

T.R.
GEBZE TECHNICAL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

JOINT AUTOENCODER-BASED PREDICTIVE MAINTENANCE
METHODS FOR INDUSTRIAL MACHINERY

KÜRŞAT İNCE
A THESIS SUBMITTED FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
DEPARTMENT OF COMPUTER ENGINEERING

GEBZE

2023

T.R.

GEBZE TECHNICAL UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

**JOINT AUTOENCODER-BASED
PREDICTIVE MAINTENANCE METHODS
FOR INDUSTRIAL MACHINERY**

KÜRŞAT İNCE

**A THESIS SUBMITTED FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY**

DEPARTMENT OF COMPUTER ENGINEERING

THESIS SUPERVISOR

ASST. PROF. DR. YAKUP GENÇ

GEBZE

2023

T.C.

GEBZE TEKNİK ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

**ENDÜSTRİYEL MAKİNELER İÇİN
BİRLEŞİK OTOKODLAYICI TABANLI
KESTİRİMCİ BAKIM YÖNTEMLERİ**

KÜRŞAT İNCE

DOKTORA TEZİ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

DANIŞMANI

DR. ÖĞR. ÜYESİ YAKUP GENÇ

GEBZE

2023



DOKTORA JÜRİ ONAY FORMU

GTÜ Fen Bilimleri Enstitüsü Yönetim Kurulu'nun 19/07/2023 tarih ve 2023/40 sayılı kararıyla oluşturulan jüri tarafından 02/08/2023 tarihinde tez savunma sınavı yapılan Kürşat İNCE'nin tez çalışması Bilgisayar Mühendisliği Anabilim Dalında DOKTORA tezi olarak kabul edilmiştir.

JÜRİ

ÜYE

(TEZ DANIŞMANI) : Dr. Öğr. Üyesi Yakup Genç

ÜYE : Prof. Dr. F. Erdoğan Sevilgen

ÜYE : Prof. Dr. Berrin Yanıkoğlu

ÜYE : Doç. Dr. Mehmet Göktürk

ÜYE : Prof. Dr. H. Rahmi Topçu

Onay

Gebze Teknik Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun

...../...../..... tarih ve/..... sayılı kararı.

İMZA/MÜHÜR

SUMMARY

This thesis explores predictive maintenance in industrial systems using deep learning, focusing on fault classification and remaining useful life (RUL) prediction. The objectives include creating a deep learning framework that includes fault classification, RUL prediction, and integrates operating conditions for accurate real-world representation. Addressing limited labeled data, the thesis designs a fault detection algorithm for precise labeling during training. The methodology employs an autoencoder-based architecture, showing remarkable efficacy in fault detection and RUL prediction, leading to significant advancements in predictive maintenance. Another notable contribution is the innovative fault detection-based RUL labeling, overcoming data labeling challenges and enhancing predictive maintenance in industrial machinery. Deep learning often requires substantial labeled data, which can be scarce in industrial applications. The proposed fault detection algorithm plays a pivotal role in precise data labeling, enabling effective deployment of deep learning models in real-world industrial settings. Overall, this research offers an effective deep learning framework for fault detection and RUL prediction, contributing substantially to predictive maintenance literature. Demonstrated results underscore the framework's potential to predict RUL accurately across diverse systems. A new dataset supports future research in health index generation, fault identification, remaining useful life prediction, and failure prognosis in industrial systems.

Keywords: Predictive maintenance, Deep learning, Remaining useful life, Fault classification, Remaining useful life prediction, Fault detection algorithm, Joint autoencoder-based prediction.

ÖZET

Bu tezde, endüstriyel sistemlerde hata sınıflandırmasını ve kalan faydalı ömür (KFÖ) kestirimini içeren kestirimci bakım yöntemleri, derin öğrenme modellerine odaklanarak araştırılmaktadır. Tezin amacı hata sınıflandırmasını, KFÖ tahminini içeren ve çalışma koşullarını entegre eden bir derin öğrenme çerçevesi oluşturmayı içerir. Sınırlı etiketli verileri ele alan tez, eğitim öncesinde kullanılmak üzere veri etiketleme için bir hata tespit algoritması sunar. Metodoloji, otokodlayıcı tabanlı bir mimari kullanır ve arıza tespiti ve KFÖ kestirimine dikkate değer bir etkinlik gösterir ve kestirimci bakımda önemli ilerlemelere yol açar. Dikkate değer bir katkı, endüstriyel makinelerde veri etiketleme zorluklarının üstesinden gelen ve kestirimci bakımı geliştiren yenilikçi hata algılama tabanlı KFÖ etiketlemesidir. Derin öğrenme genellikle, endüstriyel uygulamalarda kıt olabilen, önemli miktarda etiketlenmiş veri gerektirir. Önerilen hata tespit algoritması, hassas veri etiketlemede önemli bir rol oynayarak, gerçek dünyadaki endüstriyel ortamlarda derin öğrenme modellerinin etkili bir şekilde uygulanmasını sağlar. Sonuç olarak, bu araştırma, arıza tespiti ve KFÖ kestirimi için etkili bir derin öğrenme çerçevesi sunarak kestirimci bakıma büyük ölçüde katkıda bulunur. Elde edilen sonuçlar, çerçevenin çeşitli sistemlerde KFÖ'ü doğru bir şekilde tahmin etme potansiyeli gösterir. Hazırlanan yeni veri seti, endüstriyel sistemlerde sağlık indeksi oluşturma, arıza tanımlama, kalan faydalı ömür kestirimi ve arıza tespiti konularda gelecekteki araştırmaları desteklemektedir.

Anahtar Kelimeler: Kestirimci bakım, Derin öğrenme, Kalan faydalı ömür, Hata sınıflandırma, Kalan faydalı ömür kestirimi, Hata tespit algoritması, Birleşik otokodlayıcı temelli kestirim.

ACKNOWLEDGMENTS

I extend my sincere gratitude to my esteemed supervisor, Dr. Yakup Genc, whose unwavering guidance, expertise, and encouragement have been invaluable throughout this remarkable journey of completing my PhD thesis. Gebze Technical University provided the essential academic foundation for my research, and I am deeply appreciative of the conducive environment it offered. I extend special recognition to the esteemed members of the Steering Committee, Prof. Dr. Berrin Yanıkoğlu and Prof. Dr. Fatih Erdoğan Sevilgen, for their valuable insights and guidance, which have played a pivotal role in shaping the direction and quality of this thesis. The collaboration with HAVELSAN Naval Combat Management Technologies Center has been a defining aspect of my research experience. Their support and resources greatly enriched the scope and depth of this study, contributing to its overall excellence. I am indebted to Dr. Orhan Aksoy, Muhammet Altun, and Göktürk Aşıcı for their unwavering general support, advice, and assistance throughout this journey. I am grateful to Assoc. Prof. Gazi Koçak, a respected member of ITU Maritime Faculty, for his insightful contributions that significantly enhanced the quality of this research. I also extend my appreciation to Kerem Palabıyık of Four.One Technology and Dr. Atilla Yardımcı for their valuable collaboration and input. To my cherished family, friends, and colleagues at HAVELSAN, your unwavering support has been my constant motivation, and for that, I am truly thankful. Reflecting on my personal journey, I've come to understand the importance of self-awareness, the necessity to persevere in the face of challenges, and the significance of rewarding oneself for achievements along the way. Each of you has left an indelible mark on this thesis, and I am honored to acknowledge your contributions with the deepest gratitude.

TABLE OF CONTENTS

	<u>Page</u>
SUMMARY	v
ÖZET	vi
ACKNOWLEDGMENTS	vii
TABLE OF CONTENTS	viii
LIST OF ABBREVIATIONS and ACRONYMS	xi
LIST OF FIGURES	xii
LIST OF TABLES	xiv
1. INTRODUCTION	1
1.1. Motivations	1
1.2. Problem Definition	3
1.3. Contributions	4
1.4. Outline	4
2. LITERATURE REVIEW	6
2.1. Introduction	6
2.2. Prognostics and Health Management	8
2.2.1. Approaches	10
2.2.2. General Framework for Data-Driven Prognostics	11
2.2.3. Application Areas and Research Problems	13
2.3. Methods for Data-Driven Prognostics	16
2.3.1. Data Preprocessing	17
2.3.2. Model Development	19
Machine Learning	20
Deep Learning	21
Hyper-parameter Optimization	23
Evaluation Metrics	25
2.4. Public Datasets for Prognostics Research	27
2.5. Challenges in Prognostics Research	31
2.6. Summary	33

3. PRELIMINARY STUDIES	38
3.1. Automobile Brake System Leakage Detection	38
3.1.1. Introduction and Problem Definition	38
3.1.2. Methodology	40
3.1.3. Experiments	42
3.1.4. Results and Discussion	43
3.2. Experimental Filtration System Remaining Useful Life Prediction	46
3.2.1. Introduction and Problem Statement	46
3.2.2. Methodology	47
3.2.3. Experiments	50
3.2.4. Results and Discussion	51
3.3. Robotic Test Bench Fault Detection and Classification	54
3.3.1. Introduction and Problem Definition	55
3.3.2. Methodology	57
3.3.3. Experiments	59
3.3.4. Results and Discussion	61
3.4. Summary	64
4. TURBOFAN ENGINE REMAINING USEFUL LIFE PREDICTION	65
4.1. Introduction and Problem Definition	65
4.2. Proposed Framework	66
4.2.1. Data Labeling	67
4.2.2. Data Preprocessing	71
4.2.3. Training and Evaluation	72
4.3. Joint Autoencoder-Regressor Architecture: Base Model	73
4.3.1. Implementation	74
4.3.2. Results	76
4.3.3. Discussions	81
4.4. InceptionTime-Based Model	82
4.4.1. Implementation	83
4.4.2. Results and Discussion	85
4.5. Joint Autoencoder-Regressor Architecture: Possible Improvements	87
4.6. Summary	90

