldsymbol{h}}^{t-1} + \overrightarrow{\boldsymbol{b}_f}), \tag{2.2.19}$$

$$\overrightarrow{\boldsymbol{a}}^t = \tanh(\overrightarrow{W_c}\overrightarrow{x}^t + \overrightarrow{U_c}\overrightarrow{\boldsymbol{h}}^{t-1} + \overrightarrow{\boldsymbol{b}_c}), \tag{2.2.20}$$

$$\overrightarrow{\boldsymbol{c}}^t = \overrightarrow{\boldsymbol{f}}^t \circ \overrightarrow{\boldsymbol{c}}^{t-1} + \overrightarrow{\boldsymbol{i}}^t \circ \overrightarrow{\boldsymbol{a}}^t, \tag{2.2.21}$$

$$\overrightarrow{\boldsymbol{h}}^t = \overrightarrow{\boldsymbol{o}}^t \circ \tanh(\overrightarrow{\boldsymbol{c}}^t), \tag{2.2.22}$$

and

$$\overleftarrow{\boldsymbol{i}}^t = \sigma(\overleftarrow{W_i}\overleftarrow{x}^t + \overleftarrow{U_i}\overleftarrow{\boldsymbol{h}}^{t+1} + \overleftarrow{\boldsymbol{b}_i}), \tag{2.2.23}$$

$$\overleftarrow{\boldsymbol{o}}^t = \sigma(\overleftarrow{W_o}\overleftarrow{x}^t + \overleftarrow{U_o}\overleftarrow{\boldsymbol{h}}^{t+1} + \overleftarrow{\boldsymbol{b}_o}), \tag{2.2.24}$$

$$\overleftarrow{\boldsymbol{f}}^t = \sigma(\overleftarrow{W_f}\overleftarrow{x}^t + \overleftarrow{U_f}\overleftarrow{\boldsymbol{h}}^{t+1} + \overleftarrow{\boldsymbol{b}_f}), \tag{2.2.25}$$

$$\overleftarrow{\boldsymbol{a}}^t = \tanh(\overleftarrow{W_c}\overleftarrow{x}^t + \overleftarrow{U_c}\overleftarrow{\boldsymbol{h}}^{t+1} + \overleftarrow{\boldsymbol{b}_c}), \tag{2.2.26}$$

$$\overleftarrow{\boldsymbol{c}}^t = \overleftarrow{\boldsymbol{f}}^t \circ \overleftarrow{\boldsymbol{c}}^{t+1} + \overleftarrow{\boldsymbol{i}}^t \circ \overleftarrow{\boldsymbol{a}}^t, \tag{2.2.27}$$

$$\overleftarrow{\boldsymbol{h}}^t = \overleftarrow{\boldsymbol{o}}^t \circ \tanh(\overleftarrow{\boldsymbol{c}}^t). \tag{2.2.28}$$

Then, the concatenated output vector $\boldsymbol{h}_{bi}^t$ is given by

$$\boldsymbol{h}_{bi}^t = \overrightarrow{\boldsymbol{h}}^t \oplus \overleftarrow{\boldsymbol{h}}^t. \tag{2.2.29}$$

Effectiveness of the BLSTM has been proven in many fields such as phoneme classification [201], speech recognition [221], human activity recognition [222], healthcare [223], Infrastructure quality [224]. The BLSTM has achieved noticeable success in PHM

applications and its subsequent fields, for example, Wang *et al.* [225] proposed an approach by utilizing the BLSTM network for predicting the RUL of aircraft engine based on simulator data. The model can detect hidden patterns from sensor data under different working conditions, degradation model, and fault patterns. Wilson *et al.* [226] introduced robust classifiers based on BLSTM for rapid fault detection in marine hydrokinetic turbine in order to reduce the cost of both operation and maintenance of this turbine. The proposed model has been validated using simulated time-series sensor data from the turbine simulation platform.

Li *et al.* [227] presented a new model of tool RUL prediction by employing limited data. Firstly, the process of the tool wear has been tracked using an adaptive time window, then in the second stage, a deep BLSTM to detect the past and future contexts relationships. Bian *et al.* [228] utilized the BLSTM to design a model for the state of charge estimation in lithium-ion batteries, by capturing the temporal information of the battery in both directions (forward and backward) and outline the long-term dependencies (from both past and future). Moreover, stacked layers of BLSTM empower the model to define the non-linear and dynamic relationship between the input measurements of the battery and the output state of charge on a layer by layer basis. Wang *et al.* [229] proposed a model by employing a BLSTM with an attention mechanism for predicting the voltage degradation of the Proton exchange membrane fuel cells. The model inputs have been extracted using random forest regression. Qiu *et al.* [230] proposed a fault diagnosis method for a rolling bearing by employing the BLSTM. Time-frequency feature with a combination of different wavelet packet transform, collected from the original vibration data, then by using the BLSTM that utilizes only the long-term memory to handle the rolling bearing data and then conduct the fault diagnosis. Li *et al.* [231] constructed a model of fault diagnosis for early gear pitting, by using the BLSTM that employs the raw vibration signals to extract the informative features for evaluating the faults degree of the early gear pitting. Wu *et al.* [232] introduced a model for tool wear prediction, by utilizing a singular value decomposition and BGRU. The Hankle matrix is used for reconstructing the raw cutting force signal, then the extracted signal features are done by the singular value decomposition of the reconstructed matrix. The current sampling extracted features as well as the previous four sampling periods are considered as the input and then the predicted value of the tool wear at the present time is achieved using BiLSTM. Cao *et al.* [233] introduced a new intelligent approach

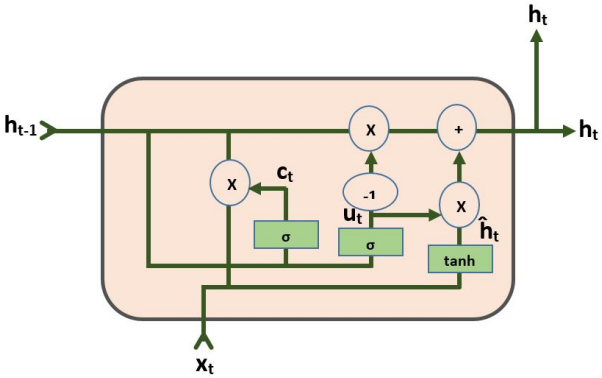


Figure 2.17: Block diagram of the GRU.

for fault diagnosing of wind turbine based on BLSTM, where ten common time-domain features are input to the BLSTM for fault diagnosing, which removes the need to manually identify the correct features and enhances the training time. The efficacy of the proposed approach is checked by three vibration signals of accelerometers in the drivetrain test rig of the wind turbine.

2.2.2.4 The Gated Recurrent Unit (GRU)

GRU is the optimized structure and the newer generation of recurrent neural networks (RNNs) proposed more recently by Cho *et al.* [234] in 2014. Similar to the Long Short-Term Memory network (LSTM), GRU is designed to introduce a gating mechanism for better control of the flow of the information through the various time-steps and to handle sequential data with its ability to encode temporal information and learn representations. And the main concept behind this design is solving the long-term dependency problem while mitigating the vanishing/exploding gradient issues [235]. Unlike the LSTM, the GRU fully exposes its memory at each time step and does not have separate memory cells. Moreover, each GRU has two gates (a) an update gate u^t that comes from combining (the forget gate with the input gate in the LSTM), and (b) a reset gate $\boldsymbol{c}^t$, while the LSTM has three gates (i.e., input, output, and the forget gate). Therefore, activations of the gates in the GRU only depend on the current input and previous output, which in general makes the GRU faster than the LSTM [234]. In short, the GRU has comparable advantages to LSTM units while being simpler. As shown in Fig. 2.17, there are two gates in GRU (an update gate u^t and a reset gate $\boldsymbol{c}^t$) to control if information needs to be updated or forgotten. In particular, the update gate decides how much of the past information can be passed

along to the future, and the reset gate specifies how much of the past information to forget [235].

(i) Reset Gate: The reset gate denoted by $\boldsymbol{c}^t$, where the current input and the previous hidden state are used to derive and calculate this gate, by multiplying them with their respective weights and then summing the results to be passed through a sigmoid function for enabling the gate to differentiate between the less important (0) and more valuable information (1) in the subsequent steps. After training the entire network through back-propagation, the weights will be updated and the network will learn to maintain only those valuable features. And then again the previous hidden state will be multiplied by the trainable weights and an element-wise multiplication will be held with the reset vector. This procedure will determine the information that should be held from the previous time steps along with the new inputs. Simultaneously, the current input will be multiplied by the trainable weights and then summed with the above product results, and then the result will pass through a *tanh* nonlinear activation function to form the new estimated candidate memory content [234, 236].

(ii) Update Gate: The update gate denoted by u^t is similar to the reset gate where the current input and the previous hidden state are used to derive and calculate the gate. The first part of the process will be the same as in the reset gate however, the multiplied weights are different to each gate (unique) as a result the vectors will be different. Then an element-wise multiplication between the update vector and the previous hidden state. The update gate here is aimed at helping the model decide how much of the past information held in the previous hidden state needs to be preserved for the future [234, 236].

Based on Fig. 2.17, at time step t, the state $\boldsymbol{h}^t$ of the j^{th} GRU is computed as

$$\boldsymbol{h}^t = (1 - \boldsymbol{u}^t)\boldsymbol{h}^{t-1} + \boldsymbol{u}^t\hat{\boldsymbol{h}}^t, \tag{2.2.30}$$

where $\boldsymbol{h}^{t-1}$ and $\hat{\boldsymbol{h}}^t$ represent the previous memory content and the new candidate memory content, respectively. The update gate, the reset gate, the new memory content, and updated hidden state are computed based on the previous hidden states

$\boldsymbol{h}^{t-1}$ and the current input $\boldsymbol{x}^t$ as follows

$$\left.\begin{aligned} \boldsymbol{u}^t &= \sigma(W_u \boldsymbol{x}^t + U_u \boldsymbol{h}^{t-1}) \\ \boldsymbol{c}^t &= \sigma(W_c \boldsymbol{x}^t + U_c \boldsymbol{h}^{t-1}) \\ \hat{\boldsymbol{h}}^t &= \tanh(W_h \boldsymbol{x}^t + \boldsymbol{c}^t \circ U_h \boldsymbol{h}^{t-1}) \\ \boldsymbol{h}^t &= \boldsymbol{u}^t \circ \boldsymbol{h}^{t-1} + (1 - \boldsymbol{u}^t) \circ \hat{\boldsymbol{h}}^t, \end{aligned}\right\} \tag{2.2.31}$$

where $\sigma(\cdot)$ denotes the sigmoid activation function of both gates; W_u, W_c, W_h, U_u, U_c, U_h are weight matrices; Operator "$\circ$" denotes the Hadamard product (entry-wise multiplication), while; $\tanh(\cdot)$ represents element-wise hyperbolic tangent activation function.

GRU's effectiveness in solving difficult sequence problems has been proven in a wide variety of applications such as speech recognition [237], machine translation [238], handwriting recognition [239], in addition to video captioning tasks, and financial sequence prediction [235]. The GRU has demonstrated exceptional performance in machine health monitoring, RUL prediction, and the other applications of PHM, therefore many approaches have been proposed in order to handle such problems. For example, Jinglong *et al.* [235] proposed RUL prediction approach of nonlinear degradation process for aero-engines based on simulated data. The proposed method used the kernel principal component analysis to extract nonlinear features, and then the GRU to estimate the RUL. Xu *et al.* [240] presented an approach for tool condition monitoring, that utilized the GRU to predict the wear in gun drilling. The efficiency of the proposed GRU model has been compared with existing models based on support vector regression and multi-layer perceptron, and the results of the GRU model outperforms the other models. Li *et al.* [241] proposed a model for rolling bearing health index and RUL prediction. The model utilized both kernel principal component analysis and exponentially weighted moving average for designing a modified health index and then stacked layers of GRU used for estimating both future health index and RUL. Xu *et al.* [242] used the GRU to propose a multi-stage tool condition monitoring for two main tasks, i.e., tool wear regression as well as, tool state classification. Firstly, the GRU as a classifier is estimating the stage of the tool condition. Then, training for different models based on each stage is conducted, in order to estimate the tool condition. To validate the performance of the proposed model, a gun-drilling experiment is conducted under various cutting conditions. Zhong *et al.* [243] proposed an exhaust gas temperature prognostic model

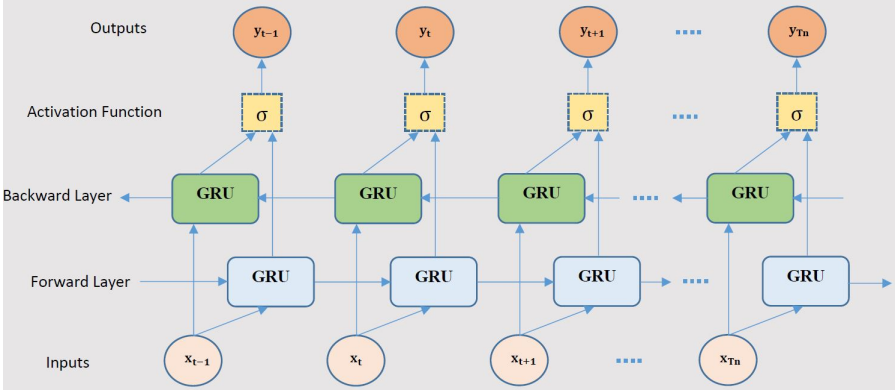


Figure 2.18: Block diagram of the BGRU.

based on the GRU network that simultaneously tackles the time series and nonlinear characteristics. To improve the prediction performance, the GRU design was defined by contrast experiments, where GRU stacked layer number, look-back timestamps, and output dimension were defined. Xiao *et al.* [244] employed the GRU network to propose an accurate method for state of charge estimation of lithium-ion batteries. The GRU learning process (training) has been improved and managed by utilizing two optimization techniques, i.e., the Nadam and AdaMax optimizers. Similarly, Yang *et al.* [245] employed the GRU to predict the battery state of charge from determining voltage, current, and temperature signals. The proposed approach utilizes the preceding state of charge and observations for providing better estimation efficiency. In addition, the proposed model is robust against the uncertain initial state of charge measures.

2.2.2.5 The Bidirectional Gated Recurrent Unit (BGRU)

The core idea of BGRU is simply to process the sequence input in two directions including forward and backward ways, which is contrary to the GRU that is a unidirectional RNN. Therefore, the BGRU employs two individual hidden layers (GRUs) as shown in Fig. 2.18, each one of them can jointly capture past (forward direction) and future (backward direction) [246]. Thus, we are able to compute the updated hidden vectors out of the forward ($\overrightarrow{\boldsymbol{h}}^t$) and backward ($\overleftarrow{\boldsymbol{h}}^t$) processes as follows

$$\overrightarrow{h}^t = \overrightarrow{\mathbb{H}}(x^t, \overrightarrow{h}^{t-1}) \tag{2.2.32}$$

$$\overleftarrow{h}^t = \overleftarrow{\mathbb{H}}(x^t, \overleftarrow{h}^{t+1}), \tag{2.2.33}$$

where $\longrightarrow$ and $\longleftarrow$ denote forward and backward process, respectively, and, function $\mathbb{H}$ is defined by Eq. (2.2.31). Then, the overall output sequence of the BGRU (denoted by h_{bi}^{t}) is computed by using an element-wise concatenation of the forward and backward hidden states at time step t, by

$$h_{bi}^{t} = \overrightarrow{h}^{t} \oplus \overleftarrow{h}^{t}. \tag{2.2.34}$$

Effectiveness of the BGRU has been proven in many fields such as object classification, speech recognition [247, 248], Biometrics human identification [249], power load forecasting [250, 251], text classification [252], pathology detection [253], multimodal object classification [254], human activity recognition [255], to mention just a few.

For machine health monitoring, RUL estimation, and other prognostic health management related fields, the BGRU is an effective and important tool especially when it combined with another technique in a hybrid model, however, it has some PHM applications as a stand alone model such as Yu *et al.* [256] proposed two steps approach for RUL estimation. The first step of this method is based on training the BGRU based autoencoder using an unsupervised way for converting high-dimensional multi-sensor readings that collected from multiple units of the same system, as historical run-to-failure readings, to low dimensional to be used for constructing the one dimensional health index to present different patterns of health deterioration of the instances. While in the second step, a comparison between the test HI curve that is derived from online sensor readings, and deterioration patterns developed in the offline period, using the technique of matching curves based on similarities. Rengasamy *et al.* [257] an approach has been introduced to improve the efficacy of machine prognostics and diagnostics. The improvement has been done by adopting two different weighted loss functions, the first one by creating a weight map based on the expected value and error acquired for each instance. In addition to the focal loss technique which is designed to predict the probabilistic outputs. The BGRU used with the previous techniques for predicting the RUL of a gas turbine engine and for fault detection in the air pressure system of heavy trucks. Remadna *et al.* [258] proposed an approach for RUL prediction of aircraft engine based on simulated data (C-MAPSS dataset). The proposed method comprises of two primary stages, where the first stage is based on using the principal component analysis and truncated singular value decomposition, for reducing the data dimensions without sacrificing relevant and important information. The second stage is by employing the BGRU to estimate the expected

engine behavior and its RUL.

Despite the fact, that all the aforementioned methods, generally have achieved good results in different PHM areas, still there are many of the data-driven based approaches that have an accuracy issue because of the deviation of the learned models from the real behavior of the system. In addition, they can hardly sustain good generalization, and that boosted the importance of developing advanced techniques that are able to handle all the issues and limitations of the previous approaches and at the same time to cope in view of the ever-growing requirements. On that basis, the third category was developed.

2.2.2.6 Hybrid (Multiple-Model) Approaches

Hybrid solutions have recently gained prominence due to their ability to overcome weaknesses and limitations of individual approaches. The key goal of the hybrid methods is to optimize the predictive potential by taking advantage of unique features and benefits of different approaches coupled to form the hybrid model [259]. To this aim, many researchers [260–270] have defined the hybrid model as an integration of both the data-driven and the physics based approaches. On the other side, many more researchers [271–284] have described the hybrid model using a more broad definition, which is: the integration of any two or more techniques that are combined to overcome the limitations of an individual method. Hence, they have implemented many hybrid model types (two or more of: different data-driven techniques, or different DNN techniques, etc.) that their designs built by this (second) definition and not on the traditional one (the data-driven and the physics). Both definitions are interrelated however, the second one is more inclusive in the sense that it applies to, includes, and covers all possible hybrid model types besides the first definition models. Aside from all of the above, we proposed a slight modification for the second definition of the hybrid model to become "the combination of any two or more various techniques that are integrated to form a new model to maximize the robustness and to minimize the limitations of the individual techniques" and this what we implemented and followed within this thesis.

The patterns of all the involved parameters and the characteristics of underlying data in the real applications of prognostic processes, are diversified and difficult to be predicted, in addition, different levels of data processing are conducted such as extraction, analysis, and modeling. This leads to the fact that one model cannot address all

the previous challenges, and this derives the need for the hybrid model. More specifically, the hybrid models that are designed based on utilizing the DNN techniques, as the exceptional success and the diversity of the deep learning techniques have made them an integral part of the modern maintenance solutions [154] yielded significant performance improvements that changed the prognostic health management to be centered on deep learning architectures. These hybrid models are divided into two main classes: series and parallel.

Series hybrid approach is well known in PHM literature especially for the (data-driven and physics based) type, it is usually, a physics-based model with prior knowledge about the process, combined with a data-driven model that handles the parameters estimation job for updating model parameters as new data becomes available [261, 283]. Many approaches with different styles for the series hybrid models have been proposed [285–288]. The series hybrid approaches are not the focus of our thesis, as all the proposed RUL estimation models in our thesis belong to the parallel hybrid model, as we will see later.

The parallel hybrid approach is the most promising category that genuinely reflects the objectives of the hybrid models especially when it comes to the hybrid of the DNN techniques. Using the parallel hybrid will ensure avoiding the shortcomings and constraints of individual approaches, in addition to compensating for their imperfections, This is achieved by utilizing the inherent design and mechanism of the parallel style as all the integrated models will receive the same input and each model then process the data in a specific way and consequently the results from all the models will be fused to form the desired output. For such style, in the case if, for any reason, missing information and/or a glitch occurred in one of the parallel models, then the other models will compensate. In contrast, the series hybrid model does not have this advantage where the input of the second stage is based on the output of the first one. In the case, if for any reason, missing information and/or a glitch occurred in one of the models, then it will propagate till the end.

Furthermore, the parallel hybrid design with multiple types of DNN (The focus of our thesis) will enable the model to collect more informative features, as each DNN technique has the ability to capture specific patterns from the input data resulting in different features, and advantages, that will be aggregated to enhance predictive efficiency [259]. Such parallel hybrid model capabilities allow for the abstraction of complex issues and allow for advanced precision in fault diagnosis and prognosis.

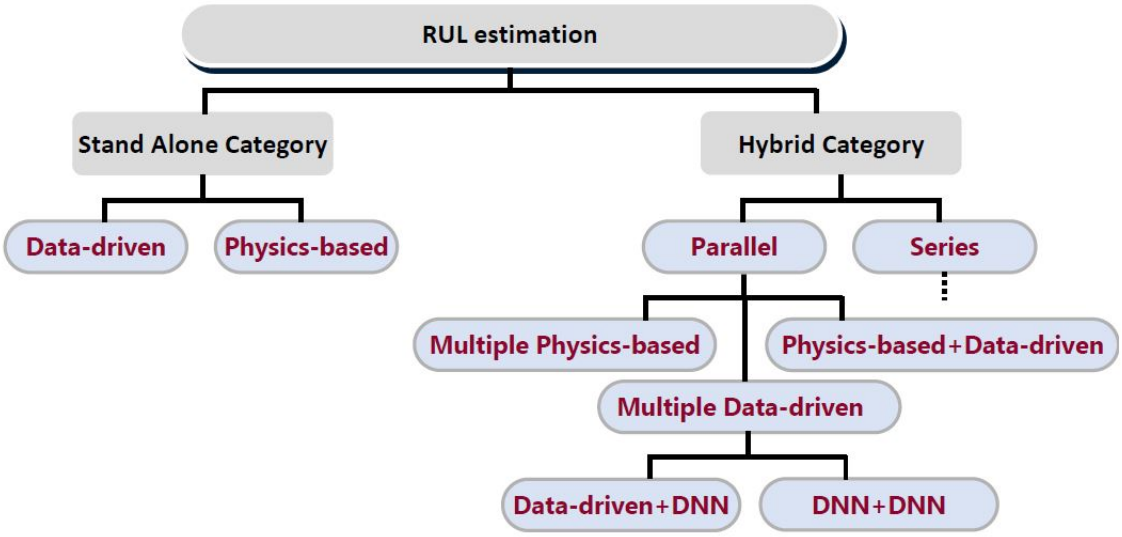


Figure 2.19: The classification of hybrid category.

It is worth mentioning that all the proposed approaches for the RUL estimation in our thesis belong to this category, and our proposed models (HDNN), (NBLSTM), (NPBGRU), (NPHM), and (MPHD) are the first parallel hybrid deep neural network model, the first noisy hybrid deep neural network models, the first multipath hybrid model, and the first noisy multipath hybrid deep neural network model, respectively for RUL estimation in the literature, as we will see in Chapters 4 5, and 6. Fig. 2.19 shows the classifying of the existing work on RUL estimation.

The exceptional performance and versatility of the deep learning hybrid models have made them essential tools in many areas such as natural language processing [289], waste classification, and recycling [290], speech recognition [291], human action recognition [292], video classification [293], age, and gender classification [294], and face verification [295], to name just a few. To the best of our knowledge, the following researches are the only few attempts that have been made towards the development of hybrid solutions for RUL estimation using deep neural networks.

For instance, Hinchi *et al.* [296] have introduced a series model through the integration of a CNN layer and an LSTM layer for bearing RUL estimation. Song *et al.* [279] proposed a hybrid model to improve the accuracy of RUL prediction for turbofan engines. This goal is achieved by combining an auto-encoder as a feature extractor, and bidirectional LSTM (BLSTM) to capture the bidirectional long-range dependencies of features. Zhao *et al.* [297] proposed a deep learning approach to deal with tool wear estimation issues. This is achieved by integrating a CNN architecture

in series with a bi-directional LSTM. Fully connected layers in addition to the linear regression layer are then used for the prediction task. Zhao *et al.* [246] combined the GRU and Bi-directional GRU (BGRU) architectures to build a hybrid model for machine health monitoring. Yu *et al.* [298] has employed vanilla LSTM and Restricted Boltzmann Machine (RBM) to explore the impact of unsupervised pre-training in residual lifetime estimation, by employing a setup of semi-supervised. Jayasinghe *et al.* [299] employed integration of temporal convolution layers and LSTM layers along with data augmentation for RUL estimation. Kong *et al.* [300] proposed a hybrid DNN framework that employed the CNN and LSTM integrated with health indicator (HI), for extracting different classes of features to effectively prognostication. Pan *et al.* [301] integrated one-dimensional CNN with LSTM to form a model for bearing fault diagnosis. The CNN output is the input of the LSTM to classify the fault types of the bearings. Li *et al.* [284] used one-dimensional CNN trained by using acoustic emission signals, and GRU networks trained by using vibration signals, for diagnosing the gear pitting faults problem. Liu *et al.* [302]utilized the feature-attention mechanism for weighting the input features in a way to distinguish the important features by getting higher weights than the others get and consequently improve the prediction performance. The next stage is based on using the BGRU for extracting the long-term dependencies from the input data (already weighted), then multilayer CNN is utilized for exploiting the sequence data local features. Finally, multilayer fully connected networks is employed for decoding the above abstractive features in order to estimate the RUL. Rao *et al.* [303] proposed two stages model, where the first one is utilized the many-to-many BLSTM in order to learn the speed related information from signals of vibration in forward and backward directions. Then, the final speed is derived consecutively by using the LSTM stage from the extracted information of the BLSTM. And finally, Li *et al.* [304] have used a parallel combination of CNN and LSTM for feature extraction. Extracted features are then combined and are provided as input to another LSTM followed by fully connected layers to predict the RUL. It is worth noting that the last reference is the only parallel hybrid model, however, it was proposed just after our proposed HDNN approach.

Chapter 3

General Degradation Modeling Frameworks

During the last two decades, many researchers have focused on modeling the degradation process of mission critical systems. In this regard, a wide range of models has been developed in the literature to analyze the degradation path, to capture the degradation dynamics of the underlying system, and consequently to support the decision making tasks. As discussed in Chapter 2, despite recent advancements in this context, most of the existing degradation works basically deal with one specific model, which is developed based on the assumption that a particular type of statistical distribution (e.g., Normal, Wiener, or Gamma distribution) governs the degradation process. However, a system may consist of multiple components or a component may have multiple degradation measures that require simultaneous consideration of multiple degradation paths. Needless to say that randomly applying a model with a certain path can be risky as every model is designed for a certain application. Such an approach becomes even more risky and complicated when multiple degradation measures are involved. To bridge the aforementioned gap, in this chapter general state-space modeling frameworks are proposed that cover a wide range of degradation scenarios. More specifically, in this chapter we propose the following two approaches:

(i) The Multiple Model Degradation Path (MMDP) estimation [305] framework, presented in Section 3.1, which takes into consideration a set of candidate models for the degradation path. Degradation prediction is then performed based

on each model in parallel. Finally, the outputs of localized filters are combined adaptively and in an intelligent fashion to form the overall estimate of the degradation process over time.

(ii) The second proposed approach (IMMPF) [306], presented in Section 3.2, which is based on implementing a combination of interactive multiple model and particle filters that run in parallel but with different stochastic characteristics modeling different potential degradation paths.

It is worthwhile to mention that our modes are capable of simultaneously handling different linear and non-linear degradation models without having any prior knowledge of the true degradation model of the system. Different simulation experiments are performed to evaluate the performance of the proposed MMDP and IMMPF frameworks. Simulation results are presented in Section 3.3.

3.1 Multiple-Model Degradation Path (MMDP) Estimation Framework

We consider the following state space model representing the underlying of CPESs.

$$x_k = \boldsymbol{f}(x_{k-1}, \boldsymbol{\mu}_k) + \boldsymbol{w}_k \tag{3.1.1}$$

$$z_k = \boldsymbol{h}(x_k) + \boldsymbol{v}_k, \tag{3.1.2}$$

where $x \in \mathbb{R}^{n_x}$ is the state variable, k denotes the time instant, $\boldsymbol{\mu}$ denotes the degradation state variable, and $z_k \in \mathbb{R}^{n_z}$ is the observation vector. It is worth mentioning that in the above state-space model, the degradation process is considered a hidden process because it cannot be measured online directly. In other words, any change of the degradation variable can affect the state vector directly while it impacts the measurement vector indirectly that is why $\boldsymbol{\mu}$ is not appearing in Eq. (3.1.2). If the degradation process is connected with the change of the performance variable $\boldsymbol{\phi}$ then the influence on the state is given by $\boldsymbol{\mu}_k = \boldsymbol{u}(\boldsymbol{\phi}_k)$, where $\boldsymbol{u}(\cdot)$ provides a function $\boldsymbol{\phi}_k$. The degradation path plays a major role in modeling and analyzing the degradation data in order to reliably estimate the state of CPESs and predict the failure mechanism of the system. The proposed MMDP is based on the following hybrid state

space model

$$
\begin{aligned}
x_k &= \boldsymbol{f}^{(j)}\big(x_{k-1}, \boldsymbol{u}(\boldsymbol{\phi}_{k-1}^{(j)})\big) + \boldsymbol{w}_k^{(j)}, && (3.1.3)\\
\boldsymbol{\phi}_k^{(j)} &= \boldsymbol{g}\big(\boldsymbol{\phi}_{k-1}^{(j)}\big) + \boldsymbol{\nu}_k^{(j)}, && (3.1.4)\\
\textit{and}\ \ z_k &= h_k(x_k) + \boldsymbol{v}_k, && (3.1.5)
\end{aligned}
$$

where superscript $j \in \{1, \dots n_f\}$ is the index representing the discrete state (or the degradation mode); $\boldsymbol{f}_k^{(j)}(\cdot)$ and $\boldsymbol{g}_k^{(j)}(\cdot)$ are, respectively, the state transition and degradation models corresponding to the degradation mode j, and; $\boldsymbol{w}_k^{(j)}$ and $\boldsymbol{v}_k$ are, respectively, the zero-mean state and observation noises with covariance matrices $\boldsymbol{Q}_k^{(j)}$ and $\boldsymbol{R}_k$. Term $\boldsymbol{\nu}_k^{(j)}$ represents the uncertainty in the degradation model j. Depending on the selected model, $\boldsymbol{\nu}_k^{(j)}$ follows a specific predefined distribution. The proposed MMDP estimation algorithm is based on the state-space model developed in Eqs. (3.1.3)- (3.1.5) and consists of a bank of n_f degradation-matched filters each corresponding to one of the candidate models in the model set. The overall state estimate $\hat{\boldsymbol{x}}_{k|k}$ is then computed from a weighted sum of n_f degradation-conditioned state estimates $\hat{\boldsymbol{x}}_{k|k}^{(j)}$, for $(1 \leq j \leq n_f)$, obtained from each of the model degradation filters. More precisely, the global state estimate $\hat{\boldsymbol{x}}_{k|k}$ at iteration $(k \geq 1)$ is defined as the expected value of the posterior distribution $p(x_k|z_{1:k})$ computed as

$$
\begin{aligned}
\hat{\boldsymbol{x}}_{k|k} &= \mathbb{E}\big\{x_k|z_{1:k}\big\} = \int x_k p(x_k|z_{1:k})\boldsymbol{d}x_k \\
&= \int x_k \sum_{j=1}^{n_f} p\big(x_k|z_{1:k}, m_k^{(j)}\big) p\big(m_k^{(j)}|z_{1:k}\big)\boldsymbol{d}x_k, && (3.1.6)
\end{aligned}
$$

where $m_k^{(j)}$ denotes the event that the degradation mode of the system at time k is mode j, for $(1 \leq j \leq n_f)$, and $\mathcal{M} = \{m^{(j)}\}$ denotes the set of all degradation models at all times. Eq. (3.1.6) can be expressed as a function of the degradation-conditioned state estimates as

$$
\hat{\boldsymbol{x}}_{k|k} = \sum_{l=1}^{n_f} \hat{x}_{k|k}^{(j)} p\big(m_k^{(j)}|z_k\big)\boldsymbol{d}x_k, \quad \text{with} \quad \hat{\boldsymbol{x}}_{k|k}^{(j)} = \int x_k p\big(x_k|z_k, m_k^{(j)}\big)\boldsymbol{d}x_k \qquad (3.1.7)
$$

the degradation-conditioned state estimate computed by assuming that the degradation mode of the system at iteration k is $m_k^{(j)}$. The degradation-conditioned state estimate $\hat{\boldsymbol{x}}_{k|k}^{(j)}$ is obtained from the filter matched to mode j. Before describing the

estimation algorithm based on Eqs. (3.1.6) and (3.1.7), first we develop a couple of candidate degradation models (i.e., Wiener process and Gamma process) to be incorporated in the degradation set $\mathcal{M} = \{m^{(\mathrm{B})}, m^{(\mathrm{G})}\}$, i.e., $n_f = 2$, where, $m^{(\mathrm{B})}$ refers to the mode that the hidden degradation path follows the Wiener process, while $m^{(\mathrm{G})}$ corresponds to the scenario where the Gamma process governs the underlying degradation path. It is very important to mention that the proposed hybrid state-space model and its corresponding estimation algorithm is general and can accommodate any number of candidate models.

3.1.1 Wiener Process (Brownian Motion)

As as we have mentioned earlier in 2.1.1.1 this model of the degradation path is a continuous-time Markov process with independent increments resembling a Brownian motion with drift, which is widely used for modeling random degradation behavior as it evolves over time [5]. The degradation path based on the Winner process can be expressed as

$$\phi_\tau = \tau_0 + \eta\tau + \sigma\mathcal{B}_\tau, \tag{3.1.8}$$

where $\tau_0 = \phi(0) \in R$ is the initial degradation value, $\eta \in R$ is the drift parameter, σ denotes the variance parameter, and $\mathcal{B}_\tau$ with $\mathcal{B}_0 = 0$ is the standard Brownian motion. To identify the state-space representation of the degradation process, we discretize Brownian motion using the properties of the Brownian motion with drift, i.e., $\theta(\tau) \sim \mathcal{N}(\eta_0 + \eta\tau, \sigma^2\tau)$ and $\Delta\theta \sim \mathcal{N}(\eta\Delta\tau, \sigma^2\Delta\tau)$. Consequently, the hidden degradation path based on the Winner process is given by $\theta_k = \theta_{k-1} + \eta T + \omega_k$ where $\theta_0 = \eta_0$ with $\theta_k = \theta_{kT}$ the value of the performance variable at time kT, and the error sequence $\omega_k \sim \mathcal{N}(0, \sigma^2 T)$, for $k \geq 0$. As the result, the modal state model corresponding to the Brownian motion $m^{(1)}$ can be described as follows

$$x_k = \boldsymbol{f}^{(\mathrm{B})}\big(x_{k-1}, \boldsymbol{u}(\theta_{k-1})\big) + \boldsymbol{w}_k^{(\mathrm{B})}, \tag{3.1.9}$$

$$and \;\; \theta_k = \theta_{k-1} + \eta T + \omega_k. \tag{3.1.10}$$

3.1.2 Gamma Process

This second model of the degradation path as we stated earlier in 2.1.1.2 is a stochastic process with independent, non-negative increments following a Gamma distribution with an identical scale parameter [30]. Intuitively speaking, the Gamma process corresponds to gradual degradation that monotonically accumulates over time. The Gamma process has been used in a wide range of applications in degradation analysis due to its monotonic nature. To develop the degradation path governed by the Gamma process, we use $\{G(t), t \geq 0\}$ which represents the degradation path with shape parameters $\alpha > 0$ and scale parameter $\beta > 0$ such that $\Delta G(t) \sim \text{Gamma}(\alpha \Delta t, \beta)$. Now by using the tabulated formulas for the mean and variance of a gamma distribution, we obtain the moment-matching normal approximation $\Delta G(t) \sim \mathcal{N}(\alpha/\beta \Delta t, \alpha/\beta^2 \Delta t)$. Then the hidden degradation process can be described as $G_k = G_{k-1} + \alpha/\beta T + \boldsymbol{\nu}_k$. Consequently, the degradation state-space corresponding to the Gamma process is given by

$$x_k = \boldsymbol{f}^{(\mathrm{G})}\big(x_{k-1}, \boldsymbol{u}(G_{k-1})\big) + \boldsymbol{w}_k^{(\mathrm{G})}, \tag{3.1.11}$$

$$\textit{and} \;\; G_k = G_{k-1} + \frac{\alpha}{\beta} T + \boldsymbol{\nu}_k. \tag{3.1.12}$$

In the next sub-section, we combine the Wiener process with the Gamma process to develop the multiple model adaptive estimation framework.

3.1.3 The main steps of the proposed (MMDP)

(1) Computing Degradation-Matched State Estimate: For a non-linear system, given by Eqs. (3.1.3)-(3.1.5), we use the particle filter to compute each degradation-matched state estimate. More precisely, each degradation-matched particle filter is developed based on the following optimal Bayesian filtering recursions

$$P^{(j)}(x_k|z_{1:k-1}) = \int P(x_{k-1}|z_{1:k-1}) P^{(j)}(x_k|x_{k-1}) dx_{k-1}, \tag{3.1.13}$$

$$\textit{and} \;\; P^{(j)}(x_k|z_{1:k}) = \frac{P(z_k|x_k) P^{(j)}(x_k|z_{1:k-1})}{P(z_k|z_{1:k-1})}, \tag{3.1.14}$$

where $P^{(j)}(x_k|x_{k-1})$ denotes the transitional density function matched to degradation mode $m^{(j)}$ and is computed based on Eqs. (3.1.3) and (3.1.4). After computation of the state estimate matched to degradation mode j, for $(1 \leq j \leq n_f)$, the next step

is to compute the corresponding weights for each mode-matched filter and form the fused estimate which is described next.

(2) Hypothesis Test (Weight Update): The goal of the weight update step is to determine the conditional probability density function $p(m_k^{(j)}|z_{1:k})$ corresponding to mode $m_k^{(j)}$, for $(1 \leq j \leq n_f)$, given all the observations $z_{1:k}$ up to the current time, i.e., $p\big(m_k^{(j)}|z_{1:k}\big)$ After applying the Bayes' rule, $p\big(m_k^{(j)}|z_{1:k}\big)$ can be expressed as

$$\mu_k^{(j)} \triangleq p\big(m_k^{(j)}|z_{1:k}\big) = \frac{p\big(z_k, z_{1:k-1}, m_k^{(j)}\big)}{p\big(z_k, z_{1:k-1}\big)} = \frac{p\big(z_k|z_{1:k-1}, m_k^{(j)}\big)p\big(z_{1:k-1}, m_k^{(j)}\big)}{p\big(z_k, z_{k-1}\big)}. \quad (3.1.15)$$

The second term in the nominator of Eq. (3.1.15) is factorized as follows

$$p\big(z_{1:k-1}, m_k^{(j)}\big) = p\big(m_k^{(j)}|z_{1:k-1}\big)p\big(z_{1:k-1}\big), \quad (3.1.16)$$

which results in the following conditional density function

$$\begin{aligned} \mu_k^{(j)} &= \frac{p\big(z_k|z_{1:k-1}, m_k^{(j)}\big)p\big(m_k^{(j)}|z_{1:k-1}\big)p\big(z_{1:k-1}\big)}{p\big(z_k|z_{1:k-1}\big)p\big(z_{1:k-1}\big)} = \frac{p\big(z_k|z_{1:k-1}, m_k^{(j)}\big)p\big(m_k^{(j)}|z_{1:k-1}\big)}{p\big(z_k|z_{1:k-1}\big)} \\ &= \frac{p\big(z_k|z_{1:k-1}, m_k^{(j)}\big)p\big(m_k^{(j)}|z_{1:k-1}\big)}{\sum_{j=1}^{n_f} p\big(z_k|z_{1:k-1}, m^{(j)}\big)p\big(m^{(j)}|z_{1:k-1}\big)}, \end{aligned} \quad (3.1.17)$$

where the denominator is a normalizing factor to ensure that $p(m_k^{(j)}|z_{1:k})$ is a proper probability density function (PDF). In the numerator of Eq. (3.1.17), term $p(z_k|z_{1:k-1}, m_k^{(j)})$, is the likelihood function corresponding to mode j which is computed from the particle filter matched to this degradation model.

(3) Computing the Overall State Estimate: The final step of the MMDP state estimation approach is to update the global state estimate $\hat{\boldsymbol{x}}_{k|k}$ and its corresponding error covariance matrix $\boldsymbol{P}_{k|k}$ from the available n_f mode-matched state estimates. It is performed on the basis of the following expressions

$$\hat{\boldsymbol{x}}_{k|k} = \sum_{j=1}^{n_f} \mu_k^{(j)} \hat{x}_{k|k}^{(j)} \quad \text{and} \quad \boldsymbol{P}_{k|k} = \sum_{j=1}^{n_f} \mu_k^{(j)} \left[\left(\hat{x}_{k|k}^{(j)} - \hat{x}_{k|k} \right) \left(\hat{x}_{k|k}^{(j)} - \hat{x}_{k|k} \right)^T \right]. \quad (3.1.18)$$

This completes the proposed MMDP framework. Next, we present our simulation example to evaluate its performance.

3.2 Interactive Multiple Model Particle Filters (IMMPF) Framework

We consider the following state space model representing the CPESs, which is the same state space model as in section 3.1.

$$x_k = \boldsymbol{f}(x_{k-1}, \boldsymbol{\mu}_k) + \boldsymbol{w}_k \tag{3.2.1}$$

$$\text{and } z_k = \boldsymbol{h}(x_k) + \boldsymbol{v}_k, \tag{3.2.2}$$

The degradation path plays a major role in modeling and analyzing the degradation data in order to reliably estimate the state of CPESs and predict the failure mechanism of the system. The proposed IMMPF is based on the following hybrid state space model

$$x_k = \boldsymbol{f}^{(j)}\big(x_{k-1}, \boldsymbol{u}(\boldsymbol{\phi}_{k-1}^{(j)})\big) + \boldsymbol{w}_k^{(j)}, \tag{3.2.3}$$

$$\boldsymbol{\phi}_k^{(j)} = \boldsymbol{g}\big(\boldsymbol{\phi}_{k-1}^{(j)}\big) + \boldsymbol{\nu}_k^{(j)}, \tag{3.2.4}$$

$$\text{and } z_k = h_k(x_k) + \boldsymbol{v}_k, \tag{3.2.5}$$

The proposed IMMPF estimation algorithm consists of a bank of n_f degradation-matched filters, each corresponding to one of the candidate models in the model set. In our approach three particle filters are implemented and run in parallel based on the following set of candidate degradation models: Gamma DP; Brownian DP; and Inverse Gaussian DP. and provide filtering results for different dynamic models and mix the outputs based on their mixing probabilities.

Before describing the estimation algorithm, first we develop the group of candidate degradation models (i.e., Wiener process, Gamma process and Inverse Gaussian process.) to be incorporated in the degradation set $\mathcal{M} = \{m^{(\mathrm{B})}, m^{(\mathrm{G})}, m^{(\mathrm{IG})}\}$, where, $m^{(\mathrm{B})}$ represents the hidden degradation path based on the Wiener process, $m^{(\mathrm{G})}$ for the Gamma process, and $m^{(\mathrm{IG})}$ for Inverse Gaussian process.

The Wiener Process (Brownian Motion) and Gamma Process are used as the first two degradation paths, as briefly described below:

Wiener Process (Brownian Motion): The degradation path based on the Winner process can be expressed as

$$\phi_\tau = \tau_0 + \eta\tau + \sigma\mathcal{B}_\tau, \tag{3.2.6}$$

Consequently, the modal state model for the Brownian motion $m^{(1)}$ can be described as follows

$$x_k = \boldsymbol{f}^{(\mathrm{B})}\big(x_{k-1}, \boldsymbol{u}(\theta_k)\big) + \boldsymbol{w}_k^{(\mathrm{B})}, \tag{3.2.7}$$

$$\text{and } \theta_k = \theta_{k-1} + \eta T + \omega_k. \tag{3.2.8}$$

Gamma Process: The degradation state-space corresponding to the Gamma process is given by

$$x_k = \boldsymbol{f}^{(\mathrm{G})}\big(x_{k-1}, \boldsymbol{u}(G_{k-1})\big) + \boldsymbol{w}_k^{(\mathrm{G})}, \tag{3.2.9}$$

$$\text{and } G_k = G_{k-1} + \frac{\alpha}{\beta} T + \boldsymbol{\nu}_k. \tag{3.2.10}$$

More details on the Wiener Process and Gamma Process are provided in Sections 3.1.1 and 3.1.2.

3.2.1 Inverse-Gaussian Process

As we stated earlier 2.1.1.3 it is similar to the Gamma process in terms of the monotone degradation path with independent, non-negative increment that follows an IG distribution. Compared with the Gamma process, the IG process has many useful properties and it is very flexible in incorporating covariates and random effects [41]. Its flexibility comes from the inverse relation between the Wiener process and IG process. The later has been used in a range of applications in degradation analysis where the two previously mentioned processes have failed, such as the GaAs laser degradation analysis [42]. To develop the degradation path governed by the IG process, we use $\{IG(k), k \geq 0\}$ with $IG(0) = 0$, which represents the degradation path with shape parameter $\mu\boldsymbol{\Lambda(k)}$ and scale parameter $\lambda\boldsymbol{\Lambda^2(k)}$ such that $\Delta IG(k) \sim IG(\mu\boldsymbol{\Lambda(k)}, \lambda\boldsymbol{\Lambda^2(k)})$. When $\mu > 0$, $\lambda > 0$ and monotonic increasing function of time k with $\boldsymbol{\Lambda(0)}$ =0, where $\boldsymbol{\Lambda(k) = k^q}$, where $(q > 0)$ and using different values of q in this power law function will lead to modeling various patterns of degradation process [43]. The mean of $IG(k)$ is $\mu\boldsymbol{\Lambda(k)}$. Its variance is $\mu^3\boldsymbol{\Lambda(k)}$

/ $\lambda\boldsymbol{\Lambda}(\boldsymbol{k})$. Consequently, the degradation state-space corresponding to the Inverse-Gaussian process is given by

$$x_k = \boldsymbol{f}^{(\mathrm{IG})}\big(x_{k-1}, \boldsymbol{u}(IG_{k-1})\big) + \boldsymbol{w}_k^{(\mathrm{IG})} \tag{3.2.11}$$

$$\text{and} \quad IG_k = IG_{k-1} + \boldsymbol{\nu}_k. \tag{3.2.12}$$

Below we combine the three modes to describe the IMMPF framework. The proposed hybrid state-space model and its corresponding estimation algorithm is general and can incorporate any number of candidate modes.

3.2.2 The IMMPF Algorithm

For the non-linear system (3.2.3)-(3.2.5), we use the particle filter to compute the estimate of the degradation states for each mode. More precisely, each degradation-matched particle filter is developed based on the optimal Bayesian filtering recursion.

$$P^{(j)}(x_k|z_{1:k-1}) = \int P(x_{k-1}|z_{1:k-1})P^{(j)}(x_k|x_{k-1})dx_{k-1} \tag{3.2.13}$$

$$\text{and} \quad \boldsymbol{P}^{(j)}(x_k|z_{1:k}) = \frac{P(z_k|x_k)P^{(j)}(x_k|z_{1:k-1})}{P(z_k|z_{1:k-1})}, \tag{3.2.14}$$

where $P^{(j)}(x_k|x_{k-1})$ denotes the transitional density function matched to degradation mode $m^{(j)}$ and is computed using Eqs. (3.2.3)-(3.2.5). Please refer to 2.1.4.1 for further details on the particle filters. After computing the state estimates for three degradation modes j, $(1 \leq j \leq n_f)$, the next step is to combine three state estimates through localized Gaussian approximation. Considering that the filter is in steady state, the associated estimates, i.e., $(\mu_{k-1}^{(i)}, \hat{\boldsymbol{x}}_{k-1|k-1}^{(i)}, \boldsymbol{P}_{k-1|k-1}^{(i)})$, for $(1 \leq i \leq n_f)$, are computed for iteration $(k-1)$. Iteration k for the mixing stage is outlined below:
(1) Interaction/Mixing Stage: Let p_{ij} denote the (3×3) model transition probability from model i to model j, $(1 \leq i, j \leq 3)$. Suppose $\mu_{k-1}^{(i)}$ is the probability of model $m^{(i)}$ being in effect at the time step $(k-1)$, then the mixing probability $\mu_k^{(i|j)}$ for models $m^{(i)}$ and $m^{(j)}$ is computed as

$$\bar{c_k}^{(j)} = \sum_{i=1}^{n_f} p_{ij}\mu_{k-1}^{(i)} \qquad \text{and} \qquad \mu_k^{(i|j)} = \frac{1}{\bar{c_k}^{(j)}} p_{ij}\mu_{k-1}^{(i)}, \tag{3.2.15}$$

where $\bar{c_k}^{(j)}$ is the normalization factor. As state estimates $\hat{\boldsymbol{x}}_{k-1|k-1}^{(j)}$ and associated

covariance matrices $\boldsymbol{P}_{k-1|k-1}^{(j)}$ are known from the previous step, we can compute the initial state vector and associated covariance matrix for mode j in the current iteration as follows

$$\hat{\boldsymbol{x}}_{k-1|k-1}^{(0j)} = \sum_{i=1}^{n_f} \mu_k^{(i|j)} \hat{\boldsymbol{x}}_{k-1|k-1}^{(i)} \tag{3.2.16}$$

$$\boldsymbol{P}_{k-1|k-1}^{(0j)} = \sum_{i=1}^{n_f} \mu_k^{(i|j)} \left\{ \boldsymbol{P}_{k-1|k-1}^{(i)} + \left[\hat{\boldsymbol{x}}_{k-1|k-1}^{(i)} - \hat{\boldsymbol{x}}_{k-1|k-1}^{(0j)}\right]\left[\hat{\boldsymbol{x}}_{k-1|k-1}^{(i)} - \hat{\boldsymbol{x}}_{k-1|k-1}^{(0j)}\right]^T \right\}. \tag{3.2.17}$$

where $j \in \{1, \dots n_f\}$ refers to the model number.

(2) Particle Filter Stage: For $(1 \leq j \leq n_f)$, filter j generates N_p particles from the initial state vector $\hat{\boldsymbol{x}}_{k-1|k-1}^{(0j)}$ and covariance matrix $\boldsymbol{P}_{k-1|k-1}^{(0j)}$ based on the Gaussian distribution as follows

$$\mathbb{X}_k^{(i,j)} \sim \mathcal{N}(\hat{\boldsymbol{x}}_{k-1|k-1}^{(0j)}, \boldsymbol{P}_{k-1|k-1}^{(0j)}) \; |_{i=1}^{N_p} |_{j=1}^{N_m=n_f} \; . \tag{3.2.18}$$

Given the particles $\mathbb{X}_k^{(i,j)}$, $(\; i \in \{1, \dots N_p\})$, the corresponding weights $W_k^{(i,j)}$ are updated by

$$W_k^{(i,j)} \propto p\big(z_k \big| \mathbb{X}_k^{(i,j)}\big) = \mathcal{N}(z_k - h(\mathbb{X}_k^{(i,j)}), R). \tag{3.2.19}$$

The weights are normalized as $\bar{W}_k^{(i,j)} = W_k^{(i,j)} \; / \; \sum_{i=1}^{N_p} W_k^{(i,j)}$. Consequently, the posterior is approximated with the particle set $\{\; \mathbb{X}_k^{(i,j)}, \; W_k^{(i,j)} \;\}_{i=1}^{N_p} \rightarrow \mathcal{N}(x_k^{(j)}, \boldsymbol{P}_k^{(j)})$. The state estimate for mode $m_k^{(j)}$ is given by

$$\hat{x}_{k|k}^{(j)} = \sum_{i=1}^{N_p} \bar{W}_k^{(i,j)} \mathbb{X}_k^{(i,j)} \tag{3.2.20}$$

$$\text{and} \qquad \hat{\boldsymbol{P}}_{k|k}^{(j)} = \sum_{i=1}^{N_p} \bar{W}_k^{(i,j)} (\mathbb{X}_k^{(i,j)} - \hat{x}_{k|k}^{(j)})(\mathbb{X}_k^{(i,j)} - \hat{x}_{k|k}^{(j)})^T \tag{3.2.21}$$

(3) Hypothesis Test (Weight Update): The goal of the adaptive weight update step is to find the conditional probability density function $p(m_k^{(j)}|z_{1:k})$ corresponding to mode $m_k^{(j)}$, for $(j \in \{1, \dots n_f\}\;)$, given all the observations $z_{1:k}$ up to the current

time, i.e., $p\big(m_k^{(j)}|z_{1:k}\big)$ which can be expressed as

$$\mu_k^{(j)} \triangleq p\big(m_k^{(j)}|z_{1:k}\big) = \frac{\mu_{k-1}^{(j)} p\big(z_k|x_k, z_{1:k-1}, m_k^{(j)}\big)}{\sum_{j=1}^{n_f} \mu_{k-1}^{(j)} p\big(z_k|x_k, z_{1:k-1}, m_k^{(j)}\big)}. \tag{3.2.22}$$

Taking into account the Markov transition probability p_{ij} from mode i to mode j, i.e., $(p\big(m_k^{(i)}|m_{k-1}^{(j)}\big) = p_{ij}$ for $i, j \in \mathcal{M})$. In this case, $p(m_k^{(j)}|z_{1:k-1}) = \sum_{i=1}^{n_f} p_{ij} p(m_{k-1}^{(j)}|z_{1:k-1})$, therefore, the corresponding weight for the filter matched to mode $m_k^{(j)}$, for $(1 \leq j \leq n_f)$, is given by

$$\mu_k^{(j)} = p\big(z_k|x_k, z_{1:k-1}, m_k^{(j)}\big) \frac{\sum_{i=1}^{n_f} p_{ij} \mu_{k-1}^{(i)}}{\sum_{j=1}^{n_f} p\big(z_k|x_k, z_{1:k-1}, m_k^{(j)}\big) p_{ij} \mu_{k-1}^{(i)}}. \tag{3.2.23}$$

As the adaptive weight of each mode-matched filter is found, then we can combine the local state estimates and form the global state estimate as described in the next step.

(4) Computing Overall State Estimate: The final step of the IMMPF state estimation approach is to update the global state estimate $\hat{\boldsymbol{x}}_{k|k}$ and its corresponding error covariance matrix $\boldsymbol{P}_{k|k}$ from the available n_f mode-matched state estimates. It is performed using the following expressions

$$\hat{\boldsymbol{x}}_{k|k} = \sum_{j=1}^{n_f} \mu_k^{(j)} \hat{x}_{k|k}^{(j)} \tag{3.2.24}$$

and

$$\boldsymbol{P}_{k|k} = \sum_{j=1}^{n_f} \mu_k^{(j)} \left[\left(\hat{x}_{k|k}^{(j)} - \hat{x}_{k|k} \right) \left(\hat{x}_{k|k}^{(j)} - \hat{x}_{k|k} \right)^T \right]. \tag{3.2.25}$$

This completes the proposed IMMPF framework. Next, results from Monte Carlo simulations are presented.

3.3 Simulations

In this section, different simulation results are presented to evaluate the proposed MMDP and IMMPF frameworks.

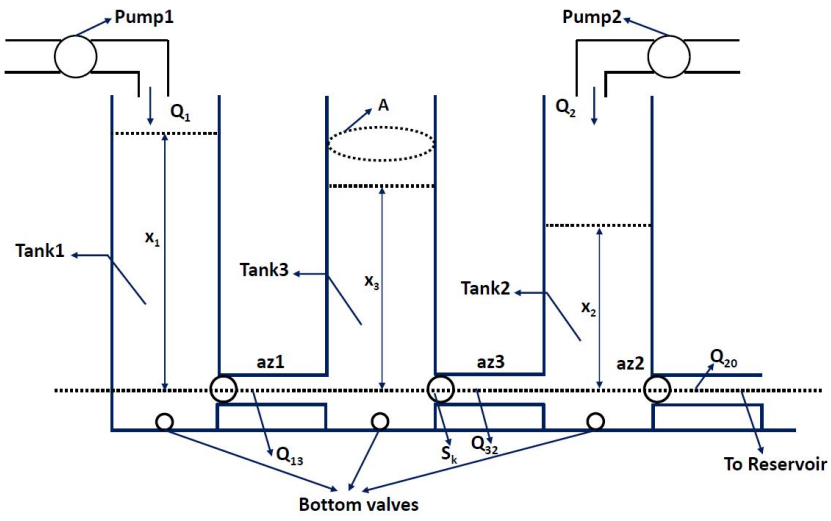


Figure 3.1: The complete structure of the three-tank DTS200 Model.

3.3.1 Evaluation of the MMDP Framework

For the proposed degradation modelings, we considered the following three-tank model called DTS200 that was developed and provided by a German company called Amira Automation [307].

3.3.1.1 The Three Tank DTS200 Model

As shown in Fig. 3.1 the DTS200 model is a closed system comprised of three cylindrical tanks T1, T2, and T3 having the same cross section area A (m^2). These tanks are connected together through cylindrical pipes of S_k (m^2) cross section area. An outflow valve with a circular cross section S_k (m^2) located on the right side of T2. The system's outflowing liquid is stored in a reservoir under the three tanks. Two pumps P1and P2 are there to control the inflow to tank T1 and T2, respectively. These pumps are supplied through the reservoir with a liquid that returns to the system, and they are automatically switched off when the T1or T2 liquid level reaches a specified upper limit. In addition, the system has five more valves, two of them to control the characteristic of the flow between the neighboring tanks, while the other three valves are at each tank's bottom to drain any tank manually if needed. The process input signals are the pumps flow rates denoted by Q_1 and Q_2 (m^3/s), while the output signals are the levels of the tanks T1 and T2.

The following equations illustrate the dynamic mechanism of the water tank system [307] to evaluate the proposed MMDP

$$x_{1,k} = x_{1,k-1} + \Delta T\big(\frac{Q_1 - Q_{13}}{A}\big) + w_1, \tag{3.3.1}$$

$$x_{2,k} = x_{2,k-1} + \Delta T\big(\frac{Q_2 + Q_{32} - Q_{20}}{A}\big) + w_2, \tag{3.3.2}$$

$$\text{and} \quad x_{3,k} = x_{3,k-1} + \Delta T\big(\frac{Q_{13} - Q_{32}}{A}\big) + w_3, \tag{3.3.3}$$

where Q_{13}, Q_{32} and Q_{20} are unknown quantities computed based on

$$q = azS_k\text{sign}(\Delta x)\sqrt{(2g|\Delta x|)} \tag{3.3.4}$$

as follows

$$Q_{13} = az_1S_k\text{sign}(x_{1,k} - x_{3,k})\sqrt{(2g|x_{1,k} - x_{3,k}|)}, \tag{3.3.5}$$

$$Q_{32} = az_2S_k\text{sign}(x_{3,k} - x_{2,k})\sqrt{(2g|x_{3,k} - x_{2,k}|)}, \tag{3.3.6}$$

$$\text{and} \quad Q_{20} = az_2S_k\sqrt{(2gx_{2,k})} \tag{3.3.7}$$

with $x_{1,k}$, $x_{2,k}$, and $x_{3,k}$ denoting the state variables (tanks water levels); az_1, az_2, and az_3 are the outflow coefficients (dimensionless, real values ranging from 0 to 1; Q_{13}, Q_{32} and Q_{20} the flow rates (m^3/s); g the earth acceleration (m/s^2). The observation models are given by

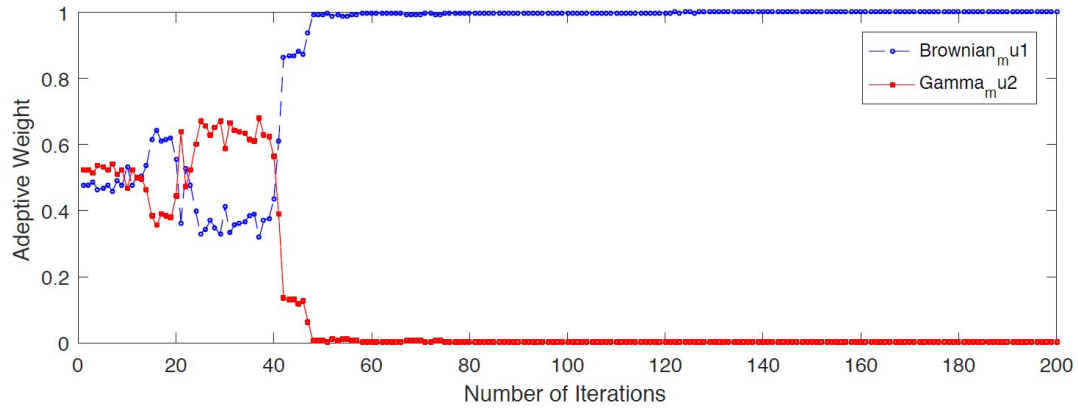
$$y_{1,k} = x_{1,k} + v_{1,k}, \tag{3.3.8}$$

$$y_{2,k} = x_{2,k}^2 + v_{2,k}, \tag{3.3.9}$$

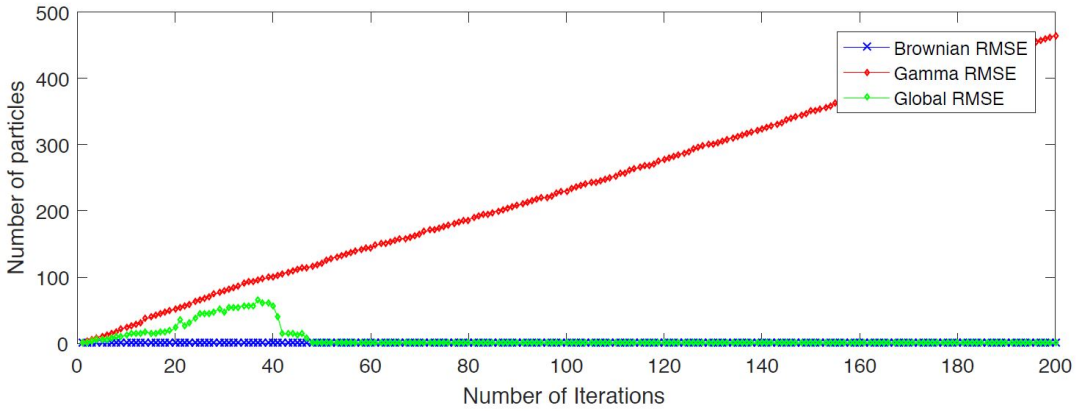
$$\text{and} \quad y_{3,k} = x_{3,k} + v_{3,k} \tag{3.3.10}$$

where the states x_1 and x_3 are directly measured with Gaussian noises $(v_{1,k}, v_{3,k})$ while state x_2 is indirectly measured with non-Gaussian noise $(v_{2,k})$ [307]. In our model, the hidden degradation process is related to the decrease of the flow coefficient $az1$, which indicates that the connection pipe between tank 1 and tank 3 is jammed. Then the degradation path function can be described using Eqs. (3.1.8) and (3.1.12).

In our simulations, the degradation process follows the Wiener process (Brownian motion), which is used to generate the degradation data. Our proposed approach does not have any a priori knowledge of the degradation model. Fig. 3.2 (a) illustrates the



(a)



(b)

Figure 3.2: (a) Adaptive weights associated with the Brownian and Gamma models. (b) The RMSE results obtained based on the proposed MMDP framework, and stand-alone Gamma and Browning matched filters.

adaptive weight corresponding to each of the constituent models. Fig. 3.2(a) shows that the evolution of the adaptive weight for the Brownian path, which varies over time initially but converges to 1 after some iterations. On the other hand, the adaptive weight for the Gamma path starts with some initial value picked at random and evolves to 0 over time. The proposed filter, therefore, successfully recognizes the hidden degradation process. Fig. 3.2 (b) illustrates the Root-Mean-Square Error (RMSE) results obtained based on three implemented filters as follows: (i) The proposed MMDP filter; (ii) Stand-alone particle filter matched to Brownian degradation model, and; (iii) Stand-alone particle filter where the degradation path is matched to the Gamma process. All the filters start from the same initial value equal to the starting point of the true state. Fig. 3.2 (b) shows how the error evolves over time

for the Brownian path, Gamma path, and also for the proposed MMDP model. For the Brownian model as well as the proposed model, the error converges to zero. For the Gamma path, the error continues to increase. Similar results were achieved for several numbers of Monte Carlo runs.

3.3.2 Evaluation of the IMMPF Framework

The results of the MMDP approach developed in Section 3.1 were remarkable; however, there is some limitation with this approach, where, the approach can accurately find the right degradation path without any prior knowledge even if the system may consist of multiple components or a component may have multiple degradation measures. Nevertheless, once the approach has found the degradation path then it will continue considering the same degradation path, while the behavior of degradation may alter over time, thus in this section, we proposed the IMMPF Framework in Section 3.2 that can tackle this issue and bridge the gap. To test the performance of the IMMPF framework, we consider the same tank system as in 3.3.1 by using the discretized version of the commonly used three-vessel water tank system DTS200 [308], designed by Amira automation corporation, Germany. The system dynamics are given by

$$x_{1,k} = x_{1,k-1} + \Delta T\big(\frac{Q_1 - Q_{13}}{A}\big) + w_1, \tag{3.3.11}$$

$$x_{2,k} = x_{2,k-1} + \Delta T\big(\frac{Q_2 + Q_{32} - Q_{20}}{A}\big) + w_2, \tag{3.3.12}$$

$$\text{and} \quad x_{3,k} = x_{3,k-1} + \Delta T\big(\frac{Q_{13} - Q_{32}}{A}\big) + w_3, \tag{3.3.13}$$

where $\{Q_{13}, Q_{32}, Q_{20}\}$ are unknown quantities computed using

$$Q_{13} = az_1 S_k \text{sgn}(x_{1,k} - x_{3,k})\sqrt{2g|x_{1,k} - x_{3,k}|}, \tag{3.3.14}$$

$$Q_{32} = az_2 S_k \text{sgn}(x_{3,k} - x_{2,k})\sqrt{2g|x_{3,k} - x_{2,k}|}, \tag{3.3.15}$$

$$\text{and } Q_{20} = az_2 S_k \sqrt{2g x_{2,k}}, \tag{3.3.16}$$

Table 3.1: Specifications of the experimental setup based on the three-vessel water tank system, DTS200 [308].

Variables	Description	Values
az_1, az_2, az_3	Outflow coefficients	0.4904, 0.6114, 0.4502
Q_{13}, Q_{32}, Q_{20}	Flow rates in m^3/s	——
Q_1,Q_2	Supplying flow rates	4.0×10^{-5}, $1.4 \times 10^{-5} m^3/s$
A	Cross sectional area of water tank	$1.54 \times 10^{-2} m^2$
S_k	Cross sectional area of connection pipe	$5.0 \times 10^{-5} m^2$
g	Acceleration due to gravity	9.81m/s
T	Sampling interval	1s

with $\{x_{1,k}, x_{2,k}, x_{3,k}\}$ denoting the state variables. The observation model is given by

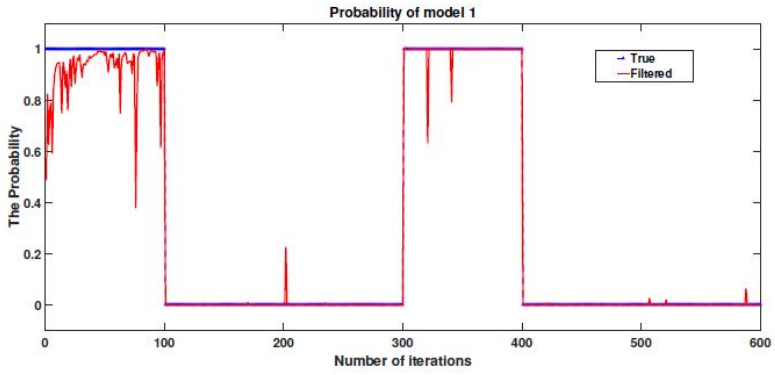
$$y_{1,k} = x_{1,k} + v_{1,k}, \tag{3.3.17}$$

$$y_{2,k} = x_{2,k}^2 + v_{2,k}, \tag{3.3.18}$$

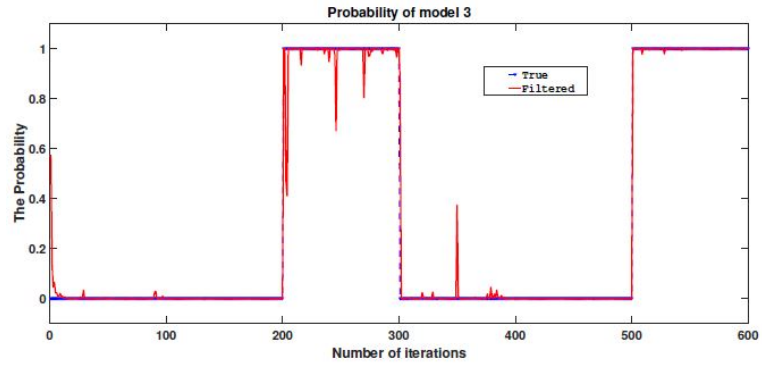
$$\text{and} \quad y_{3,k} = x_{3,k} + v_{3,k} \tag{3.3.19}$$

The degradation path is modeled as in Eqs. (3.2.6), (3.2.10)and (3.2.12).

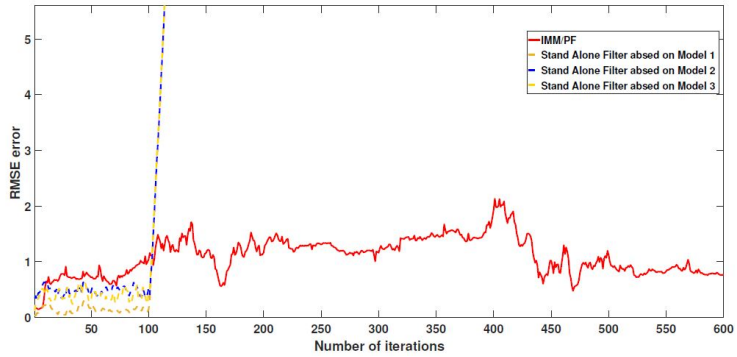
In our simulations, the system is simulated 600 time steps, and the active model during the steps (1-100) and (301-400) is set to model 1 (*Brownianpath*), and during the steps (101-200) and (401-500) is for model 2 (*InverseGaussianpath*), while during the steps (201-300) and (5001-600) is for model 3 (*Gammapath*), The purpose of forcing the model transitions instead of simulating them randomly is to demonstrate the properties of IMM-filter. It also reflects the fact that in real problems we cannot determine the model transition probability matrix accurately. Our proposed approach does not have any prior knowledge of the degradation model. Fig. 3.3(a) shows the filtered estimate for the probability of model 1 (*Brownianpath*) in each time step, which is almost zero in the following periods (101-200), (201-300), (401-500) and (5001-600), but its probability is almost 1 during (101-200) and (401-500), while Fig. 3.3(b) displays the desired results of model 3 (*theGammapath*) for the designated time interval. The proposed filter, therefore, successfully recognizes the hidden degradation process. Fig. 3.3(c) shows how the RMSE error evolves to infinity over time for each stand-alone filter (Brownian path, Inverse Gaussian path, and Gamma path) while bounded for our proposed model IMMPF.



(a)



(b)



(c)

Figure 3.3: Descriptioin of the figure goes here.(a) The Probability for model 1 (Brownian). (b) The Probability for model 3 (Gamma). (c) The Root-Mean-Square Error for stand-alone filters and the proposed IMMPF.

3.4 The Summary

In this chapter, the multiple model adaptive estimation framework (MMAE), and the interacting multiple model (IMM) have been utilized coupled with a set of candidate

degradation models for the degradation path, to perform degradation prediction based on each model in parallel. Each degradation model is used to perform a spectrum reliability analysis of the system. The innovations and likelihoods of the various models are integrated to compute adaptive weights for the degradation models. The local predictions obtained from the degradation models are combined based on the computed weights to establish a single degradation estimate for that time instant. For a future time, the process is iterated to model the dynamic nature of the system's degradation.

Chapter 4

Hybrid Parallel Deep Neural Network Models for RUL Estimation

This chapter and the following chapters focus on the second dimension considered in the thesis, i.e., the RUL estimation problem. The leading industrial and manufacturing companies try to use the advancements in predictive analytics and machine learning to come up with new data science methodologies and sophisticated algorithms to move from showing what happened (the traditional CBM solution and Preventive Maintenance) to predicting what will happen in the future (now performing Predictive Maintenance). In Chapter 2, we presented different categories of the RUL prediction models, and we discussed the benefits and potentials of hybrid modeling (in particular the one designed based on different deep neural network architectures) as an upcoming and promising category. Although several multiple model solutions have been introduced for RUL estimation, most of such approaches are developed based on the same deep neural network architectures. Although limited research works utilized different deep neural network architectures for the development of hybrid models, the thesis proposes the first parallel hybrid deep neural network models designed for RUL estimation. The proposed models establish a new important category in this field that utilizes the most successful techniques of the DNN in a parallel fashion and based on different architectures. In this chapter, we present four different hybrid parallel models using many DNN techniques, which are CNN, LSTM, GRU, FC, BLSTM, and BGRU architectures. The proposed approaches have achieved the

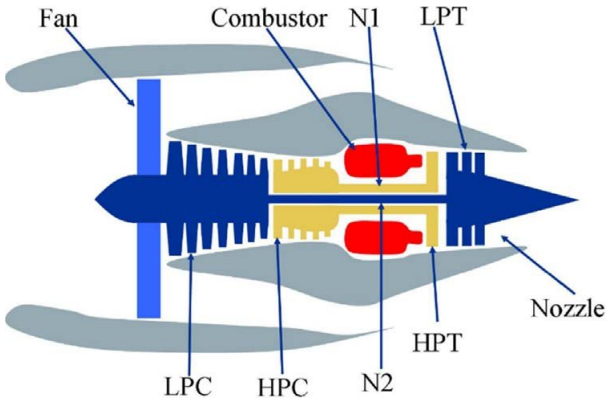


Figure 4.1: A simplified diagram of the simulation engine in C-MAPSS [311].

best results among all the existing methods, especially in complex situations where different operating conditions and fault mods are involved.

4.1 NASA C-MAPSS Data Set

The proposed approaches are implemented and evaluated based on the degradation datasets of the turbofan engine provided in References [309] and [310]. The NASA C-MAPSS dataset is the most popular simulated dataset for RUL prediction, which was produced on the Commercial Modular Aero-Propulsion System Simulation (C-MAPSS). Fig. 4.1 shows a simplified diagram of the simulation engine in C-MAPSS. The C-MAPSS is a software for simulating several scenarios of the degradation behavior and the faults impact of the main five rotating components (fan, Low Pressure Compressor, High Pressure Compressor (HPC), High Pressure Turbine, and Low Pressure Turbine) found in a large commercial turbofan engine, throughout different combinations of operating conditions and failure modes.

According to these operating conditions and failure modes, the dataset can be divided into four sub-datasets (referred to as the FD001 to FD004) each sub-dataset is further divided into training and test subsets. The datasets are outlined in Table 4.1. The datasets are arranged in an N-by-26 matrix, where N corresponds to the number of data points in each dataset. Each row is a snapshot of data taken during a single operating time cycle, which includes 26 columns and each column represents a different variable. The 26 columns of data consist of two index values representing the engine number and the current operational cycle number, three operational settings that have a substantial effect on engine performance, as well as 21 sensor values, as

Table 4.1: Data Set Details (Simulated From C-MAPSS) [312].

Dataset	C-MAPSS			
	FD001	**FD002**	**FD003**	**FD004**
Train Trajectories	100	260	100	249
Test Trajectories	100	259	100	248
Conditions	1	6	1	6
Fault Modes	1	1	2	2

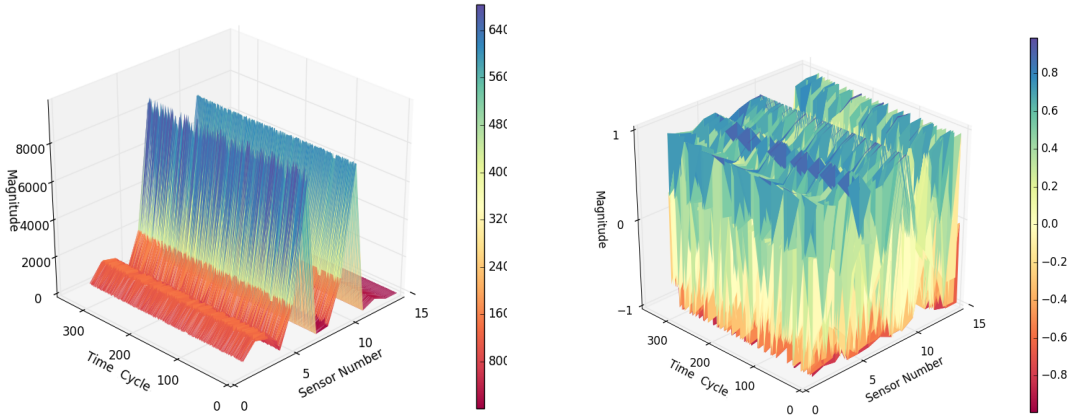
shown in Table 4.1, details of which can be found in [311]. Each trajectory within the datasets simulates the lifetime of an engine. While each engine is simulated with different initial conditions, the operational status of each engine is healthy in the early stage and begins to degrade as time progresses until a failure occurs. For each engine trajectory within the training sets, the last data entry corresponds to the moment the engine is declared unhealthy. While trajectories in the test sets terminate at some time prior to failure and the target is to predict the number of cycles until the end of product life for each engine commonly referred to as the RUL. The actual RUL values of the test trajectories for the C-MAPSS dataset was made available to the public.

4.1.1 Operating Conditions and Fault Modes

The data points are grouped into different clusters according to the impact of different factors such as various operating conditions and fault modes on a unit's performance [309]. Firstly, clustering based on the operating conditions leads into two different groups, i.e., (FD001 and FD003), which are simulated only under the single condition of operating at the sea level. On the other hand, we have (FD002 and FD004), which are simulated based on various (six to be exact) conditions. Now, based on the other factor (i.e., fault modes), we end up with different groups, where (FD001 and FD002) are formed using a single degradation factor (the HPC), whilst, (FD003 and FD004) are simulated based on two degradation factors, i.e., fan and HPC degradations, which means more complex and difficult scenarios for the RUL estimation.