5. SHIP ENGINE ROOM FAULT CLASSIFICATION	91
5.1. Introduction	91
5.2. Kongsberg Engine Room Simulator Dataset	93
5.3. Problem Definition	94
5.4. Methods and Techniques	95
5.5. Experiments and Results	97
5.5.1. Fault Classification	97
5.5.2. Emission Estimation	98
5.5.3. Results and Discussion	99
5.6. Summary	103
6. CONCLUSIONS AND FUTURE WORK	106
6.1. Conclusions	106
6.2. Future Work	107
REFERENCES	107
BIOGRAPHY	121
APPENDICES	122
A. Publications within the Scope of the Thesis Study	122
B. Systematic Review Queries	123

LIST OF ABBREVIATIONS and ACRONYMS

<u>Abbreviations</u>	:	<u>Explanations</u>
----------------------	---	---------------------

<u>and Acronyms</u>

ACC	:	Accuracy
AE	:	Autoencoder
C-MAPSS	:	Commercial Modular Aero-Propulsion System Simulation
CNC	:	Computer Numeric Control
CNN	:	Convolutional Neural Network
DBN	:	Deep Belief Networks
GMM	:	Gaussian Mixture Model
GP	:	Gaussian Process
GRU	:	Gated Recurrent Unit
LDA	:	Linear Discriminant Analysis
LSTM	:	Long-short Term Memory Network
MAE	:	Mean Absolute Error
MCC	:	Matthews Correlation Coefficient
MSE	:	Mean Squared Error
OC	:	Operating Condition
PCA	:	Principle Component Analysis
PHM	:	Prognostics and Health Management
RBM	:	Restricted Boltzmann Machine
RF	:	Random Forest
RMSE	:	Root Mean Squared Error
RNN	:	Recurrent Neural Network
RUL	:	Remaining Useful Life

LIST OF FIGURES

<u>Figure No:</u>		<u>Page</u>
2.1:	General framework for data-driven prognostics.	11
2.2:	Example deterioration of a system.	13
2.3:	Confusion matrix for classification tasks.	25
3.1:	The Filling Process: Through the filling hose, (1) air is vacuumed off, (2) then fluid is pumped in.	39
3.2:	Sample chassis readings from the filling dataset.	41
3.3:	The experimental rig for PHME20 Data Challenge.	47
3.4:	Data processing workflow for clog detection.	47
3.5:	Fault estimation for <i>PwL_Fault</i> RUL assignment method.	49
3.6:	Linear, PwL, and <i>PwL_Fault</i> RUL assignment strategies.	50
3.7:	The Fuse Test Bench for PHME21 Data Challenge.	56
3.8:	Confusion matrices per fold for fault classification.	60
4.1:	RUL prediction framework for C-MAPSS Dataset.	67
4.2:	Visualization of training and testing data construction in Algorithm 1.	71
4.3:	Visualization of Algorithm 1 score values.	71
4.4:	The proposed joint autoencoder-regression architecture for estimating RUL with time series data.	73
4.5:	Architecture and hyper-parameters of the optimized model.	76
4.6:	RUL prediction performance of algorithms on C-MAPS FD001 test data.	79
4.7:	Plots showing typical results of RUL prediction of algorithms.	79
4.8:	Inception Module and Inception Network.	83
4.9:	JAER-IT architecture with Gaussian noise layer.	87
5.1:	General data processing workflow for fault classification.	96
5.2:	Proposed joint autoencoder classifier architecture.	97
5.3:	Confusion matrices for varying amount of training data.	100
5.4:	Contribution of features to the fault classification task.	102

B.1:	Keyword frequencies of publications between 2018 – 2023.	124
B.2:	Word cloud of publications between 2018 – 2023.	124



LIST OF TABLES

<u>Table No:</u>	<u>Page</u>
2.1:	Recents studies on PHM application domains. 17
2.2:	Recents studies on public datasets. 28
2.3:	Summary of the Literature Review. 35
3.1:	Features in the Filling Dataset. 40
3.2:	Hyper-parameters for RF and GB models. 42
3.3:	Random forest and gradient boosting results for leakage detection. 44
3.4:	CNN architecture results for leakage detection. 45
3.5:	Model parameters for PHME20 models. 51
3.6:	MAE scores using 25% of the training data. 52
3.7:	MAE scores using 50% of the training data. 52
3.8:	MAE scores using 75% of the training data. 53
3.9:	MAE scores using full training data. 53
3.10:	Statistical features derived from test-bench sensors. 56
3.11:	GA search parameters for hyper-parameter optimization. 61
3.12:	Results of classification models for different window sizes. 62
3.13:	Model results after GA optimizations. 63
4.1:	C-MAPSS Dataset information. 66
4.2:	The JAER-CNN model hyper-parameters for GA optimization. 75
4.3:	JAER-CNN evaluation results on C-MAPSS FD001 data. 77
4.4:	JAER-CNN evaluation results on C-MAPSS FD003 data. 80
4.5:	Comparison of JAER-CNN model with some recent deep learning work on C-MAPSS. 81
4.6:	The JAER-IT model hyper-parameters for GA optimization. 84
4.7:	JAER-IT evaluation results on C-MAPSS FD001 data. 86
4.8:	Performance improvements on JAER-IT with C-MAPSS FD001 data. 86
4.9:	Results of the base JAER-IT models and Gaussian noise added models. 88

4.10:	Results of the pruned models.	88
4.11:	Comparison of JAER architectures with some recent deep learning work on C-MAPSS.	89
5.1:	Initial Conditions for MC90-V Dataset.	94
5.2:	Classification results for different training percentages.	100
5.3:	Classification results for varying amount of train and test data.	101
5.4:	NO _x estimations for different faults and ship loads.	103
5.5:	SO _x estimations for different faults and ship loads.	103
5.6:	Mean and standard deviations of NO _x emissions in the dataset for different faults and ship loads.	104
5.7:	Mean and standard deviations of SO _x emissions in the dataset for different faults and ship loads.	104

1. INTRODUCTION

The emergence of cutting-edge technology, including advanced sensors, the Internet of Things (IoT), and artificial intelligence/machine learning, has led to the rise of Industry 4.0, a transformative industrial revolution. The industrial machineries are now more complex than ever, which make their maintenance planning crucial in several aspects. Firstly, it ensures the reliability of digital systems, such as AI and IoT, by conducting regular updates and troubleshooting. It also plays a vital role in cybersecurity, protecting digital systems and data from potential threats. Secondly, maintenance optimizes data-driven processes by monitoring sensors and data storage systems. Furthermore, maintenance ensures the efficient operation of advanced technologies and facilitates adaptation to emerging advancements. Last but not least, it contributes to cost optimization and sustainability by minimizing downtime and maximizing equipment lifespan.

The thesis puts some light on predictive maintenance, especially for engines, using the deep learning techniques. This chapter gives a brief introduction to the thesis, including motivation, problem definition with respect to predictive maintenance, our contribution to the literature and the thesis outline.

1.1. Motivations

The progress made in technology, including advancements in sensors, the Internet of Things (IoT), and artificial intelligence/machine learning, has led to the emergence of Industry 4.0. First Industrial Revolution was characterized by the transition from manual labor to mechanization, driven by steam power and factory production. Second Industrial Revolution was driven by advancements in electricity, steel, and assembly line production, shaping modern industrialization. Third Industrial Revolution, a.k.a. digital revolution, has witnessed rapid advancements in electronics, telecommunications, and computer technologies. Finally Fourth Industrial Revolution is characterized by the fusion of digital technologies, such as artificial intelligence, big data, robotics, and

the Internet of Things (IoT). It encompasses advancements in fields like automation, biotechnology, renewable energy, and smart infrastructure, with a focus on interconnected "everything" and the transformation of industries and societies as a whole.

Maintenance of complex systems in the Industry 4.0 is possible by leveraging digitalization, predictive analytics, remote monitoring, and collaborative platforms, thus organizations can optimize asset performance, minimize downtime, and improve maintenance efficiency. Based on the collection, management, and analysis of the monitoring data, predictive maintenance (PdM) focuses on assets' condition in order to predict when maintenance should be performed. This enables organizations to meet safety compliances, preemptive corrective actions, and increased asset life. Benefits of PdM include, elimination of unscheduled equipment downtime caused by equipment or system failure, increased labor utilization, increased production capacity, reduced maintenance costs, and increased equipment lifespan.

As the systems become more complex and vast amount of monitoring data become available, model-based approaches are replaced with data-driven approaches. Increase in the data volume challenged researches to study advanced technologies and develop intelligent systems to deal with maintenance requirements of these complex systems.

For the last decade, deep learning has emerged as a dominant approach for analyzing big data. Initially stemming from the multilayer perceptron, it has evolved to encompass non-linear structures like autoencoders, convolutional neural networks, and recurrent neural networks. These advanced architectures have found extensive application in industrial big data analytics. Given that historical maintenance data is often abundant and intricate in nature, deep learning proves to be a valuable tool for uncovering hidden patterns within the data. Consequently, it holds promise for developing innovative Predictive Maintenance (PdM) techniques.

The thesis aims to explore deep learning architectures specifically for predictive maintenance, with a focus on fault classification and remaining useful life prediction. The forthcoming section will present our research questions and objectives.

1.2. Problem Definition

Prognostics research focuses on predicting future equipment performance and failures to optimize maintenance strategies. Several challenges are specific to this research area. Acquiring comprehensive and high-quality data can be difficult, impacting the accuracy of prognostic models. Selecting relevant features and preprocessing data to handle missing values, outliers, and noise is crucial. Limited failure data poses a challenge, affecting the performance of robust prognostic models. Generalizability across different equipment types and operating conditions is difficult.

In light of these challenges, our research aims to provide solutions and develop effective deep learning methods for predictive maintenance. To guide our investigation, we have formulated the following research questions:

- RQ1: How can we leverage publicly available datasets to extract the maximum value for prognostics?
- RQ2: Which deep learning techniques are best suited for prognostic research?
- RQ3: How can we handle the limitations of scarce failure data to extract the required features and labels?
- RQ4: How can we effectively incorporate operating conditions into deep learning architectures for improved prognostic performance?

To address these research questions, we have set the following objectives:

- OBJ1: Develop a comprehensive deep learning framework for predictive maintenance modeling, encompassing fault classification and remaining useful life prediction.
- OBJ2: Design a fault detection algorithm to facilitate the generation of accurate labels for training and evaluation purposes.

- OBJ3: Devise an approach to effectively integrate operating conditions into the deep learning models, enabling a more accurate representation of real-world scenarios.
- OBJ4: Whenever possible, apply and validate the developed prognostic framework on real-life data. Alternatively, investigate and analyze relevant public datasets to assess the performance and generalizability of the proposed methods.

By accomplishing these objectives, our research aims to contribute to the field of predictive maintenance by overcoming some of the existing challenges and providing effective deep learning techniques for prognostic analysis.

1.3. Contributions

Our contributions to the literature are as follows:

- We introduce a deep learning-based joint autoencoder architecture, which extracts spatial and temporal features for fault detection and remaining useful life prediction. Since operating conditions and environmental factors are important, data regarding these factors are also integrated to the architecture. We demonstrate the effectiveness of this architecture on two prognostics benchmarks.
- Deep learning requires vast amount of labeled data. It is not always possible to have labeled data in industrial systems. A new fault detection algorithm is proposed, which helps the framework to label the data accordingly.

1.4. Outline

The structure of the thesis is as follows: Chapter 1 provides an introduction that outlines the motivation and contribution of this thesis. Chapter 2 reviews relevant literature on maintenance policy development, predictive maintenance, the impact of Industry 4.0, and the application of deep learning. This chapter also discusses several public datasets commonly used as benchmarks in the field of predictive maintenance.

Chapter 3 case studies from predictive maintenance datathlons conducted as part of our research. Chapter 4 presents a case study focused on implementing a predictive maintenance framework for remaining useful life (RUL) prediction using the Turbofan Engine Dataset. Chapter 5 showcases a case study that utilizes the predictive maintenance framework for fault classification in the ship engine room. Chapter 6 concludes the thesis, providing a summary of the findings and suggesting opportunities for further research.



2. LITERATURE REVIEW

This chapter contains an extensive review of the predictive maintenance research specific to various application areas with deep learning methodology. The chapter starts with maintenance approaches and continues with prognostics and health management, including predictive maintenance.

2.1. Introduction

The Oxford Dictionary defines maintenance as “The act of keeping something in good condition by checking or repairing it regularly” [1]. Holmberg et al. define maintenance as the technologies or acts that is to achieve better maintainability, reliability, and availability of machine/equipment [2]. Maintenance improves the equipment’s economic life, lifespan, and overall productivity. Well-maintained equipment is less likely to break down, which leads to increased uptime, more working hours, fewer repair and maintenance costs, shorter return-of-investment times, and increased revenue. According to a recent survey conducted by Aberdeen Group on IT infrastructures, the average cost of system downtime is approximately \$260,000 per hour [3]. Another survey conducted in the automobile industry reveals that downtime costs range from \$22,000 to \$50,000 per minute, equivalent to about \$1.3 million to \$3 million per hour [4]. Implementing a successful predictive maintenance program in manufacturing can yield several benefits. It can increase system uptime by 9%, reduce risks related to safety, health, environment, and quality by 14%, lower overall maintenance costs by 12%, and extend the system’s lifespan by 20% [4].

Based on how we deal with break-down of the equipment, Scheffer and Girdhar define maintenance approaches as follows [5]:

- **Breakdown Maintenance:** In this approach, the maintenance policy allows machine/equipment to fail. The maintenance staff has not taken any precautions, and allow the machine/equipment run-to-failure. The advantage is that equipment is utilised for its maximum life time. The disadvantage is that, this approach

provides a short term benefit. Other than the unplanned shutdowns, if there exists too many failures, that will result a *crisis management* on the system, which in turn lower the morale of the maintenance staff. Such a breakdown may cause secondary failures. Another disadvantage is the need to store different kinds of spare parts in the repository. Failure can be acceptable if the machine/equipment shutdown does not affect production, and if labor and spare cost to replace machine/equipment is low.

- **Preventive Maintenance:** In this approach, maintenance is planned and applied at the time intervals specified by the machine/equipment manufacturer, e.g., based on calendar days or based on runtime hours. The maintenance policy allows to repair or replace the parts before an expected failure, which reduces unplanned downtimes. Machine/equipment shutdowns lower the production during maintenance process. Breakdowns or diminished performance occur if the maintenance is not carried out by a maintenance staff. Incorrect repair procedure may be applied, or spare parts may be installed improperly.
- **Predictive Maintenance:** In this approach, the condition of the machine/equipment is assessed for possible performance loss via the special hardware or sensors on the machine/equipment. Thus it is also called condition based maintenance. A maintenance is scheduled if a possible failure is detected for the near future. The life span is extended as a maintenance is planned for just before the breakdown. This allows acquisition of the spare parts from the manufacturer until the maintenance date. The disadvantage is that the assessment requires specialized hardware or sensors to monitor the condition. The maintenance staff must have special training to use/understand the specialized hardware.
- **Proactive Maintenance:** In this approach, failures are diagnosed to understand the root cause of the failure. Each failure is analyzed and proactive measures are taken to ensure that they are not repeated. Skillful and knowledgeable staff is required for root cause analysis. The analysis may also require design changes

which may or may not be carried out by the system beneficiary.

In terms of their effect on the production, the machine/equipment is classified as “critical”, “essential”, or “general purpose” [5]. “Critical” means the machine/equipment can affect plant safety, the capital cost is high, maintenance is expensive, or takes long time. “Essential” means the machine/equipment slows down the production, and the maintenance requires moderate expenditure, expertise, and time to repair. “General purpose” means the machine/equipment shutdown is not critical or essential at all, there are plenty of spare parts, and secondary damage is minimal. Critical systems benefit from predictive or proactive maintenance approaches. Essential systems use preventive or less sophisticated predictive maintenance approaches, general purpose system benefits from breakdown approach. The next section will dive into predictive maintenance and methods.

2.2. Prognostics and Health Management

Prognostics and Health Management (PHM) is the general term that defines predictive maintenance as an engineering process, which includes condition monitoring and data collection, fault diagnostics, prognostics, and data analysis. Modern systems generally work under complex operational conditions [6]. Equipments withstand a variety of mechanical and thermal stress during the operation of machinery [7]. In general terms, predictive maintenance is condition-driven maintenance, in which maintenance decisions are given by analyzing the monitoring data. In the new industrial era, the systems are monitored via special hardware or the sensors, which can be classified as follows:

Vibration monitoring: A vibration sensor is used to measure vibration of the equipment in multitude of dimensions. Vibration monitoring is useful to detect mechanical defects especially in rotating machinery. It can detect misalignments in the shaft, mechanical loose of the parts, structural problems in bearing or gear, shaft skews, etc. Vibration sensor is one of the most frequently used sensor for condition assessment [8].

Thermography: Thermal imaging is used to detect thermal, mechanical or struc-

tural defects in the electrical or mechanical equipment. It can be used to detect problems in generators, overhead lines, boilers, and carbon fiber structures [5].

Oil analysis: Lubrication is important for the moving parts. Lubrication oil is analyzed for physical condition, chemical characteristics, and worn machine components (e.g. size and number of free particles) to assess the condition of equipment, especially contact surface, as in bearings and gears [5, 8].

Acoustic emission: Rotating machinery, such as bearings and gears, generate noise when they are put to work. Likewise pipes emit sound when something flows in them. Acoustic signals from microphones and ultrasonic sensors are useful to detect, locate, and monitor cracks in structures, stream traps, or pipelines [9].

One or more sensors can be used together to monitor different components of the equipment. Number of sensors can be chosen to provide a coverage on the machine/equipment, e.g. vibration sensors can be placed on each joint of a robotic arm, or thermometer can be used for each engine cylinder.

Diagnostics and prognostics are two common applications of predictive maintenance. Diagnostics, also known as fault detection and diagnosis, refers to the process of identifying and analyzing faults or problems in a system or device. It involves identifying deviations or abnormalities in the behavior or performance of the system and determining the underlying causes of those issues [10]. Fault diagnostics can be applied to various domains, including mechanical systems, electrical systems, and more. The goal is to detect and diagnose faults early so that maintenance teams can take timely corrective actions to prevent unplanned downtime, prevent further damage to the equipment, improve system reliability, and reduce downtime.