4.1.2 Data Normalization

As stated previously, C-MAPSS dataset contains time-series measurements from 21 sensors that used to monitor engine performance. Nonetheless, some sensor readings are not considered to be informative for RUL estimation, since they have almost



(a) (b)

Figure 4.2: The illustration of the data from the 14 selected sensors of FD004 before and after normalization. (a) Raw sensor data (before). (b) Normalized sensor data (after).

constant outputs in the engine's lifetime. Thus, a group of 14 sensor outcomes has been selected following Reference [313]. Due to having different ranges for collected sensor measurements, a normalization phase is required prior to any testing and training, to unify the values to be within a specific range and to provide unbiased involvement from the readings of each sensor. In this regard, to provide a standard range across all the features, the Min-Max normalization was employed, i.e.,

$$\bar{x}_i = \frac{2(x_i - \min\ x_i)}{\max\ x_i - \min\ x_i} - 1, \tag{4.1.1}$$

where x_i represents the time sequence for i_{th} sensor measurements, and $\bar{x}_i$ represents the normalized data. The normalization help to provide equal participation from all features associated with all operating conditions [314]. Fig. 4.2 is an illustrative example to visualize the need for normalization and to show the effects of data normalization. More specifically, Fig. 4.2 presents the readings from different sensors obtained from the FD004 dataset when normalization is applied and in the absence of normalization. Note that the normalized data is distributed within the $[-1, 1]$ range.

4.1.3 Performance Evaluation

In this research, to evaluate the performance of the proposed RUL models and provide comparisons with their counterparts, two objective performance measures, i.e., Scoring function, and Root Mean Square Error (RMSE), are adopted. The considered performance measures are briefly outlined below:

(1) *Scoring Function*: It is a performance measure defined by the PHM community at the International Conference on Prognostics and Health Management in 2008 and is given by

$$S = \sum_{i=1}^{M_{te}} s_i, \text{ where } \quad s_i = \begin{cases} e^{-\frac{h_i}{13}} - 1 & for \quad h_i < 0 \\ e^{\frac{h_i}{10}} - 1 & for \quad h_i \geq 0, \end{cases} \tag{4.1.2}$$

where S is the computed score, M_{te} is the total number of testing data samples, and $h_i = \overline{RUL}_i - RUL_i$ (estimated RUL - true RUL, with respect to the i_{th} data point).

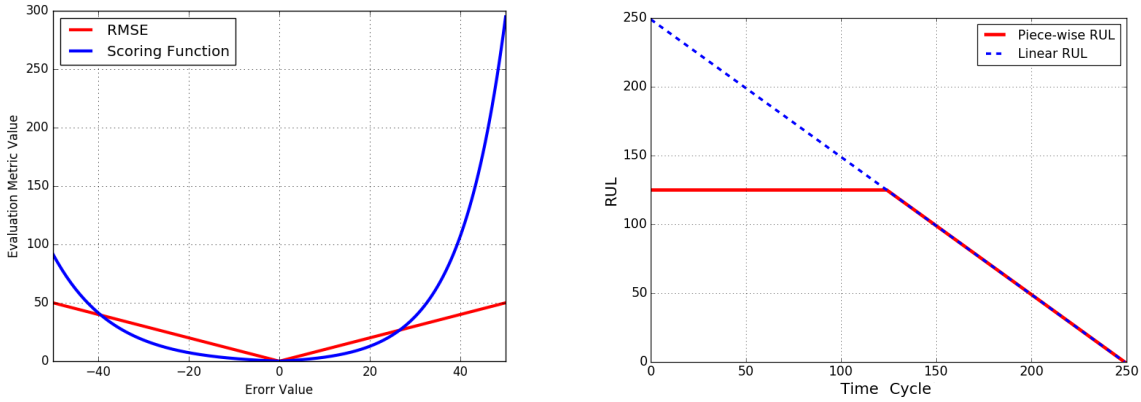
(2) *The RMSE*: A common metric that is widely used as a performance measure for evaluating the estimation accuracy of the RUL estimation. The formulation of the RMSE is given by

$$\text{RMSE} = \sqrt{\frac{1}{M_{te}} \sum_{i=1}^{M_{te}} h_i^2}. \tag{4.1.3}$$

Fig. 4.3(a) demonstrates comparisons of the RMSE and the score function. It should be noted that the scoring function favors early predictions (i.e., the estimated RUL value is smaller than the actual RUL value) more than late predictions (i.e., the estimated RUL value is larger than the actual RUL value), as it penalizes late predictions more than early ones. Whilst, the RMSE assigns equal weights to both early and late predictions. The lower the evaluation metrics are, the better performance the proposed method can achieve.

4.1.4 RUL Target Function

For the training of neural networks, the output (ground truth) corresponding to the input data should be precisely known. In prognostic health management applications,



(a) (b)

Figure 4.3: (a) The comparison of scoring function and RMSE function. (b) Illustration of piece-wise linear RUL target function.

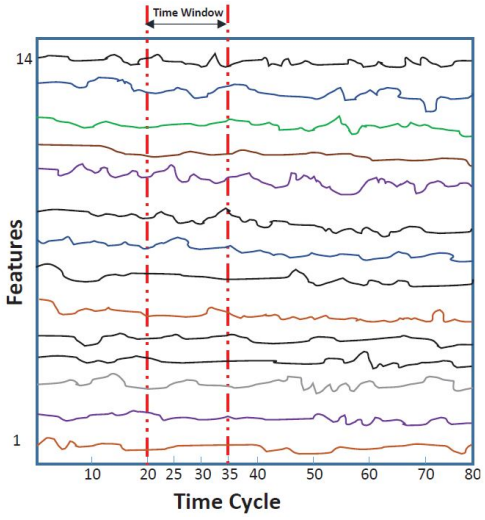
however, accurate knowledge of the target RUL used for training the network is typically not available and is estimated based on a physics-based model [313]. Different solutions [156, 192, 214–216, 313, 315–318] have been proposed to address this issue. Among these solutions, references [156, 192, 214, 313, 316, 318] have used a piece-wise linear degradation model for determining the target RUL. The intention behind this approach is that the system is healthy during the initial stage of its operation. Degradation increases when the system approaches its "end-of-life". For this reason, it is reasonable to model the RUL as a constant value when the system is relatively new. It degrades linearly with the passage of time. For this dataset, the piece-wise linear degradation model has been validated to be effective [156, 192, 214, 313, 316, 318, 319]. It has a constant RUL phase with a value of 125 estimated from the observed data during the degradation phase, the system degrades linearly (Fig. 4.3(b)). The piece-wise linear RUL target function has the advantage of preventing the algorithm from overestimating the RUL. Moreover, the linear degradation model is the most natural choice to be used in cases where prior knowledge of a suitable degradation model is not available. This completes the description of the C-MAPSS dataset.

4.2 The General Settings of the Proposed Frameworks

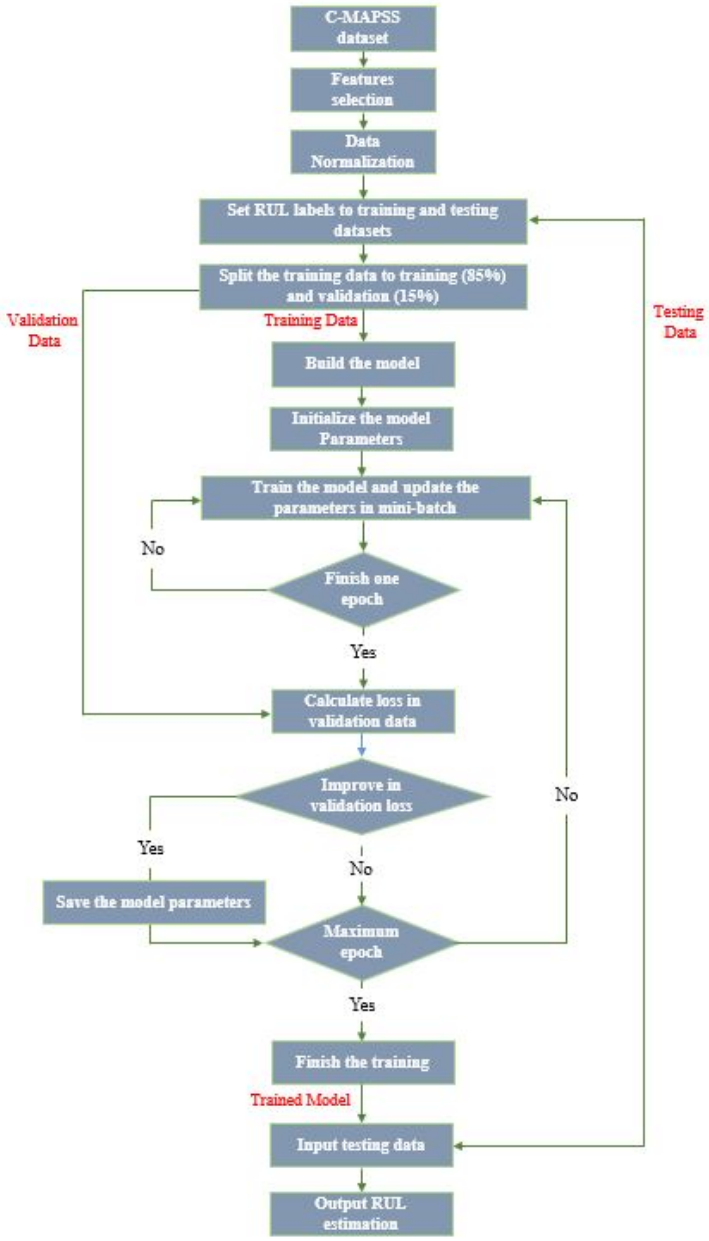
In this chapter, all the proposed frameworks consist of two parallel paths, the first path usually based on (one, multiple, or mixed) of the following DNN techniques (LSTM, GRU, BLSTM, and BGRU). The second path is based on the CNN technique. The two parallel paths are followed by fully connected multilayer neural networks (fusion layers), acting as the fusion center combining the extracted features from each of the two parallel paths in order to estimate the targeted RUL. In these approaches, we consider the RUL prediction of a system/subsystem (like the engine of an airplane) with N units based on run-to-fail maintenance strategy or any particular event measurements defining the end-life of the unit. The number of sensors L attached to each component of the system is used to collect information that serves as features for RUL estimation. For each unit n, where $(1 \leq n \leq N)$, the observation vector is defined as $\boldsymbol{x}^t = [x_1^t, \ldots, x_L^t]^T \in \mathbb{R}^{L \times 1}$ at $t > 0$. The collected data from the sensors corresponding to unit n are represented in the following matrix form

$$\boldsymbol{X}_n = [\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^t, \ldots, \boldsymbol{x}^{T_n}] \qquad \in \mathbb{R}^{L \times T_n}, \tag{4.2.1}$$

where superscript T denotes transpose operation, and T_n is the failure time of a specific unit (n). The RUL estimation is an example of the multi-variate time series-based problems, and as the temporal sequence data offers more informative features than a multivariate sample [320], a sliding window technique is adopted to prepare samples to use the temporal multivariate information for better feature extraction. In this regard, first, consider r_{tw} denoting the size of the sliding window, and r_f as the number of selected features. Then $(r_f \times r_{tw})$ represents the 2D input matrix to the proposed models. In particular, the 2D matrix $(r_f \times r_{tw})$ is supplied as an input into the CNN path. At the same time, each column of the matrix will simultaneously be fed as an input into the other parallel path at each time moment. Different lengths (30 & 15) of r_{tw} were examined and tested with sliding step size of 1. Fig. 4.4(a) shows data samples of length 15 time window for 14 selected sensors. Fig. 4.4(b) describes the complete procedures of the proposed approaches.



(a)



(b)

Figure 4.4: (a) Training sample of 15 time window (14 selected features). (b) Flowchart of the proposed models.

4.3 Frameworks of The Proposed Networks

In this section, the key structures used to develop the proposed frameworks are presented. Fig. 4.5 presents the proposed parallel hybrid model referred to as the Hybrid Deep Neural Network Model (HDNN) [321, 322]. The proposed HDNN framework is

the first parallel hybrid deep neural network model designed for RUL estimation. The exceptional results of this approach, its simple design, the reduced number of involved parameters, and its impressive performance in handling complex datasets consisting of several operating conditions and fault modes, opened the door for the development of other parallel hybrid models for RUL estimation. In this regard, the thesis proposed three alternative approaches shown in Figs. 4.6, 4.7, and 4.8. In what follows, we describe different components used in developing the three paths of the proposed frameworks.

4.3.1 The First Parallel Path

As stated previously, the first parallel path in each of the proposed models is designed uniquely based on different DNN architectures. For the first proposed model, i.e., the"HDNN", the first path is designed based on the LSTM architecture to overcome the vanishing and exploding gradient problem of the traditional RNN. Additionally, the LSTM architecture can learn order dependence in sequence prediction problems, which is achieved by employing a memory state mechanism with a special gating system (see Section 2.2.2.2). The LSTM takes each measurement sequence (formed based on a time window with a fixed length) and then models the whole sequence based on the targeted RUL value. The structure of the incorporated LSTM is many-to-one as shown in Fig. 2.13. Three LSTM layers are stacked in the LSTM path of the HDNN model for the extraction of temporal features. Each of the first two layers is defined by 32 cell structures, while the third layer is based on the 64 cell structure. Repeated cells within the LSTM layer have the same structure and parameter values.

The first path in the second proposed approach [83], shown in Fig. 4.6, is based on the BLSTM and LSTM architectures. The BLSTM is the modified and extended version of the LSTM architecture. As illustrated in Fig. 2.16, the BLSTM model consists of two LSTM layers included to process the sequence input in two directions (i.e., forward and backward directions). The intuitive rationale behind such a design is to utilize all available information from the past and future of a specific time frame. This path begins with a BLSTM layer defined by a 32-cell structure with return sequences and a dropout rate of 0.3. The output is to be concatenated and then fed to the next LSTM layer. The BLSTM layer followed by two LSTM layers which are defined using 32 and 64 cell structures, respectively. The associated setting values for the repeating cells are considered to be the same.

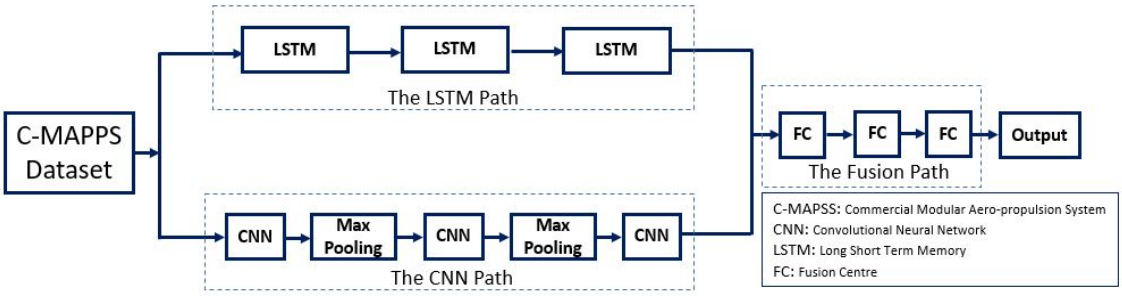


Figure 4.5: The proposed hybrid deep neural network (HDNN) framework.

In the third proposed approach [323,324], shown in Fig. 4.7, the GRU architecture is used to design the first parallel path. GRU is an optimized and a newer generation of RNNs that has been inspired by the LSTM design. As shown in Fig. 2.17, the GRU architecture transfers information only using the hidden state without the incorporation of the cell state. Moreover, it has only two gates, the reset gate and the update gate making the GRU faster than LSTM but with comparable performance and a somewhat more streamlined version (see Section 2.2.2.4). In this path, we utilized two stacked GRU layers to extract the temporal features. In this regard, 32 and 64 cell structures with return sequences are adopted for the first and the second GRU layers, respectively. In addition, dropout is used with a rate of 0.3 value. The same cell configuration, in terms of parameter values and structure, is used within this layer.

Finally, as shown in Fig. 4.8, in the fourth proposed approach [323,324], we utilized the BGRU to design the first parallel path. The BGRU processes the sequence input in two directions including forward and backward ways, which is different from the GRU that uses a unidirectional RNN. The BGRU, therefore, employs two individual hidden layers (GRUs) as shown in Fig. 2.18, each one of them can jointly capture past (forward direction) and future (backward direction) [325]. We used 32 and 64 cell structures, respectively, with return sequences for the two BGRU layers in this path, used for extracting temporal features. Furthermore, the dropout of 0.3 is used.

4.3.2 The CNN Path (The Second Parallel Path)

The proposed methods have a common design for the second parallel path, which is based on using the CNN technique, as shown in Figs. 4.5 - 4.8. This path is employed as another powerful feature extraction mechanism, that extracts another

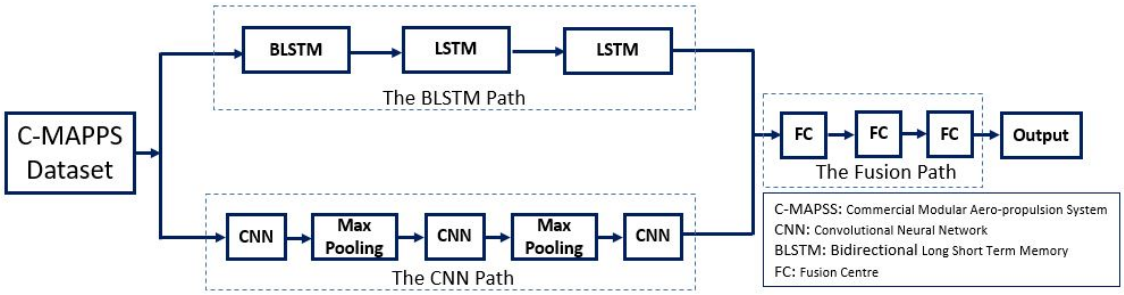


Figure 4.6: The proposed hybrid deep neural network (BiLSTM) framework.

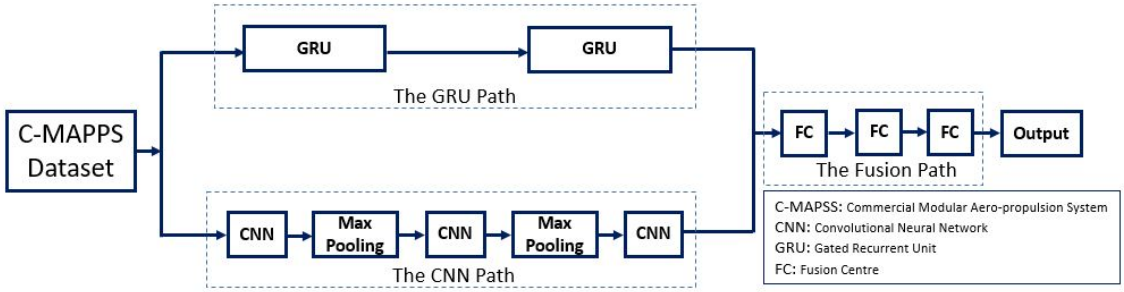


Figure 4.7: The proposed hybrid deep neural network (GRU) framework.

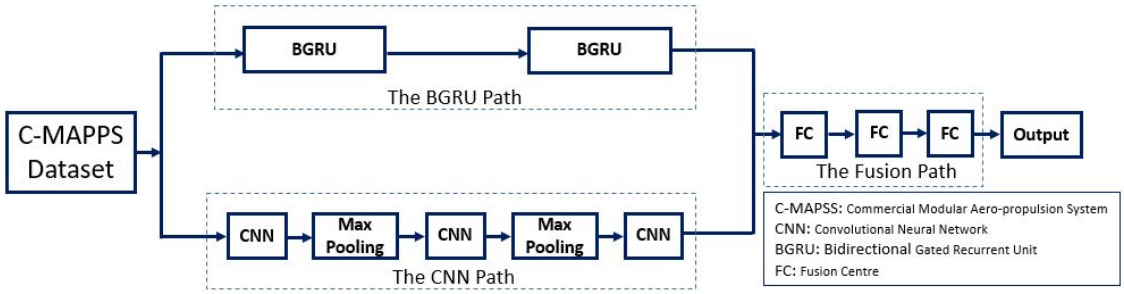


Figure 4.8: The proposed hybrid deep neural network (BiGRU) framework.

class of features as compared with the first parallel path. This path comprises three CNN layers, the first two of which are followed by max pooling layers. 10 filters of the same configurations, each one with the size of (9×1) are applied for the convolution layers, and a filter of size is (2×1) is selected for the max pooling layers. The last layer within the CNN path consists of one CNN filter of (3×1) dimensions. The first two approaches used *tanh* as an activation function, while the third and fourth approaches used the Rectified Linear Unit (ReLU) as the activation function for all the layers. Fig. 4.9 shows the detailed architecture of the CNN path for the proposed approaches for RUL estimation.

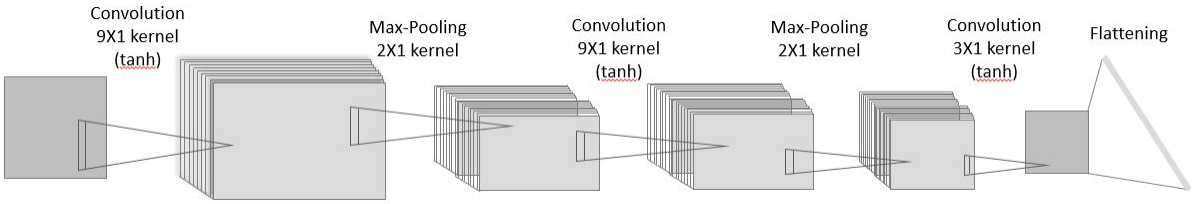


Figure 4.9: The detailed architecture of the CNN path.

4.3.3 The Fusion Path (The Third Path)

This path represents the fusion center that integrates the extracted features from the previous parallel paths to perform the regression task for generating the targeted output of the RUL prediction. The input of this path is vector constructed based on the flattened output of the CNN path (to be in 1D) concatenated with the output features of the first parallel path.

This path in all the proposed methods consists of three fully connected layers. In the first proposed model "HDNN", each of the first two fusion layers has 100 neurons and uses a *tanh* activation function, similarly, for the second proposed model "BLSTM" the first two FC layers based on 103 neurons with *tanh* activation function. While in both models the third FC layer is designed based on a 1 neuron and the utilized activation function is the Rectified Linear Unit (ReLU).

For the third and the fourth proposed models ("GRU" & "BiGRU"), the first and the second FC layers are built using 87 and 107 neurons, respectively. The last FC layer has a 1 neuron and all the layers in both models have used the (ReLU) activation function. A dropout rate of 0.3 has been used in all the proposed approaches. This completes the description of the different paths and components of the proposed frameworks.

4.3.4 The Overfitting Issue

One of the main problems in deep neural networks is overfitting. This occurs when the model, instead of learning a mapping from inputs to outputs and learning the general distribution of the training data, starts to: (i) Memorize the training dataset; (ii) Learn the associated noise patterns, and; (iii) Learn the predicted outcomes for each data point [326].

In contrast, underfitting occurs when the model is not powerful enough for capturing the underlying patterns of neither the training data nor new data. In this case,

the model has to be improved or modified in such a way to be able to handle the complexity of the dataset. There are several different methods for handling overfitting, the most important methods are Dropout and Early Stopping [327] as described below.

4.3.5 The Dropout Technique

Dropout is a simple and effective form of regularization approach. Dropout is, typically, used to reduce data overfitting during the training stage. The term dropout comes from randomly dropping out neurons as well as all of its connections (its incoming and outgoing weights) temporarily during the training step [326]. The remaining neurons will handle the required job of the missing neurons, consequently, the network is less responsive to specific neuronal weights. This potentially leads to better generalization, more resistance to overfitting to the training data, and prevents repeated extraction of the same features in some cases [156].

4.3.6 The Early Stopping Technique

Dividing a training dataset into a training set and a validation set is a common practice to use the validation set for evaluating the model performance after each epoch (training iteration). Early stopping is a method to monitor and avoid overfitting in NN models by stopping the training at the moment where the overfitting starts. As shown in Fig. 4.10, the early stopping technique is implemented by monitoring the model's efficiency during the training step, and the best way to measure that is by computing the error or the loss function value during the training. The error is computed repeatedly for both the training and the validation sets throughout the entire training cycle. As time goes on, the error/loss of both training and validation data will decrease as the model learns. Nevertheless, this will no longer be the case if the model is being trained too long, where the computed error on the training set will continue to decline, but eventually, the computed error on the validation set would increase again [328]. This is the point at which the overfitting starts as well as the early stopping works. In addition, at this stage, we have the best generalization for both training and validation sets.

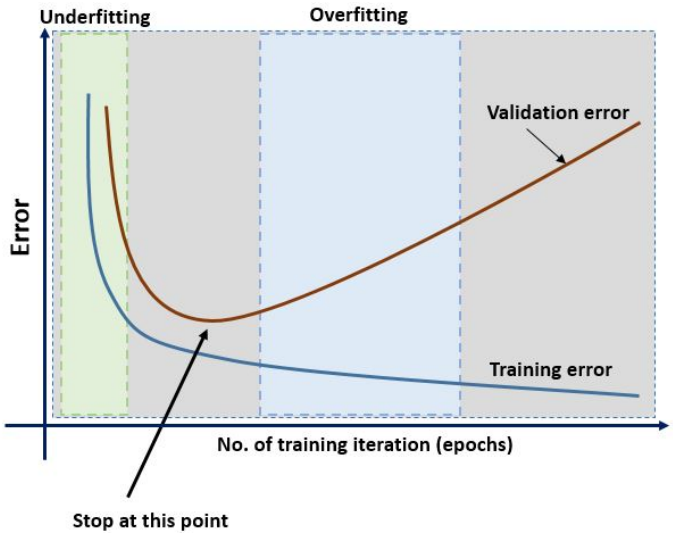


Figure 4.10: The Early Stopping Technique.

4.3.7 Training Process

The objective of the training process is to obtain optimal parameters (weights and biases) so that the cost function is minimized [329]. In other words, the training involves determining the most suitable values for the weights and biases in the network for enabling the model to solve a particular problem. As stated earlier in Section 2.2.1.3, there are several types of training, in this thesis, the adopted type is supervised training because the desired outputs (actual RUL) of the data are already provided in the dataset. Then the network analyses and processes the input data, and compares their corresponding outputs to the desired values. In the proposed approaches, each training dataset was randomly divided into 85 percent and 15 percent, respectively, as training and validation sets. For the training set, all the available measurement data points of the engines are utilized as samples for the training, and each one of them is associated with the related RUL label as the targeted value. The piecewise degradation method is employed to determine the RUL label for every training sample. For each training iteration (epoch), the training dataset is randomly divided into small batches of samples called mini batches, and the batch size is set equal to 512 in all the proposed models. Backpropagation and mini batch gradient decent [330] are used for training the networks, as they are working on finding the weights that minimize the loss function, as well as for deciding how much amount each network weight needs to update, after the comparison of the estimated output with the desired one. In addition to that, the adaptive moment estimation (Adam) is used as

the optimization technique for updating the weights and minimizing the model error rate [331]. The maximum number of training epochs is 250. The used learning rate is 0.001. The Mean Squared Error (MSE) is adopted as the loss function which is given by

$$\text{MSE} = \frac{1}{M_{tr}} \sum_{i=1}^{M_{tr}} h_i^2, \tag{4.3.1}$$

where M_{tr} is the total number of training data samples, and $h_i = \overline{RUL}_i - RUL_i$, i.e., the difference between the estimated RUL and the true RUL with respect to the i_{th} data point. The early stopping technique and the dropout have been used to avoid overfitting problems.

4.4 Simulations

In this section, we evaluate the performance of the proposed frameworks by employing the C-MAPSS dataset with all the settings and details of section 4.1.

4.4.1 Results

In this subsection, we present various experimental results to evaluate the performance of the proposed frameworks for RUL estimation.

4.4.1.1 The RUL Estimation Results

Samples of the RUL prediction results over the four datasets (i.e., FD001-FD004) from the proposed models are presented in Figs. 4.11, and 4.12. For better visualization of the results, in Figs 4.11(a)-(d) testing engines are sorted in ascending order (from small to large). Figs. 4.11(a)-(d) show the prediction results associated with the last recorded data point over the four datasets. It is worth mentioning that the number of test cases in each dataset is different ranging from, 100 testing engines in FD001 and FD003 to 256 and 248 engines in FD002 and FD004, respectively. It is observed that the predicted RUL values closely follow their ground truth. Two key points can be highlighted: (i) First, it can be observed that the accuracy for engines with smaller RUL is noticeably higher. This is particularly important as smaller RUL translates to the closeness of a potential failure, where better accuracies are required

Table 4.2: The results of 30 time window size.

TW=30		HDNN	BiLSTM	GRU	BiGRU
FD001	Score	245	252.915	262.656	**234.406**
	RMSE	13.017	12.848	12.831	**12.108**
FD002	Score	1282.42	1184.925	1070.577	**1063.415**
	RMSE	**15.24**	16.384	16.182	16.233
FD003	Score	287.72	257.857	292.329	**231.727**
	RMSE	12.22	12.972	**12.006**	12.217
FD004	Score	1527.42	**1467.793**	1889.597	1735.845
	RMSE	**18.156**	18.171	19.015	18.67

to perform maintenance actions at optimum times to avoid catastrophic failures, and; (ii) The results shown in Figs. 4.11(b) and (d) are significantly interesting as these are corresponding to the most complex scenarios and typically most of the existing solutions fail to provide reliable results for these two cases.

Figs. 4.12(a)-(d) predicted RUL values for a sample unit of each dataset (selected at random). First, we would like to point out that as the complete failure history of each testing engine is not available for prognostic performance evaluation, the last parts of the engine units lifetime are not included in the figures. The actual RUL values are provided in the dataset, therefore, their associated RUL labels can be computed accordingly. In par with our previous results, it is observed that the proposed frameworks perform well across all four datasets, specially FD002 and FD004 (Figs. 4.12(b) and (d)), which are considered extremely complex scenarios. More interestingly, the proposed frameworks manage to provide accurate RUL estimates values closely following their ground truth when the units are close to failure.

4.4.1.2 The Results with Different Time Window Size

The time window size has a major role in collecting informative features, where the larger the time window size, the more amount of information can be covered, which is the foundation for further feature extraction. In other words, the larger the time window size, the better the RUL prediction. (30) window size is the common value of the window size used in most of the RUL estimation studies that employed C-MAPSS datasets. Table 4.2 shows the results obtained from the proposed frameworks using 30 time window size for FD001 to FD004. In particular, both the RMSE and the score values for FD002 and FD004 are significantly better than the values reported in the literature. To further show the efficiency of our proposed frameworks, we examine the effects of the window size on the results. As such, we used a smaller window size of 15. Table 4.3 shows that the results obtained from smaller time window size are

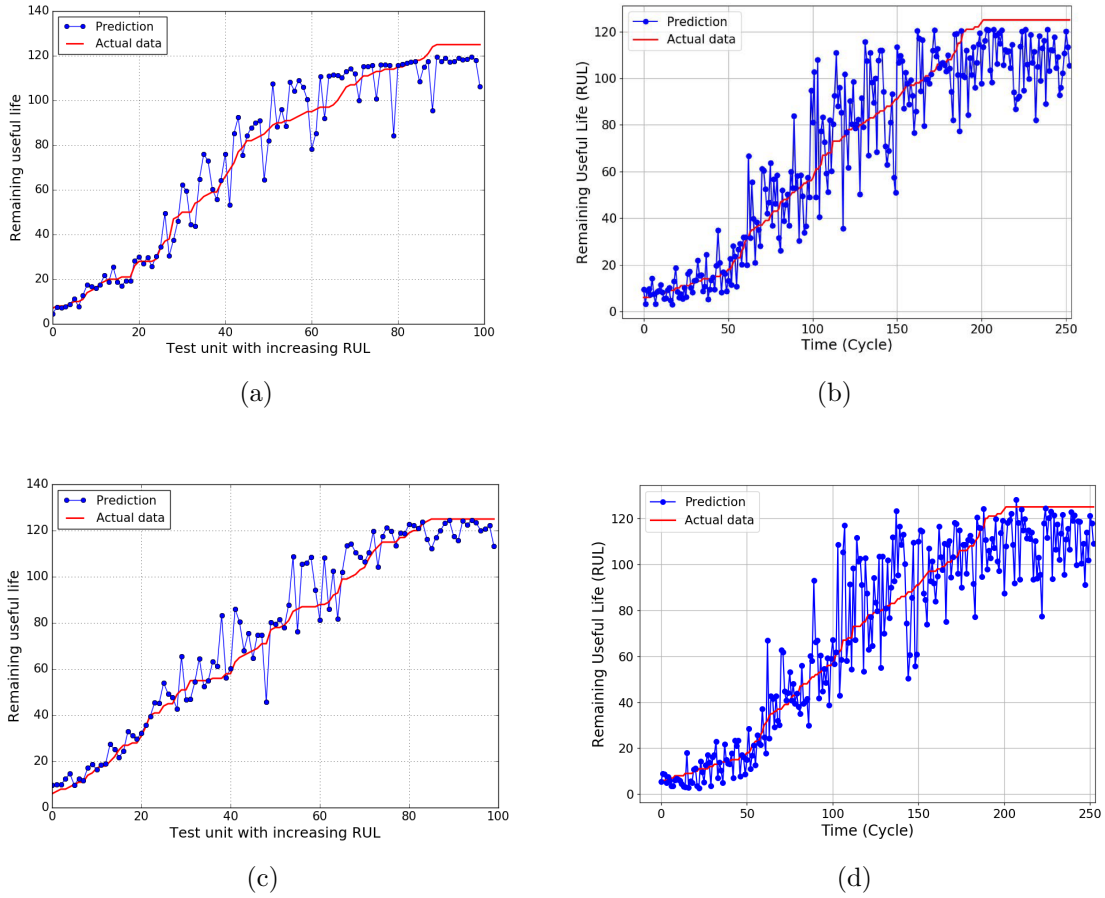


Figure 4.11: The prediction for the last recorded data point of different testing engine units in FD001-FD004. (a) Prediction for the 100 testing engine units in FD001 for the BiGRU model. (b) Prediction for the 256 testing engine units in FD002 for BiLSTM model (c) Prediction for the 100 testing engine units in FD003 for the HDNN model. (d) Prediction for the 248 testing engine units in FD004 for the GRU model.

Table 4.3: The results of 15 time window size.

TW=15		**HDNN**	**BiLSTM**	**GRU**	**BiGRU**
FD001	**Score**	849.517	647.688	770.360	**637.495**
	RMSE	17.095	16.92	17.223	**17.005**
FD002	**Score**	1966.415	1824.353	1511.547	**1489.844**
	RMSE	**17.411**	19.715	19.048	19.060
FD003	**Score**	752.363	884.505	841.322	**657.177**
	RMSE	16.078	18.160	**17.526**	17.824
FD004	**Score**	2549.64	**2805.555**	3183.081	3004.6
	RMSE	**20.265**	20.22	21.399	20.769

again superior. It is worth mentioning that as expected the score and RMSE results obtained using a smaller window size of 15 (Table 4.3) are increased as compared to a larger window size of 30 (Table 4.2).

Table 4.4: Comparison of results obtained from the proposed models with those reported by 7 state-of-the-art methodologies.

TW=30		**HDNN**	**BiLSTM**	**GRU**	**BiGRU**	**CapsNet [332]**	**DAG [304]**	**CNNTW [333]**	**DBiLSTM [225]**	**D-LSTM [214]**	**DCNN [156]**	**TEMPCO [299]**
FD001	**Score**	245	252.915	262.656	234.406	276.34	229	**224.16**	295	338	273.7	1220
	RMSE	13.017	12.848	12.831	12.108	12.58	**11.96**	12.18	13.65	16.14	12.61	23.57
FD002	**Score**	1282.42	1184.925	1070.577	**1063.415**	1229.72	2730	2494.35	4130	4450	10412	3100
	RMSE	**15.24**	16.384	16.182	16.233	16.30	20.34	19.58	23.18	24.49	22.36	20.45
FD003	**Score**	287.72	257.857	292.329	**231.727**	283.81	535	1279.85	317	852	284.1	1300
	RMSE	12.22	12.972	12.006	12.217	**11.71**	12.46	15.67	12.74	16.18	12.64	21.17
FD004	**Score**	1527.42	**1467.793**	1889.597	1735.845	2625.64	3370	4523.32	5430	5550	12466	4000
	RMSE	**18.156**	18.171	19.015	18.67	18.96	22.43	22.12	24.86	28.17	23.31	21.03

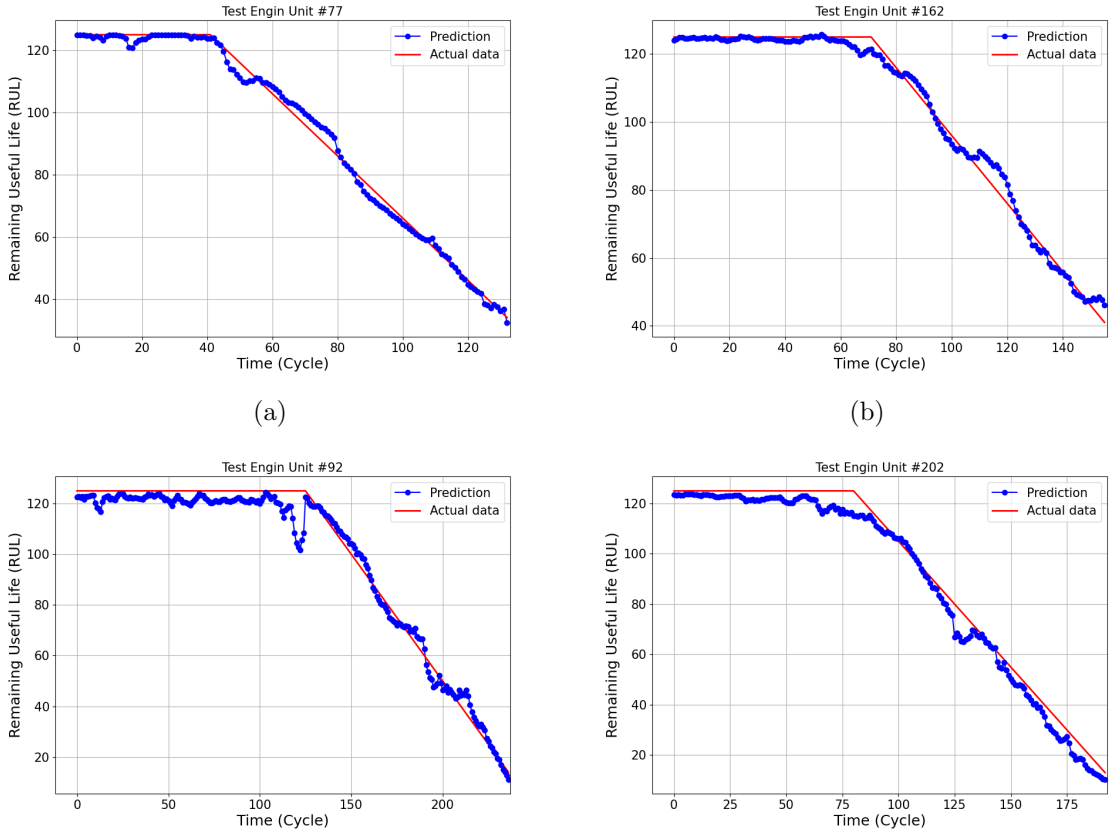


Figure 4.12: Different examples of lifetime RUL prediction for a sample engine unit of each dataset. (a) The testing engine Unit 77 in FD001 for the HDNN model. (b) The testing engine Unit 162 in FD002 for the BiGRU model. (c) The testing engine Unit 92 in FD003 for the GRU model. (d) The testing engine Unit 202 in FD004 for the BiLSTM model.