Prognostics is the process of predicting the remaining useful life (RUL) of a system or component based on its current condition and historical data. It aims to predict the time remaining until a fault or failure occurs, allowing for proactive maintenance and minimizing downtime. Prognostics is widely applied in various industries, including manufacturing, transportation, aerospace, energy, and health care. It helps organizations transition from reactive maintenance strategies to proactive and predictive maintenance,

enabling them to optimize resource allocation, extend equipment lifetimes, and reduce the risk of unexpected failures [10].

With the development of industrial requirements, demands of both fault diagnosis and RUL prediction are increasing rapidly. When fault diagnosis is implemented, since most machines cannot be halted immediately, RUL prediction is also needed to give a basic idea of how long could the machine operate. Also, if the failure type and RUL are known in advance, then the corresponding measurements or maintenance can be prepared earlier to decrease the downtime and increase production [10]. In modern digital manufacturing, nearly 79.6% of the downtime of a machine tool is caused by its mechanical failures [11]. Thus, early fault detection is highly desired to increase the product quality, expedite the progress of unscheduled maintenance, and shorten the machine downtime. With such information available, engineers can schedule maintenance and repairs in advance, which can effectively prevent machinery performance degradation, malfunctions, or even catastrophic failures [6].

2.2.1. Approaches

PHM analysis methods can be categorized into physics-based, data-driven, and hybrid approaches. Physics-based approach (also known as model-based approach) relies on understanding the underlying physical processes and mechanisms of asset degradation and failure [12]. It involves building mathematical models based on the physics, mechanics, and materials science principles that govern the behavior of the asset under different operating conditions and stress levels. Therefore, prior knowledge of the system's failure mechanism is needed in advance [13]. Popular model-based approaches include particle filter, Paris–Erdogan model, Eyring model, and Weibull distribution [12, 14–17]. Model-based approaches can perform well but their use is often restricted in practice because of insufficient understanding of the degradation mechanisms for building the model and estimating its parameters [17]. Interested readers are suggested to see [18] for review of model-based methods.

As the system complexity increases, building a physical model for the system

becomes harder. The explosive growth of monitoring data brings new opportunities to PdM. Data-driven approaches do not depend on physical nature of the failure mechanism of the systems [13]. Since, data-driven approaches are able to learn degradation characteristics of machinery from massive monitoring data and build the corresponding models automatically, they are gaining more and more attention in the field of PdM [19]. A recent review of data-driven approaches can be found in [20, 21]. Finally, hybrid-model approaches use both model-based and data driven-based approaches together for PdM model construction. Readers are suggested to review Zonta et al. [22] for a review.

2.2.2. General Framework for Data-Driven Prognostics

The general framework for data-driven prognostics provides a structured approach to leverage historical data and develop models that can offer insights into future system behavior. A general framework for data-driven prognostics (Figure 2.1) typically consists of the following steps [23]:

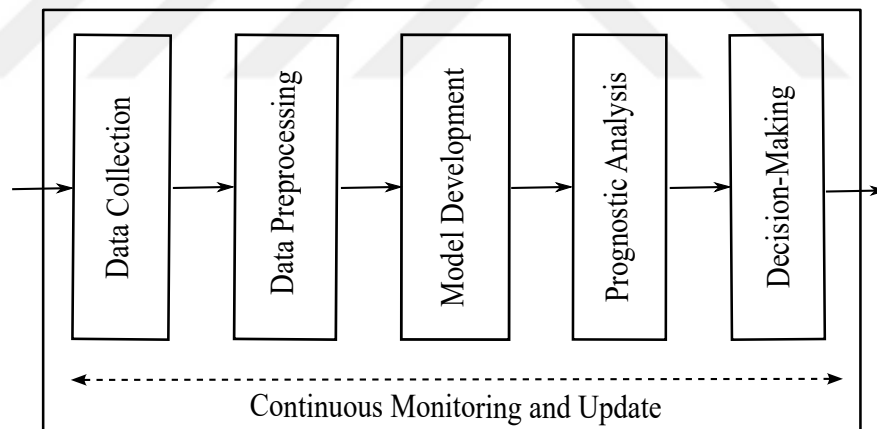


Figure 2.1: General framework for data-driven prognostics.

Data Acquisition: Prognostics relies on collecting and analyzing relevant data about the system or component being monitored. This can include sensor readings, operational parameters, environmental conditions, maintenance records, and other relevant information.

Data Preprocessing: Data preprocessing is a crucial step that involves transforming raw data to enhance its quality, handle missing values and outliers, and ensure

compatibility with prognostic modeling techniques. It includes tasks such as data cleaning, normalization, feature scaling, feature engineering, and dimensionality reduction. By preprocessing the data effectively, the accuracy, reliability, and generalizability of prognostic models are improved. This enables the extraction of valuable insights from historical data, leading to more accurate predictions about future performance or failure, and facilitating optimized maintenance strategies in various industries

Model Development: In this step, prognostic models are developed using machine learning, statistical modeling, or other data-driven approaches. The choice of modeling technique depends on the nature of the data and the specific prognostic task. Common modeling techniques include regression models, classification models, time series analysis, or ensemble methods.

Prognostic Analysis: The developed models are then used to analyze the current condition of the system and make predictions about its future behavior. This analysis involves estimating the fault class, forecasting the remaining useful life, determining the probability of failure within a certain time frame, or providing a degradation trend.

Decision-Making: The prognostic analysis results can guide maintenance decisions, such as scheduling maintenance actions based on the predicted remaining useful life. This proactive approach helps to optimize maintenance activities, reduce costs, and improve system availability.

Continuous Monitoring and Update: Fault prognostics is an ongoing process that requires continuous monitoring and updating of the prognostic models. As new data becomes available, the models can be retrained or updated to improve their accuracy and reliability.

The deterioration of a system's health condition follows a P-F curve, where 'P' represents potential failure and 'F' denotes functional failure (Figure 2.2), which signifies the system's breakdown [24, 25]. Prognostics allows us to analyze monitoring data and evaluate the health status, degradation, and anticipated lifespan based on the P-F curve. The primary objective of prognostics is to predict the remaining useful life (RUL) of a system using historical data before a failure occurs. RUL refers to the

time remaining from the assessment point until the end of the machine's useful life. Predicting RUL involves forecasting the remaining operational time of the machinery before it becomes inoperable [26].

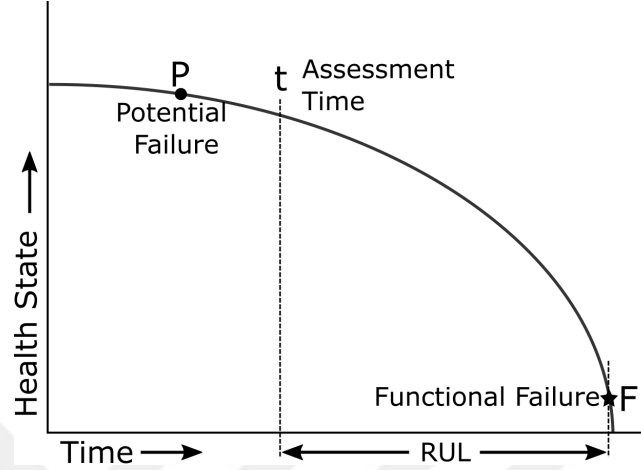


Figure 2.2: Example deterioration of a system.

2.2.3. Application Areas and Research Problems

In the existing literature, predictive maintenance mostly applied to bearings and gears, engines, milling machines, lithium-ion batteries, fuel cells and hard disks.

Bearing and gears: Bearings, gears and gearboxes play critical roles in mechanical systems like wind turbines, vehicles and many other industrial applications [10]. Predictive maintenance techniques for bearings and gearboxes involve monitoring parameters such as vibration, temperature, and lubrication [7, 12, 27]. By analyzing this data, potential issues like misalignment, wear, or inadequate lubrication can be detected early. Fault detection, fault classification and RUL prediction are some of the applications on bearings, gears, and gearboxes. Interested readers are suggested to see [28–30] for some recent reviews.

Milling machines: Milling machines have extensive usage in manufacturing operations. Predictive maintenance for milling machines encompasses the monitoring of parameters like vibration, cutting forces, lubrication, and tool wear. Through the analysis of this data, possible problems such as spindle imbalance, misalignment, or

tool breakage can be detected [31]. By implementing predictive maintenance, timely actions such as realigning, tool replacements, or adjusting cutting parameters can be undertaken. This ensures precise machining, reduces instances of unplanned downtime, and optimizes productivity by maximizing the machine's operational efficiency [19, 32].

Lithium-ion batteries: Due to the continuous improvements in their cycling efficiencies and energy and power densities, as well as the reduction in their costs, lithium-ion (Li-ion) rechargeable batteries have been increasingly used as energy storage devices in portable devices such as cell phones, laptops, hybrid and electric vehicles, and energy storage systems [33]. Compared with traditional batteries, lithium-ion battery has many advantages including high output voltage, high energy density, low self-discharge, long cycle life, high reliability [34]. Predictive maintenance for lithium-ion batteries involves monitoring parameters such as voltage, current, temperature, and cell imbalance [16]. By analyzing this data, potential issues like capacity degradation, thermal runaway, or cell aging can be detected. Predictive maintenance enables proactive measures such as cell balancing, temperature control, or capacity restoration, optimizing battery performance, preventing unexpected failures, and ensuring the reliability of energy storage systems [34, 35]. Interested readers are suggested to see [36] for recent review.