Table 4.5: Performance comparison with 4 methods that do not use the piece-wise linear degradation model with the proposed frameworks

Dataset		**HDNN**	**BiLSTM**	**GRU**	**BiGRU**	**ResCNN [334]**	**Semi-S [335]**	**BiLSTM-ED [256]**	**Rulclipper [315]**
FD001	**Score**	245	252.915	262.656	234.406	**212.48**	231	273	216
	RMSE	13.017	12.848	12.831	**12.108**	12.16	12.56	14.74	13.27
FD002	**Score**	1282.42	1184.925	1070.577	**1063.415**	2087.77	3366	3099	2796
	RMSE	**15.24**	16.384	16.182	16.233	20.85	22.73	22.07	22.89
FD003	**Score**	287.72	257.857	292.329	231.727	**180.76**	251	574	317
	RMSE	12.22	12.972	**12.006**	12.217	12.01	12.10	17.48	16
FD004	**Score**	1527.42	**1467.793**	1889.597	1735.845	3400	2840	3202	3132
	RMSE	**18.156**	18.171	19.015	18.67	24.97	22.66	23.49	24.33

4.4.1.3 Comparison with Existing Methods

In the previous section, we have presented the results obtained from implementing the proposed frameworks. Here, to better position the reported results within the existing

literature we present comprehensive comparison results with existing state-of-the-art RUL estimation solutions. Most of such existing studies on the RUL estimation problem have used the C-MAPSS datasets for evaluation and comparison, which is the same dataset used in our proposed models. Table 4.4 illustrates the results reported by the latest researches and is compared with the proposed frameworks. The solutions included for comparison purposes are ranging from simple deep learning structures using one type of deep learning architecture such as CNN in [156, 333] or LSTM in [214] to more complex designs such as BLSTM in [225] or others that are in the same level of complexity such as the approach in [304], in addition to some other different techniques as in [299, 332]. Although some of our proposed architectures might appear more complicated in design than some of the aforementioned techniques, they are not intricate to a degree beyond which our comparison might not easily prove to be relevant and fair. In other words, the number of parameters involved in the design of the proposed architectures is much less than some of the approaches included in our comparison (the number of parameters involved in our designs is ranging from about $48,000$ in the GRU approach to around $88,000$ in $BiGRU$, while about $85,000$ and more than 2 million parameters are utilized in [225] and [299], respectively). Furthermore, when comparing our results to the others, one may think that our applied improvement did not conclude the best outcomes, however, with regard to our pioneer "HDNN" model by the time it was published, its results were the best in the sense that it was the first published hybrid parallel model with almost the best achieved results in the literature. While the other approaches we are comparing ours with did not conclude as good results as ours except two approaches of them [304,332] which came up with close results to ours. It is worth mentioning here that these two approaches were published (submitted and accepted) several months after ours.

As for the other three approaches of ours (BiLSTM, GRU, and BiGRU) even though, they were published as supportive approaches besides the main ones within different research papers, they achieved great results when compared with their counterparts. In addition, our comparison was designed to be simple and similar to those used in the literature [156, 192, 214, 225, 299, 313, 336], regardless of other designs and various complexity degrees. Furthermore, for an equitable comparison, we classified the RUL prediction approaches into two categories: (i) The piece-wise linear degradation approaches, as it is still the natural common choice in the literature, and;

(ii) Solutions developed without using the piece-wise linear degradation model. Table 4.4 displays the RUL prediction results obtained from techniques that employ the piece-wise linear degradation model. Table 4.5 compares the results of the proposed models with methodologies that use alternative approaches to the piece-wise linear degradation model. The proposed methods achieve competitive results across most of the datasets. Despite the good results of (ResCNN) method, it is important to note that (ResCNN) has a number of limitations as compared with the piece-wise linear degradation method, such as lacking the imbalance of signal data, that affects the prediction accuracy [334].

4.4.1.4 Monte Carlo Simulations with Additive Noise

Effectiveness and robustness of proposed methods on RUL prediction are further evaluated through Monte Carlo (MC) simulations. In particular, 100 MC runs are performed where at each run, sensor measurements are contaminated by additive stochastic noise based on a specific level of signal to noise ratio (SNR). From the four proposed models, we selected two models to evaluate using MC simulations. Table 4.6 presents the MC simulation results obtained from the proposed HDNN framework using noisy sensor measurements (based on different SNR values which are 30, 25, and 20 dB). While table 4.7 presents the MC simulation results obtained from the proposed BiGRU framework using noisy sensor measurements (based on different SNR values which are 30, 25, and 20 dB). It is noteworthy that our proposed approaches are the only approaches in the literature that fully evaluated using MC simulations based on different levels of SNR. The achieved RMSE and the score values show a remarkably stable performance of the proposed models. It can be clearly noted that the algorithms' performance degrades as the value of the SNR reduces especially below 20dB since the training model is based on clean data (not noisy) and hence the model sees fewer variations in the data during training which makes the model not stable enough to deal with a high rate of noise (low SNR values). This motivates further research to improve the proposed models.

Table 4.6: Monte Carlo simulation results of (HDNN) model: (a) The results under SNR =30 dB. (b) The results under SNR =25dB. (c) The results under SNR =20 dB.

HDNN SNR=30dB	Metrics	
	RMSE (mean ± std)	Score (mean ± std)
FD001	13.059 ± 0.118	247.40 ± 5.31
FD002	16.650 ± 0.597	1599.293 ± 238.426
FD003	12.229 ± 0.044	287.914 ± 3.450
FD004	19.831 ± 0.676	2253.937 ± 476.30

a

HDNN SNR=25dB	Metrics	
	RMSE (mean ± std)	Score (mean ± std)
FD001	13.248 ± 0.067	247.68 ± 3.28
FD002	18.241 ± 0.769	2027.03 ± 251.288
FD003	12.816 ± 0.078	278.022 ± 3.701
FD004	21.006 ± 0.916	2228.392 ± 400.042

b

HDNN SNR=20dB	Metrics	
	RMSE (mean ± std)	Score (mean ± std)
FD001	13.261 ± 0.117	248.087 ± 5.431
FD002	19.233 ± 1.773	2701.205 ± 390.1
FD003	12.784 ± 0.146	279.277 ± 7.24
FD004	21.413 ± 1.199	3071.59 ± 560.08

c

Table 4.7: Monte Carlo simulation results of (BiGRU) model: (a) The results under SNR =30 dB. (b) The results under SNR =25dB. (c) The results under SNR =20 dB.

BiGRU SNR=30dB	Metrics	
	RMSE (mean ± std)	Score (mean ± std)
FD001	12.151 ± 0.031	236.004 ± 1.616
FD002	17.382 ± 0.511	1271.335 ± 149.803
FD003	12.241 ± 0.048	232.066 ± 1.902
FD004	19.288 ± 0.562	2039.764 ± 260.368

a

BiGRU SNR=25dB	Metrics	
	RMSE (mean ± std)	Score (mean ± std)
FD001	12.174 ± 0.052	237.105 ± 2.755
FD002	18.243 ± 0.771	1802.528 ± 180.63
FD003	12.28 ± 0.082	234.584 ± 3.112
FD004	21.114 ± 0.882	2278.694 ± 408.126

b

BiGRU SNR=20dB	Metrics	
	RMSE (mean ± std)	Score (mean ± std)
FD001	12.201 ± 0.133	238.571 ± 5.166
FD002	20.852 ± 1.821	2493.663 ± 446.482
FD003	12.304 ± 0.135	237.142 ± 5.609
FD004	21.539 ± 1.67	3099.344 ± 484.758

c

4.5 The Summary

In this chapter, a new important category in the field of RUL estimation models has been introduced, which is the parallel hybrid design with multiple types of DNNs. The proposed models have been designed based on two parallel paths of different NN techniques. Both paths receive the same input data, and each one of them then processing the data in a specific fashion. Consequently, the results from both parallel

paths are fused by the third path that acts as a fusion center designed based on fully connected layers to find the RUL. Four different approaches are proposed by integrating CNN, LSTM, GRU, BLSTM, and BGRU architectures. The proposed methods have been tested on NASA's C-MAPSS dataset that simulates the degrading health of a commercial aero engine. Comparisons with several state-of-the-art methodologies have been conducted and the results demonstrate the outstanding performance of the proposed methods.

Chapter 5

The Noisy and Hybrid Deep Neural networks for Remaining Useful Life Estimation

The RUL estimation frameworks proposed in the previous chapter (Chapter 4) considerably outperformed their state-of-the-art counterparts. There are, however, potential venues to further improve the performance of the proposed frameworks, i.e., to improve on the robustness and the generalization behavior of the models. In other words, the achieved results can be improved by enhancing the learning capabilities of the models (through enhanced training) considering robustness and generalization by design [337]. However, improving the training step to ensure better performance and generalization behavior for a model is a challenging task. This is the case as the training usually introduces issues such as memorizing the training dataset rather than learning the general mapping from inputs to outputs. Furthermore, if the training step is performed based on a small dataset, there is a less chance to properly identify the structure of the input space and its relation to the output. These issues during the training stage can lead to either having an "underfit model" when fewer data points are provided preventing the model to sufficiently learn the problem (i.e., unlearnable mapping function). Alternatively, we can have an "overfit model" which happens when the model memorizes the training dataset. Several approaches have been proposed to address these issues and ensure the robustness and the generalization behavior. For example, regularization [337], weight decay [338] pruning methods [339], information minimization [340], constructive methods [341], dropout 4.3.5,

early stopping 4.3.6, and adopting noisy training, to mention but a few.

Among the aforementioned solutions to improve the training stage, adopting noisy training or adding random noise is an effective approach to increase the robustness of the network. Several studies [337, 342–350] have shown that adding small amounts of noise during the training step contributes to a better generalization behaviour and improved learning capabilities. What makes noisy training one of the most effective techniques, is that it possesses the following unique characteristics: It has a regularization effect [349]; It can be considered as some form of data augmentation [344], and; It can reduce generalization error and reduce the over-fitting problem [337]. Although the most common and widely studied approach is adding noise to the inputs, random noise can be considered during the training phase in other parts of the model, e.g., noise can be added to the weights (adopting weight noise). This strategy is considered as equivalent to the traditional form of regularization, encouraging the stability of learning the general mapping from inputs to outputs [337]. It has also been shown that adopting the weight noise can improve the learning rate [351]. However, some researchers [352] have argued that the weight noise is less effective than the input noise for regression problems, while they showed to have similar effects for the classification problems. Alternatively, noise can be added to the gradients (injecting gradient noise) focusing more on robustness of the optimization process. Injecting gradient noise has shown to be effective to avoid the over-fitting issue and can minimize the training loss by enabling better parameter space exploration. The latter is essential in case of optimizing several layers within a complex structured NN [345]. Finally, noise can be added to the activations and to the outputs [353].

In this chapter, we will employ different noise injecting strategies to further improve the approaches proposed in Chapter 4. The proposed approaches in this section will be evaluated and tested using the same C-MAPSS dataset (described in Sections 4.1- 4.3).

5.1 The Noise Injection Strategy

Noise injection during the training stage is a generic strategy that can be implemented, no matter what type of NN is utilized. Many researches have investigated and shown that injecting noise is an effective and successful method to improve the performance of different NN architectures (including MLP, LSTM, GRU, and CNN). Murray and

Edwards [354] have investigated and analyzed the effect of adding noise to the MLP and shown that this method has enhanced the learning ability, generalization, and the network's fault tolerance. Many other studies [355, 356] have shown the same results. Along the same direction, it was shown [237, 357–359] that injecting noise to the RNN in general or the LSTM and BLSTM, will improve their efficiency and increase their learning ability throughout the training process. CNN is no different from the other approaches, where References [360–362] have adopted noise injection and demonstrated potential improvements that can be achieved resulting in state-of-the-art outcomes.

It is worth mentioning that noise injection can be applied regardless of the problem type being tackled. Hence, adding noise to models that handle problems of classification or regression is acceptable. However, some noise injection ways are more effective than others for specific data or problem types [352, 353].

Gaussian Noise Layer: Gaussian noise is the most common form of noise used throughout the training stage in the literature. The Gaussian noise can be provided by using a pseudorandom number generator, however, in our proposed approaches, the noise injection is performed via an alternative and effective technique, which is the inclusion of Gaussian noise layers. A Gaussian noise layer applies additive zero-centered Gaussian noise as a stand alone layer to accomplish the noise injection task. A noise layer's output should be in the same form as that of the input, the only change is the addition of noise to the values.

The Gaussian noise layer can be added and used in different ways and locations of the model. An example is the traditional use of noise when a noise layer is used as an input layer for adding the noise directly to the input variables. In addition, the Gaussian noise layer can be added between the hidden layers within the model. In this case, it could be included before the activation function, or after that to form some sort of noisy activation function. The injection of noise is necessary to have a consistent effect on the neural network model, and this can be done by standardizing input variables or by normalizing them prior to adding the noise [353]. Moreover, the amount of noise injection (standard deviation) is a dynamic hyperparameter. A very small amount of noise does not have any effect, while too much noise leads to poor learning for the mapping function.

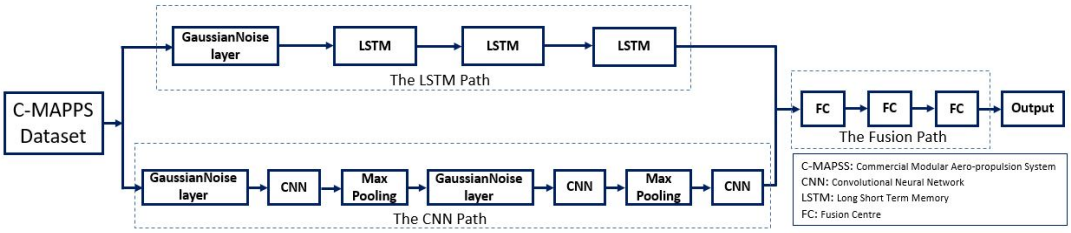


Figure 5.1: The NLSTM framework.

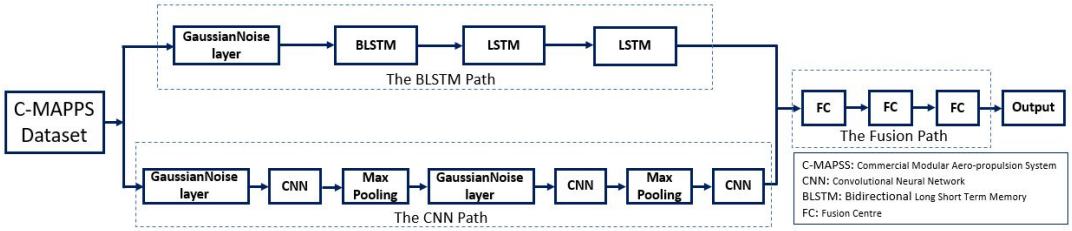


Figure 5.2: The NBLSTM framework.

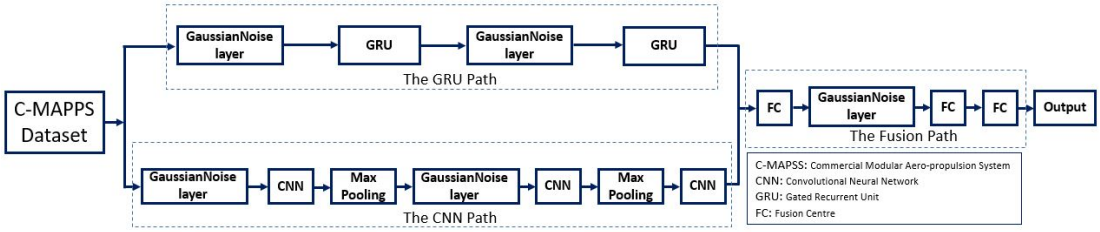


Figure 5.3: The NGRU framework.

5.2 Frameworks of The Proposed Networks

In this section, structures of the proposed frameworks are presented, the presented methods as shown in Figs. 5.1- 5.4, make use of three integrated paths, i.e., the first two paths are in parallel and they are constructed based on several DNN architectures along with different noisy layers. Their combined output is then fed into the third path (fusion center) to integrate the obtained features of the parallel paths to find the RUL. The proposed models can be classified into two groups, based on the way of adopting the noise layers, where the first and the second approaches (NLSTM & NBLSTM) [83] as shown in Figs. 5.1 and 5.2 are using the noise layer as an input layer to the first parallel path, while as input layer and between the hidden layers of the CNN path. The other two approaches (NGRU & NPBGRU) [323, 324] as shown in Figs. 5.3 and 5.4 are using fully noisy architectures as we adopted the noise layer in all the integrated paths. In the following subsections, the main components of the three paths of the proposed frameworks are presented.

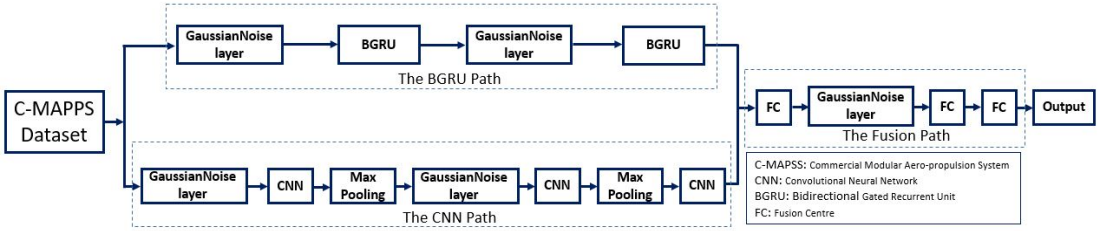


Figure 5.4: The NPBGRU framework.

5.2.1 The First Parallel Path

In the second proposed approach, as shown in Fig. 5.2, the first path is similar to the previous approach as it has three layers for temporal features extraction, preceded by a noisy layer contaminated with zero-mean Gaussian noise with a standard deviation of (0.01). However, the first layer followed the noisy layer, is a BLSTM layer defined by a 32-cell structure with return sequences, a dropout rate of 0.3, and output to be concatenated and then fed to the next LSTM layer. The following layers are two LSTM layers defined by 32 and 64 cell structures, respectively. The associated setting values for the repeating cells used within the LSTM layers are considered to be the same. The third proposed approach, as shown in Fig. 5.3, has utilized two GRU layers for extracting temporal features, each one preceded by a Gaussian noisy layer with zero-mean and standard deviation of (0.01 & 0.1) respectively. 32 and 64 cell structures are adopted for the first and the second GRU layers, respectively. Furthermore, the dropout of 0.3 is used. The related parameter values are assumed to be the same for the repeated cells used in the GRU layers.

Finally, the fourth proposed approach, as shown in Fig. 5.4, has followed the design of the third method, but based on using BGRU. More specifically, the fourth model has two Gaussian noise layers each one followed by one BGRU for extracting temporal features. The BGRU layers are defined by 32 and 64 cell structures, respectively, with return sequences. Furthermore, the dropout of 0.3 is used. The related parameter values are assumed to be the same for the repeated cells used in the BGRU layers.

5.2.2 The Noisy CNN Path (The Second Parallel Path)

The proposed methods have a common design for the second parallel path, which is based on using the CNN technique, as shown in Figs. 5.1- 5.4. The CNN path is employed for extracting another class of features as compared with the first parallel path. This path consists of three CNN layers, the first two of which are both preceded

by Gaussian noise layers that have zero mean and $(0.01, 0.1)$ standard deviations, respectively. In addition, they are followed by max pooling layers. 10 filters of the same configurations, each one with the size of (9×1) are applied for the convolution layers, and a filter of size is (2×1) is selected for the max pooling layers. The last layer within the CNN path consists of one CNN filter of (3×1) dimensions. The first two approaches used *tanh* as an activation function, while the third and fourth approaches used the Rectified Linear Unit (ReLU) as the activation function for all the layers.

5.2.3 The Fusion Path (The Third Path)

This path is designed based on three different fully connected layers, used as a fusion center to conduct the regression task to estimate the RUL. For the first two proposed approaches (NLSTM & NBLSTM) we have not adopted noise injection in this path. The first two FC layers have 100 neurons and use *tanh* activation function in the first approach, while, 103 neurons with *tanh* activation function in the second one. In both models, the third FC layer is designed based on a 1 neuron and the utilized activation function is the Rectified Linear Unit (ReLU).

For the third and the fourth proposed models (NGRU & NPBGRU) one Gaussian noisy layer with zero mean and 0.01 standard deviation is injected between the first two FC layers, and these FC layers are built using 87 and 107 neurons, respectively. The last FC layer has a 1 neuron in both models and all the layers have used the (ReLU) activation function. A dropout rate of 0.3 has been used in all the proposed approaches.

5.3 Noisy Training

. As we have already shown, that adopting noise injection during the training has an effective impact to improve deep neural networks in terms of performance, generalization, and robustness. Additionally, noise injection can be applied regardless of the problem type (classification or regression) being tackled. However, only a few research works have employed noisy training for the problem of RUL estimation. For instance, Chen *et al.* [363] have employed Gaussian noises during the training phase to overcome the issue of over-fitting and enhance the robustness and generalization ability of the network for rolling bearing fault severity identification.

The proposed models, to the best of our knowledge, are the first parallel hybrid DNN models for RUL estimation, that adopting the Gaussian noise layer in the way that we proposed. More specifically, other than using the traditional method of adding noise at the input stage of the model, we proposed two different ways of applying the noise throughout the use of the noise layer. The first way is for the first two proposed models (NLSTM & NBLSTM) as we applied the Gaussian noise layer at the input of each parallel path (First & CNN paths) in addition to another Gaussian noise layer between the first and the second CNN layers in the CNN path. The second way, we called it, a fully noisy training, which is achieved by applying Gaussian noise layers to each of the proposed (NGRU & NPBGRU) models' paths. The injected noise level has a clear effect on the training process and the generalization capability offered by the noisy training [347]. Different combinations of noise levels are investigated during the training and the results on the test data are provided in Section 5.4.1.4.

To train the proposed models, each training set was randomly divided into 85 percent and 15 percent, respectively, as training and validation sets. The validation set was used in each training epoch to assess the model performance. The mean squared error (MSE) is the loss function. The other settings are the same as in Sections 4.2 and 4.3.7.

5.3.1 Grid Search Technique

Machine learning involves data prediction and classification, and to do that, different machine learning models have been employed according to the used datasets. Machine learning models are parameterized to adapt their behavior to a particular problem. These models can have several parameters (internal configuration parameters) and hyperparameter (external configuration parameters). Finding the best combination of parameters can be treated as a search problem. In a machine learning model, "parameters" are the model's internal configuration variables, the values of which are estimated from the given data [364, 365]. During the training process, model parameters are learned from historical training data by optimizing a loss function using certain methods such as gradient descent. Examples of these parameters include the weights and biases in an ANN and the coefficients in linear regression or logical regression.

On the other hand, "Hyperparameters" are considered as the model's external configuration values that cannot be estimated from the training data, but they need

to be defined prior to the training process [366]. Hyperparameters used to optimize the model's performance by affecting the speed and accuracy of the model's learning process. Then, different systems require a different number of hyperparameters. In brief, the hyperparameters are like an algorithm's settings that can be adjusted to improve the model's outcomes. Learning rate, batch-size, number of hidden nodes and layers, and activation function, are examples of the hyperparameters. Knowing the best values of the hyperparameters is not a straightforward process and there is no standard mechanism to do so. However, there are different hyperparameter optimization algorithms among which the grid search, described below, is the commonly used approach.

Grid Search is the most widely used strategy that aims to identify a suitable set of hyperparameters for a specific model. The key idea of the grid search is simply building a model for each possible combination of different hyperparameters and then evaluating those models on the validation dataset in order to select the model that provides the best results [366]. To understand the grid search method, let us consider the following example. Suppose, we have a machine learning model M with hyperparameters $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, and $\boldsymbol{x}_3$. Using the grid search, firstly one needs to define a specific range for the values of all the hyperparameters ($\boldsymbol{x}_1$, $\boldsymbol{x}_2$, and $\boldsymbol{x}_3$). Then, the grid search approach will develop different versions of the underlying model using all the possible combinations of the hyperparameter. This range of values for the hyperparameter is called the "grid". Therefore, if we define the grid as follows:

$$\boldsymbol{x}_1 = [10, 20, 30, 40, 50]$$

$$\boldsymbol{x}_2 = [100, 200, 300, 400, 500]$$

$$\boldsymbol{x}_2 = [15, 25, 45, 65, 85],$$

then the grid search will start with the combination of $[10, 100, 15]$, and ends with $[50, 500, 85]$. The search will be through all the combinations between those sets, which shows one of the main disadvantages of this technique that is computationally intensive and consequently needs a longer time. That would lead to seeking different strategies, however, it is always a good idea to start with a grid search for any problem, as it has the ability to provide strong initial speculations [367].

Table 5.1: Details of the incorporated dataset from C-MAPSS [312].

Dataset	C-MAPSS			
	FD001	**FD002**	**FD003**	**FD004**
Train Trajectories	100	260	100	249
Test Trajectories	100	259	100	248
Conditions	1	6	1	6
Fault Modes	1	1	2	2

5.3.2 Hyperparameters Optimization

For the choice of the hyperparameters such as the number of RNN layers, number of the cells in each layer, number of CNN layers, number of FC layers, and the number of neurons in each layer, batch size, standard deviation value, and dropout rate, we followed two strategies: (i) Grid search, which has a simple concept as we previously explained and easy to implement [368]. During the search, we fix some values such as the number of the RNN layers, CNN layers, FC layers while searching for the rest. The choice of the fixed hyperparameters was based on two objectives: (1) The framework must be minimal in terms of the number of trainable parameters, and; (2) The performance metrics (Section 4.1.3) must be minimized. (ii) The second strategy, which is a common practice in the design of deep neural techniques, and many researches called it "trial-and-error" strategy [369, 370]. As there are no absolute rules for selecting the hyperparameters that function for each dataset and any problem, the values of the hyperparameters have to be determined by trial and error for each specific problem. And that has been achieved by performing several experiments on the train set and observing the results of the approach on the validation set, and the hyperparameters with the best validation prediction performance are considered.

5.4 Simulations

In this section, we evaluate the performance of the proposed frameworks by employing the C-MAPSS dataset with all the settings and details as in section 4.1. Various experiments and comparison results are conducted and the results are reported. Table 5.1 shows the details of the incorporated dataset from C-MAPSS.

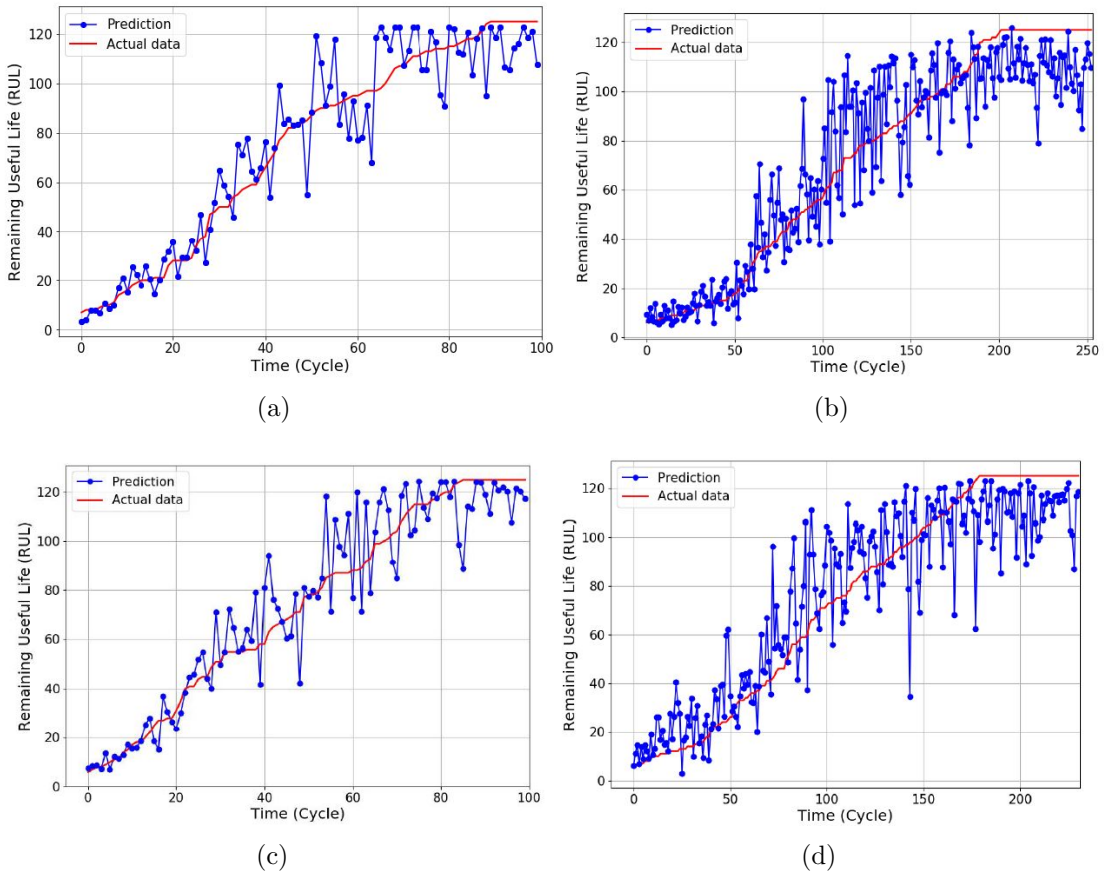


Figure 5.5: The prediction for the last recorded data point of different testing engine units in FD001-FD004. (a) Prediction for the 100 testing engine units in FD001 for the NLSTM model. (b) Prediction for the 256 testing engine units in FD002 for NBLSTM model (c) Prediction for the 100 testing engine units in FD003 for NGRU model. (d) Prediction for the 248 testing engine units in FD004 for the NPBGRU model.

5.4.1 Results

In this subsection, results of different experimental scenarios is reported.

5.4.1.1 The RUL Estimation Results

Figs. 5.5, and 5.6 illustrate the results of the RUL estimation through all the available datasets (i.e., FD001 to FD004) from the proposed models. In Figs. 5.5(a)-(d), In order to analyze the prediction of all engine units of the 4 datasets, we summarize the last recorded measurement sample of all of them, and to better represent the results, sorting is performed in ascending order. Accurate prediction of the RUL is clear, i.e., the blue lines (predicted RUL) are almost overlapping with the red lines (true values).

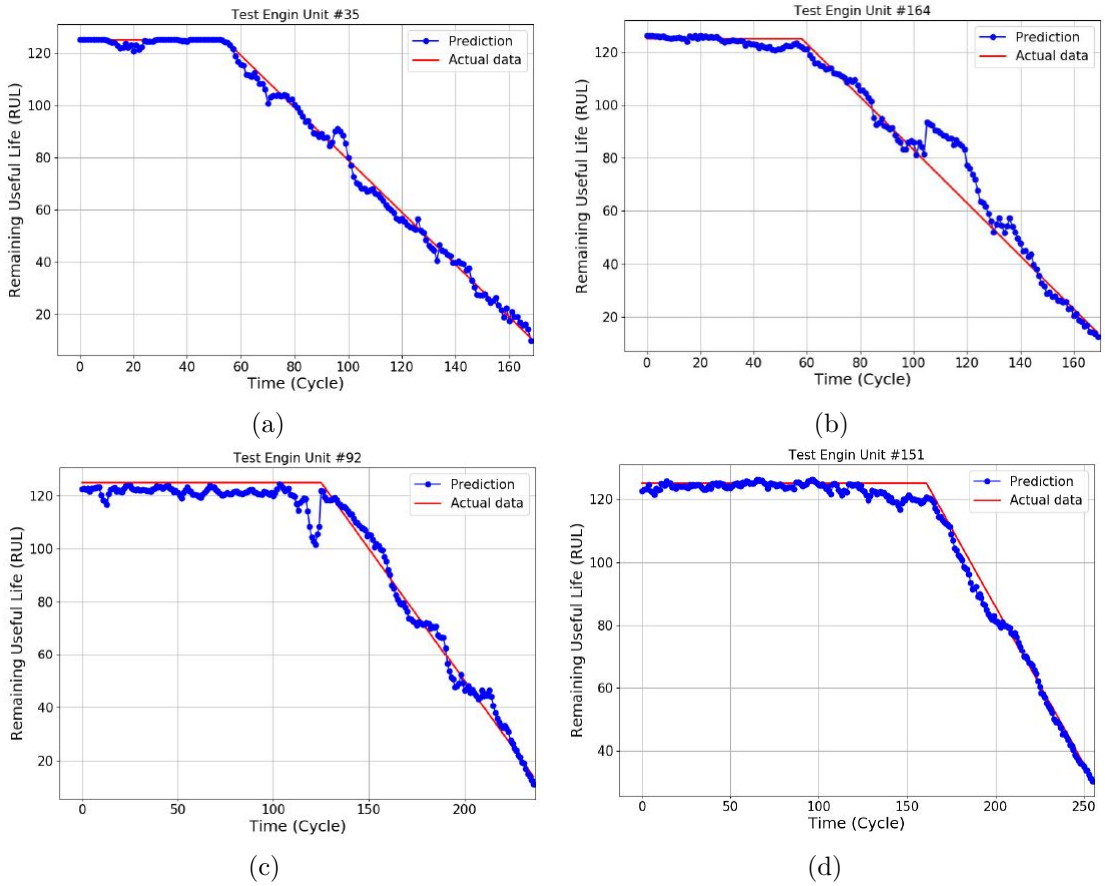


Figure 5.6: Different examples of lifetime RUL prediction for a sample engine unit of each dataset. (a) The testing engine Unit 35 in FD001 for the NLSTM model. (b) The testing engine Unit 164 in FD002 for the NBLSTM model. (c) The testing engine Unit 92 in FD003 for the NGRU model. (d) The testing engine Unit 151 in FD004 for the NPBGRU model.

Furthermore, the prediction accuracy for engines with a smaller RUL can be observed to be significantly higher due to the fact that it is at that point the engine units are close to a possible failure. This is a very important point to be observed for health management in industrial applications, resulting in increased operating performance and safety.

Fig. 5.6 presents RUL prediction results of engine units (35, 164, 92, and 151) selected randomly from each of the 4 datasets in the proposed models. It is noted that the predicted RUL values precisely follow the actual values, which is pointing to the prediction quality of the proposed model.

Table 5.2: Performance results obtained based on window size of length 30.

TW=30		NLSTM	NBLSTM	NGRU	NPBGRU
FD001	Score	247.01	238.34	255.02	**191.80**
	RMSE	13.231	12.321	12.629	**10.44**
FD002	Score	1206.255	1056.629	988.216	**899.762**
	RMSE	16.293	15.038	16.528	**14.651**
FD003	Score	276.514	226.482	217.452	**197.463**
	RMSE	12.782	11.364	11.404	**10.59**
FD004	Score	1434.832	1357.20	1873.684	**1306.5**
	RMSE	17.474	17.752	18.868	**16.78**

Table 5.3: Performance results obtained based on the window size of length 15.

TW=15		NLSTM	NBLSTM	NGRU	NPBGRU
FD001	Score	641.563	626.77	711.642	**572.828**
	RMSE	17.36	16.353	17.238	**15.63**
FD002	Score	1838.621	1645.566	1510.261	**1277.886**
	RMSE	19.29	18.358	18.63	**17.296**
FD003	Score	919.541	794.115	658.102	**577.831**
	RMSE	17.805	16.014	17.442	**15.622**
FD004	Score	2800.148	2705.716	3006.851	**2631.909**
	RMSE	**19.047**	19.755	21.196	19.611

5.4.1.2 The Effects of Different Time Window Size

The main pillar of a high quality prediction model is to extract more informative features, thus using a larger size of time window leads to more precise RUL estimation. Table 5.2 shows the results achieved by the proposed approach using a window size of (30), which is used in most C-MAPSS-based RUL estimation studies. To further investigate the efficacy of the proposed frameworks and the impact of the window size on them, a smaller window size of 15 is utilized, with excellent results as shown in Table 5.3. It should be noted that these results are even better than most of those reported in the literature for the 30 window size.

5.4.1.3 Effects of Operating Conditions and Fault Modes on RUL Estimation Results

Fig. 5.7 illustrates the distribution histogram of the test engines prediction error for the 4 datasets using NPBGRU model, where the error between the predicted RUL and the actual RUL, represented by x-axis, while the y-axis represents the number of engines in the error region. From Fig. 5.7, it can be noted that the error periods of FD001 and FD003 datasets are smaller than those of the FD002 and FD004 datasets. The prediction error associated with FD001 and FD003 is distributed between [-20, 20], whilst, the prediction errors were concentrated between [-40, 40] for the other

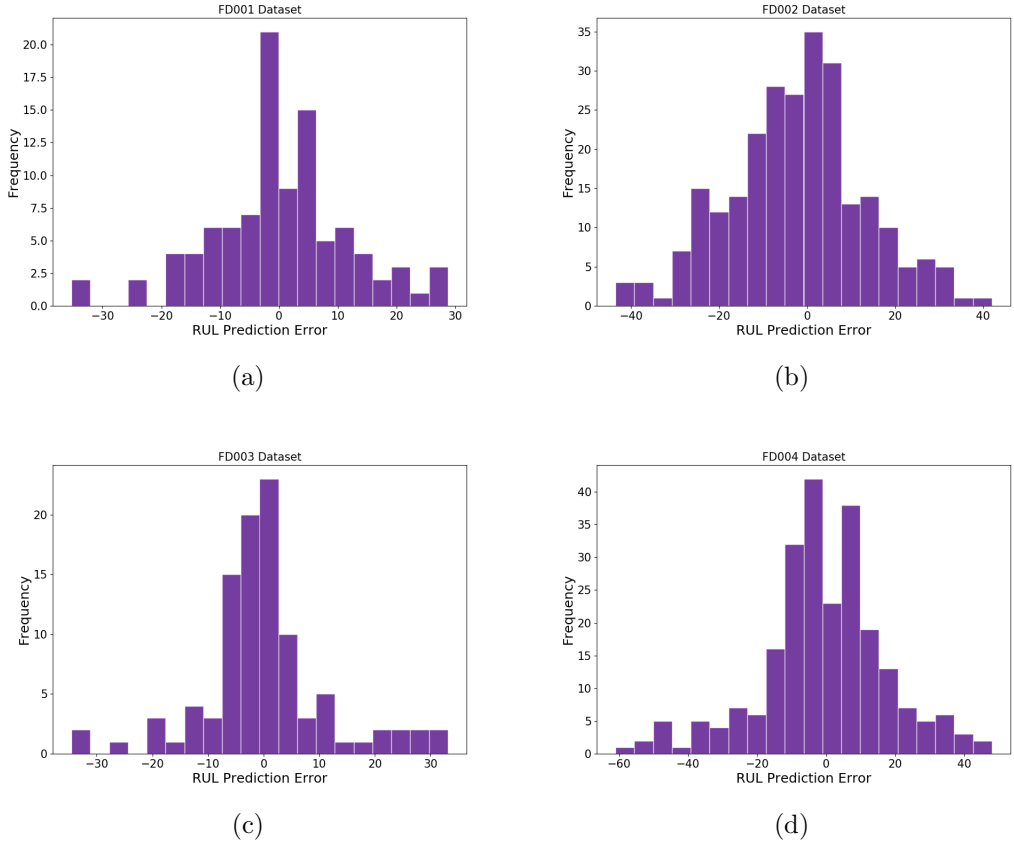


Figure 5.7: Distribution histogram of prediction error for NPBGRU. (a)FD001 Dataset. (b) FD002 Dataset. (c) FD003 Dataset. (d) FD004 Dataset.

two datasets. The reason behind this behavior is that the FD002 and FD004 datasets are simulated based on six operating conditions making them more complex than the other two datasets, leaving it more challenging to predict the RUL. Finally, it is worth noting that we can conclude from the previous observations that the number of operating conditions has a greater impact than the number of fault modes on the reliability of the results.