Fuel cells: Fuel cells are clean and efficient energy conversion devices used in various applications. Predictive maintenance for fuel cells involves monitoring parameters such as temperature, pressure, and gas flow rates. By analyzing this data, potential issues like catalyst degradation, membrane damage, or fuel leakage can be identified in advance. Predictive maintenance helps optimize fuel cell performance, maximize efficiency, and ensure uninterrupted power generation. It reduces the risk of unexpected failures, extends system lifespan, and improves overall operational reliability [37].

Hard disks: Hard disks are essential components in computer systems and data storage devices. Predictive maintenance for hard disks involves monitoring parameters such as temperature, seek error rates, and SMART (Self-Monitoring, Analysis, and

Reporting Technology) data. By analyzing this information impending disk failures can be predicted [38].

Turbofan engines: Turbofan engines, which are frequently employed for aircraft propulsion, undergo predictive maintenance that includes monitoring several parameters like vibration, temperature, pressure, and fuel consumption [39]. By consistently analyzing operational data, potential problems such as imbalance, wear, or degradation of engine components can be identified at an early stage [17]. Predictive maintenance methods facilitate the detection of trends, patterns, irregularities, and RUL predictions enabling maintenance teams to plan interventions promptly. Taking proactive maintenance measures supported by predictive analytics aids in enhancing engine performance, preventing failures, and minimizing downtime and maintenance expenses in the field of aviation and aerospace [40].

For these application areas, the relevant literature defines several research problems, some of which may be relevant in more than one application area. Recent studies on these research problems along with their application areas are given in Table 2.1.

Fault Detection and Classification: Fault detection and classification involve identifying the presence of faults or abnormalities in equipment or systems and categorizing them into specific fault types or classes. The classification step aims to assign the detected faults to specific fault types or categories, enabling targeted maintenance actions and further analysis to address the identified issues.

Remaining Useful Life Prediction: Remaining Useful Life (RUL) prediction refers to predicting the remaining operational lifespan of an asset or system before it reaches a critical or failure state. RUL prediction takes into account the current health condition of the asset, historical performance data, degradation patterns, and other relevant factors. Accurate RUL prediction helps optimize maintenance scheduling, minimize downtime, and avoid costly failures.

Uncertainty Estimation: Uncertainty estimation involves quantifying and assessing the uncertainty associated with predictions or estimates made in predictive maintenance and prognostics. Since the future behavior of complex systems can be

uncertain, it is important to provide not only a single prediction or estimate but also a measure of confidence or uncertainty. Uncertainty estimation aids decision-making by highlighting areas where additional data or analysis may be needed and helps manage risks associated with maintenance actions.

Capacity Estimation: Capacity estimation for batteries involves assessing the available energy storage capacity of a battery. It is particularly relevant in the context of battery health management, where accurate capacity estimation helps in determining the remaining capacity of a battery compared to its original or nominal capacity. Battery capacity estimation techniques utilize data from battery management systems, voltage and current measurements, discharge tests, or impedance spectroscopy to estimate the current state of charge or state of health of the battery. This information is valuable for optimizing battery usage, predicting battery degradation, and planning maintenance or replacement strategies for batteries in various applications.

Degradation Trajectory: Degradation trajectory (P-F Curve, see Figure 2.2) refers to the pattern or path that describes the deterioration of an asset or system over time. It represents the progression of the asset's condition from an initial healthy state to a degraded state. It helps in understanding the aging process, identifying degradation patterns, and predicting the remaining useful life (RUL) of the asset.

2.3. Methods for Data-Driven Prognostics

As researchers gain access to larger amounts of data, prognostics algorithms have evolved to leverage this extensive data pool [65]. In recent times, deep learning techniques, known for their superior performance in tasks like image recognition and natural language processing, have also been utilized in predictive maintenance and prognostics [66]. This section summarizes methods for data preprocessing and model development steps of the general framework.

Table 2.1: Recents studies on PHM application domains.

Research Problems	Recent Studies
Bearing & Gears	
Degradation trajectory	[41]
Fault Detection / Classification	[10, 42–45]
RUL prediction	[7, 12–15, 23, 27, 46–48]
Uncertainty estimation	[32, 49]
CNC Machinery	
Fault Detection / Classification	[42]
RUL prediction	[40, 50]
Hard disks	
RUL prediction	[38]
Lithium-Ion Batteries	
Capacity estimation	[51]
Degradation trajectory	[52]
RUL prediction	[34, 53–55]
Turbofan engines	
Degradation trajectory	[56]
Maintenance Decisions	[57]
RUL prediction	[6, 17, 23, 39, 58–63]
Uncertainty estimation	[49, 64]

2.3.1. Data Preprocessing

Data preprocessing is essential for accurate and reliable prognostic models. It involves a series of techniques to enhance data quality, handle missing values, outliers, and noise, and ensure compatibility with prognostic modeling methods.

Data cleaning is a critical step that involves identifying and addressing missing data, outliers, and noise. Missing values can be imputed using various techniques, and outliers or noisy data can be detected and treated to avoid skewing the results. Data normalization and scaling are often employed to ensure features are on a common scale, preventing certain features from dominating the model training process. *Data normalization* is applied to scale the features to a common range, which is particularly

important when the features have different scales. *Data standardization* transforms the data to have zero mean and unit variance, which can help certain modeling techniques, such as neural networks or support vector machines, converge faster or improve performance.

Dimensionality reduction techniques such as principal component analysis (PCA), linear discriminant analysis (LDA), or autoencoders (AE) help reduce the complexity of high-dimensional data, improving model performance and interpretability. PCA aims to transform high-dimensional data into a lower-dimensional space while retaining the most important information. PCA identifies the principal components, which are new variables that capture the maximum variance in the data. These components are orthogonal to each other, allowing for a reduction in dimensionality without significant loss of information. By projecting the data onto the principal components, PCA enables visualization, data compression, and feature extraction. It finds applications in various fields, including image processing, pattern recognition, and data visualization, helping to uncover underlying patterns and simplify complex datasets [67].

LDA is another dimensionality reduction technique and also a classification algorithm. Unlike PCA, LDA takes into account the class labels associated with the data samples. LDA aims to find a lower-dimensional space that maximizes the separation between classes while minimizing the within-class similarity. By projecting the data onto this space, LDA creates new discriminant features that preserve class information. It can effectively reduce dimensionality while retaining the discriminative power for classification tasks. LDA finds applications in areas such as pattern recognition, face recognition, and document classification. It helps to enhance the separability of classes and improve classification accuracy in supervised learning scenarios [68]. Autoencoders will be introduced later in this section.

Once the data is collected, relevant features or parameters are extracted to characterize the system's condition using *feature extraction* methods. These features could be derived from sensor data, such as temperature, vibration, or current readings, or they could be derived from other system measurements or indicators. Tri-domain features

from time-series data are regularly used by the researchers. They refer to a set of features that capture information from three distinct domains: time, frequency, and statistical. These features are commonly used in signal processing and analysis tasks. The time domain features describe the signal characteristics in the time dimension, such as mean, standard deviation, or temporal patterns. The frequency domain features represent the signal's frequency content, including spectral characteristics like power spectrum or dominant frequencies. The statistical domain features encompass statistical properties of the signal, such as skewness, kurtosis, or entropy. By incorporating information from multiple domains, tri-domain features provide a comprehensive representation of the signal, enabling more robust and informative analysis in various applications, including audio processing, image analysis, and condition monitoring [69].

By applying these data preprocessing techniques, the quality and suitability of the data for prognostic modeling are enhanced, which leads to more accurate and reliable predictions about future outcomes, enabling informed decision-making and optimized maintenance strategies.

2.3.2. Model Development

Model development involves the creation of algorithms or models that can analyze historical data and perform diagnostic or prognostic analysis on the equipment/systems. This typically requires the selection of an appropriate modeling technique, such as machine learning or deep learning. The chosen model is trained using labeled data, where the input features represent the system's condition, and the output is the target variable, such as predicted fault class, remaining useful life or failure probability. Model development includes other tasks such as hyper-parameter optimization, and model evaluation. The resulting model can then be used to make accurate predictions and inform maintenance decision-making, optimizing maintenance schedules and reducing downtime.

Machine Learning

This section contains information on and references to classical machine learning models that are widely used in PHM model development.

Random Forest is an ensemble learning technique, which is commonly used for various tasks such as classification and regression. It involves constructing multiple decision trees during the training phase. Random Forest is considered a nonlinear modeling tool that addresses certain limitations of regular decision trees. By employing random feature selection, it overcomes issues like low accuracy and prevents overfitting [70]. To optimize its performance, hyper-parameter tuning is often employed. Adjusting parameters such as maximum depth, maximum number of nodes, and minimum number of points per leaf node can control the size and complexity of the decision trees within the Random Forest implementation. Random Forest is particularly well-suited for addressing failure classification problems in scenarios where there is unclear prior knowledge or incomplete data. It can handle such situations effectively and provide reliable results.

Gradient Boosting, which is an ensemble learner like random forest, constructs a model in a sequential manner. It differs in that it allows for the optimization of any differentiable loss function during the generalization process [71]. The resulting model is a combination of individual models, making gradient boosting highly effective for both classification and regression tasks. Although the predictive ability of each individual model in gradient boosting is relatively weak and susceptible to overfitting, the technique excels by combining numerous such weak models into an ensemble. This aggregation of models leads to a significantly enhanced overall result. The gradient is used to minimize a loss function. In each round of training, a weak learner is built and its predictions are compared to the correct outcome that we expect. The distance between prediction and truth represents the error rate of our model. These errors can now be used to calculate the gradient. The gradient is basically the partial derivative of the loss function - so it describes the steepness of our error function. The gradient can be used to find the direction in which to change the model parameters in order to reduce

the error in the next round of training by “descending the gradient”.

Gaussian Process is a stochastic process that encompasses a collection of random variables, such as time series data. Each of these data points follows a multivariate normal distribution, and the joint distribution of all these variables forms the distribution of a Gaussian process. In machine learning, Gaussian process-based algorithms utilize lazy learning and a kernel function to measure the similarity between samples. This approach allows for the prediction of values for unseen data points based on the training data. What sets Gaussian process models apart is their non-parametric nature. They do not rely on predefined parameters, which makes them highly expressive. Additionally, Gaussian process models provide not only an estimate for the predicted value of a data point but also uncertainty information. The uncertainty is represented by a one-dimensional Gaussian distribution [72]. The Gaussian process technique finds interesting applications by establishing a probabilistic model over data and adjusting an underlying function to account for posterior probabilities in prognostics. In this way, Gaussian processes not only predict future values but also provide insights into the associated uncertainty.

Deep Learning

Deep learning is a specific branch of machine learning techniques that utilizes artificial neural networks, drawing inspiration from the structure of neurons found in the human brain. The term “deep” typically refers to the inclusion of multiple layers within artificial neural networks, similar to the layered structure observed in the human brain [19]. Deep learning methods have recently attracted the attention of the PHM researchers due to the capability of these methods to automatically extract features and their good generalization performance. [33] This section contains information on and references to deep learning architectures that are identified in the literature review.

Convolutional Neural Networks (CNN) are particularly effective in processing spatial and local patterns in data, making them well-suited for image and signal processing tasks. In the context of predictive maintenance, CNNs can be utilized to analyze

sensor data or image-based inspections for fault detection and classification. CNNs employ convolutional layers that automatically learn relevant features from the input data, capturing important patterns and spatial relationships. By extracting and leveraging these features, CNNs can accurately identify abnormal patterns indicative of faults or failures in equipment or systems.

Recurrent Neural Networks (RNN) are designed to handle sequential and time-dependent data. RNNs have feedback connections that allow them to retain information over time, making them suitable for modeling temporal dynamics. Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU) are specialized variants of RNNs that address the vanishing gradient problem and capture long-term dependencies in the data. In the context of predictive maintenance, RNNs, LSTM, and GRU can be utilized for various tasks. For example, they can model the temporal patterns and relationships in sensor data to predict the remaining useful life (RUL) of equipment or systems. By considering the historical data and capturing long-term dependencies, these models can provide accurate estimations of the time to failure, enabling proactive maintenance strategies.

Deep Belief Networks (DBN) are powerful generative models that consist of multiple layers of restricted Boltzmann machines (RBMs). RBMs are unsupervised learning models that learn hierarchical representations of data. In the context of predictive maintenance, DBNs are employed for feature learning from sensor data. They can automatically discover informative features that capture the underlying patterns and anomalies in equipment or system behavior. By learning a hierarchy of representations, DBNs can model complex relationships and uncover hidden structures in the data, enabling more accurate fault detection and prognostics.

Autoencoders are unsupervised learning models used for feature extraction, dimensionality reduction, and anomaly detection. They consist of an encoder network that maps the input data to a lower-dimensional representation (encoding) and a decoder network that reconstructs the input from the encoded representation. Autoencoders have two advantages: dimension reduction and automatically selecting relevant fea-

tures, both of which reduce overfitting [48]. In the context of predictive maintenance, autoencoders can be employed to extract relevant features from sensor data. By training the autoencoder on normal operating conditions, it can reconstruct the normal patterns accurately. Anomalies or deviations from the reconstructed data can indicate potential faults or abnormal behavior, allowing for early detection and intervention.

Deep learning concepts offer powerful tools for analyzing sensor data, detecting faults, predicting remaining useful life, and identifying anomalies in predictive maintenance applications. Deep learning offers several advantages stemming from two key aspects. Firstly, it eliminates the need for complex and time-consuming feature extraction and dimensionality reduction processes. Instead, it operates directly on raw data with minimal preprocessing. Secondly, the method harnesses the power of deep hidden layers to generate high-level abstractions and representations of the data. Through the hierarchical structure of these layers, the model can automatically learn intricate and meaningful features, facilitating a more comprehensive understanding of the data [17].

Hyper-parameter Optimization

Hyper-parameter optimization refers to the process of finding the best set of hyper-parameters for a machine learning models. Hyper-parameters are parameters that are set before the learning process begins, and optimizing them involves searching for the values that yield the best performance and generalization ability of the model. Optimization approaches offer unique characteristics and trade-offs in terms of computational cost, search efficiency, and exploration-exploitation balance.

Grid Search: Grid search is a systematic approach to hyper-parameter optimization. It involves defining a grid of hyper-parameter values and exhaustively evaluating the model's performance for each combination of values [61]. The grid typically covers a range of possible values for each hyper-parameter, and the model's performance is assessed using a performance metric (e.g., accuracy or loss). Grid search allows for a comprehensive exploration of hyper-parameter space but can be computationally

expensive, especially when dealing with a large number of hyper-parameters or a wide range of values. Despite its computational cost, grid search provides a simple and intuitive way to search for the optimal set of hyper-parameters.

Random Search: Random search is an alternative approach to hyper-parameter optimization that involves randomly sampling from the hyper-parameter space. Instead of predefined grids, random search selects hyper-parameters randomly within user-defined ranges. By exploring various combinations in a stochastic manner, random search can discover effective hyper-parameter configurations without exhaustively evaluating all possibilities. It is less computationally intensive compared to grid search but does not guarantee full coverage of the hyper-parameter space.

Particle Swarm Optimization: Particle swarm optimization (PSO) is a population-based stochastic optimization algorithm inspired by social behavior, particularly the flocking of birds or schooling of fish. In PSO, a group of particles (representing potential solutions) moves through a multidimensional search space, aiming to find the optimal solution. Each particle adjusts its position and velocity based on its own best-known position and the best position found by the swarm. In the context of hyper-parameter optimization, PSO can be used to search for the optimal set of hyper-parameters by treating it as an optimization problem. PSO has the advantage of efficient exploration of the search space and can handle continuous and discrete hyper-parameters.

Genetic Algorithms: Genetic algorithms (GA) are population-based search and optimization algorithms inspired by the principles of natural evolution [73]. In a genetic algorithm, a population of potential solutions (chromosomes) evolves over generations through processes such as selection, crossover, and mutation. These processes mimic the principles of natural selection, genetic recombination, and genetic variation. In the context of hyper-parameter optimization, genetic algorithms can be applied by representing each chromosome as a set of hyper-parameters and evaluating their performance using a fitness function. Genetic algorithms provide a robust and flexible approach to explore and exploit the search space, allowing for a global search for the optimal set of hyper-parameters.

Evaluation Metrics

Evaluation metrics are quantitative measures used to assess the performance or quality of a machine learning model. They allow for objective comparisons between different approaches and help determine the success of a model or system in achieving its intended goals. Commonly used evaluation metrics depend on the specific task or problem being addressed. The following metrics are used commonly in the literature and in our thesis studies.

Confusion Matrix: A confusion matrix is a tabular representation that shows the performance of a classification model by summarizing the predicted and actual class labels. It provides a clear breakdown of true positive (TP), true negative (TN), false positive (FP), and false negative (FN) predictions, as shown in Figure 2.3. Each row of the matrix corresponds to the actual class, while each column corresponds to the predicted class. The values in the matrix allow for calculations of evaluation metrics such as accuracy, precision, recall, and F1-score, providing insights into the model's performance and potential areas of improvement.

		Actual Labels	
		Positive	Negative
Predicted Labels	Positive	True Positive (TP)	False Positive (FP)
	Negative	False Negative (FN)	True Negative (TN)

Figure 2.3: Confusion matrix for classification tasks.

Accuracy (ACC): Accuracy is a metric that measures the proportion of correctly classified instances or predictions.

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \quad (2.1)$$

Precision: The ratio of true positive predictions to the total predicted positive instances. It quantifies the correctness of positive predictions.

$$precision = \frac{TP}{TP + FP} \quad (2.2)$$

Recall: The ratio of true positive predictions to the total actual positive instances. It quantifies the completeness of positive predictions.

$$recall = \frac{TP}{TP + FN} \quad (2.3)$$

F1-Score (F1): A balanced metric that combines precision and recall, providing a single value that considers both measures.

$$F1 = 2 \times \frac{precision \times recall}{precision + recall} \quad (2.4)$$

Matthews Correlation Coefficient (MCC): Another balanced metric that is more informative than other metrics [74]. It takes all the balance ratios of the all the measures into account.

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \quad (2.5)$$

Mean Squared Error (MSE): An evaluation metric commonly used in regression tasks, measuring the average squared difference between true values y and predicted values $\hat{y}$. Square root of MSE (root mean squared error, RMSE) is another common metric.

$$MSE(y, \hat{y}) = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad (2.6)$$

$$RMSE(y, \hat{y}) = \sqrt{MSE(y, \hat{y})} \quad (2.7)$$

Mean Absolute Error (MAE): Another evaluation metric commonly used in regression tasks to quantify the average magnitude of errors between predicted and

actual values. It provides a measure of the average absolute difference between true values y and predicted values $\hat{y}$.

$$MAE(y, \hat{y}) = \left(\frac{1}{n}\right) \sum_{i=1}^n |y_i - \hat{y}_i| \quad (2.8)$$

These metrics are just a few common examples, and there are many other evaluation metrics available, each suitable for specific tasks and objectives. The choice of evaluation metric depends on the nature of the problem and the desired outcome. It's important to select an appropriate metric that aligns with the goals of the project and provides meaningful insights into the performance of the system or model being evaluated.