5.4.1.4 Effects of Training based on Different Noise Levels

It has been shown [347, 371] that noisy training can lead to improvements in terms of performance and generalization of the neural network, especially in the case of conditions mismatching between the training and testing data (e.g., training based on clean data while the test data are noisy [347]). However, adding a large amount of noise will cause the mapping function to be too challenging to learn and then

Table 5.4: The Effects of Training based on Different Noise Levels.

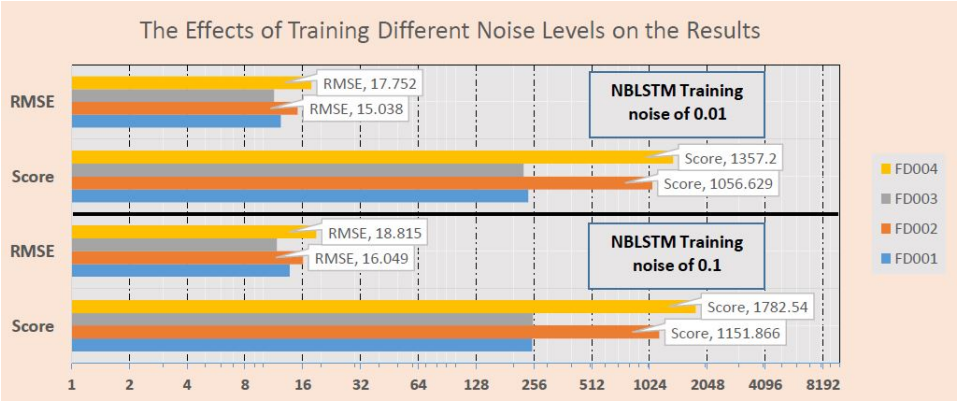
Sub-datasets		NBLSTM (0.01)	NBLSTM (0.1)	NPBGRU (0.01)	NPBGRU (0.1)
FD001	**Score**	238.34	250.934	191.80	209.604
	RMSE	12.321	13.708	10.44	12.014
FD002	**Score**	1056.629	1151.866	899.762	1048.117
	RMSE	15.038	16.049	14.651	15.972
FD003	**Score**	226.482	253.42	197.463	229.415
	RMSE	11.364	11.728	10.59	11.52
FD004	**Score**	1357.20	1782.54	1306.5	1849.184
	RMSE	17.752	18.815	16.78	18.666

Table 5.5: The Effects of Increasing the Noise Level From (0.01 to 0.1).

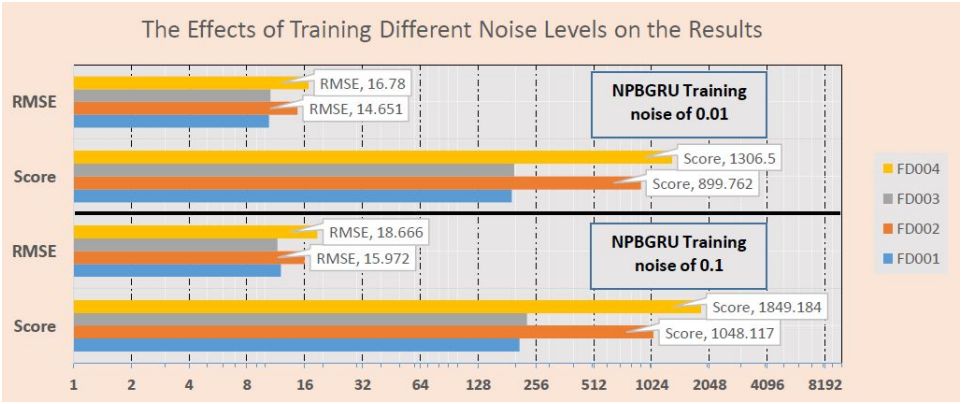
		NBLSTM (%)	NPBGRU (%)
FD001	**Score**	5.284	9.286
	RMSE	11.257	15.077
FD002	**Score**	9.013	16.49
	RMSE	6.723	9.016
FD003	**Score**	11.894	16.181
	RMSE	3.203	8.782
FD004	**Score**	31.340	41.537
	RMSE	5.988	11.24

can yield information loss. It is worthy of note that the proposed models have been trained based on two different Gaussian noise levels (0.01, 0.1). More specifically, the adopted noise levels of the training were (0.01 for the input layers of the parallel paths, and the second layer of FC path) in addition to (0.1 to the second layers of the two parallel paths). Then to show the stability of the proposed model, the noise level (for the input layers of the parallel paths, and the second layer of the FC path) is increased to 10 times (0.1) and the test results were highly competitive.

Table 5.4 and Fig. 5.8 show the effects of adopting different noise levels on the results of (NBLSTM & NPBGRU proposed models), they displayed that increasing the noise has affected mostly on FD004 and approximately in the same level on FD002, which was expected due to the complexity inherent in having six operating conditions and two fault modes. In addition to that, the effects of increasing the noise in the NPBGRU proposed model were clearly more than those in the proposed NBLSTM model as in Table 5.5, and that due to adopting what we called it, the fully noisy training in NPBGRU. One important point to highlight, based on Reference [372], is that most of the publications (70%) used only FD001 dataset to validate their algorithms since it is the easiest and the basic scenario among the rest. This fact raises a big question mark about their actual performance on more challenging datasets such as FD004, not to mention FD004 with additional noise as is the cases of our proposed methods.



(a)



(b)

Figure 5.8: The Effects of Training Different Noise Levels. (a) On the Results of NBLSTM Model. (b) On the Results of NPBGRU Model.

5.4.1.5 Comparison with the Proposed Models of Chapter 4

The proposed models of this chapter are the improved models of chapter 4, where we adopted two different styles of noisy training to improve the proposed models of the previous chapter. The first style was by using the noise layers within the parallel paths as in Fig. 5.2, while the second noisy style was by adopting the noise injection in every path (Fully Noisy) as in Fig. 5.4.

Table 5.6 shows the results of two proposed models before and after adopting the noisy training and the percentages of the improvements. It can be clearly seen that the improvements with the fully noisy style are better than the other noisy training style, however the fully noisy is more sensitive to the change of the noise level as we stated earlier in Subsection 5.4.1.4.

Table 5.6: Comparison with the Proposed Models of Chapter 4: (a) The results of BiLSTM and the NBLSTM. (b) The results of BiGRU and the NPBGRU.

	Without Noise	With Noise	% of Improvement
	BiLSTM	**NBLSTM**	
FD001	252.915	238.34	5.76
	12.848	12.321	4.1
FD002	1184.925	1056.629	10.83
	16.384	15.038	8.22
FD003	257.857	226.482	12.17
	12.972	11.364	12.4
FD004	1467.793	1357.20	7.53
	18.171	17.752	2.31

(a)

	Without Noise	With Noise	% of Improvement
	BiGRU	**NPBGRU**	
FD001	234.406	191.8	18.18
	12.108	10.44	13.78
FD002	1063.415	899.762	15.39
	16.233	14.651	9.75
FD003	231.727	197.463	14.79
	12.217	10.59	13.32
FD004	1735.845	1306.5	24.73
	18.67	16.78	10.12

(b)

5.4.1.6 Comparison with State-of-The-Art Methodologies

A comprehensive comparative study is conducted to validate the effectiveness and superiority of the proposed approaches along with other state-of-the-art RUL prediction methods reported in the literature. In particular, we compare against a group of models that reported the best results in the literature and they followed various styles in the design of their structures. Table 5.7 demonstrates the results that prove that adopting the noisy training can improve the prediction ability, where the proposed models (NPBGRU) and (NBLSTM) have shown impressive outcomes specifically in complex situations such as (FD002 & FD004). More specifically, the $NPBGRU$ proposed model achieved the best reported results within the existing literature in all different cases (FD001- FD004). With regards to the RMSE values, the results are improved as follows 12.71%, 10.12%, 9.56%, and 11.5% for the datasets from FD001 to FD004, respectively. Likewise, with regard to the score value, the proposed methods have shown improvements as follows 14.6%, 26.83%, 30.42%, and 50.24% for the datasets from FD001 to FD004, respectively.

Furthermore, the results of the other proposed models (NBLSTM, NGRU, and NLSTM) were competitive, and in most cases, they were better than the other approaches. Also when it comes to methods that employ techniques other than the piece-wise linear model, the proposed methods achieve remarkably better than most of the available approaches. Table 5.8 particularly shows the comparison between the proposed $NPBGRU$ model and another 4 models that utilized different ways than the piece-wise linear degradation method.

Table 5.7: Performance comparison of the proposed models with 7 state-of-the-art methodologies.

TW=30		**NPBGRU**	**NBLSTM**	**NGRU**	**NLSTM**	**CapsNet [332]**	**DAG [304]**	**CNNTW [333]**	**DBiLSTM [225]**	**D-LSTM [214]**	**DCNN [156]**	**TEMPCO [299]**
FD001	**Score**	**191.8**	238.34	255.02	247.01	276.34	229	224.16	295	338	273.7	1220
	RMSE	**10.44**	12.321	12.629	13.231	12.58	11.96	12.18	13.65	16.14	12.61	23.57
FD002	**Score**	**899.762**	1056.629	988.216	1206.255	1229.72	2730	2494.35	4130	4450	10412	3100
	RMSE	**14.651**	15.038	16.528	16.293	16.30	20.34	19.58	23.18	24.49	22.36	20.45
FD003	**Score**	**197.463**	226.482	217.452	276.514	283.81	535	1279.85	317	852	284.1	1300
	RMSE	**10.59**	11.364	11.404	12.782	11.71	12.46	15.67	12.74	16.18	12.64	21.17
FD004	**Score**	**1306.5**	1357.20	1873.684	1434.832	2625.64	3370	4523.32	5430	5550	12466	4000
	RMSE	**16.78**	17.752	18.868	17.47	18.96	22.43	22.12	24.86	28.17	23.31	21.03

Table 5.8: Comparing performance obtained from four approaches that exclude the need for utilization of the piece-wise linear degradation model with the proposed NBLSTM model based on the C-MAPSS datasets.

Dataset		NPBGRU	ResCNN [334]	Semi-S [335]	BiLSTM-ED [298]	Rulclipper [315]
FD001	**Score**	**191.80**	212.48	231	273	216
	RMSE	**10.44**	12.16	12.56	14.74	13.27
FD002	**Score**	**899.762**	2087.77	3366	3099	2796
	RMSE	**14.651**	20.85	22.73	22.07	22.89
FD003	**Score**	197.463	**180.76**	251	574	317
	RMSE	**10.59**	12.01	12.10	17.48	16
FD004	**Score**	**1306.5**	3400	2840	3202	3132
	RMSE	**16.78**	24.97	22.66	23.49	24.33

Table 5.9: Noisy testing results of (NPBGRU).

SNR=30dB	Metrics	
	RMSE (mean ± std)	**Score (mean ± std)**
FD001	10.48± 0.0301	192.63± 1.602
FD002	15.937± 0.506	1125.103 ± 150.779
FD003	10.591 ± 0.043	197.912 ± 1.814
FD004	17.488 ± 0.54	1781.951 ± 258.866

5.4.1.7 Performance Evaluation with Additive Noise

When it comes to evaluating the effects of noise and uncertainty in predictive modeling, Monte Carlo simulations are considered as the gold standard. Thus, it has been utilized to produce noisy data with a particular value of Signal-to-Noise Ratio (SNR) to be added to the test data, in order to verify the robustness and the stability of the proposed methods. It must be noted that adding noise during the testing phase may rarely exist in the literature, to the best of our knowledge, there are only two studies that have provided thorough results considering noisy testing for all the datasets [321,322]. The conducted results in Tables 5.9-5.12 (NPBGRU & NBLSTM were selected as sample) prove the robustness of the proposed models against noise, as the RMSE and the score values show exceptional stability for different noise scenarios (SNR of 30 & 20) across all the datasets. Nevertheless, the efficiency of the models against the noise was less in cases of FD002 and FD004, as they are more complex than others. Finally, it is also noteworthy and important to mention that the conducted results even with 20 SNR are still uncommonly stable and better than the majority of the published results for free noise testing.

Table 5.10: Monte Carlo simulation results of (NBLSTM).

SNR=30dB	Metrics	
	RMSE (mean ± std)	**Score (mean ± std)**
FD001	12.325± 0.0343	238.682± 1.96
FD002	16.779 ± 0.539	1293.753 ± 224.666
FD003	11.37 ± 0.043	226.765 ± 2.1
FD004	18.125 ± 0.551	1824.109 ± 264.905

Table 5.11: Noisy testing results of (NPBGRU).

SNR=20dB	Metrics	
	RMSE (mean ± std)	**Score (mean ± std)**
FD001	10.5± 0.11	193.75± 5.09
FD002	18.87 ± 1.614	2417.305 ± 434.161
FD003	10.609 ± 0.106	201.356 ± 5.087
FD004	18.705 ± 1.271	2837.086 ± 483.844

Table 5.12: Monte Carlo simulation results of (NBLSTM).

SNR=20dB	Metrics	
	RMSE (mean ± std)	**Score (mean ± std)**
FD001	12.35± 0.11	240.415± 6.201
FD002	19.222 ± 1.783	2538.112 ± 463.042
FD003	11.42 ± 0.156	228.928 ± 6.691
FD004	20.831 ± 1.312	2885.596 ± 488.744

5.5 The Summary

In this chapter, the approaches proposed in Chapter 4 have been improved by adopting effective noisy training techniques. The main objective is to ensure that the network is less likely to memorize the training dataset and learn the general mapping from inputs to outputs instead. This will in turn result in improved network stability and thus less generalization error, high robustness, and faster learning. To achieve the noisy training, the noise injection strategy has been implemented by adding Gaussian noise layers in all the paths of the proposed approaches in different styles and levels. The proposed models are experimentally validated using NASA's C-MAPSS dataset, and many experiments based on different scenarios are conducted, such as investigating the effects of operating conditions and fault modes, training based on different noise levels, different time window sizes (15 and 30), and experimenting the effects of adding noise during the testing phase. The proposed models are also compared with state-of-the-art prognostic approaches and exhibited outstanding results with remarkable stability and high robustness against potential uncertainties.

Chapter 6

Multipath Parallel Hybrid Deep Neural Networks

As discussed in Chapters 4 and 5, the proposed parallel hybrid models of different deep neural network architectures achieved significantly appreciable performances in the field of RUL estimation. The success of these models stemmed from a profound concept [353], i.e., "the more informative features that you collect, the better the results you will achieve," as extracted features represent the characteristics of an observed phenomenon and characterize the underlying problem from the available data. The effectiveness of all machine learning algorithms literally depends on how you present the data. Hence, the purpose of extracting more features is to provide more knowledge about the available data and then enhance the learning process of the data attributes in addition to learn the features themselves. Capitalizing on these concepts and to achieve better prediction results, this chapter goes beyond the conventional structure of hybrid models with two parallel paths and develops/proposes integrated multipath parallel hybrid frameworks. In this regard, we capitalize on thesis developments in Chapters 4 and 5 and construct the multipath frameworks based on the most successful classes of the artificial neural networks. The models proposed in this chapter consist of three parallel paths and the outputs of these paths are then combined by a fusion center to find the RUL.

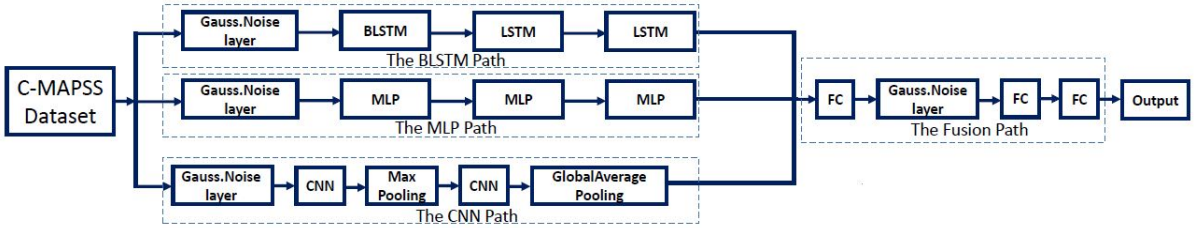


Figure 6.1: The NMPM framework.

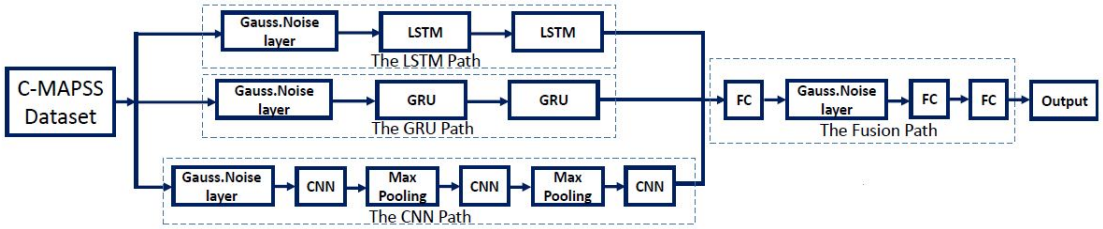


Figure 6.2: The MPHD framework.

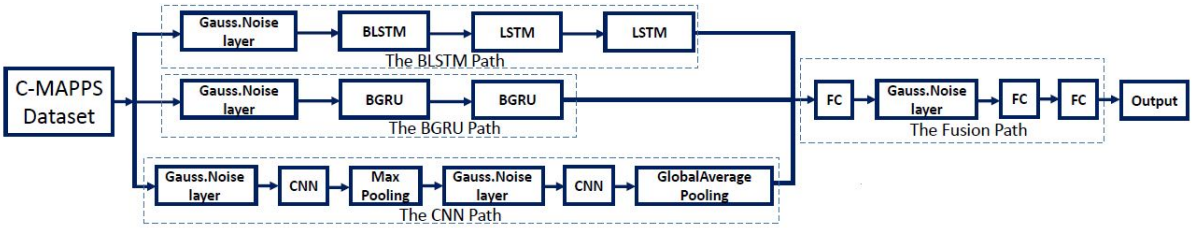


Figure 6.3: The NPHM framework.

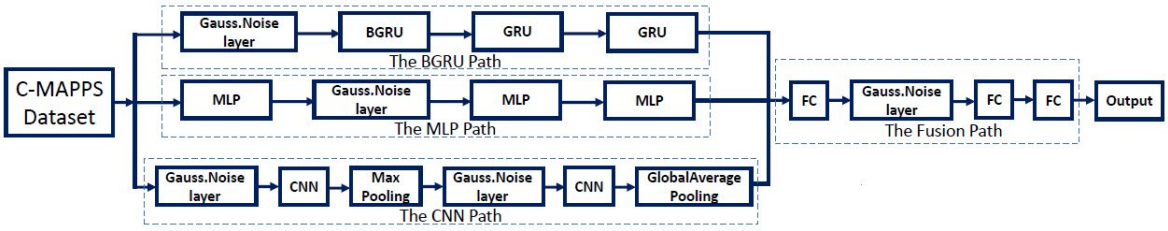


Figure 6.4: The TDHA framework.

6.1 Frameworks of The Proposed Networks

In this section, we present the overall structure of the proposed multipath frameworks. As shown in Figs. 6.1- 6.4, the proposed multipath models make use of four integrated paths, i.e., the first three paths are in parallel and constructed based on different DNN architectures along with different noisy layers. Their combined output is then fed into the fourth path (fusion center) to integrate the extracted features to form the RUL predictions. The fusion path is designed similarly across the proposed

multipath frameworks. Furthermore, the uses of the injected noise are different in these models, especially in the proposed NPHM and TDHA frameworks. In the following subsections, we elaborate on the main components of the four paths of the proposed multipath frameworks.

6.1.1 The First Parallel Path

In each proposed model, different DNN architectures along with specific settings have been utilized. As can be seen in Fig. 6.1, in the first proposed model, referred to as the NMPM, the first parallel path is designed based on the following four layers:

- The first layer is a Gaussian noise layer that has a zero-mean and standard deviation of (0.01).
- The second layer is the $BLSTM$ layer defined by a 16-cell structure with return sequences.
- The third and the fourth components are two $LSTM$ layers each defined by a 10 cell structure.

The repeating cells within the $LSTM$ layers have the same structure and parameter values.

The MPHD Framework: As shown in Fig. 6.2, the proposed MPHD approach has the first path started with a noisy layer contaminated with zero-mean Gaussian noise with a standard deviation of (0.01). The noisy layer is followed by two $LSTM$ layers defined by a 10 and 28 cell structure, respectively. The associated setting values for the repeating cells used within the $LSTM$ layers are considered to be the same.

The NPHM Framework: Fig. 6.3 shows the proposed NPHM framework. The first path of the NPHM architecture, similar to the NMPM approach, has three layers for temporal features extraction, preceded by a noisy layer contaminated with zero-mean Gaussian noise with a standard deviation of (0.01). The three layers started with a $BLSTM$ layer defined by a 16-cell structure with return sequences. The BLSTM component is followed by two $LSTM$ layers each defined by 10 cell structure. The repeating cells within the $LSTM$ layer have the same structure and parameter values.

The TDHA Framework. As shown in Fig. 6.4, the first parallel path in the TDHA framework is designed based on the following four layers:

- The first layer is a Gaussian noise layer that has a zero-mean and standard deviation of (0.01).

- The second layer is a *BGRU* layer defined by a 16-cell structure with return sequences.
- The third and the fourth components are two *GRU* layers each one defined by 10 cell structures. The repeating cells within the *GRU* layer have the same structure and parameter values.

Finally, a batch normalization layer is used within the TDHA framework for accelerating and improving the learning process [373]. This completes the description of the first path of the proposed four multipath frameworks.

6.1.2 The Second Parallel Path

The NMPM and TDHA methods have a common design for the second parallel path, which is based on the MLP technique. As shown in Figs. 6.1 and 6.4, the MLP path is employed for extracting another class of features as compared to the first parallel path. This path consists of three MLP layers, in addition to one noisy layer contaminated with zero-mean Gaussian noise and a standard deviation of (0.01). In the NMPM model, this path starts with the noisy layer, while in the TDHA model, the path starts with the MLP layer followed by the noise layer. All the MLP layers in both models are designed based on the same settings, which are 30, 27, and 10 respectively. A dropout rate of 0.15 and $ReLU$ activation function is used in both models.

The second path in the MPHD and NPHM models is developed based on the *GRU* and the *BGRU* architectures, respectively. In both models, the path starts with a Gaussian noise layer of zero mean and a standard deviation of 0.01. The noisy layer is then followed by two *GRU* layers defined by a 10 and 28 cell structure, respectively, in the MPHD model (Fig. 6.2). In the NPHM framework, after the noisy layer, two *BGRU* layers defined by a 20 and 20 cell structure, respectively, are incorporated as shown in Fig. 6.3.

6.1.3 The Third Parallel Path

In all the models, CNN architecture is used to construct the third parallel path. The rationale behind using CNN in all the four models is its exceptional capability to extract spatial and temporal features.

In the NMPM approach, the third path starts with a Gaussian noise layer of zero mean and (0.01) standard deviation. This layer is followed by a CNN layer that has 10 filters of size (11×1), and then one max pooling layer that has (2×1) filter is utilized. A second CNN layer is then incorporated that has 100 filters of size (11×1). The last layer of this path is a global average pooling layer. The ReLU is used as the activation function for all the CNN layers.

In the MPHD framework, the first four layers of the third parallel path have the same settings and design as those in the NMPM model. However, the path ends with two layers (one max pooling layer that has (2×1) filter, and one CNN layer of 1 filter of size (3×1). The third parallel path in both NPHM and TDHA models has a common design based on two CNN layers, two Gaussian noise layers, one max pooling layer, and one global average pooling layer. Where, the path starts with a Gaussian noise layer of zero mean and (0.01) standard deviation followed by a CNN layer of 10 filters with the size of (11×1). Afterward, we have a max pooling layer that has (2×1) filter, followed by another Gaussian noise layer of zero mean and (0.1) standard deviation. This is then appended by another CNN layer that is built based on 40 filters of size (11×1). The ReLU is the activation function for all the CNN layers. Finally, the last layer of the third path in NPHM and TDHA models is a global average pooling layer.

6.1.4 The Fusion Path (The Fourth Path)

This path is designed to be a fusion center to conduct the regression task for the RUL estimation. All the proposed models have a common design that is based on three fully connected layers and one Gaussian noisy layer with zero mean and 0.01 standard deviation that is injected between the first two FC layers. The first two FC layers are built in both (NMPM and NPHM) by using 100 and 101 neurons, respectively. While, 111 and 111 for the proposed MPHD, in addition to 117 and 117 in the proposed TDHA. The last FC layer in all the proposed models has a 1 neuron. All the layers have used the (ReLU) activation function. A dropout rate of 0.3 has been used in all the proposed approaches.

6.2 Noisy Training

For the training, the same "Noisy Training" technique based on the concept of noise injection is utilized as introduced in Section 5.3. We have implemented the noisy training for all of the four proposed approaches. For training the fourth approach, the TDHA framework, batch normalization [373, 374] is used additionally and has improved the outcomes. For the values of the hyperparameters, we adopted the same strategies as described in Section 5.3.

6.2.1 Batch Normalization

Batch normalization is one of the most successful technological developments in the area of deep learning [374] that allows deep neural networks to be trained more efficiently and stably. The basic concept behind batch normalization is to normalize each layer's input in the network and not just the input layer [375]. This is performed in addition to the re-parametrization of the underlying optimization problem such that it is more stable and smoother. The objective is to make the gradients more reliable and predictive by enabling the training algorithm to take larger steps without running the risk of facing vanishing gradient issue (which can result in unexpected changes) or the exploding gradients problem [374]. Hence, the batch normalization has many advantages for training deep neural networks including:

(a) Stabilizing the learning process and reducing the number of training epochs needed to train the model. This in turn results in accelerating the training process [353].
(b) Adding robustness to various hyperparameter settings such as the learning rate and initialization scheme.
(c) Preventing the exploding and vanishing gradients problems.
(d) Reducing the generalization error.

It is worthwhile to mention that the batch normalization may not work perfectly with all deep learning architectures due to certain factors such as the model design layout, batch size, and learning type (online learning vs. batch learning) [376–378]. In the proposed TDHA model, the batch normalization layer has been utilized between the BGRU layer (the second layer) and the first GRU layer (the third layer).

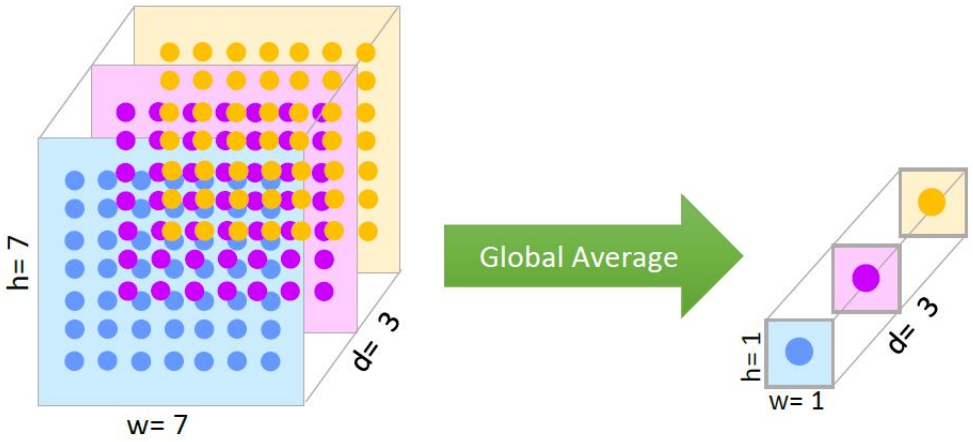


Figure 6.5: The Global Average Pooling Technique.

Table 6.1: Details of the incorporated dataset from C-MAPSS [312].

Dataset	C-MAPSS			
	FD001	**FD002**	**FD003**	**FD004**
Train Trajectories	100	260	100	249
Test Trajectories	100	259	100	248
Conditions	1	6	1	6
Fault Modes	1	1	2	2

6.2.2 Global Average Pooling

Generally speaking, pooling layers are downsampling approaches that summarize the available features in patches of the feature map. Similar to max pooling layers, the global average pooling layers are used to minimize the overfitting issues. However, the global average performs a special type of dimensionality reduction by downsampling the entire feature map to a single value, which is the average output of each feature map in the previous layer [379]. Let the feature map dimensions to be $(h \times w \times d)$ then the global average pooling will reduce the size to $(1 \times 1 \times d)$ dimensions as shown in Fig. 6.5. The global average pooling layers reduce each $(h \times w)$ feature map to a single value by taking the average of all $h \times w$ values. The global average pooling does not have any trainable parameters to optimize, therefore, overfitting at this layer is avoided.

6.3 Simulations

In this section, we use the same dataset (details of which is shown in Table 6.1) as described in Chapters 4 and 5 to evaluate the proposed multipath frameworks.

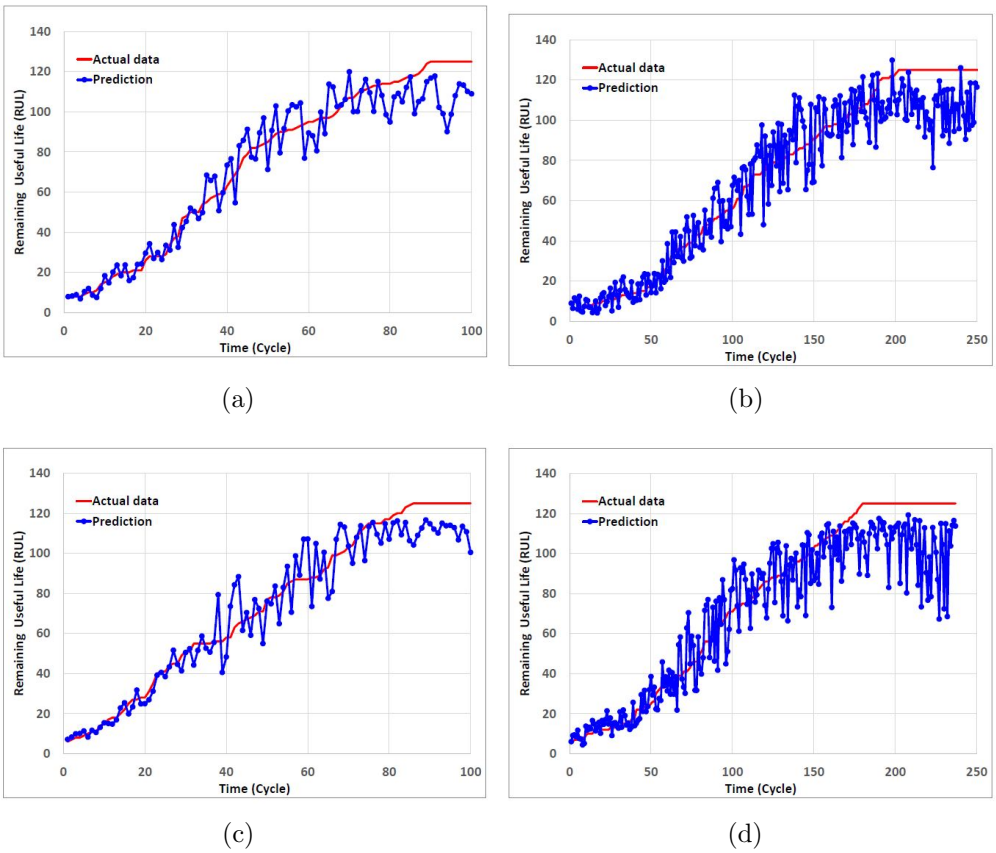


Figure 6.6: The prediction for the last recorded data point of different testing engine units in FD001-FD004. (a) Prediction for the 100 testing engine units in FD001 for TDHA model. (b) Prediction for the 256 testing engine units in FD002 for MPHD model (c) Prediction for the 100 testing engine units in FD003 for the NPHM model. (d) Prediction for the 248 testing engine units in FD004 for NMPM model.

6.3.1 Results

In this section, different comparisons and results are reported to evaluate the proposed multipath frameworks. In the computation of the following reported results, we use data normalization, evaluation metrics, and RUL target function similar to the ones described in Chapters 4 and 5.

6.3.1.1 The RUL Estimation Results

As stated earlier, the main objective of this research is to demonstrate the framework's capability to perform highly accurate prognostic. Figs. 6.6, and 6.7 illustrate the results of the RUL estimation through all the available datasets (i.e., FD001 to FD004). Applying the same procedures of chapters 4 and 5, Figs. 6.6(a)-(d), show

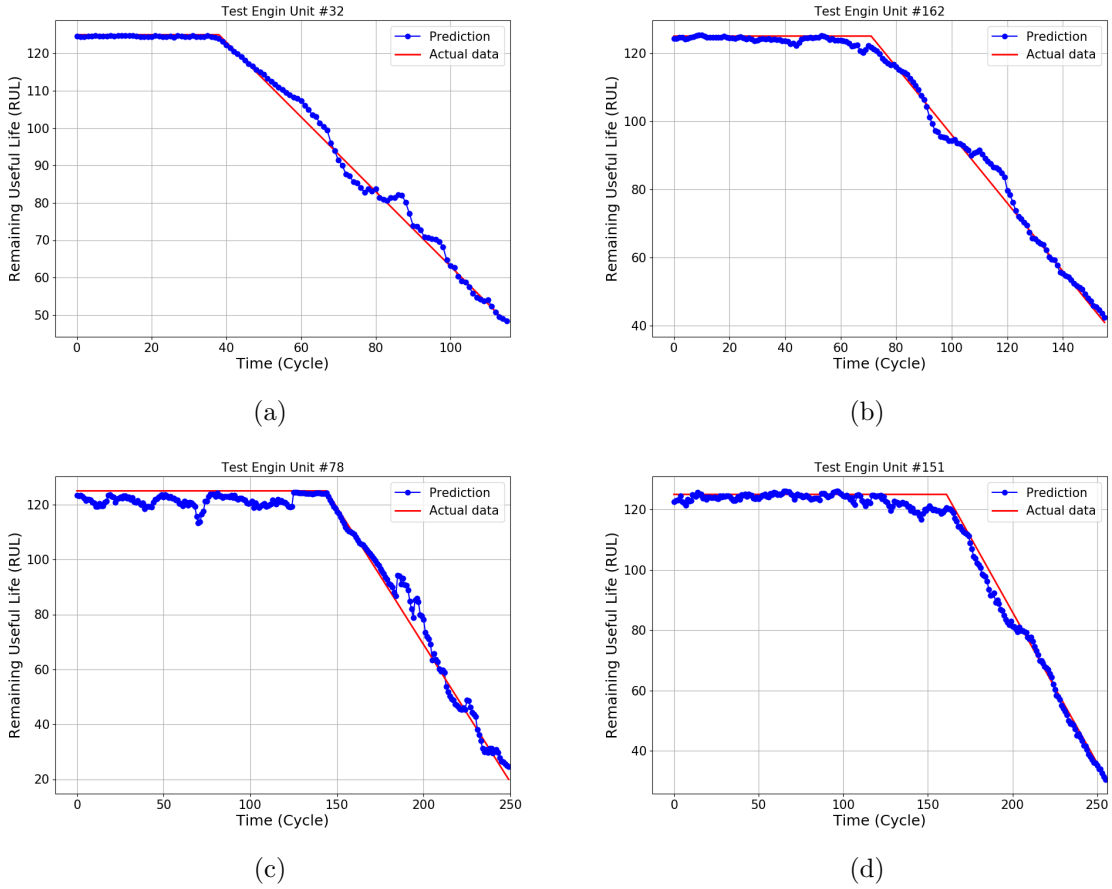


Figure 6.7: Different examples of lifetime RUL prediction for a sample engine unit of each dataset. (a) The testing engine Unit 32 in FD001 for TDHA model. (b) The testing engine Unit 162 in FD002 for the MPHD model. (c) The testing engine Unit 78 in FD003 for the NPHM model. (d) The testing engine Unit 151 in FD004 for NMPM model.

the prediction results corresponding to the last recorded measurement sample based on all the available 4 datasets (sorting in ascending order is performed for better visualization of the results). It could be noted that the predicted RUL values closely follow their corresponding truth values, and the results reflected the capabilities of the developed multipath hybrid models to deliver excellent prognostic performance, especially, in the region where the RUL value is small when the engine unit is working close to failure and the accurate prognostic is highly needed. It is important to highlight that the results are shown in Figs. 6.6(b) and (d) are noteworthy, as they belong to the two challenging complex datasets (FD002 & FD004) where reliable outcomes are not commonly reported in the literature for these two scenarios.

Furthermore, Figs. 6.7(a)-(d) illustrate the predicted RUL values from sample

Table 6.2: Performance results obtained based on the window size of length 30.

TW=30		MPHD	NPHM	NMPM	TDHA
FD001	Score	**177.824**	182.597	188.63	180.71
	RMSE	**10.003**	10.272	10.315	10.118
FD002	Score	798.021	846.083	876.16	**793.064**
	RMSE	**13.856**	14.164	14.322	14.009
FD003	Score	**176.366**	179.851	189.381	177.092
	RMSE	10.376	**10.226**	10.441	10.233
FD004	Score	1148.19	944.514	962.403	**939.270**
	RMSE	16.436	15.979	16.156	**14.90**

Table 6.3: Performance results obtained based on the window size of length 15.

TW=15		MPHD	NPHM	NMPM	TDHA
FD001	Score	473.011	480.643	485.3	471.436
	RMSE	13.374	13.772	13.866	13.436
FD002	Score	1003.406	1058.44	1103.801	1039.251
	RMSE	15.395	15.855	15.891	15.548
FD003	Score	441.91	459.029	466.733	447.082
	RMSE	14.024	13.711	14.001	13.883
FD004	Score	2031.952	1687.331	1759.477	1655.608
	RMSE	18.637	17.841	18.158	17.418

units (32, 162, 78, and 151) selected in a random fashion from each of the 4 available datasets, and the results are inline with our previous results and the proposed models clearly perform well over all four datasets.

6.3.1.2 The Effects of Different Time Window Size

The size of the time window plays an important role in predicting insightful outcomes, it has been proved that the longer the sliding window becomes, the more information it contains [156,316], which is the foundation for further feature extraction, that leads to lower scores and RMSE. As mentioned earlier in the previous chapters, (30) window size is the common value of the window size used in most of the RUL estimation studies that used the C-MAPSS datasets. Table 6.2 presents the results achieved by the proposed frameworks using a 30-time window size. To further, demonstrate the effectiveness and efficiency of the proposed frameworks we investigate the window size effects on the outcomes. As such, we test a smaller window size of 15. Table 6.3 displays that the results from smaller time window size were again superior.

6.3.1.3 The Effect of Training based on Different Noise styles

Table 6.4 shows the effect of adopting noisy training. The performance and the accuracy of the proposed models have been remarkably improved after implementing the

Table 6.4: The Effect of Noisy Training

Dataset		MPHD	WMPHD	NPHM	WNPHM	NMPM	WNMPM	TDHA	WTDHA
FD001	Score	177.824	223.83	182.597	236.483	188.63	236.87	180.70	232.63
	RMSE	10.003	11.66	10.272	12.52	10.315	12.08	10.118	12.1
FD002	Score	798.021	989.59	846.083	1071.3	876.16	1098.68	793.064	1004.06
	RMSE	13.856	15.44	14.164	16.17	14.322	16.23	14.009	15.88
FD003	Score	176.366	211.05	179.851	216.75	189.38	226.97	177.092	212.81
	RMSE	10.376	12.17	10.226	12.061	10.441	12.21	10.233	12.02
FD004	Score	1148.19	1569.15	944.514	1304.08	962.403	1310.4	939.270	1329.05
	RMSE	16.436	18.547	15.979	18.308	16.156	18.11	14.90	17.14

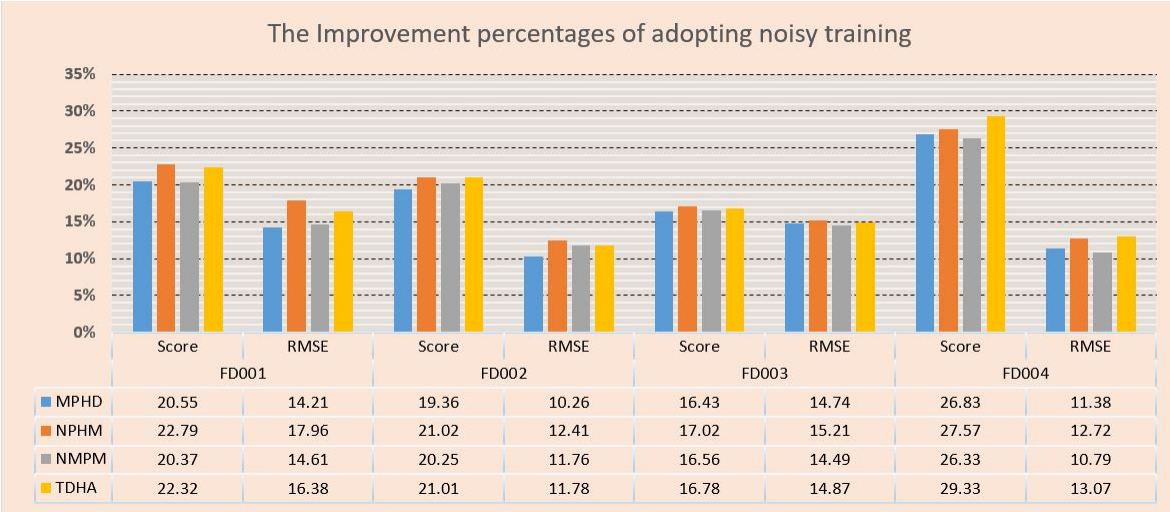


Figure 6.8: The Improvement percentages for adopting noisy training.

effective noisy training technique, and the reason behind that is the powerful ability to enhance network robustness by eliminating the memorization effect of a deep neural network which in turn leads to boost the exploration performance of the learning algorithms [347]. Where (WMPHD, WNPHM, WNMPM, and WTDHA) denote the proposed models but without using the noise. Fig. 6.8 shows the improvement percentages after adopting the noisy training.

6.3.1.4 The Effects of Multiple Parallel Paths

There is no doubt that the features in the used dataset will have a direct impact on the predictive models and the achieved results, where the success of the machine learning algorithms depends on several interrelated factors and properties, however, the most important part in that is how to present the data in order to extract more informative features that describe the inherent structures of the used data, and finally resulting in better accuracy of the proposed model on the unseen data. Up to the

Table 6.5: The Effects of Multiple Parallel Paths.

Dataset		NPHM	OBiGRU	OBiLSTM	TDHA	OBiGRU1	OMLP
FD001	Score	182.592	200.33	227.404	**180.70**	203.552	199.638
	RMSE	10.272	11.46	11.851	**10.118**	11.51	11.50
FD002	Score	846.083	997.868	1061.46	**793.064**	972.003	964.89
	RMSE	14.164	15.23	15.35	**14.009**	15.206	14.91
FD003	Score	179.851	198.41	212.316	**177.092**	199.824	200.79
	RMSE	**10.226**	10.863	11.18	10.233	11.077	10.98
FD004	Score	944.514	1289.76	1347.92	**939.270**	1196.881	1186.714
	RMSE	15.979	16.905	17.52	**14.90**	16.685	16.233

time when this thesis was written, the best achieved results in the literature were obtained using one or two of some of the most common/ successful techniques in the field of RUL estimation (CNN, LSTM, BLSTM, GRU, BGRU, and MLP) within this thesis new different models have been introduced using, for the first time three of those techniques in parallel for each model. The purpose of using three techniques for each model was to ensure the best investment of them in terms of extracting as much informative, relevant, and nonredundant features. Hence, the use of multiple parallel paths based on different DNNs architectures in these newly proposed models improved the models performance, increased their efficiency, and consequently introduced effective solutions that achieved the best results in the literature.

To demonstrate the effectiveness of the proposed methods, we examined the effects of implementing the three parallel paths using different combinations compared with the proposed methods. For each combination, the CNN path was fixed while the other two paths were added alternately. It is worthwhile to mention here that the settings for each path were fixed for each trail following the same settings of the proposed original model. Tables 6.5 shows a sample of the recorded outcomes of two models (NPHM & TDHA) the outcomes show clearly that the best achieved results were those of the models which used three parallel paths.

6.3.1.5 Performance Evaluation with Additive Noise

Monte Carlo simulation has been implemented to produce noisy data based on a specific level of signal-to-noise ratio (SNR) to be added to the test data, in order to evaluate the robustness of the three parallel paths proposed models for the RUL estimation mission. Two different SNR levels, i.e., 30, and 20 dB, are considered each based on 100 Monte Carlo runs. Tables 6.6-6.9 display the results obtained from the proposed (NPHM) & (TDHA) models (were selected as sample) based on 30, and 20 dB of SNR. It can be clearly observed that the multipath parallel models are stable

Table 6.6: Noisy testing results of (NPHM).

SNR=30dB	**Metrics**	
	RMSE (mean ± std)	**Score (mean ± std)**
FD001	10.309 ± 0.0288	183.117 ± 1.277
FD002	15.218 ± 0.41	1036.55 ± 128.639
FD003	10.231 ± 0.036	180.189 ± 1.109
FD004	16.368 ± 0.42	1374 ± 208.953

Table 6.7: Noisy testing results of (TDHA).

SNR=30dB	**Metrics**	
	RMSE (mean ± std)	**Score (mean ± std)**
FD001	10.215± 0.017	182.343 ± 1.087
FD002	14.924 ± 0.33	913.836 ± 104.9
FD003	10.26 ± 0.028	180.02 ± 1.0
FD004	15.857 ± 0.366	1240 ± 198.77

Table 6.8: Monte Carlo simulation results of (NPHM).

SNR=20dB	**Metrics**	
	RMSE (mean ± std)	**Score (mean ± std)**
FD001	10.313 ± 0.106	184.07 ± 4.11
FD002	17.518 ± 1.35	1794.27 ± 384.66
FD003	10.54 ± 0.097	184.68 ± 4.709
FD004	18.15 ± 1.24	2197.877 ± 401.9

Table 6.9: Monte Carlo simulation results of (TDHA).

SNR=20dB	**Metrics**	
	RMSE (mean ± std)	**Score (mean ± std)**
FD001	10.242 ± 0.1	183.79 ± 3.85
FD002	17.18 ± 1.314	1688.98 ± 381.7
FD003	10.51 ± 0.097	181.902 ± 4.488
FD004	17.8 ± 1.15	2113.62 ± 394.545

and robust against noise, where the RMSE and the score values show remarkable stability especially with FD001 and FD003, where the (std) values did not exceed 0.106 and 0.097 in terms of RMSE, respectively, in addition to 4.11 and 4.709 in terms of the score values, respectively. However, the model was more sensitive to noise in cases of FD002 and FD004, and that was expected and it is logical to happen as we deal with six different operating conditions for FD002 and FD004, which would increase the level of noise. Moreover, the achieved results for FD002 and FD004 when SNR is set to 20 are still remarkably stable and better than most of the reported results for normal testing without any noise.

Table 6.10: Comparison with the Proposed Models of Chapter 5: (a) The results of MPHD, TDHA, NPBGRU, and NBLSTM at TW=30. (b) The results of MPHD, TDHA, NPBGRU, and NBLSTM at TW=15.

TW=30		FD001	FD002	FD003	FD004
MPHD	Score	**177.824**	798.021	**176.366**	1148.19
	RMSE	**10.003**	**13.856**	10.376	16.436
TDHA	Score	180.71	**793.064**	177.092	**939.27**
	RMSE	10.118	14.009	**10.233**	**14.9**
NPBGRU	Score	191.8	899.762	197.463	1306.5
	RMSE	10.44	14.651	10.59	16.78
NBLSTM	Score	238.34	1056.629	226.482	1357.2
	RMSE	12.321	15.038	11.364	17.752
Improvement Percentages	Score	7.29%	11.86%	10.68%	28.11%
	RMSE	4.19%	5.43%	3.37%	11.20%

(a)

TW=15		FD001	FD002	FD003	FD004
MPHD	Score	473.011	**1003.406**	**441.91**	2031.952
	RMSE	**13.374**	**15.395**	14.024	18.637
TDHA	Score	471.436	1039.251	447.088	**1655.608**
	RMSE	13.436	15.548	**13.883**	**17.418**
NPBGRU	Score	572.828	1277.886	577.831	2631.909
	RMSE	15.63	17.296	15.622	19.611
NBLSTM	Score	626.77	1645.566	794.115	2705.716
	RMSE	16.353	18.358	16.014	19.755
Improvement Percentages	Score	17.70%	21.48%	23.52%	37.09%
	RMSE	14.43%	10.99%	11.13%	11.18%

(b)

6.3.1.6 Comparison with the Proposed Models of Chapter 5

Although the proposed models of the previous chapter (chap. 5) were successful and effective, the new proposed approach of utilizing three parallel paths plays an imperative role in improving the RUL perdition models in terms of accuracy, generalization, and robustness.

Tables 6.10 (a & b) illustrate the impact of using three parallel paths on the prediction performance, in which TW is set to 30 and then to 15. The improvements ranged from (7.29%) to (37.09%) in terms of the score values, and from (3.37%) to (11.2%) in terms of the RMSE values. It is important to highlight that the improvement percentages when the (TW=15) are notable which gives a clear indication that the extracted features in the case of three parallel paths are more informative than approaches of the previous chapter, however, the TW is small.

Tables 6.11 (a & b) prove how the three parallel paths models are stable and robust to noise than the models of chapter 5, where two levels (30 & 20)dB of SNR have been used and the standard deviation has been reduced between (11.78%) to (51.49%) in terms of score values and from (8.49%) to (63.46%) in terms of the RMSE values. Finally, the most remarkable point is that the number of involved parameters in the proposed three parallel paths models is around (42000) which is almost half of the number of involved parameters in the proposed models of chapter 5.

Table 6.11: Comparison with the Proposed Models of Chapter 5: (a) The results of MPHD, TDHA, NPBGRU, and NBLSTM at SNR=30. (b) The results of MPHD, TDHA, NPBGRU, and NBLSTM at SNR=20.

SNR =30 dB		Standard deviation (std)				% of Improvement
		MPHD	TDHA	NPBGRU	NBLSTM	
FD001	Score	1.03	1.087	1.602	1.96	35.71
	RMSE	0.011	0.017	0.0301	0.0343	63.46
FD002	Score	109.33	104.9	150.779	224.666	30.43
	RMSE	0.316	0.33	0.506	0.539	37.55
FD003	Score	0.88	1.0	1.814	2.1	51.49
	RMSE	0.031	0.028	0.043	0.043	34.88
FD004	Score	242.636	198.77	258.866	264.905	23.22
	RMSE	0.382	0.366	0.54	0.551	32.22

(a)

SNR =20 dB		Standard deviation (std)				% of Improvement
		MPHD	TDHA	NPBGRU	NBLSTM	
FD001	Score	3.717	3.85	5.09	6.201	26.97
	RMSE	0.062	0.1	0.11	0.11	43.64
FD002	Score	384.104	381.7	434.161	463.042	12.08
	RMSE	1.306	1.314	1.614	1.783	19.08
FD003	Score	4.601	4.488	5.087	6.691	11.78
	RMSE	0.12	0.097	0.106	0.156	8.49
FD004	Score	516.836	394.545	483.844	488.744	18.46
	RMSE	1.201	1.15	1.271	1.312	9.52

(b)

6.3.1.7 Comparison with Existing Methods

To illustrate and evaluate the efficiency of the proposed three parallel paths solutions, the prediction results of these models are compared with seven different published studies, which are: CapsNet [332], DAG [304], CNNTW [333], DBiLSTM [225], D-LSTM [214], DCNN [156]and TEMPCO [299]. These studies represent the latest and most successful solutions in the literature. Table 6.12 presents the performance comparison results of the proposed multipath models and the other seven approaches. It can be easily noted that the proposed solutions have the highest prediction accuracy across all methods by achieving the lowest score and the lowest RMSE values, which implies that the proposed methods are performing significantly better in the RUL prediction of turbofan engines. Here, two points can be highlighted: (i) achieving lower score values represents the earlier prediction of RUL, which promotes more efficiency, effectiveness, and safety in real life PHM applications, where, the score function measure has a higher penalty for late estimation) [311], (ii) the models have achieved these outstanding outcomes with only 42000 parameters.

The results of the score values have been improved as follows; 20.67%, 35.51%, 37.86%, and 64.23% for FD001, FD002, FD003, and FD004, respectively. While, in terms of the RMSE values, the proposed models achieved 16.36%, 14.99%, 12.67%, and 21.41% improvements for FD001, FD002, FD003, and FD004, respectively.

All the aforementioned approaches have employed the piece-wise linear degradation model. However group of different approaches have utilized alternative ways for modeling the degradation behavior. Table 6.13 compares the results of the proposed multipath with other methodologies that use different models other than the piece-wise linear degradation model. The proposed methods consistently achieve significantly better results across most of the datasets, i.e., 17.74%, 33.54%, 14.85%, and 34.25% in terms of the RMSE value for FD001, FD002, FD003, and FD004, respectively. While about 16.31%, 62.01%, 2.43%, and 66.93% in terms of the score value for FD001, FD002, FD003, and FD004, respectively.

6.4 The Summary

In this chapter, the principle of collecting different classes of informative features has been intensively invested by the design of multipath hybrid RUL estimation frameworks. In particular, three parallel paths are constructed based on different DNN

Table 6.12: Performance comparison of the proposed models with 7 state-of-the-art methodologies.

TW=30		**MPHD**	**NPHM**	**NMPM**	**TDHA**	**CapsNet [332]**	**DAG [304]**	**CNNTW [333]**	**DBiLSTM [225]**	**D-LSTM [214]**	**DCNN [156]**	**TEMPCO [299]**
FD001	**Score**	**177.824**	182.597	188.63	180.71	276.34	229	224.16	295	338	273.7	1220
	RMSE	**10.003**	10.272	10.315	10.118	12.58	11.96	12.18	13.65	16.14	12.61	23.57
FD002	**Score**	798.021	846.083	876.16	**793.1**	1229.72	2730	2494.35	4130	4450	10412	3100
	RMSE	**13.856**	14.164	14.322	14.009	16.30	20.34	19.58	23.18	24.49	22.36	20.45
FD003	**Score**	**176.366**	179.851	189.38	177.092	283.81	535	1279.85	317	852	284.1	1300
	RMSE	10.376	**10.226**	10.441	10.233	11.71	12.46	15.67	12.74	16.18	12.64	21.17
FD004	**Score**	1148.19	944.514	962.403	**939.270**	2625.64	3370	4523.32	5430	5550	12466	4000
	RMSE	16.436	15.979	16.156	**14.90**	18.96	22.43	22.12	24.86	28.17	23.31	21.03

Table 6.13: Comparing performance obtained from four approaches that exclude the need for utilization of the piece-wise linear degradation model with the proposed three parallel paths models based on the C-MAPSS datasets.

Dataset		MPHD	NPHM	NMPM	TDHA	ResCNN [334]	Semi-S [335]	BiLSTM-ED [298]	Rulclipper [315]
FD001	**Score**	**177.824**	182.597	188.63	180.71	212.48	231	273	216
	RMSE	**10.003**	10.272	10.315	10.118	12.16	12.56	14.74	13.27
FD002	**Score**	798.021	846.083	876.16	**793.064**	2087.77	3366	3099	2796
	RMSE	**13.856**	14.164	14.322	14.009	20.85	22.73	22.07	22.89
FD003	**Score**	**176.366**	179.851	189.381	177.092	180.76	251	574	317
	RMSE	10.376	**10.226**	10.441	10.233	12.01	12.10	17.48	16
FD004	**Score**	1148.19	944.514	962.403	**939.270**	3400	2840	3202	3132
	RMSE	16.436	15.979	16.156	**14.90**	24.97	22.66	23.49	24.33

architectures (i.e., LSTM, BLSTM, GRU, BGRU, MLP, and CNN) along with different noisy layers. Their combined output is then fed into the fourth path (fusion center) to incorporate the obtained features to compute estimates of the RUL. Different settings, different styles of noisy training, and different effective techniques such as (the global average pooling and batch normalization) are employed to design and propose four different multipath frameworks, i.e., NMPM, MPHD, NPHM, and TDHA. Moreover, several experiments have been conducted to examine the effects of having multiple (more than two) parallel paths based on different DNNs architectures. The proposed models are evaluated and tested by utilizing the (C-MAPSS) dataset, which is provided by NASA. The proposed multipath models outperform previous studies especially in cases where the datasets are obtained from complex environments.

Chapter 7

Contributions and Future Research Directions

Generally speaking, Prognostic Health Management (PHM) consists of the following two main pillars: (i) Diagnostics, which is an evaluation stage based on observed symptoms to asses a system's current and past health state. Assessments are then utilized to recognize and isolate faults/failures, and; (ii) Prognostic, which is an estimation stage to forecast future health states of the system. Prognostic, typically, also involves estimation of the time when the system will no longer be operational. The prognostic, beside predicting the impact of known failure modes on assets' life, also focuses on how other failure modes can be initiated by the known failure modes. It is worth mentioning that diagnostics are crucial for effective and accurate prognostic since a successful prognostic approach starts with robust diagnostics, as the uncertainties of the current system condition influence any possible predictions. However, no prediction or estimation is made in most diagnostic techniques.

In light of the above introduction, the objective of this thesis was to develop efficient and powerful models to identify and capture the degradation dynamics of a manufacturing/industrial system. Moreover, the thesis focused on designing innovative algorithms capable of handling different datasets to effectively obtain as much information as possible from the available datasets. In this regard, we focused on implementation of novel data-driven models to learn general mappings from inputs to outputs leading to better network stability, less generalization error, high robustness, and faster learning.

7.1 Summary of Contributions

The following is a brief overview of the key contributions provided by this thesis.

- **Multiple Model Degradation Path (MMDP) Estimation** [305]: Despite the recent advances in degradation modeling of mission critical systems, most of the proposed models for predicting the degradation process are based on the assumption that a particular type of statistical distribution governs the degradation process (e.g., Normal, Wiener or Gamma). In practice, a system may consist of multiple components or a component may have multiple degradation measures that require simultaneous consideration of multiple degradation paths. To bridge this gap, we introduced a new category for degradation modeling by proposing a generalized hybrid non-linear filtering framework that is capable of simultaneously handling different linear and non-linear degradation paths. The proposed generalized hybrid approach covers a wide range of scenarios based on the state-space modeling, without having any prior knowledge of the true degradation model of the system. The proposed model (MMDP) combines the Multiple Model Adaptive Estimation Framework (MMAE) with particle filtering, to derive a degradation model optimized to the dynamics of the system under consideration. A set of candidate models for the degradation path are taken into consideration, the MMDP then performs degradation prediction based on each model in parallel, and then combines the outputs of localized filters adaptively to form the overall estimate of the degradation process over time. In this study, only two models (degradation paths) are developed, however, the proposed hybrid state-space model and its corresponding estimation algorithm is general and can accommodate any number of candidate models. Although the proposed MMDP represents a new successful category for degradation modeling that provides near-optimal results, this approach has some limitations. For example, once the approach has found the degradation path then it will continue considering the same degradation path over time, however, the behaviour of the degradation process may alter over time. We have, therefore, introduced a new approach in Chapter 1 that can resolve this problem and bridge the gap.

- **Interactive Multiple Model Particle Filters (IMMPF) Framework** [306]: The proposed IMMPF estimation algorithm consists of a bank of parallel particle filters each of them representing a separate model of the system under consideration. In other words, three particle filters are implemented and run in parallel based on the following set of candidate degradation models: Gamma Degradation Path (DP); Brownian DP; and Inverse Gaussian DP. It is worth noting that the aforementioned three models are selected as proof of concept, the proposed IMMPF can accommodate any number of candidate models. For each filter, at the beginning of each cycle, the initial estimate is a mixture of all recent estimates obtained from available models. This mixing mechanism allows the IMMPF to take full account of the history of the modes, leading to faster and more reliable estimation for the changed states. The switching probabilities and the likelihood of each model control the interactions between the models. The IMMPF result is a combined state vector, which is the sum of the state vectors for each of the modes weighted by their model probabilities. The proposed framework provides precise results without requiring any prior knowledge of the true degradation model of the system.

- **Hybrid Parallel Deep Neural Network Models for RUL Estimation** [83, 321–324]:Remaining Useful Life (RUL) is a crucial measure used in the maintenance domain within manufacturing and industrial systems. Accurate RUL estimation enables improved decision-making for operations and maintenance. Capitalizing on the recent success of multiple-model (also referred to as hybrid or mixture of experts) deep learning techniques, the thesis proposed four (i.e., HDNN, BiLSTM, GRU, and BiGRU) hybrid deep neural network frameworks for RUL estimation. All the proposed frameworks consist of two parallel paths, one of them is based on three CNN layers combined with two max pooling layers and has common settings among all the models. The second path used three stacked LSTM layers, one BLSTM layer and two LSTM layers, two GRU layers, and two BGRU layers, for the HDNN, BiLSTM, GRU, and BiGRU, respectively. In each model, the parallel paths are followed by a third path representing the fusion center that integrates the extracted features to perform the regression task for generating the targeted output of the RUL prediction. The fusion path in all the proposed methods consists of three fully connected layers based on different settings and activation functions in addition to 0.3 dropout

rate. Different techniques have been used to train the models such as piece-wise degradation method; Backpropagation and mini batch gradient descent, and; Adaptive moment estimation (Adam). The adopted loss function was the mean squared error (MSE). All the proposed models have been tested and evaluated using the NASA C-MAPSS dataset. Although the prediction results of the proposed models were superior to state-of-the-art results, still there were some issues to be solved and improved such as robustness and the generalization behavior of the models.

- **The Noisy and Hybrid Deep Neural networks for Remaining Useful Life Estimation** [83,323,324]: One approach to improve deep neural networks in terms of performance, generalization, and robustness, is to add random noise during the training stage. Such an approach (adding noise/noisy training) is one of the most effective techniques to address the aforementioned issues because: It has regularization effects [349]; introduces some form of data augmentation [344]; has the ability to reduce generalization error, and; has the potentials to reduce over-fitting problems. Four models have been proposed by utilizing different noise injection strategies to improve the previously proposed approaches. The proposed models can be classified into two main categories (i.e., partial noisy training and fully noisy training) based on the way of adopting the noise layers. The first and the second approaches (NLSTM & NBLSTM) [83] used partial noisy training as they have a noise layer as an input layer to the first parallel path and have no additional noise layers within the fusion center. On the other hand, the other two approaches (NGRU & NPB-GRU) [323,324] use fully noisy architectures as we adopted the noise layer in all the integrated paths. However, the CNN path in all the proposed models used a fully noisy style. The grid search strategy is used to identify a suitable set of hyperparameters for each model. The results of the proposed models after investigating different effects such as the effects of different time window size, the effects of operating conditions and fault modes, and the effects of training based on different noise levels, have shown the effectiveness of adopting the noise injection strategy. The proposed models have achieved state-of-the-art results and performed remarkably in all different cases. With regards to the RMSE values, the results are improved between (2.31% to 13.78%) and between (5.76% to 24.73%) in terms of the score values.

- **Multipath Parallel Hybrid Deep Neural Networks** [380–382]: Enhancing the learning ability (training) of the prediction model is always the target to achieve the main objective of the neural network, which is generalizing the good performance from the training dataset to any new data the model would use to make predictions. There is no doubt that extracting more informative features will translate to achieving better results [353]. Four new models have been proposed by implementing the following effective strategies: (i) Utilizing the most successful classes of the artificial neural networks, which are LSTM, BLSTM, GRU, BGRU, MLP, and CNN; (ii) Using the most promising category of RUL estimation, which is the hybrid solution, and; (iii) Using the parallel architecture to ensure collecting different features from constituent DNNs. All the proposed models (NMPM, MPHD, NPHM, and TDHA) were designed based on three parallel paths and the outputs of these paths are then combined by a fusion center to form the RUL estimates. To develop the proposed methodologies, the following techniques have been incorporated: (a) Different styles of noisy training are used to show the effectiveness of the proposed models; (b) Global average pooling is employed to minimize overfitting issues and improve the training, and; (c) The batch normalization is used to stabilize the learning process of the proposed TDHA. Different experiments have been conducted to examine the effects of having multiple (more than two) parallel paths based on different DNNs architectures. The proposed models achieved the best results in the literature.

7.1.1 Summary of Achieved Results

For the diagnostic part of the thesis, the following two approaches are proposed:

- **Multiple Model Degradation Path (MMDP) Estimation**: The model was successful to recognize and find the right hidden degradation process without having any prior knowledge about the degradation model. The results have been illustrated in two ways. First, by using the adaptive weight corresponding to each of the constituent models, where the adaptive weight for the right degradation path converged to 1 after some iterations while the others converged to 0 over time. Second, by using the Root-Mean-Square Error (RMSE) measure for the involved filters in addition to the proposed MMDP model, where the error associated with the right path and the proposed model has converged to

Table 7.1: All the proposed models in Chapters 4, 5, and 6.

	Hybrid Parallel Deep Neural Network Models for RUL Estimation				The Noisy and Hybrid Deep Neural networks for RUL Estimation				Multipath Parallel Hybrid Deep Neural Networks for RUL Estimation				% of Improvement with HDNN
	HDNN	BiLSTM	GRU	BiGRU	NLSTM	NBLSTM	NGRU	NPBGRU	MPHD	NPHM	NMPM	TDHA	
FD001	245	252.915	262.656	234.406	247.01	238.34	255.02	191.8	177.824	182.597	188.63	180.71	27.42
	13.017	12.848	12.831	12.108	13.231	12.321	12.629	10.44	10.003	10.272	10.315	10.118	23.15
FD002	1282.42	1184.925	1070.577	1063.415	1206.255	1056.629	988.216	899.762	798.021	846.083	876.16	793.064	38.16
	15.24	16.384	16.182	16.233	16.293	15.038	16.528	14.651	13.856	14.164	14.322	14.009	9.08
FD003	287.72	257.857	292.329	231.727	276.514	226.482	217.452	197.463	176.366	179.851	189.381	177.092	38.7
	12.22	12.972	12.006	12.217	12.782	11.364	11.404	10.59	10.376	10.226	10.441	10.233	16.32
FD004	1527.42	1467.793	1889.597	1735.845	1434.832	1357.20	1873.684	1306.5	1148.19	944.514	962.403	939.270	38.51
	18.156	18.171	19.015	18.67	17.474	17.752	18.868	16.78	16.436	15.979	16.156	14.90	17.93

zero, while it continued to increase for the rest of the paths.

Cons: The proposed MMDP is based on a set of candidate models for the degradation path, performs degradation prediction based on each model in parallel, and involves state-space modeling. Thus, we have a challenge in state-space modeling design and design of the degradation process model (degradation path). Another limitation is scenarios where the degradation process has an unknown structure, i.e., non of the degradation paths matches the degradation of the system. Finally, the main issue is that, once the approach has found the degradation path then it will continue considering the same degradation path over time, while the behavior of degradation may change over time.

- **Interactive Multiple Model Particle Filters (IMMPF) Framework**: The proposed model can effectively recognize and fetch the right hidden degradation process in each time step, without having any prior knowledge about the degradation model. The proposed model is superior to the MMDP approach, in the sense that the model will keep tracking the changes in the degradation process over time.

  Cons: The main issue with this design is that the model can only fetch and handle one degradation path at a time. Another limitation is that if the degradation process doesn't have any match in the candidate degradation-matched filters. In addition to the challenge of the state space model design.

For the prognostic part of the thesis, 12 approaches have been proposed as outlined below:

- **Parallel Hybrid Deep Neural Networks Models**: With regards to the proposed models for RUL estimation, Table 7.1 shows the increasing improvements achieved from the first proposed model (HDNN) to the last proposed model, which reflects the effectiveness and evolution of our adopted model designing techniques. It is worth mentioning that Table 7.1 also shows improvement percentages in terms of score and RMSE values. The score value improvements are between 27.42% to 38.7%, and between 9.08% to 23.15% in terms of the RMSE values. Furthermore, there was an impressive improvement in terms of the involved parameters, as it was around (77,000) in the first proposed model (HDNN) and only (42,000) involved parameters in the new proposed multipath models.

Cons: The proposed approaches are based on some assumptions that need to be improved in the future as discussed in the next section.

7.2 Future Research Directions

In continuation of the research works presented in the thesis, the following potential directions could be followed:

- One direction for future research in the proposed (IMMPF) framework, is to improve the model to be able to fetch and handle multiple degradation paths at a time and investigate different scenarios such as the involving of many degradation paths based on different percentages of involvement. Furthermore, a neural network model can be utilized as one of the candidate models to solve the problem of missing degradation-matched filters.

- The piece-wise linear degradation model utilized in this thesis is considered as one of the assumptions and limitations that need to be improved in the future. One direction for future research is to use alternative models for degradation derived from sensors' readings.

- There is no doubt that uncertainty quantification is an extremely critical problem when it comes to using the neural networks in sensitive fields such as aerospace, transportation, and manufacturing. One direction to measure the uncertainty of a model is to resort to the Bayesian construction of DNNs. A fruitful direction for future research is to extend the proposed models to incorporate uncertainty quantification, e.g., via the development of Bayesian DNNs.

- In order to be capable of learning the most suitable feature representations from the data, one can consider the exploration of other deep learning techniques such as Deep belief networks or Capsule Networks. Furthermore, one can study how features can be extracted more efficiently to further boost prediction accuracy.

- Examining different types of injected noise, and study the effects of that on the model generalization and robustness are other directions for future research. Additionally, one can consider exploring more effective methods to find the optimum hyper-parameter settings.

- The batch normalization can not be utilized for all the DNNs techniques and models due to dependence on different variables such as the model design layout, batch size, and learning type. In batch normalization, the mean and its standard deviation are computed across the batch and are the same for each example in the batch. In other words, it normalizes the input features across the batch dimension. This issue might be solved by using layer normalization where the normalization statistics are computed across each feature and are independent of other training scenarios.

Bibliography

[1] R. Kothamasu, S. Huang, and W. Verduin, "System health monitoring and prognostics - a review of current paradigms and practices," *International Journal of Advanced Manufacturing Technology*, vol. 28, no. 9, pp. 1012-1024, 2006.

[2] Ryan Arsenault, "The rising cost of downtime," " https://www.aberdeen.com/techpro-essentials/stat-of-the-week-the-rising-cost-of-downtime," 2016.

[3] dvanced Technology Services, Inc., "Downtime costs auto industry $ 22k /minute - Survey," " https://news.thomasnet.com/companystory," 2006.

[4] Jonathan Trout, "Predictive maintenance survey 2019," "https://www.reliableplant.com/Read/31707/predictive-maintenance-survey-2019," 2019.

[5] Z. Ye, and M. Xie, "Stochastic modelling and analysis of degradation for highly reliable products," *Applied Stochastic Models in Business and Industry,* vol. 31, no. 1, pp. 16-32, 2015.

[6] Z. Huang, Z. Xu, X. Ke, W. Wang, and Y. Sun, "Remaining Useful Life Prediction for an Adaptive Skew-Wiener Process Mode," *Mech. Syst. Signal Process.*, vol. 87, pp. 294-306, 2017.

[7] 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, 2011.

[8] Z. Tian, L. Wong, and N. Safaei, "A Neural Network Approach for Remaining Useful Life Prediction Utilizing both Failure and Suspension Histories," *Mechanical Systems and Signal Processing*, vol. 24, no. 5, pp. 1542–1555, 2010.

[9] C. Hu, B. Youn, P. Wang, and J. Yoon, "Ensemble of Data-driven Prognostic Algorithms for Robust Prediction of Remaining Useful Life," *Reliability Engineering and System Safety*, vol. 103, pp. 120-135, 2012.

[10] K. Liu, N. Gebraeel and J. Shi, "A Data-Level Fusion Model for Developing Composite Health Indices for Degradation Modeling and Prognostic Analysis," *IEEE Transactions on Automation Science and Engineering*, vol. 10, no. 3, pp. 652-664, 2013.

[11] P. Lim, C. Goh, K. Tan, and P. Dutta, "Estimation of Remaining Useful Life based on Switching Kalman Filter Neural Network Ensemble," *Proceedings of the Annual Conference of the Prognostics and Health Management Society (PHM)*, pp. 2-9, 2014.

[12] M. Kan, A. Tan, and J. Mathew, "A Review on Prognostic Techniques for Non-stationary and Non-linear Rotating Systems," *Mechanical Systems and Signal Processing*, vol. 62-63, PP. 1-20, 2015.

[13] F. Balali, H. Seifoddini, and A. Nasiri, "Data-driven predictive model of reliability estimation using degradation models: a review," *Life Cycle Reliability and Safety Engineering,* vol. 9, pp. 113–125, 2020.

[14] R. KANG, W. GONG, and Y. CHEN, "Model-driven degradation modeling approaches:Investigation and review," *Chinese Journal of Aeronautics,* 2020; https://doi.org/10.1016/j.cja.2019.12.006.

[15] R. Precup, P. Angelov, B. Costa, and M. Mouchaweh, "An overview on fault diagnosis and nature-inspired optimal control of industrial process applications," *Computers in Industry,* vol. 74, pp. 75-94, 2015.

[16] H. Li, R. Li, and R. Yuan, "Reliability modeling of multiple performance based on degradation values distribution," *Advances in Mechanical Engineering,* vol. 8, no. 10, 2016. DOI: 10.1177/1687814016673755

[17] F. Balali, H. Seifoddini, and A. Nasiri, "Data-driven predictive model of reliability estimation using degradation models: a review," *Life Cycle Reliability and Safety Engineering,* 2020. https://doi.org/10.1007/s41872-020-00111-6

[18] Z. Zhang, X. Si, C. Hu, and Y. Lei, "Degradation Data Analysis and Remaining Useful Life Estimation: A Review on Wiener-Process-Based Methods," *European Journal of Operational Research,* vol. 271, no. 3, pp.775-796, 2018.

[19] X. Wang, "Wiener processes with random effects for degradation data," *Journal of Multivariate Analysis,* vol. 101, no. 2, 2010. https://doi.org/10.1016/j.jmva.2008.12.007.

[20] D. Pan, J. Liu, F. Huang, J. Cao, and A. Alsaedi, "A Wiener process model with truncated normal distribution for reliability analysis," *Applied Mathematical Modelling,* vol. 50, pp. 333-346, 2017.

[21] D. Pan, Y. Wei, H. Fang, and W. Yang, "A reliability estimation approach via Wiener degradation model with measurement errors," *Applied Mathematics and Computation,* vol. 320, pp. 131-141, 2018.

[22] T. Tsai, C. Lin, Y. Sung, P. Chou, C. Chen and Y. Lio, "Inference From Lumen Degradation Data Under Wiener Diffusion Process," *in IEEE Transactions on Reliability,* vol. 61, no. 3, pp. 710-718, 2012.

[23] H. Hao, and C. Su, "A Bayesian Framework for Reliability Assessment via Wiener Process and MCMC," *Mathematical Problems in Engineering,* Article ID 486368, 8 pages, 2014. DOI:10.1155/2014/486368.

[24] G. Jin, D. Matthews, and Z. Zhou, "A Bayesian framework for on-line degradation assessment and residual life prediction of secondary batteries inspacecraft," *Reliability Engineering and System Safety,* vol. 113, pp. 7-20, 2013.

[25] H. Li, D. Pan and C. L. P. Chen, "Reliability modeling and life estimation using an expectation maximization based wiener degradation model for momentum wheels," *in IEEE Transactions on Cybernetics,* vol. 45, no. 5, pp. 969-977, 2015. doi: 10.1109/TCYB.2014.2341113

[26] S. Mishra, and O. Vanli, "Remaining Useful Life Estimation with Lamb-Wave Sensors Based on Wiener Process and Principal Components Regression," *Journal of Nondestructive Evaluation,* vol. 35, no. 3, 2015. DOI: 10.1007/s10921-015-0328-2.

[27] X. Wang, S. Lin, S. Wang, Z. He, and C. Zhang, "Remaining useful life prediction based on the Wiener process for an aviation axial piston pump," *Journal of Nondestructive Evaluation,* vol. 29, no. 3, 2015. DOI: 10.1016/j.cja.2015.12.020.

[28] Y. Cheng, H. Zhu, K. Hu, J. Wu, X. Shao, and Y. Wang, "Reliability prediction of machinery with multiple degradation characteristics using double-Wiener process and Monte Carlo algorithm," *Mechanical Systems and Signal Processing,* vol. 134, 2019. 106333

[29] Q. Dong, L. Cui, and S. Si, "Reliability and Availability Analysis of Stochastic Degradation Systems Based on Bivariate Wiener Processes," *Applied Mathematical Modelling,* vol. 79, pp. 414-433, 2020.

[30] J. Noortwijk, "A survey of the application of gamma processes in maintenance," *Reliability Engineering and System Safety,* vol. 94, pp. 2-21, 2009.

[31] N. Gorjian, L. Ma, M. Mittinty, P. Yarlagadda, and Y. Suni, "A review on degradation models in reliability analysis," *Engineering Asset Lifecycle Management,* pp. 369-384, 2010.

[32] Z. Pan, and N. Balakrishnan, "Reliability modeling of degradation of products with multiple performance characteristics based on gamma processes," *Reliability Engineering and System Safety,* vol. 96, issue 8, pp. 949-957, 2011.

[33] Q. Qiu, and L. Cui, "Gamma process based optimal mission abort policy," *Reliability Engineering and System Safety,* vol. 190, 2019. 106496.

[34] B. Sun, M. Yan, Q. Feng, Y. Li, Y. Ren, K. Zhou, and W. Zhang, "Gamma Degradation Process and Accelerated Model Combined Reliability Analysis Method for Rubber O-Rings," *in IEEE Access,* vol. 6, pp. 10581-10590, 2018. doi: 10.1109/ACCESS.2018.2799853.