2.4. Public Datasets for Prognostics Research

Public datasets play a crucial role in advancing prognostics research by enabling benchmarking, reproducibility, method development and validation, resource accessibility, diversity, and collaboration. They contribute to the growth of the field, facilitate knowledge sharing, and accelerate the development of effective prognostics techniques for various applications and industries. Table 2.2 summarizes some of the recent studies on public prognostics datasets. Rest of the section we summarize the public dataset we encountered during literature review.

Case Western Reserve University (CWRU) Bearing Dataset [75] consists of vibration signals recorded from bearings subjected to various fault conditions, including inner race, outer race, and ball faults, as well as healthy bearings. The dataset encompasses four different bearing conditions: normal, inner race fault, outer race fault, and ball fault, each at different severity levels. The signals were collected using accelerometers placed on the bearing housing, and the dataset provides both raw time-domain signals and preprocessed features for analysis. The dataset offers a comprehensive set of vibration signals with different fault types and severity levels, making it a valuable resource for developing and evaluating fault diagnosis and prognosis algorithms. It has become a standard benchmark dataset in the field, enabling researchers to compare and assess the

Table 2.2: Recents studies on public datasets.

Datasets	Recent Studies
Bearing & Gears	
CWRU Bearing Dataset [75]	[42–45]
IMS Bearing Dataset [76]	[10]
MaFaulDa Dataset [77]	[44]
PRONOSTIA Dataset [78]	[7, 14, 15, 27, 46–49, 69, 79]
NASA Spiral Bevel Gear Damage Dataset [80]	[13]
XJTU-SY Bearing Dataset [81]	[41, 82–84]
CNC Machinery	
NASA Milling dataset [85]	[40]
PHM 2010 Data Challenge Dataset [86]	[42, 50]
Hard disk	
Backblaze Dataset [87]	[38]
Baidu Dataset	[38]
Lithium-Ion Batteries	
MIT-Stanford Li-ion Battery Dataset [88]	[55]
NASA Battery Dataset [89]	[34, 51, 53, 54]
RWTH Battery Dataset [90]	[52]
Turbofan engine	
C-MAPSS Dataset [91]	[6, 17, 39, 57–63]

performance of different algorithms and methodologies.

Intelligent Maintenance Systems (IMS) Bearing Dataset [76] was provided by Intelligent Maintenance Systems of University of Cincinnati, and hosted on Prognostic Data Repository of NASA¹. The dataset consists of vibration signals obtained from accelerometers placed on bearings under various operating conditions and with different fault scenarios. It is composed of three subsets of bearing degradation runs. During these runs, four double row bearings were installed on a shaft. Accelerometers were installed on the bearing housings. An oil circulation system was designed to lubricate the bearings. A magnetic plug was installed in the oil feedback pipe to collect debris.

¹<https://www.nasa.gov/content/prognostics-center-of-excellence-data-set-repository>

The run was stopped automatically by an electrical switch when the accumulated debris exceeded a certain level. After the runs, the bearings were checked and their fault patterns were recorded in detail.

Initially published in IEEE international conference of Prognostics and Health Management 2012 Data Challenge, the NASA FEMTO Bearing Dataset [78], also known as the PRONOSTIA, is a widely used dataset in the field of predictive maintenance. It was developed by FEMTO-ST Institute² in France. The dataset is specifically focused on bearing fault detection and diagnosis. The dataset consists of vibration signals recorded from accelerometers installed on bearings subjected to various fault conditions. The dataset provides valuable information for studying the behavior of bearings under different fault types, such as inner race, outer race, and ball faults. It is designed to aid researchers and practitioners in developing and evaluating fault detection and diagnosis algorithms for predictive maintenance applications. It includes multiple data files, each corresponding to a specific fault type and severity level. The signals are typically recorded at high sampling rates to capture detailed information about bearing conditions. Researchers often use the dataset to develop and benchmark fault detection and diagnosis algorithms, as well as to explore feature extraction, signal processing, and machine learning techniques for predictive maintenance tasks related to bearing health monitoring.

Another public dataset is NASA Milling dataset [85], which was designed to support research on tool wear monitoring and prediction in milling processes. The dataset contains sensor measurements collected during milling operations, providing valuable information for developing models and algorithms related to tool condition monitoring. The milling dataset captures data from various sensors, including acoustic emission sensors, force sensors, and temperature sensors. These sensors monitor the milling process and collect measurements that can be used to analyze the condition of the cutting tool, detect tool wear, and predict tool failure. Researchers can utilize

²Franche-Comté Electronics Mechanics Thermal Science and Optics–Sciences and Technologies Institute

the milling dataset to explore different approaches for tool wear monitoring, develop algorithms for real-time tool wear detection, and investigate the relationship between sensor measurements and tool conditions.

Computer-controlled machining tools that use a rotating cutting tool to shape and carve materials with a high level of precision, accuracy, and automation are known as CNC milling machines. The cutting tools condition directly drives the product quality. So the process is monitored via various sensors, such as dynamometer, accelerometer, and acoustic. PHM Society has provided such a monitoring data for PHM 2010 Data Challenge [86] , in which the researchers are challenged to development of prognostics and health management techniques for CNC milling machines. Participants were provided with datasets containing sensor measurements, such as vibration, temperature, and cutting parameters, collected during the operation of CNC milling machinery.

The MIT-Stanford Lithium-ion Battery Dataset [88] is a comprehensive dataset created by researchers at MIT for studying the lifetime prediction of lithium-ion batteries. It consists of 124 commercial lithium iron phosphate/graphite cells that were subjected to fast-charging conditions. The cells exhibit a wide range of cycle lives, varying from 150 to 2,300 cycles. The dataset focuses on the behavior of the batteries during their early cycles, prior to significant capacity degradation. Discharge voltage curves from these early cycles are used as features for machine learning analysis. The dataset includes information on the cycle life of each cell, allowing for the development of predictive models and classification algorithms. The Dataset highlights the importance of combining deliberate data generation with data-driven modeling to gain insights into the behavior and lifespan of complex systems like lithium-ion batteries. It provides a valuable resource for researchers in the field of battery technology and predictive maintenance, enabling advancements in battery reliability, performance, and lifetime prediction.

NASA provides another public lithium-ion battery dataset [89]. This dataset was collected from a custom-built battery prognostics testbed located at the NASA Ames Prognostics Center of Excellence (PCoE). The dataset focuses on lithium-ion (Li-ion)

batteries and encompasses three distinct operational profiles: charging, discharging, and Electrochemical Impedance Spectroscopy. The batteries were subjected to varying temperatures during these profiles. Discharges were conducted at different current load levels until the battery voltage reached specific preset thresholds. Notably, certain thresholds were intentionally set lower than the manufacturer's recommendation of 2.7 V to induce deep discharge aging effects in the batteries. The batteries underwent repeated charge and discharge cycles to expedite the aging process. This dataset provides valuable insights into battery behavior and aging under controlled test conditions, facilitating research and advancements in battery prognostics and health management.

NASA Turbofan Engine Degradation Simulation Dataset [91], or C-MAPSS Dataset, consists of multiple instances of run-to-failure data for turbofan engines. These engines were simulated using the C-MAPSS (Commercial Modular Aero-Propulsion System Simulation) Software, which is a thermodynamic simulation model specifically designed for turbofan engines. Consequently, the dataset is also referred to as the C-MAPSS Dataset.

The NASA Turbofan Engine Degradation Simulation Dataset, aka C-MAPSS dataset, is one of the common benchmarks among prognostics researchers. The dataset provides a comprehensive set of measurements collected from turbofan engines under various operating conditions and degradation scenarios. It includes sensor readings such as temperature, pressure, and rotational speed, along with corresponding information about the engine's remaining useful life. The dataset allows for the development and evaluation of algorithms and models that aim to predict engine degradation and estimate RUL, aiding in the development of proactive maintenance strategies and improving the reliability and safety of aircraft engines.

2.5. Challenges in Prognostics Research

Prognostics research aims to predict the future performance or failure of equipment or machinery to optimize maintenance strategies. Here are some frequent challenges specific to prognostics research area:

- **Data availability and quality:** Prognostics research relies on historical data related to equipment performance, maintenance records, sensor readings, and other relevant variables. However, obtaining comprehensive and high-quality data can be challenging, particularly for older or legacy systems. Incomplete or inconsistent data can affect the accuracy and reliability of prognostic models.
- **Feature selection and data preprocessing:** Predictive maintenance often involves a large number of input variables—mostly in time-series signals—and determining which features are most relevant for prognosis can be a challenge. Selecting the right features and preprocessing the data to handle missing values, outliers, or noise is crucial for developing accurate prognostic models [11, 48].
- **Limited failure data:** In predictive maintenance, failures are relatively rare events, especially for reliable systems or when maintenance practices are effective. Limited failure data can make it challenging to develop robust prognostic models, as the imbalance between failure and non-failure cases can impact the model's performance [92].
- **Complex failure mechanisms:** Understanding and modeling the underlying failure mechanisms of complex systems can be challenging. Equipment failures may be influenced by multiple interacting factors, including environmental conditions, usage patterns, component interactions, and system dependencies. Capturing and modeling these complexities accurately is crucial for effective prognostic research [13].
- **Real-time monitoring and prediction:** Prognostic research in predictive maintenance often requires real-time monitoring of equipment health and making predictions in near real-time. Implementing robust and scalable monitoring systems, integrating sensor data, and performing real-time analysis can be technically demanding.
- **Generalizability and transferability:** Prognostic models developed for one type of

equipment or specific operational conditions may not necessarily generalize well to different equipment types or operating environments. Achieving models that are transferable across different operating conditions, assets or systems requires careful consideration of the variations and factors that influence prognosis across different contexts [33, 45].