[35] C. Zhang, X. Lu, Y. Tan, and Y. Wang, "Reliability demonstration methodology for products with Gamma Process by optimal accelerated degradation testing," *Reliability Engineering and System Safety,* vol. 142, pp. 369-377, 2015.

[36] F. Duan, and G. Wang, "Optimal design for constant-stress accelerated degradation test based on gamma process," *Communications in Statistics - Theory and Methods,* pp. 1-25, 2018 .DOI:10.1080/03610926.2018.1459718.

[37] M. Cholette, H. Yu, P. Borghesani, L. Ma, G. Kent, "Degradation modeling and condition-based maintenance of boiler heat exchangers using gamma processes," *Reliability Engineering and System Safety,* vol. 183, pp. 184-196, 2019.

[38] R. Edirisinghe, S. Setunge, and G. Zhang, "Application of Gamma Process for Building Deterioration Prediction," *Journal of Performance of Constructed Facilities,* vol. 27, no. 6, pp. 763-773, 2013.

[39] M. Mahmoodian, and A. Alani, "Modeling Deterioration in Concrete Pipes as a Stochastic Gamma Process for Time-Dependent Reliability Analysis," *Journal of Pipeline Systems Engineering and Practice,* vol. 5, issue 1, 2014.

[40] K. Zhang, and J. Xiao, "Time-dependent reliability analysis on carbonation behavior of recycled aggregate concrete based on gamma process," *Construction and Building Materials,* vol. 158, pp. 378-388 issue 1, 2018.

[41] Z. Ye, and N. Chen, "The inverse Gaussian process as a degradation model," *Technometrics,* vol. 56, no. 3, pp. 302-311, 2014.

[42] X. Wang, and D. Xu, "An Inverse Gaussian Process Model for Degradation Data," *Technometrics,* vol. 52, no. 2, pp.188-197, 2010.

[43] W. Peng, Y. Li, Y. Yang, H. Huang, M. Zuo, "Inverse Gaussian process models for degradation analysis: A Bayesian perspective," *Technometrics,* vol. 130, pp.175-189, 2014.

[44] H. Qin, S. Zhang, and W. Zhou, "nverse Gaussian process-based corrosion growth modeling and its application in the reliability analysis for energy pipelines," *Frontiers of Structural and Civil Engineering ,* vol. 7, no. 3, 2013. DOI: 10.1007/s11709-013-0207-9.

[45] Z. Ye, L. Chen, L. Tang, and M. Xie, "Accelerated Degradation Test Planning Using the Inverse Gaussian Process," *in IEEE Transactions on Reliability,* vol. 63, no. 3, pp. 750-763, 2014. doi: 10.1109/TR.2014.2315773.

[46] X. Chen, X. Sun, X. Ding, and J. Tang, "The inverse Gaussian process with a skew-normal distribution as a degradation model," *Communication in Statistics- Simulation and Computation,* 2019. DOI: 10.1080/03610918.2018.1527351.

[47] A. Xu, J. Hu, and P. Wang, "Degradation modeling with subpopulation heterogeneities based on the inverse Gaussian process," *Applied Mathematical Modelling,* vol. 81, pp. 177–193, 2020.

[48] W. Peng, Y. Li, Y. Yang, J. Mi, and H. Huang, "Bayesian Degradation Analysis With Inverse Gaussian Process Models Under Time-Varying Degradation Rates," *in IEEE Transactions on Reliability,* vol. 66, no. 1, pp. 84-96, 2017. doi: 10.1109/TR.2016.2635149.

[49] A. Shahraki, O. Yadva, and H. Liao, "A Review on Degradation Modelling and Its Engineering Applications," *International Journal of Performability Engineering,* vol. 13, no. 3, pp. 299-314, 2017. DOI:10.1555/10.23940/ijpe.17.03.p6.299314.

[50] C. Lu, and W. Meeker, "Using Degradation Measures to Estimate a Time-to-Failure Distribution," *Technometrics,* vol. 35, no. 2, pp. 161, 1993. Doi:10.2307/1269661.

[51] M. Guida, F. Postiglione, and G. Pulcini, "A Bayesian approach for non-homogeneous gamma degradation processes," *Communications in Statistics - Theory and Methods,* vol. 0, no. 0, pp. 1-18, 2018. DOI: 10.1080/03610926.2018.1440306.

[52] S. Mercier, "Probabilistic Construction and Properties of Gamma Processes and Extensions," *International Conference on Mathematical Methods on Reliability,* 2017. HAL Id: hal-01577025.

[53] Q. Guan,Y. Tang, and A. Xu, "Reference Bayesian analysis of inverse Gaussian degradation process," *Applied Mathematical Modelling,* vol. 74, pp. 496-511, 2019.

[54] J. Guo, C. Wang ,J. Cabrera, and E. Elsayed, "Improved inverse Gaussian process and bootstrap: Degradation and reliability metrics," *Reliability Engineering and System Safety,* vol. 178, pp. 269–277, 2018.

[55] Z. Ma, S. Wang, H. Liao, and C. Zhang, "Engineering-driven performance degradation analysis of hydraulic piston pump based on the inverse Gaussian process," *Reliability Engineering and System Safety,* vol. 35, issue 7, pp. 2278-2296, 2019.

[56] D. Pan, J. Liu, and J. Cao, "Remaining useful life estimation using an inverse Gaussian degradation model," *Neurocomputing,* vol. 185, pp. 64-72, 2016.

[57] W. Peng, S. Zhu, and L. Shen, "The transformed inverse Gaussian process as an age- and state-dependent degradation model," *Applied Mathematical Modelling,* vol. 75, pp. 837-852, 2019.

[58] Q. Guan, Y. Tang, and A. Xu, "Objective Bayesian analysis for competing risks model with Wiener degradation phenomena and catastrophic failures," *Applied Mathematical Modelling,* vol. 74, pp. 422–440, 2019.

[59] M. Giorgio, M. Guida, and G. Pulcini, "The transformed gamma process for degradation phenomena in presence of unexplained forms of unit-to-unit variability," *Quality and Reliability Engineering International,* vol. 34, issue 4, pp. 543-562, 2018. DOI: 10.1002/qre.2271

[60] M. Oumouni, F. Schoefs, and B. Castanier, "Modeling time and spatial variability of degradation through gamma processes for structural reliability assessment," *Structural Safety,* vol. 76, pp. 162–173, 2019.

[61] F. Duan, G.Wang, and H. Wang, "Inverse Gaussian Process Models for Bivariate Degradation Analysis: A Bayesian Perspective," *Communication in Statistics- Simulation and Computation,* vol. 47, issue 1, 2017. DOI: 10.1080/03610918.2017.1280162.

[62] H. Gao , L. Cui , and Q. Qiu, "Reliability modeling for degradation-shock dependence systems with multiple species of shocks," *Reliability Engineering and System Safety,* vol. 185, pp. 133-143, 2019.

[63] D. Liu, S. Wang, C. Zhang, M. Tomovic, "Bayesian model averaging based reliability analysis method for monotonic degradation dataset based on inverse Gaussian process and Gamma process," *Reliability Engineering and System Safety,* vol. 180, pp. 25-38, 2018.

[64] J. Sari, M. Newby, A. Brombacher, and L. Tang, "Bivariate constant stress degradation model: LED lighting system reliability estimation with two-stage modeling," *Quality and Reliability Engineering International,* vol. 25, no. 8, pp. 1067-1084, 2009.

[65] J. Zhou, Z. Pan, and Q. Sun, "Bivariate degradation modeling based on gamma process," *Proceedings of the World Congress on Engineering,* vol. 3, pp. 1783-1788, 2010.

[66] W. Peng, Y. Li, Y. Yang, S. Zhu, and H. Huang, "Bivariate Analysis of Incomplete Degradation Observations Based on Inverse Gaussian Processes and Copulas," *in IEEE Transactions Reliability,* vol. 65, no. 2, pp. 624-639, 2016.

[67] L. Rodriguez-Picon, A. Rodriguez-Picon, and A. Alvarado-Iniesta, "Degradation modeling of 2 fatigue-crack growth characteristics based on inverse Gaussian processes: A case study," *in IEEE Transactions Reliability,* vol. 35, issue 3, pp. 504-521, 2018.

[68] L. Shen, Y. Zhang, K. Song, and B. Song, "Failure analysis of a lock mechanism with multiple dependent components based on two-phase degradation model," *Engineering Failure Analysis,* vol. 104, pp. 1076–1093, 2019.

[69] Z. Yang, S. Li, C. Chen, H. Mei, and Y. Liu, " Reliability prediction of rotary encoder based on multivariate accelerated degradation modeling," *Measurement,* vol. 152, 2019, 107395.

[70] S. Mireh, A. Khodadadi, and F. Haghighi, "Joint Modeling of Linear Degradation and Multiple Dependent Competing Risks Data under a Step-Stress Accelerated Degradation Test," *Journal of Statistical Theory and Applications,* vol. 17, Issue 2, pp. 340-358, 2018.

[71] X. Si, "An Adaptive Prognostic Approach via Nonlinear Degradation Modeling: Application to Battery Data," *in IEEE Transactions on Industrial Electronics,* vol. 62, no. 8, pp. 5082-5096, 2015. doi: 10.1109/TIE.2015.2393840

[72] R. Singleton, E. Strangas, and S. Aviyente, "Extended Kalman Filtering for Remaining-Useful-Life Estimation of Bearings," *in IEEE Transactions on Industrial Electronics,* vol. 62, no. 3, pp. 1781-1790, 2015. doi: 10.1109/TIE.2014.2336616.

[73] X. Si, T. Li and Q. Zhang, "A General Stochastic Degradation Modeling Approach for Prognostics of Degrading Systems With Surviving and Uncertain Measurements," *in IEEE Transactions on Reliability,* vol. 68, no. 3, pp. 1080-1100, 2019. doi: 10.1109/TR.2019.2908492.

[74] D. He, E. Bechhoefer, J. Ma, and R. Li, "Particle filtering based gear prognostics using one-dimensional health index," *Annual Conference of the Prognostics and Health Management Society,Montreal, QC, Canada,* 2011.

[75] M. Orchard and G. Vachtsevanos, "A particle-filtering approach for on-line fault diagnosis and failure prognosis," *Institute of Measurement and Control Transactions,* vol. 31, no. 3-4, pp. 221-246, 2009.

[76] P. Wang and R. Gao, "Particle Filtering-Based System Degradation Prediction Applied to Jet Engines," *Annual Conference of the Prognostics and Health Management Society,* vol. 5, no. 067, pp. 658-663, 2014.

[77] Y. Qian and R. Yan, "Remaining Useful Life Prediction of Rolling Bearings Using an Enhanced Particle Filter," *in IEEE Transactions On Instrumentation and Measurement,* vol. 64, no. 10, pp. 2696-2707, 2015.

[78] A. Malhi, R. Yan and R. X. Gao, "Prognosis of Defect Propagation Based on Recurrent Neural Networks," *in IEEE Transactions on Instrumentation and Measurement,* vol. 60, no. 3, pp. 703-711, 2011. doi: 10.1109/TIM.2010.2078296.

[79] K. Medjaher, D. Tobon-Mejia, and N. Zerhouni, "Remaining useful life estimation of critical components with application to bearings," *in IEEE Transactions on Reliability,* vol. 61, no. 2, pp. 292-302, 2012.

[80] D. Wijayasekara, O. Linda, M. Manic and C. Rieger, "FN-DFE: Fuzzy-Neural Data Fusion Engine for Enhanced Resilient State-Awareness of Hybrid Energy Systems," *in IEEE Transactions on Cybernetics,* vol. 44, no. 11, pp. 2065-2075, 2014. doi: 10.1109/TCYB.2014.2323891.

[81] X. Si, C. Hu, Q. Zhang and T. Li, "An Integrated Reliability Estimation Approach With Stochastic Filtering and Degradation Modeling for Phased-Mission Systems," *in IEEE Transactions on Cybernetics,* vol. 47, no. 1, pp. 67-80, 2017. doi: 10.1109/TCYB.2015.2507370.

[82] X. Wang, B. Jiang, N. Lu, and C. Zhang, "Dynamic fault prognosis for multivariate degradation process," *Neurocomputing,* vol. 275, pp. 1112-1120, 2018.

[83] A. Al-dulaimi, S. Zabihi, A. Asif, A. Mohammadi, "NBLSTM: Noisy and Hybrid CNN and BLSTM-based Deep Architecture for Remaining Useful Life Estimation," *ASME Journal of Computing and Information Science in Engineering,* no. 1182, 2019.

[84] 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.

[85] N. Daroogheh, A. Baniamerian, N. Meskin, and K. Khorasani, "Prognosis and health monitoring of nonlinear systems using a hybrid scheme through integration of PFs and neural networks," *IEEE Transactions On On Systems,* vol. 47, no. 8, pp. 1990-2004, 2017.

[86] Y. Qian and R. Yan, "Ensemble neural network-based particle filtering for prognostics," *Mechanical Systems and Signal Processing,* vol. 41, no. 1-2, pp. 288-300, 2013.

[87] J. Deutsch, M. He, and D. He, "Remaining Useful Life Prediction of Hybrid Ceramic Bearings Using an Integrated Deep Learning and Particle Filter Approach," *Applied Sciences,* vol. 7, no. 7, pp. 649, 2017.

[88] X. Li and Y. Zhang, "Numerically robust implementation of multiple model algorithms," *in IEEE Transactions on Aerospace and Electronic Systems,* vol. 36, no. 1, pp. 266-278, 2000. doi: 10.1109/7.826329.

[89] R. Hallouzi, M. Verhaegen, and S. Kanev, "Multiple model estimation: A convex model formulation," *in International Journal of Adaptive Control and Signal Processing,* vol. 23, no.3, 2009. DOI: 10.1002/acs.1034.

[90] R. Nikoukhah, S. Campbell, K. Horton and F. Delebecque, "Auxiliary signal design for robust multimodel identification," *in IEEE Transactions on Automatic Contro,* vol. 47, no. 1, pp. 158-164, 2002. doi: 10.1109/9.981737.

[91] F. Alrowaie, R.Gopaluni, and K. Kwok, "Fault detection and isolation in stochastic non-linear state-space models using particle filters," *Control Engineering Practice,* vol. 20, Issue 10, pp. 1016-1032, 2012.

[92] F. Cadini, E. Zio and G. Peloni, "Particle Filtering for the Detection of Fault Onset Time in Hybrid Dynamic Systems With Autonomous Transitions," *in IEEE Transactions on Reliability,* vol. 61, no. 1, pp. 130-139, 2012. doi: 10.1109/TR.2011.2182224.

[93] D. Rupp, G. Ducard, E. Shafai, and H. Geering, "Extended Multiple Model Adaptive Estimation for the Detection of Sensor and Actuator Faults," *in Proceedings of the 44th IEEE Conference on Decision and Control,* pp. 3079-3084, 2005.doi: 10.1109/CDC.2005.1582634

[94] B. Pourbabaee, N. Meskin, and K. Khorasani, "Sensor Fault Detection, Isolation, and Identification Using Multiple-Model-Based Hybrid Kalman Filter for Gas Turbine Engines," *in IEEE Transactions on Control Systems Technology,* vol. 24, no. 4, pp. 1184-1200, 2016.doi: 10.1109/TCST.2015.2480003

[95] S. Zhao, B. Huang and F. Liu, "Fault Detection and Diagnosis of Multiple-Model Systems With Mismodeled Transition Probabilities," *in IEEE Transactions on Industrial Electronics,* vol. 62, no. 8, pp. 5063-5071, 2015. doi: 10.1109/TIE.2015.2402112.

[96] Y. Zhang and X. Li, "Detection and diagnosis of sensor and actuator failures using IMM estimator," *in IEEE Transactions on Aerospace and Electronic Systems,* vol. 34, no. 4, pp. 1293-1313, 1998. doi: 10.1109/7.722715.

[97] X. Wang and V. Syrmos, "Interacting Multiple Particle Filters For Fault Diagnosis of Non-linear Stochastic Systems," *IEEE American Control Conference,* pp. 4274-4279, 2008.

[98] M. Compare, P. Baraldi, P. Turati, and E. Zio, "Interacting multiple-models, state augmented Particle Filtering for fault diagnostics," *Probabilistic Engineering Mechanics,* vol. 40, pp.12-24, 2015.

[99] M. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking," *in IEEE Transactions on Aerospace and Electronic Systems,* vol. 50, no. 2, pp. 174-188, 2002. doi: 10.1109/78.978374.

[100] P. M. Djuric et al., "Particle filtering," *in IEEE Signal Processing Magazine,* vol. 20, no. 5, pp. 19-38, 2003. doi: 10.1109/MSP.2003.1236770.

[101] D. Samarjit, K. Amit, and V. Namrata, "Particle filter with a mode tracker for visual tracking across illumination changes," *IEEE Transactions on Image Processing* 2012; 21 (4):2340–2346.

[102] D. Marko, S. Milorad, and M. Nenad, "A variable neighborhood search particle filter for bearings-only target tracking," *Computers and Operations Research* 2014; 52(B):192–202.

[103] D. Arnaud, G. Simon, A. Christophe, "On sequential Monte Carlo methods for Bayesian filtering," *Kluwer Academic Publishers, Statistics and Computing* 2000; 10: 197–208

[104] M. Arulampalam, M. Simon, G. Neil, and C. Tim, "A tutorial on particle filters for online nonlinear/non-gaussian Bayesian tracking," *IEEE Transactions on Signal Processing* 2002.50(2):174-188.

[105] R. Douc and O. Cappé, "Comparison of resampling schemes for particle filtering," *in Proc. Intell. Symp. Image Signal Processing Anal.* 2005; 64-69.

[106] N. J. Gordon, D. J. Salmond, and A. F. M. Smith, "Novel approach to nonlinear/non-gaussian Bayesian state estimation," *IEE Proceedings, Part F: Radar and Signal Processing* 1993; 140(2): 107–113.

[107] S. Zhao, B. Huang, and F. Liu, "Identification and Convergence Analysis of a Class of Continuous-Time Multiple-Model Adaptive Estimators," *IFAC Proceedings Volumes,* vol. 41,Issue 2, 2008, pp. 8605-8610, 2015. doi: 10.1109

[108] D. Magill, "Optimal adaptive estimation of sampled stochastic processes," *in IEEE Transactions on Automatic Control,* vol. 10, no. 4, pp. 434-439, 1965. doi: 10.1109/TAC.1965.1098191

[109] J. Ru, and X. Li, "Variable-Structure Multiple-Model Approach to Fault Detection, Identification, and Estimation," *in IEEE Transactions on Control Systems Technology,* vol. 16, no. 5, pp. 1029-1038, 2006. doi: 10.1109/TCST.2007.916318.

[110] P. Maybeck, "Multiple model adaptive algorithms for detecting and compensating sensor and actuator/surface failures in aircraft flight control systems," *Mathematics,*1999. Corpus ID: 122215529. DOI:10.1002/(SICI)1099-1239(19991215)9:14¡1051::AID-RNC452¿3.0.CO;2-0

[111] P. Maybeck, and K. Hentz, "Investigation of moving-bank multiple model adaptive algorithms," *IEEE Conference on Decision and Control,* pp. 1874-1881, 1985. doi: 10.1109/CDC.1985.268907.

[112] J. Vasquez, and P. Maybeck, "Enhanced motion and sizing of bank in moving-bank MMAE," *in IEEE Transactions on Aerospace and Electronic Systems,* vol. 40, no. 3, pp. 770-779, 2004. doi: 10.1109/TAES.2004.1337453.

[113] P. Eide, and P. Maybeck, "An MMAE failure detection system for the F-16," *in IEEE Transactions on Aerospace and Electronic Systems,* vol. 32, no. 3, pp. 1125-1136, 1996. doi: 10.1109/7.532271.

[114] P. Lu, and E. Kampen, "Selective Reinitialisation Multiple Model Adaptive Estimation for Fault Detection and Diagnosis," *AIAA Guidance, Navigation, and Control Conference,* 2014. DOI: 10.2514/6.2014-0965.

[115] W. Zhang, S. Wang, and Y. Zhang "Multiple-Model Adaptive Estimation with A New Weighting Algorithm," *Complexity,* vol. 2018, pp. 1-11, 2018. DOI: 10.1155/2018/4789142

[116] M. Almasri, N. Tricot, and R. Lenain, "Multiple Model Adaptive Estimation for Blocked Wheel Fault Detection on Mobile Robots," *Computer Science,*2018, Corpus ID: 58005404.

[117] P. Lu, L. Eykeren, E. Kampen, C. Visser, and Q. Chu, "Double-model adaptive fault detection and diagnosis applied to real flight data," *Control Engineering Practice,* vol. 36, pp. 39-57, 2015.

[118] M. Vaezi and A. Izadian, "Multiple-model adaptive estimation of a hydraulic wind power system," *Annual Conference of the IEEE Industrial Electronics Society,* pp. 2111-2116, 2013. doi: 10.1109/IECON.2013.6699457.

[119] J. Farber, and D. Cole, "Using Multiple-Model Adaptive Estimation And System Identification For Fault Detection In Nuclear Power Plants," *in Proceedings of the ASME International Mechanical Engineering Congress and Exposition IMECE,* 2018, 87616.

[120] Z. Renwick, D. Tilbury, and E. Atkins, "A Reinforcement Learning Based Adaptive Supervisor for Multiple Model Adaptive Estimation and Control," *IEEE International Conference on Electro/Information Technology (EIT),* vol. 36, pp. 0216-0221, 2018. doi: 10.1109/EIT.2018.8500247.

[121] H. Blom and Y. Bar-Shalom, "The interacting multiple model algorithm for systems with markovian switching coefficients," *in IEEE Transactions on Automatic Control,* vol. 33, no. 8, pp. 780-783, 1988. doi: 10.1109/9.1299.

[122] W. Vianna, and T. Yoneyama, "Interactive Multiple-Model Application for Hydraulic Servovalve Health Monitoring," *Annual Conference of the Prognostics and Health Management Society,* vol. 6, no. 15, 2015.

[123] X Li, and L. Jilkov, "Expected-mode augmentation for multiple-model estimation," *in International conferenceoninformationfusion,* 2000.

[124] Y. Yan, M. Mallick, J. Zhang, and J. Liu, "Prognostics by interacting multiple model estimator," *IEEE International Conference on Prognostics and Health Management (ICPHM),* pp. 1-8, 2016. doi: 10.1109/ICPHM.2016.7542822

[125] Y. Zhang and X. Li "Detection and Diagnosis of Sensor and Actuator Failures Using IMM Estimator," *in IEEE Transactions on Aerospace and Electronic Systems,* vol. 34, no. 4, pp. 1293-1313, 1998

[126] S. Kim, J. Choi, and Y. Kim, "Fault detection and diagnosis of aircraft actuators using fuzzy-tuning IMM filter," *in IEEE Transactions on Aerospace and Electronic Systems,* vol. 44, no. 3, pp. 940-952, 2008. doi: 10.1109/TAES.2008.4655354.

[127] S. Zhao, B. Huang and F. Liu, "Detection and Diagnosis of Multiple Faults With Uncertain Modeling Parameters," *in IEEE Transactions on Control Systems Technology,* vol. 25, no. 5, pp. 1873-1881, 2017.

[128] V. Judalet, S. Glaser, D. Gruyer, and S. Mammar, "Fault Detection and Isolation via the Interacting Multiple Model Approach Applied to Drive-By-Wire Vehicles," *Sensors,* vol. 18, no. 7, 2018. 2332. DOI: 10.3390/s18072332.

[129] J Sikorska, M. Hodkiewicz, and L.Ma, "Prognostic modeling options for remaining useful life estimation by industry by industry," *Mechanical Systems and Signal Processing,* vol. 25, pp. 1803–1836, 2011. 2332.

[130] R. Kothamasu, S. Huang, and W. VerDuin, "System health monitoring and prognostics - a review of current paradigms and practices," *Int J Adv Manuf Technol,* vol. 28, pp. 1012–1024, 2006.

[131] D. Liu, J. Zhou, H. Liao, Y. Peng and X. Peng, "A Health Indicator Extraction and Optimization Framework for Lithium-Ion Battery Degradation Modeling and Prognostics," *in IEEE Transactions on Systems, Man, and Cybernetics: Systems,* vol. 45, no. 6, pp. 915-928, 2015.

[132] M. Chookah, M. Nuhi, and M. Modarres, "A probabilistic physics-of-failure model for prognostic health management of structures subject to pitting and corrosion fatigue," *Reliability Engineering and System Safety,* vol. 96, no. 12, pp. 1601–1610, 2011.

[133] S. Hong, Z. Zhou, and C. Lv, "Storage lifetime prognosis of an intermediate frequency (if) amplifier based on physics of failure method," *Chemical Engineering,* vol. 33, pp. 1117–1122, 2013.

[134] J. Fan, K. Yung, and M. Pecht, "Physics of failure based prognostics and health management for high power white light emitting diode lighting," *IEEE Transactions on Device and Materials Reliability,* vol. 11, no. 3, pp. 407–416, 2011.

[135] J. Lee, F. Wu, W. Zhao, M. Ghaffari, L. Liao, and D. Siegel, "Prognostics and health management design for rotary machinery systems - Reviews, methodology and applications," *Mechanical Systems and Signal Processing,* vol. 42, no. 1-2, pp. 314–334, 2014

[136] J. Ramuhalli, P. Bond, J. Hines, and B. Upadhyaya, " A review of prognostics and health management applications in nuclear power plants," *International Journal of Prognostics and Health Management,* vol. 6, no. 16, pp. 1–22, 2015.

[137] O. Dragomir, R. Gouriveau, F. Dragomir, E. Minca, and N. Zerhouni, " Review of prognostic problem in condition-based maintenance," *IFAC and in collaboration with the IEEE Control Systems Society. European Control Conference,* pp.1585-1592, 2009.

[138] J. Luo, M. Namburu, K. Pattipati, L. Qiao, M. Kawamoto, and S. Chigusa, " Model Based Prognostic Techniques," *IEEE Systems Readiness Technology Conference,* pp. 330-340, 2003.

[139] S. Yin, X. Li, H. Gao, and O. Kaynak, "Data-based techniques focused on modern industry: An overview," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 1, pp. 657–667, 2015.

[140] F. Ahmadzadeh and J. Lundberg, "Remaining useful life estimation: review," *International Journal of System Assurance Engineering and Management,* vol. 5, pp. 461–474, 2014.

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

[142] A. Widodo and B. Yang, "Support vector machine in machine condition monitoring and fault diagnosis," *Mechanical systems and signal processing*, vol. 21, no. 6, pp. 2560–2574, 2007.

[143] V. Muralidharan and V. Sugumaran, "A comparative study of naive bayes classifier and bayes net classifier for fault diagnosis of monoblock centrifugal pump using wavelet analysis," *Applied Soft Computing*, vol. 12, no. 8, pp. 2023–2029, 2012.

[144] M. Chen, U. Challita, W. Saad, C. Yin and M. Debbah, "Artificial Neural Networks-Based Machine Learning for Wireless Networks: A Tutorial," *in IEEE Communications Surveys & Tutorials,* vol. 21, no. 4, pp. 3039-3071, 2019.

[145] A. Tenney, M. Glauser, C. Ruscher and Z. Berger "Application of Artificial Neural Networks to Stochastic Estimation and Jet Noise Modeling," *AIAAJ Journal,* vol. 58, no. 2, 2020. https://doi.org/10.2514/1.J058638.

[146] M. Tkac and R. Verner "Artificial neural networks in business: Two decades of research," *Applied Soft Computing,* vol. 38, pp. 788-804, 2016.

[147] G. Marcus, "Deep Learning: A Critical Appraisal," *Artificial Intelligence (cs.AI) ,* arXiv:1801.00631v1 [cs.AI], 2018.

[148] J. Wang, Y. Ma, L. Zhang, R. Gao, and D. Wu , "Deep learning for smart manufacturing: Methods and applications," *Journal of Manufacturing Systems,* vol. 48, Part C, pp. 144-156, 2018.

[149] P. Ongsulee, "Artificial intelligence, machine learning and deep learning," *in international Conference on ICT and Knowledge Engineering (ICT&KE),* pp. 1-6, 2017.

[150] A. Karoly, R. Fuller, and P. Galambos, "Unsupervised Clustering for Deep Learning:A tutorial survey," *in Acta Polytechnica Hungarica,* vol. 15, no. 8, pp. 29-53, 2018. DOI: 10.12700/APH.15.8.2018.8.2.

[151] S. Paliwal, "Machine Learning on Cloud Platforms: Machine Learning on Cloud Platforms(Google CloudPlatform, Microsoft Azure , Amazon web Wervices)- Volume 1," *Amazon.com Services LLC ,* 2017.ASIN: B082DB4H55.

[152] Y. Jaafra, J. Laurent, A. Deruyver, M. Naceur, "A Review of Meta-Reinforcement Learning for Deep Neural Networks Architecture Search," *arXiv:1812.07995v1 [cs.LG],* 2018.

[153] S. Khan, T. Yairi, "A review on the application of deep learning in system health management," *Mechanical Systems and Signal Processing*, vol. 107, pp. 241-265, 2018.

[154] R. Zhao, R. Yan, Z. Chen, K. Mao, P. Wang, and R. Gao "Deep Learning and its Applications to Machine Health Monitoring," *Mechanical Systems and Signal Processing*, vol. 115, pp. 213-237, 2019.

[155] S. Trenn, "Multilayer Perceptrons: Approximation Order and Necessary Number of Hidden Units," *in IEEE Transactions on Neural Networks*, vol. 19, no. 5, pp. 836-844, 2008.

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

[157] H. Faris, I. Aljarah, and S. Mirjalili, "Training feedforward neural networks using multi-verse optimizer for binary classification problems," *Applied Intelligence*, vol. 45, pp. 322–332, 2016.

[158] J. Fengi and S. Lu, "Performance Analysis of Various Activation Functions in Artificial Neural Networks," *Journal of Physics: Conf. Series*, vol. 1237, issue 2, 2019. doi:10.1088/1742-6596/1237/2/022030.

[159] N. Buduma and N. Locascio, " Multilayer Perceptron: Architecture Optimization and Training," *International Journal of Interactive Multimedia and Artificial Intelligence*, vol. 4, no.1, pp. 26-30, 2016.

[160] N. Buduma and N. Locascio, "Fundamentals of Deep Learning Designing Next-Generation Machine Intelligence Algorithms," *O'Reilly Media, Inc*, 2017. ISBN: 9781491925614.

[161] S. Basu, N. Das, R. Sarkar, M. Kundu, M. Nasipuri, and D. Basu, "An MLP based Approach for Recognition of Handwritten 'Bangla' Numerals," *arXiv:1203.0876v1 [cs.CV]* , 2012.

[162] P. Naraei, A. Abhari and A. Sadeghian, "Application of multilayer perceptron neural networks and support vector machines in classification of healthcare data," *Future Technologies Conference (FTC)*, pp. 848-852, 2016.

[163] D. Das, A. Tripathi, A. Shah, and S. Mehta, " Application of Multilayer Perceptron for Forecasting of Selected IIPs of India –An Empirical Analysis," *International Journal of Computer Sciences and Engineering* , vol. 6, issue 11, 2018.

[164] A. Thakur and D. Mishra, " Hyper spectral image classification using multilayer perceptron neural network and functional link ANN," *International Conference on Cloud Computing, Data Science and Engineering - Confluence* , pp. 639-642, 2017.

[165] O. Sezer, M. Ozbayoglu, and E. Dogdu, "An Artificial Neural Network-based Stock Trading System Using Technical Analysis and Big Data Framework," *arXiv:1712.09592v1 [cs.CE]*, 2017.

[166] R. Huang, L. Xi, X. Li, C. Liu, H. Qiu, and J. Lee, "Residual life predictions for ball bearings based on self-organizing map and back propagation neural network methods," *Mechanical Systems and Signal Processing*, vol. 21, issue 1, pp. 193-207, 2007.

[167] J. Kim, J. Yu, M. Kim, K. Kim, and S. Han "Estimation of Li-ion Battery State of Health based on Multilayer Perceptron: as an EV Application," *IFAC-PapersOnLine*, vol. 51, issue 28, pp. 392-397, 2018.

[168] L. Jedlinski and J. Jonak, "Early fault detection in gearboxes based on support vector machinesand multilayer perceptron with a continuous wavelet transform," *Applied Soft Computing*, vol. 30, pp. 636–641, 2015.

[169] X. Hu, J. Vian, J. Slepski and D. Wunsch, "Vibration analysis via neural network inverse models to determine aircraft engine unbalance condition," *Proceedings of the International Joint Conference on Neural Networks*, vol. 4, pp. 3001-3006, 2003.

[170] L. de Almeida, J. Bizarria, F. Bizarria, and M. Mathias, "Condition-based monitoring system for rolling element bearing using a generic multi-layer perceptron," *Journal of Vibration and Control*, vol. 21, no. 16, 2014. DOI: 10.1177/1077546314524260.

[171] O. Geramifard, J. Xu, C. Pang, J. Zhou, and X. Li, "Data-driven approaches in health condition monitoring — A comparative study," *IEEE ICCA*, pp. 1618-1622, 2010.

[172] I. Loboda, Y. Feldshteyn, and V. Ponomaryov, "Neural Networks for Gas Turbine Fault Identification:Multilayer Perceptron or Radial Basis Network," *International Journal of Turbo and Jet-Engines*, vol. 29, issue 1, 2012.

[173] S. Zolfaghari, S. Noor, M. Mehrjou, M. Marhaban, and N. Mariun, "Broken Rotor Bar Fault Detection and Classification Using Wavelet Packet Signature Analysis Based on Fourier Transform and Multi-Layer Perceptron Neural Network," *Applied Sciences* , vol. 8, issue 1, 2018. https://doi.org/10.3390/app8010025.

[174] M. Heidari and S. Shateyi, "Wavelet support vector machine and multi-layer perceptron neural network with continues wavelet transform for fault diagnosis of gearboxes," *Journal of Vibroengineering* ,vol. 19, no. 1, pp. 125-137, 2017. DOI: 10.21595/jve.2016.16813 .

[175] Y. Cun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard and L. Jackel, "Handwritten Digit Recognition with a Back-Propagation Network," *Proceedings of the International Conference on Neural Information Processing Systems*,vol. 2, pp. 396-404, 1990.

[176] W. Zhang, G. Peng and C. Li, "Rolling Element Bearings Fault Intelligent Diagnosis Based on Convolutional Neural Networks Using Raw Sensing Signal,," *Advances in Intelligent Information Hiding and Multimedia Signal Processing*, vol. 64, pp 77-84, 2017.

[177] O. Abdel-Hamid, A. Mohamed, H. Jiang, L. Deng, G. Penn and D. Yu, "Convolutional Neural Networks for Speech Recognition," *in IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, no. 10, pp. 1533-1545, 2014.

[178] F. Longueira ND S. Keene, "A Fully Convolutional Neural Network Approach to End-to-End Speech Enhancement," *arXiv:1807.07959v1 [cs.SD]*, 2018.

[179] A. Mahendran and A. Vedaldi, "Visualizing deep convolutional neural networks using natural pre-images," *International Journal of Computer Vision volume*, vol. 120, no. 3, pp. 233–255, 2016.

[180] J. Acquarelli, E. Marchiori, L. Buydens, T. Tran, and T. Laarhoven, "Spectral-Spatial Classification of Hyperspectral Images: Three Tricks and a New Learning Setting," *Remote Sensing*, vol. 10, issue 7, 2018. https://doi.org/10.3390/rs10071156.

[181] 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.

[182] A. Krizhevsky, I. Sutskever and G. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Advances in Neural Information Processing Systems*, pp. 1097-1105, 2012.

[183] A. Conneau, H. Schwenk, L. Barrault, and Y. LeCun, "Very Deep Convolutional Networks for Natural Language Processing," *arXiv:1606.01781v2 [cs.CL]*, 2017.

[184] R. Yamashita, M. Nishio, R. Do, and K. Togashi, "Convolutional neural networks: an overview and application in radiology," *Insights into Imaging*, vol. 9, Issue 4, pp. 611-629, 2018.

[185] O. Abdel-Hamid, A. Mohamed, H. Jiang and G. Penn, "Applying Convolutional Neural Networks concepts to hybrid NN-HMM model for speech recognition," *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4277-4280, 2012.

[186] Y. Zhang, D. Zhao, J. Sun, G. Zou, and W. Li, "Adaptive Convolutional Neural Network and Its Application in Face Recognition," *Neural Processing Letters*, vol. 43, pp. 389–399, 2016.

[187] K. Chen and W. Tao, "Once for All: A Two-Flow Convolutional Neural Network for Visual Tracking," *in IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 12, pp. 3377-3386, 2018.

[188] Y. Zheng, Q. Liu, E. Chen, Y. Ge, and J. Zhao, "Time Series Classification using Multi-channels Deep Convolutional Neural Networks," *Proceedings of the International Conference on Web-Age Information Management*, vol. 8485, pp. 298-310, 2014.

[189] Z. Chen, C. Li, and R. Sanchez, "Gearbox Fault Identification and Classification with Convolutional Neural Networks," *Shock and Vibration*, vol. 2, pp. 1-10, 2015.

[190] D. Lee, V. Siu, R. Cruz, and C. Yetman, " Convolutional Neural Net and Bearing Fault Analysis," *Proceedings of the International Conference on Data Mining(DMIN)*, pp. 194-200, 2016.

[191] J. Wang, J. Zhuang, L. Duan, and W. Cheng, "A Multi-scale Convolution Neural Network for Featureless Fault Diagnosis," *Proceedings of the International Symposium of Flexible Automation (ISFA)*, pp. 65-70, 2016.

[192] G. Babu, P. Zhao, and X. Li, "Deep Convolutional Neural Network based Regression Approach for Estimation of Remaining Useful Life," *Proceedings of the International Conference on Database Systems for Advanced Applications*, vol. 9642, pp. 214-228, 2016.

[193] W. Sun, R. Zhao, R. Yan, S. Shao, and X. Chen, "Convolutional Discriminative Feature Learning for Induction Motor Fault Diagnosis," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 3, pp. 1350-1359, 2017.

[194] L. Ren, Y. Sun, H. Wang, and L. Zhang "Prediction of Bearing Remaining Useful Life With Deep Convolution Neural Network," *IEEE Access*, vol. 6, pp. 13041-13049, 2018.

[195] V. Pham, S. Han, M. Do and H. Choi, "A wavelet packet spectral subtraction and convolutional neural network based method for diagnosis of system health," *Journal of Mechanical Science and Technology*, vol. 33, no. 12, pp. 5683-5687, 2019.

[196] C. Maior, M. Santos, J. Santana,A. Negreiros, I. Lins, and E. Droguett, "Convolutional Neural Network for remaining useful life prediction based on vibration signal," *European Safety and Reliability Conference,* vol. 33, no. 12, pp. 5683-5687, 2019.

[197] A. Sherstinsky, "Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Networkl," *arXiv:1808.03314v6 [cs.LG],* 2020. DOI:10.1016/j.physd.2019.132306.

[198] S. Raschka and V. Mirjalili, "Machine Learning and Deep Learning with Python, scikit-learn and TensorFlow," *Packt Publishing Ltd,* 2017, ISBN 978-1-78712-593-3.

[199] M. Schuster , K. Paliwal, "Bidirectional recurrent neural networks," *in IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673-2681, 1997.

[200] F. Gers, J. Schmidhuber, and F. Cummins, "Learning to Forget: Continual Prediction with LSTM," *Neural computation*, vol. 12, no. 10, pp. 2451-2471, 2000.

[201] A. Graves, and J. Schmidhuber, "Framewise Phoneme Classi

cation with Bidirectional LSTM and other Neural Network Architectures," *Neural Networks*, vol. 18, no. 5, pp. 602-610, 2005.

[202] W. Di, A. Bhardwaj, and J. Wei, "Deep Learning Essentials: Your hands-on Guide to the Gundamentals of Deep Learning and Neural Network Modeling," *Packt Publishing - ebooks Account*, 2018, ISBN: 9781785880360.

[203] R. Staudemeyer and E. Morris, "Understanding LSTM – a tutorial into Long Short-Term Memory Recurrent Neural Networks," *arXiv:1909.09586v1 [cs.NE]*, 2019.

[204] T. Fischer and C. Krauss, "Deep learning with long short-term memory networks for financial market predictions," *European Journal of Operational Research*, vol. 270, Issue 2, pp. 654-669, 2018.

[205] J. Lei, C. Liu, and D. Jiang, "Fault diagnosis of wind turbine based on Long Short-term memory networks," *Renewable Energy*, vol. 133, pp. 422-432, 2019.

[206] A. Graves, D. Eck, N. Beringer, and J. Schmidhuber, "Biologically Plausible Speech Recognition withLSTM Neural Nets," *Ijspeert A.J., Murata M., Wakamiya N. (eds) Biologically Inspired Approaches to Advanced Information Technology. BioADIT*, vol. 3141, pp. 127-136, 2004. Online ISBN 978-3-540-27835-1.

[207] J. Chen and D. Wang, "Long short-term memory for speaker generalization in supervised speech separation," *The Journal of the Acoustical Society of America*, vol. 141, no. 4705, 2017; https://doi.org/10.1121/1.4986931

[208] T. Young, D. Hazarika, S. Poria and E. Cambria, "Recent Trends in Deep Learning Based Natural Language Processing [Review Article]," *in IEEE Computational Intelligence Magazine*, vol. 13, no. 3, pp. 55-75, 2018.

[209] M. Sundermeyer, H. Ney and R. Schlüter, "From Feedforward to Recurrent LSTM Neural Networks for Language Modeling," *in IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 3, pp. 517-529, 2015.

[210] S. Wang and J. Jiang, "Learning Natural Language Inference with LSTM," *arXiv:1512.08849v2 [cs.CL]* , 2016.

[211] M. Baccouche, F. Mamalet, C. Wolf, C. Garcia, and A. Baskurt, "Sequential Deep Learning for Human Action Recognition," *International Workshop on Human Behavior Understanding HBU: Human Behavior Understanding*, vol. 7065, pp. 29-39, 2011.

[212] L. Sun, T. Su, C. Liu, and R. Wang, "Deep LSTM Networks for Online Chinese Handwriting Recognition," *Proceedings of the IEEE International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pp. 271-276, 2016.

[213] Q. You, H. Jin, Z. Wang, C. Fang, and J. Luo, "Image Captioning with Semantic Attentio," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4651-4659, 2016.

[214] S. Zheng, K. Ristovski, A. Farahat, and C. Gupta, "Long Short-Term Memory Network for Remaining Useful Life Estimation," *Proceedings of the IEEE International Conference on Prognostics and Health Management (ICPHM)*, pp. 88-95, 2017.

[215] 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.

[216] P. Malhotra, V. TV, A Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff, "Multi-Sensor Prognostics using an Unsupervised Health Index based on LSTM Encoder-Decoder," *ACM SIGKDD Workshop on Machine Learning for Prognostics and Health Management*, 2016.

[217] Y. Zhou, Y. Huang, J. Pang, and K. Wang, "Remaining useful life prediction for supercapacitor based on long short-term memory neural network," *Journal of Power Sources*, vol 440, 2019. 227149.

[218] J. Liu, Q. Li, W. Chen, Y. Yan, Y. Qiu, and T. Cao, "Remaining useful life prediction of PEMFC based on long short-term memory recurrent neural networks," *International Journal of Hydrogen Energy*, vol. 44, Issue 11, pp. 5470-5480, 2019.

[219] F Wang, X. Liu, G. Deng, X. Yu, H. Li, and Q. Han, "Remaining Life Prediction Method for Rolling Bearing Based on the Long Short-Term Memory Network," *Neural Processing Letters*, vol. 50, pp. 2437–2454, 2019.

[220] J. Wu, K. Hu, Y. Cheng, H. Zhu, X. Shao, and Y. Wang, "Data-driven remaining useful life prediction via multiple sensor signals and deep long short-term memory neural network," *ISA Transactions* , vol. 97, pp. 241-250, 2020.

[221] A. Graves, N. Jaitly, and A. Mohamed, ""Hybrid speech recognition with deep bidirectional lstm," *in Automatic Speech Recognition and Understanding (ASRU)*, IEEE Workshop, pp. 273-278, 2013.

[222] D. Tao, Y. Wen and R. Hong, ""Multicolumn Bidirectional Long Short-Term Memory for Mobile Devices-Based Human Activity Recognition," *in IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 1124-1134, 2016.

[223] R. Saini, P. Kumar, B. Kaur, P. Roy, D. Dogra, and K. Santosh, ""Kinect sensor-based interaction monitoring system using the BLSTM neural network in healthcare," *International Journal of Machine Learning and Cybernetics*, vol. 10, pp. 2529–2540, 2019.

[224] Q. Zou,Q. Li, H. Yi, Y. Yu, and C. Wu, ""A water quality prediction method based on the multi-time scale bidirectional long short-term memory network," *Environmental Science and Pollution Research*, 2020.

[225] J. Wang, G. Wen, S. Yang and Y. Liu, "Remaining Useful Life Estimation in Prognostics Using Deep Bidirectional LSTM Neural Network," *Prognostics and System Health Management Conference* , pp. 1037-1042, 2018

[226] D. Wilson, S. Passmore, Y. Tang, and J. Vanzwieten, ""Bidirectional Long Short-Term Memory Networks for Rapid Fault Detection in Marine Hydrokinetic Turbines," *IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 495-500., 2018.

[227] H. Li, W. Wang, Z. Li, L. Dong, and Q. Li, ""A novel approach for predicting tool remaining useful life using limited data," *Mechanical Systems and Signal Processing*, vol. 143, 2020. 106832.

[228] C. Bian, H. He, and S. Yang, ""Stacked bidirectional long short-term memory networks for state-of-charge estimation of lithium-ion batteries," *Energy*, vol. 191, 2020, 116538.

[229] F. Wang, T. Mamo, and X. Cheng, ""Bi-directional long short-term memory recurrent neural network with attention for stack voltage degradation from proton exchange membrane fuel cells," *Journal of Power Sources*, vol. 46, 2020, 228170.

[230] D. Qiu, Z. Liu, Y. Zhou and J. Shi, ""Modified Bi-Directional LSTM Neural Networks for Rolling Bearing Fault Diagnosis," *IEEE International Conference on Communications (ICC)*, pp. 1-6, 2019.

[231] X. Li, J. Li, C. Zhao, Y.i Qu, and D. He, ""Early Gear Pitting Fault Diagnosis Based on Bi-directional LSTM," *Prognostics and System Health Management Conference*, 2019.

[232] X. Wu, J. Li, Y. Jin, and S. Zheng "Modeling and analysis of tool wear prediction based on SVD and BiLSTM," *The International Journal of Advanced Manufacturing Technology*, vol. 106, pp. 4391–4399, 2020.

[233] L. Cao, Z. Qian, H. Zareipour, Z. Huang and F. Zhang, ""Fault Diagnosis of Wind Turbine Gearbox Based on Deep Bi-Directional Long Short-Term Memory Under Time-Varying Non-Stationary Operating Conditions," *in IEEE Access*, vol. 7, pp. 155219-155228, 2019.

[234] K. Cho, B. Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation," *arXiv:1406.1078v3 [cs.CL],* 2014.

[235] C. Jinglong, J. Hongjie, C. Yuanhong, and L. Qian, "Gated recurrent unit based recurrent neural network for remaining useful life prediction of nonlinear deterioration process," *Reliability Engineering and System Safety*,vol. 185, pp. 372-382, 2019.

[236] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555,* 2014.

[237] A. Graves, A. Mohamed, and G. Hinton, "Speech Recognition with Deep Recurrent Neural Networks," *arXiv:1303.5778v1 [cs.NE]* , 2013

[238] I. Sutskever, O. Vinyals, and Q. Le, "Sequence to Sequence Learning with Neural Networks," *arXiv:1409.3215v3 [cs.CL]* , 2014.

[239] L. Chen, R. Yan, L. Peng, A. Furuhata and X. Ding, "Multi-layer recurrent neural network based offline Arabic handwriting recognition," *International Workshop on Arabic Script Analysis and Recognition (ASAR)*, pp. 6-10, 2017.

[240] H. Xu, C. Zhang, G. Hong, J. Zhou, J. Hong and K. S. Woon, "Gated Recurrent Units Based Neural Network For Tool Condition Monitoring," *International Joint Conference on Neural Networks (IJCNN)*, pp. 1-7, 2018.

[241] X. Li, H. Jiang, X. Xiong, and H. Shao, "Rolling bearing health prognosis using a modified health index based hierarchical gated recurrent unit network ," *Mechanism and Machine Theory*, vol. 133, pp. 229-249, 2019.

[242] H. Xu, G. S. Hong, J. H. Zhou, J. Hong and K. S. Woon, "Coarse-to-Fine Tool Condition Monitoring using Multiple Gated Recurrent Units," *Annual Conference of the IEEE Industrial Electronics Society,* pp. 3737-3742, 2019.

[243] S. Zhong, Z. Li, L. Lin and Y. Zhang, "Aero-Engine Exhaust Gas Temperature Prognostic Model Based on Gated Recurrent Unit Network," *IEEE International Conference on Prognostics and Health Management (ICPHM),* pp. 1-5, 2018.

[244] B. Xiao, Y. Liu and B. Xiao, "Accurate State-of-Charge Estimation Approach for Lithium-Ion Batteries by Gated Recurrent Unit With Ensemble Optimizer," *in IEEE Access,* vol. 7, pp. 54192-54202, 2019.

[245] F. Yang, W. Li, C. Li, Q. Miao, "State-of-charge estimation of lithium-ion batteries based on gated recurrent neural network," *Energy,* vol. 175, pp. 66-75, 2019.

[246] R. Zhao, D. Wang, R. Yan, K. Mao, F. Shen, and J. Wang, "Machine Health Monitoring Using Local Feature-based Gated Recurrent Unit Networks," *in IEEE Transactions on Industrial Electronics*, vol. 65, no. 2, pp. 1539-1548, 2018.

[247] D. Wang and K. Mao, "Multimodal Object Classification using Bidirectional Gated Recurrent Unit Networks," *Proceedings of the IEEE International Conference on Data Science in Cyberspace (DSC)*, pp. 685-690, 2018.

[248] N. Zerarin, S. Abdelhamids, H. Bouzgouh, and C. Raymondchristian, "Bidirectional deep architecture for Arabic speech recognition," *Open Computer Science*, vol. 9, issue. 1, 2019. DOI: https://doi.org/10.1515/comp-2019-0004.

[249] H. M. Lynn, S. B. Pan and P. Kim, "A Deep Bidirectional GRU Network Model for Biometric Electrocardiogram Classification Based on Recurrent Neural Networks," *in IEEE Access*, vol. 7, pp. 145395-145405, 2019.

[250] X. Tang, Y. Dai, T. Wang and Y. Chen, "Short-term power load forecasting based on multi-layer bidirectional recurrent neural network," *in IET Generation, Transmission & Distribution*, vol. 13, no. 17, pp. 3847-3854, 2019.

[251] Y. Deng, L. Wang, H. Jia, X. Tong and F. Li, "A Sequence-to-Sequence Deep Learning Architecture Based on Bidirectional GRU for Type Recognition and Time Location of Combined Power Quality Disturbance," *in IEEE Transactions on Industrial Informatics*, vol. 15, no. 8, pp. 4481-4493, 2019.

[252] Q. Tang, J. li, J. Chen, H. Lu, Y. Du and K. Yang, "Full Attention-Based Bi-GRU Neural Network for News Text Classification," *International Conference on Computer and Communications (ICCC)*, pp. 1970-1974, 2019.

[253] T. Fan, J. Zhu, Y. Cheng, Q. Li, D. Xue and R. Munnoch, "A New Direct Heart Sound Segmentation Approach using Bi-directional GRU," *International Conference on Automation and Computing (ICAC)*, pp. 1-5, 2018.

[254] D. Wang and K. Mao, "Multimodal Object Classification Using Bidirectional Gated Recurrent Unit Networks," *IEEE Third International Conference on Data Science in Cyberspace (DSC)*, pp. 685-690, 2018.

[255] T. Alsarhan, L. Alawneh, M. Al-Zinati and M. Al-Ayyoub, "Bidirectional Gated Recurrent Units For Human Activity Recognition Using Accelerometer Data," *IEEE SENSORS*, pp. 1-4, 2019.

[256] W. Yu, Y. Kim, and C. Mechefske, "Remaining useful life estimation using a bidirectional recurrent neural network based autoencoder scheme," *Mechanical Systems and Signal Processing*, vol. 129, pp. 764–780, 2019.

[257] D. Rengasamy, M. Jafari, B. Rothwell, X. Chen, and G. Figueredo, "Deep Learning with Dynamically Weighted Loss Function for Sensor-Based Prognostics and Health Management," *Sensors*, vol. 20, no. 3, pp. 723, 2020; https://doi.org/10.3390/s20030723.

[258] I. Remadna, S. Terrissa, R. Zemouri, S. Ayad, N. Zerhouni 'Unsupervised Feature Reduction Techniques with Bidirectional GRU Neural Network for Aircraft Engine RUL Estimation," *International Conference on Advanced Intelligent Systems for Sustainable Development (AI2SD)*, pp. 496-506, 2020.

[259] 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.

[260] R. Gouriveau, K. Medjaher, and N. Zerhouni, "From Prognostics and Health Systems Management to Predictive Maintenance 1. Monitoring and Prognostics," *John Wiley & Sons*, 2016

[261] K. Javed, R. Gouriveau, and N. Zerhouni, "State of the art and taxonomy of prognostics approaches, trends of prognostics applications and open issues towards maturity at different technology readiness levels," *Mechanical Systems and Signal Processing*, vol. 94, pp. 214–236, 2017.

[262] C. Ordonez, F. Lasheras, J. Roca-Pardinas, and F. Juez, "A hybrid ARIMA–SVM model for the study of the remaining useful life of aircraft engines," *Journal of Computational and Applied Mathematics*, vol. 346, pp. 184–191, 2019.

[263] Y, Chang, H. Fang, Y. Zhang, "A new hybrid method for the prediction of the remaining useful life of a lithium-ion battery," *Applied Energy* , vol. 206, pp. 1564-1578, 2017.

[264] Y. Cheng, N. Zerhouni, C. Lu, "A hybrid remaining useful life prognostic method for proton exchange membrane fuel cell ," *International Journal of Hydrogen Energy*, vol. 43, Issue 27, pp. 12314-12327, 2018.

[265] A. Garga et al., "Hybrid reasoning for prognostic learning in CBM systems," *IEEE Aerospace Conference Proceedings (Cat. No.01TH8542)*, vol. 6, pp. 2957-2969, 2001.

[266] K. Medjaher and N. Zerhouni, "Hybrid prognostic method applied to mechatronic systems," *The International Journal of Advanced Manufacturing Technology*, vol. 69, pp. 823–834, 2013.

[267] F. Lasheras et al., "Hybrid PCA-CART-MARS-Based Prognostic Approach of the Remaining Useful Life for Aircraft Engines," *Sensors*, vol. 15, pp. 7062-7083, 2015; doi:10.3390/s150307062.

[268] H. Skima, K. Medjaher, C. Varnier, E. Dedu, and J. Bourgeois, "A hybrid prognostics approach forMEMS: From real measurements to remaining useful life estimation," *Microelectronics Reliability*, vol. 65, pp. 79-88, 2016.

[269] H. Sun, D. Cao, Z. Zhao and X. Kang, "A Hybrid Approach to Cutting Tool Remaining Useful Life Prediction Based on the Wiener Process," *in IEEE Transactions on Reliability*, vol. 67, no. 3, pp. 1294-1303, 2018.

[270] B. Wang, Y. Lei, N. Li and N. Li, "A Hybrid Prognostics Approach for Estimating Remaining Useful Life of Rolling Element Bearings," *in IEEE Transactions on Reliability*, vol. 69, no. 1, pp. 401-412, 2020.

[271] K. Yan, W. Li, Z. Ji, M. Qi and Y. Du, "A Hybrid LSTM Neural Network for Energy Consumption Forecasting of Individual Households," *in IEEE Access*, vol. 7, pp. 157633-157642, 2019.

[272] K. Yan, X.Wang, Y. Du, N. Jin, H. Huang, and H. Zhou, "Multi-step short term power consumption forecasting with a hybrid deep learning strategy," *Energies*, vol. 11, no. 11, pp. 3089, 2018.

[273] O. Eker, F. Camci, and I. Jennions "A New Hybrid Prognostic Methodology," *IJPHM*, vol. 10, no. 9, 2019.

[274] V. Atamuradov, K. Medjaher, and N. Zerhouni, "Prognostics and Health Management for Maintenance Practitioners-Review, Implementation and Tools Evaluation," *IJPHM, Special Issue on Railways & Mass*, vol. 8, no. 60, 2017.

[275] Y. Ran, X. Zhou, P. Lin, Y. Wen, and R. Dengi, "A Survey of Predictive Maintenance: Systems, Purposes and Approaches," *arXiv:1912.07383v1 [eess.SP]*, 2019.

[276] T. Xia, Y. Dong, L. Xiao, S. Du, E. Pan, and L. Xi, "Recent advances in prognostics and health management for advanced manufacturing paradigms," *Reliability Engineering and System Safety*, vol. 178, pp. 255–268, 2018.

[277] 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.

[278] Y. Ran, X. Zhou, P. Lin, Y. Wen, and R. Deng, "A Survey of Predictive Maintenance: Systems, Purposes and Approaches," *arXiv:1912.07383v1 [eess.SP]*, 2020.

[279] Y. Song, G. Shi, L. Chen, X. Huang, and T. Xia, "Remaining Useful Life Prediction of Turbofan Engine Using Hybrid Model Based on Autoencoder and Bidirectional Long Short-Term Memory," *Journal of Shanghai Jiaotong University (Science)*, vol. 23, pp. 85–94, 2018.

[280] K. Akkad and D. He, "A Hybrid Deep Learning Based Approach for Remaining Useful Life Estimation," *IEEE International Conference on Prognostics and Health Management (ICPHM)*, pp. 1-6, 2019.

[281] M. Amin, S. Imtiaz, and F. Khan, "Process system fault detection and diagnosis using a hybrid technique," *Chemical Engineering Science*, vol. 189, pp. 191–211, 2018.

[282] X. Li, L. Zhang, Z. Wang, and P. Dong, "Remaining useful life prediction for lithium-ion batteries based on a hybrid model combining the long short-term memory and Elman neural networks," *Journal of Energy Storage*, vol. 21, pp. 510–518, 2019.

[283] L. Liao and F. Köttig, "Review of Hybrid Prognostics Approaches for Remaining Useful Life Prediction of Engineered Systems, and an Application to Battery Life Prediction," *in IEEE Transactions on Reliability*, vol. 63, no. 1, pp. 191-207, 2014.

[284] X. Li, J. Li, Y. Qu, and D. He, "Gear Pitting Fault Diagnosis Using Integrated CNN and GRU Network with Both Vibration and Acoustic Emission Signals," *Applied Sciences*, vol. 9, issue 4, pp. 768, 2019.

[285] J. Celaya, A. Saxena, S. Saha, and K. Goebel, "Prognostics of power mosfets under thermal stress accelerated aging using data-driven and model-based methodologies," *in Proc. Int. Conf. Prognost. Health Management*, 2011.

[286] Y. Rosunally, S. Stoyanov, C. Bailey, P. Mason, S. Campbell, and G. Monger, "Prognostics framework for remaining life prediction of cutty sark iron structures," *in Proc. Annu. Conf. Prognost. Health Management Soc.*, 2009.

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

[288] B. Saha, K. Goebel, S. Poll and J. Christophersen, "Prognostics Methods for Battery Health Monitoring Using a Bayesian Framework," *in IEEE Transactions on Instrumentation and Measurement*, vol. 58, no. 2, pp. 291-296, 2009.

[289] D. Zhang, L. Tian, M. Hong, F. Han, Y. Ren and Y. Chen, "Combining Convolution Neural Network and Bidirectional Gated Recurrent Unit for Sentence Semantic Classification," *in IEEE Access*, vol. 6, pp. 73750-73759, 2018.

[290] Y. Chu, C. Huang, X. Xie, B. Tan, S. Kamal, and X. Xiong, "Multilayer Hybrid Deep-Learning Method for WasteClassification and Recycling," *Computational Intelligence and Neuroscience*, 2018. ID:5060857; https://doi.org/10.1155/2018/5060857.

[291] L. Li et al., "Hybrid Deep Neural Network–Hidden Markov Model (DNN-HMM) Based Speech Emotion Recognition," *Humaine Association Conference on Affective Computing and Intelligent Interaction*, pp. 312-317, 2013.

[292] E. Ijjina and C. Mohan, "Hybrid deep neural network model for human action recognition," *Applied Soft Computing*, vol. 46, pp. 936-952, 2016.

[293] Z. Wu et al., "Modeling Spatial-Temporal Clues in a Hybrid Deep Learning Framework for Video Classification," *ACM international conference on Multimedia*, pp. 461–470, 2015; https://doi.org/10.1145/2733373.2806222.

[294] M. Duan, K. Li, C. Yang, K. Li, "A hybrid deep learning CNN–ELM for age and gender classification," *Neurocomputing*, vol. 275, pp. 448-461, 2018.

[295] Y. Sun, X. Wang and X. Tang, "Hybrid Deep Learning for Face Verification," *in IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 10, pp. 1997-2009, 2016.

[296] A. Zakariae, and Z. Hinchi, "Rolling Element Bearing Remaining Useful Life Estimation based on a Convolutional Long-short-term memory Network," *Procedia Computer Science*, vol. 127, pp. 123-132, 2018.

[297] 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, pp. 273, 2017.

[298] W.Yu, Y. Kim, and C. Mechefske, "Remaining useful life estimation using a bidirectional recurrent neural network based autoencoder scheme," *Mechanical Systems and Signal Processing*,vol. 129, pp. 764-780, 2019.

[299] L. Jayasinghe, T. Samarasinghe, C. Yuen, J. Low, and S. Ge, "Temporal Convolutional Memory Networks for Remaining Useful Life Estimation of Industrial Machinery," *arXiv:1810.05644v2 [cs.LG]*, 2018.

[300] 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.

[301] H. Pan, X. He, S. Tang, and F. Meng, "An improved bearing fault diagnosis method using one-dimensional cnn and lstm.," *Journal of Mechanical Engineering*, vol. 64, 2018.

[302] H. Liu, Z. Liu, W. Jia and X. Lin, "Remaining Useful Life Prediction Using A Novel Feature-attention Based End-to-end Approach," *in IEEE Transactions on Industrial Informatics*, 2020; DOI: 10.1109/TII.2020.2983760 .

[303] M. Rao, Q. Li, D. Wei, M. Zuo, "A deep bi-directional long short-term memory model for automatic rotating speed extraction from raw vibration signals," *Measurement*, vol. 158, 2020, ID:107719.

[304] J. Li, X. Li, and D. He, "A Directed Acyclic Graph Network Combined With CNN and LSTM for Remaining Useful Life Prediction," *IEEE Access*, vol. 7, pp. 75464-75475, 2019.

[305] A. Al-Dulaimi, A. Mohammadi, and A. Asif "Generalized Degradation Model for Health Management of Mission Critical Systems," *IISE Annual Conference* pp.1496-1501, 2017.

[306] A. Al-Dulaimi, A. Mohammadi, and A. Asif "Modeling Degradation Paths based on Interactive Multiple Model Particle Filters for Prognostic Health Management," *IISE Annual Conference*, QCR-S8: Reliability Analysis - II, 2018.

[307] Z. Xu, Y. Ji and D. Zhou, "Real-time Reliability Prediction for a Dynamic System Based on the Hidden Degradation Process Identification," *IEEE Transactions Reliability,* vol. 57, no. 2, pp. 230-242, 2008.

[308] C. Xiongzi, Y. Jinsong, T. Diyin, W. Yingxun, "A Novel PF-LSSVR-based Framework for Failure Prognosis of Nonlinear Systems with Time-varying Parameters," *Chinese Journal of Aeronautics,* vol. 57, pp. 715-724, 2008.

[309] A. Saxena, and K. Goebel, "C-MAPSS Data Set," *NASA Ames Prognostics Data Repository*, 2008.

[310] A. Saxena and K. Goebel, "PHM08 Challenge Data Set," *NASA Ames Research Center, Moffett Field, CA, USA. [Online]. Available: http://ti.arc.nasa.gov/project/prognostic-data-repository*, 2008.

[311] A. Saxena, K. Goebel, D. Simon, and N. Eklund, "Damage Propagation Modeling for Aircraft Engine Run-to-failure Simulation," *Proceedings of the IEEE International Conference on Prognostics and Health Management*, pp. 1-9, 2008.

[312] P. Lim, C. Goh, K. Tan, and P. Dutta, "Multimodal Degradation Prognostics Based on Switching Kalman Filter Ensemble," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 1, pp. 136-148, 2017.

[313] C. Zhang, P. Lim, A. Qin, and K. Tan, "Multiobjective Deep Belief Networks Ensemble for Remaining Useful Life Estimation in Prognostics," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 10, pp. 2306-2318, 2017.

[314] L. Peel, "Data Driven Prognostics using a Kalman Filter Ensemble of Neural Network Models," *Proceedings of the International Conference on Prognostics and Health Management*, pp. 1-6, 2008.

[315] E. Ramasso, "Investigating Computational Geometry for Failure Prognostics," *International Journal of Prognostics and Health Management*, vol. 5, no. 1, ppt. 1-18, 2014.

[316] P. Lim, C. Goh, and K. Tan, "A time window neural network based framework for remaining useful life estimation," *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, pp. 1746-53, 2016.

[317] C. Louen, S. Ding, and C. Kandler, "A New Framework for Remaining Useful Life Estimation using Support Vector Machine Classifier," *Proceedings of the Conference on Control and Fault-Tolerant Systems (SysTol)*, pp. 228-233, 2013.

[318] C. Hsu, and J. Jiang, "Remaining Useful Life Estimation Using Long Short-Term Memory Deep Learning," *Proceedings of the IEEE International Conference on Applied System Innovation*, pp. 58-61, 2018.

[319] F. Heimes, "Recurrent Neural Networks for Remaining Useful Life Estimation," *Proceedings of the International Conference on Prognostics and Health Management*, pp. 1-6, 2008.

[320] C. Huang, H. Huang and Y. Li, "A Bidirectional LSTM Prognostics Method Under Multiple Operational Conditions," *IEEE Transactions on Industrial Electronics*, vol. 66, no. 11, pp. 8792-8802, 2019.

[321] A. Al-Dulaimi, S. Zabihia, A. Asif, and A. Mohammadi, "A multimodal and hybrid deep neural network model for Remaining Useful Life estimation," *Computers in Industry*, vol. 108, pp. 186-196, 2019.

[322] A. Al-Dulaimi, S. Zabihia, A. Asif, and A. Mohammadi, "Hybrid Deep Neural Network Model for Remaining Useful Life Estimation," *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP),* pp. 3872-3876, 2019.

[323] A. Al-Dulaimi, A. Asif, and A. Mohammadi, "Noisy parallel hybrid model of NBGRU and NCNN architectures for remaining useful life estimation," *Quality Engineering, Special Issue on Reliability Engineering*, vol. 32, issue 3, pp. 371-387 , 2020.

[324] A. Al-Dulaimi, A. Asif, and A. Mohammadi, "Remaining Useful Life Estimation via Hybrid of NBGRU and CNN," *Proceedings of the 2020 IISE Annual Conference*, 2020.

[325] Y. Zhang, R. Xiong, H. He and M. 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.

[326] 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.

[327] Y. Gal and Z. Ghahramani, "A Theoretically Grounded Application of Dropout in Recurrent Neural Networks," *arXiv:1512.05287v5 [stat.ML]*, 2015.

[328] L. Prechelti, "Early Stopping — But When," *Neural Networks: Tricks of the Trade/Springer*, vol. 7700, pp. 53-67, 2012, ISBN: 978-3-642-35288-1

[329] I. Valchanov, "Machine Learning: An Overview," " https://www.datascience.com/blog/machine-learning-overview," 2018.

[330] J. Brownlee, "A Gentle Introduction to Mini-Batch Gradient Descent and How to Configure Batch Size," https://machinelearningmastery.com/gentle-introduction-mini-batch-gradient-descent-configure-batch-size/, 2017.

[331] D. Kingma, and J. Ba, "Adam: A Method for Stochastic Optimization," *Proceedings of the International Conference on Learning Representations (ICLR)*, arXiv:1412.6980, 2014.

[332] A. Palazuelos, E. Droguett, and R. Pascual, "A novel deep capsule neural network for remaining useful life estimation," *The Institution of Mechanical Engineers: Risk and Reliability*, pp. 1-17, 2019.

[333] H. Yang, F. Zhao, G. Jiang, Z. Sun, and X. Mei, "A Novel Deep Learning Approach for Machinery Prognostics Based on Time Windows," *Applied Sciences*, vol. 9, issue 22, 2019. 4813.

[334] L. Wen, Y. Dong, and L. Gao, "A new ensemble residual convolutional neural network for remaining useful life estimation," *Mathematical Biosciences and Engineering*, vol. 16, no. 2, pp. 862-880, 2019.

[335] A. Ellefsen, E. Bjorlykhaug, V. Aesqy, S. Ushakov, and H. Zhang, "Remaining useful life predictions for turbofan engine degradation using semi-supervised deep architecture," *Reliability Engineering and System Safety*, vol. 183, pp. 240-251, 2019.

[336] Y. Liao, L. Zhang and C. Liu, "Uncertainty Prediction of Remaining Useful Life Using Long Short-Term Memory Network Based on Bootstrap Method," *Proceedings of the IEEE International Conference on Prognostics and Health Management (ICPHM)*, pp. 1-8, 2018.

[337] R. Reed and R. MarksII, "Neural smithing: supervised learning in feedforward artificial neural networks," *Mit Press*, 1999, ISBN-13: 978-0262527019.

[338] M. Kuhn, K. Johnson "Applied predictive modeling," *Springer*, vol. 26, 2013.

[339] J. Quinlan, "C4. 5: programs for machine learning," *Elsevier*, 2014, ISBN: 9780080500584.

[340] R. Kamimura and S. Nakanishi, "Improving generalization performance by information minimization," *IEICE Transactions on Information and Systems*, vol. E78-D, no. 2, pp. 163–173,1995.

[341] L. Franco, D. Elizondo, and J. Jerez, "Constructive neural networks," *Springer*, vol. 258, 2009, .

[342] Y. Jiang, R. Zur, L. Pesce, and K. Drukker, "A study of the effect of noise injection on the training of artificial neural networks," *International Joint Conference on Neural Networks*, pp. 1428-1432, 2009, doi: 10.1109/IJCNN.2009.5178981.

[343] K. Audhkhasi, O. Osoba, and B. Kosko, "Noise-enhanced convolutional neural networks," *Neural Networks*,vol. 78, pp. 15-23,2016, doi:10.1016/j.neunet.2015.09.014.

[344] I. Goodfellow, Y. Bengio, and A. Courville, " Deep learning (Adaptive Computation and Machine Learning series)," *MIT press*, 2016.

[345] A. Neelakantan, L. Vilnis, Q. Le, I. Sutskever, L. Kaiser, K. Kurach, and J. Martens, "Adding gradient noise improves learning for very deep networks," *arXiv preprint arXiv:1511.06807*, 2015.

[346] A. Rakin, Z. He, and D. Fan, " Parametric Noise Injection: Trainable Randomness to Improve Deep Neural Network Robustness Against Adversarial Attack," *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 588-597, 2019.

[347] S. Yin, C. Liu, Z. Zhang, Y. Lin, D. Wang, J. Tejedor, T. Zheng and Y. Li, "Noisy training for deep neural networks in speech recognition," *EURASIP Journal on Audio, Speech, and Music Processing*, no. 2, 2015, https://doi.org/10.1186/s13636-014-0047-0

[348] A. Graves, " Practical Variational Inference for Neural Networks," *Advances in Neural Information Processing Systems 24 (NIPS)*, pp. 2348—2356. 2011.

[349] C. Bishop, " Training with Noise is Equivalent to Tikhonov Regularization," *Neural Computation*, vol. 7, no. 1, pp. 108-116, 1995.

[350] J. Sietsma and R. Dow, "Creating artificial neural networks that generalize," *Neural Networks*, vol. 4, no. 1, pp. 67-79, 1991.

[351] R. Burton and G. Mpitsos, "Event-dependent control of noise enhances learning in neural networks," *Neural Networks*, vol. 5, no. 3, pp. 627-637, 1992.

[352] G. An, "The Effects of Adding Noise During Backpropagation Training on a Generalization Performance," *Neural Computation*, vol. 8, no. 3, pp. 643-674, 1996, doi: 10.1162/neco.1996.8.3.643.

[353] J. Brownlee, "Better Deep Learning Train Faster, Reduce Overfitting, and Make Better Predictions," *Machine learning mastery*, 2018

[354] A. Murray and P. Edwards, "Enhanced MLP performance and fault tolerance resulting from synaptic weight noise during training," *IEEE Transactions on Neural Networks*, vol. 5, no. 5, pp. 792-802,1994, doi: 10.1109/72.317730.

[355] K. Jim, C. Giles and B. Horne, "An analysis of noise in recurrent neural networks: convergence and generalization," *IEEE Transactions on Neural Networks*, vol. 7, no. 6, pp. 1424-1438, 1996, doi: 10.1109/72.548170.

[356] J. Wang, Q. Chang, Y. Liu, and N. Pal, "Weight Noise Injection-Based MLPs With Group Lasso Penalty: Asymptotic Convergence and Application to Node Pruning," *IEEE Transactions on Cybernetics*, vol. 49, no. 12, pp. 4346-4364, 2019, doi: 10.1109/TCYB.2018.2864142.

[357] R. Li, K. Shuang, M. Gu, and S. Su, "Adaptive Noise Injection: A Structure-Expanding Regularization for RNN," *arXiv:1907.10885v1 [cs.CL]*, 2019.

[358] S. Ognawala and J. Bayer, "Regularizing Recurrent Networks - On Injected Noise and Norm-based Methods," *arXiv:1410.5684v1 [stat.ML]*, 2014.

[359] T. Moon, H. Choi, H. Lee and I. Song, "RNNDROP: A novel dropout for RNNS in ASR," *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pp. 65-70, 2015, doi: 10.1109/ASRU.2015.7404775.

[360] K. Audhkhasi, O. Osoba, and B. Kosko, "Noise-enhanced convolutional neural networks," *Neural Networks*, vol. 78, pp. 15-23, 2016

[361] Z. You, J. Ye, K. Li, Z. Xu and P. Wang, "Adversarial Noise Layer: Regularize Neural Network by Adding Noise," *IEEE International Conference on Image Processing (ICIP)*, pp. 909-913, 2019, doi: 10.1109/ICIP.2019.8803055.

[362] L. Toth, G. Kovacs, and D. Compernolle, "A perceptually inspired data augmentation method for noise robust cnn acoustic models," *International Conference on Speech and Computer*, pp. 697–706, 2018.

[363] R. Chen, S. Chen, M. He, D. He and B. Tang, "Rolling bearing fault severity identification using deep sparse auto-encoder network with noise added sample expansion," *The Institution of Mechanical Engineers Part O Journal of Risk and Reliability*, vol. 231, no. 6, pp. 1-14, 2017.

[364] D. MacKay, "Comparison of approximate methods for handling hyperparameters," *Neural computation*, vol. 11, no. 5, pp. 1035-1068, 1999, MIT Press.

[365] V. Strijov and G. Weber, "Nonlinear regression model generation using hyperparameter optimization," *Computers & Mathematics with Applications*, vol. 60, Issue 4, pp. 1035-981-988, 2010.

[366] P. Liashchynskyi and P. Liashchynskyi, "Grid Search, Random Search, Genetic Algorithm: A Big Comparison for NAS," *arXiv:1912.06059v1 [cs.LG]*, 2019.

[367] K. Judd "Numerical methods in economics," *The MIT Press*, 1998, ISBN 0-262-10071-1.

[368] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kegl, "Algorithms for Hyper-Parameter Optimization," *Proceedings of the Annual Conference on Neural Information Processing Systems*, hal-00642998, 2011.

[369] P. Leeflang, D. Wittink, M. Wedel, P. Naert, "Building models for marketing decisions," *Springer Science & Business Media*, vol. 9, 2013.

[370] R. Bhagwat, M. Abdolahnejad, M. Moocarme, "Applied Deep Learning with Keras: Solve complex real-life problems with the simplicity of Keras," *Packt Publishing*, 2019.

[371] P. Gonzaleza, V. Kobera, and V. Ramirezb, "Adaptive composite filters for pattern recognition in nonoverlapping scenes using noisy training images," *Pattern Recognition Letters*, vol. 41, ppt. 83-92, 2014.

[372] E. Ramasso, and A. Saxena, "Review and Analysis of Algorithmic Approaches Developed for Prognostics on CMAPSS Dataset," *Proceedings of the Annual Conference of the Prognostics and Health Management Society*, hal-01145003, 2014.

[373] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," *arXiv:1502.03167v3 [cs.LG]*, 2015.

[374] S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry, "How Does Batch Normalization Help Optimization," *arXiv:1805.11604v5 [stat.ML]*, 2019.

[375] C. Garbin, X. Zhu, and O. Marques, "Dropout vs. batch normalization: an empirical study of their impact to deep learning," *Multimed Tools Appl*, vol. 79, pp. 12777–12815, 2020.

[376] C. Laurent, G. Pereyra, P. Brakel, Y. Zhang, and Y. Bengio, "Batch normalized recurrent neural networks," *in Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on. IEEE*, pp. 2657–2661, 2016.

[377] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent neural network regularization," *arXiv preprint arXiv:1409.2329*, 2014.

[378] T. Cooijmans, N. Ballas, C. Laurent, C. Gulcehre, and A. Courville, "Recurrent batch normalization," *arXiv preprintarXiv:1603.09025*, 2016.

[379] M. Lin, Q. Chen, S. Yan, "Network In Network," *arXiv:1312.4400v3*, 2014.

[380] A. Al-Dulaimi, A. Asif, and A. Mohammadi, "The Noisy Multipath Parallel Hybrid Model for Remaining Useful Life Estimation (NMPM)," *12th Annual Conference of the Prognostics and Health Management Society*, 2020.

[381] A. Al-Dulaimi, A. Asif, and A. Mohammadi, "Multipath Parallel Hybrid Deep Neural Networks Framework for Remaining Useful Life Estimation," *2020 IEEE International Conference on Prognostics and Health Management (ICPHM), USA* pp. 1-7, 2020, doi: 10.1109/ICPHM49022.2020.9187040.

[382] A. Al-Dulaimi, A. Asif, and A. Mohammadi, "Noisy Parallel, Hybrid Model for Remaining Useful Life Estimation (NPHM)," *Proceedings of the 2020 IISE Annual Conference*, 2020.