- **Uncertainty estimation:** Prognostic models should not only provide predictions but also quantify the uncertainty associated with those predictions. Estimating uncertainty helps decision-makers understand the confidence level of the prognostic output and assists in optimizing maintenance decisions and resource allocation [49].
- **Integration with maintenance decision-making:** The ultimate goal of prognostic research in predictive maintenance is to inform maintenance decision-making. Integrating prognostic models with existing maintenance practices, decision support systems, and operational workflows can be a challenge that requires collaboration between researchers, maintenance personnel, and stakeholders [57].

2.6. Summary

This chapter presents a literature review on deep learning methodology in the context of predictive maintenance research. It begins by introducing different maintenance approaches, such as breakdown, preventive, predictive, and proactive maintenance, emphasizing their significance in enhancing equipment lifespan, productivity, and cost-effectiveness while minimizing downtime and repair costs. The concept of Prognostics and Health Management (PHM) is introduced, which encompasses condition monitoring, fault diagnostics, prognostics, and data analysis for predictive maintenance.

The text explores various methods and sensors employed in predictive maintenance, such as vibration monitoring, thermography, oil analysis, and acoustic emission. It explains the concepts of diagnostics and prognostics, where diagnostics focus on fault detection and diagnosis, while prognostics aim to predict the remaining useful life

(RUL) of a system based on its current condition and historical data.

Specific application areas within predictive maintenance are highlighted, including bearings and gears, milling machines, lithium-ion batteries, fuel cells, hard disks, and turbofan engines. Research challenges in these areas are identified, such as fault detection and classification, RUL prediction, uncertainty estimation, capacity estimation, and degradation trajectory analysis.

The chapter categorizes predictive maintenance analysis methods into physics-based, data-driven, and hybrid approaches. Physics-based approaches rely on understanding the physical processes of asset degradation, while data-driven approaches utilize machine learning and statistical modeling techniques to learn degradation patterns from monitoring data. Hybrid approaches combine both approaches for modeling and analysis.

A data-driven prognostics framework is presented, covering steps like data acquisition, preprocessing, model development, prognostic analysis, decision-making, and continuous monitoring. The P-F curve concept, representing the degradation trajectory and aiding in RUL prediction, is introduced.

Methods for data-driven prognostics are explored, including data preprocessing techniques like cleaning, imputation, outlier detection, and normalization. Dimensionality reduction techniques such as principal component analysis (PCA) and linear discriminant analysis (LDA) are explained. Model development encompasses machine learning techniques (e.g., random forests, gradient boosting, Gaussian processes) and deep learning architectures (e.g., convolutional neural networks, recurrent neural networks, deep belief networks, autoencoders). Hyper-parameter optimization methods like grid search, random search, particle swarm optimization (PSO), and genetic algorithms (GA) are discussed. The chapter concludes by providing an overview of public datasets for prognostics research, mentioning datasets like the CWRU Bearing Dataset, NASA FEMTO Bearing Dataset, MIT-Stanford Lithium-ion Battery Dataset, and NASA Turbofan Engine Degradation Simulation Dataset.

In summary, the chapter comprehensively covers predictive maintenance research,

including application areas, methodologies, and challenges. It underscores the significance of data-driven approaches and proactive maintenance strategies, highlighting their potential benefits. Table 2.3 presents application areas, methods and tools, and public datasets for this review chapter.

Table 2.3: Summary of the Literature Review.

Application	Method	Tools	Preprocessing	Datasets	Ref.
RUL prediction	CNN	Deep CNN w/ 1D Conv.	Min-max scaling	C-MAPSS	[58]
Fault Detection and Classification	RNN	GRU	Tri-domain features	PHM2010 Ds. CWRU Bearing Ds.	[42]
RUL prediction	CNN	Multiscale CNN w/ skip connections	Wavelet transform Dimension reduction	PRONOSTIA	[46]
RUL prediction	CNN	Multiscale CNN w/ skip connections	Short-time Fourier transform	PRONOSTIA	[14]
RUL prediction	Hybrid	Self-attention LSTM	Hand crafted Standard Scaler	C-MAPSS	[39]
RUL prediction	Hybrid	RBM and LSTM	Standard Scaler	C-MAPSS	[59]
RUL prediction	RBM	RBM	FFT	NASA Spiral Bevel Gear Damage Ds.	[13]
RUL prediction	RNN	Bi-LSTM	Standard Scaler	C-MAPSS	[60]
RUL prediction	RNN	LSTM		PRONOSTIA	[7]
RUL prediction	AE	Stacked DAE Transfer Learning	FFT HHT Min-max scaling	PRONOSTIA	[27]
RUL prediction	RNN	Bi-LSTM in series	Standard Scaler	C-MAPSS	[6]
RUL prediction	Hybrid	RNN and Attention	Five band-pass energy values of frequency spectrum	PRONOSTIA	[47]
RUL prediction	RNN	Deep LSTM	Exponential data smoothing	C-MAPSS	[61]
RUL prediction	CNN	Double CNN	Min-max scaling	PRONOSTIA	[15]
RUL prediction	AE	AE Deep ANN	Tri-domain features	PRONOSTIA	[48]
Continued on next page					

Table 2.3: Continued.

Application	Method	Tools	Preprocessing	Datasets	Ref.
RUL prediction	Hybrid	AE Bi-RNN Similarity based Curve Matching		C-MAPSS	[62]
RUL prediction	GAN	Domain AdvNN LSTM	Min-max Scaling	C-MAPSS	[63]
RUL prediction	CNN	Deep sep. CNN Residual conn.	Standard Scaler	C-MAPSS	[23]
RUL prediction	ANN	AE Deep ANN	Min-max scaling	NASA Battery Ds.	[34]
Fault Detection and Classification	Hybrid	CNN Transfer Learning	Signal to image conversion	CWRU Bearing Ds.	[45]
RUL prediction	DNN	DNN		NASA Battery Ds.	[54]
Uncertainty Estimation	CNN	Multiscale CNN Bi-LSTM		PRONOSTIA C-MAPSS	[49]
Uncertainty Estimation	CNN	Recurrent CNN		PRONOSTIA	[32]
RUL prediction Maintenance Decisions	RNN	LSTM	Min-max Scaling	C-MAPSS	[57]
Fault Detection and Classification RUL Prediction	Hybrid	Joint-loss CNN Multi-output CNN		IMS Bearing Ds.	[10]
RUL prediction	CNN	Multiscale CNN	Min-max scaling	C-MAPSS	[17]
RUL prediction	CNN	CNN	Spectrum principal energy-vector	PRONOSTIA	[69]
RUL prediction	Hybrid	Temp.Self Att. Bi-GRU	Moving average filter	C-MAPSS	[40]
RUL prediction Uncertainty Estimation	Hybrid	LSTM and CNN	Min-max scaling	C-MAPSS	[64]
RUL prediction	RNN	Bi-GRU	Min-max scaling	C-MAPSS	[93]
RUL prediction	RNN	Multi-output LSTM	Standard Scaling	C-MAPSS	[94]
RUL prediction	CNN	CNN	Standard Scaling	PRONOSTIA	[79]
RUL prediction	RNN	LSTM		NASA Battery Ds.	[53]
RUL prediction	DBN	DBN		PRONOSTIA	[95]
Continued on next page					

Table 2.3: Continued.

Application	Method	Tools	Preprocessing	Datasets	Ref.
RUL prediction	Transfer Learning	Deep metric transfer learning	Min-max Scaling	PRONOSTIA	[96]
Capacity estimation	ELM	ELM		NASA Battery Ds.	[51]
Degradation trajectory	GAN	Deep AdvNN		PRONOSTIA XJTU-SY	[41]
RUL prediction	RNN	Multicellular LSTM		C-MAPSS	[97]
Fault Detection and Classification	CNN	CNN		MaFaulDa Ds. CWRU Bearing Ds.	[44]
RUL prediction	CNN	Dilated CNN		MIT-Stanford Li-ion Battery Ds.	[55]
RUL prediction	Hybrid	Transfer learning LSTM		C-MAPSS	[98]
RUL prediction	RNN	LSTM		Baidu Ds. Backblaze Ds.	[38]
Degradation trajectory	RNN	LSTM		RWTH Ds.	[52]
RUL prediction	Hybrid	Hierarchical Att. Graph CNN		C-MAPSS PHM2010 Ds.	[50]
RUL prediction w/ partial sensor malfunctions	CNN	CNN		C-MAPSS PRONOSTIA	[99]
RUL prediction	RNN	Bi-LSTM		XJTU-SY Ds.	[84]
RUL prediction	RNN	Bi-LSTM Transfer learning		C-MAPSS	[100]

3. PRELIMINARY STUDIES

This chapter presents the preliminary studies we conducted as a part of research thesis focused on industrial systems. The chapter encompasses three distinct case studies on PHM. The first case study delves into the critical domain of automobile brake system leakage detection, optimizing brake liquid filling process. The second case study focuses on the experimental filtration system, aiming to predict its remaining useful life, thereby enabling proactive maintenance strategies and cost-effective operational planning. Finally, the third case study explores fault detection and classification in robotic test benches, addressing the need for efficient diagnostics and improved performance in industrial automation.

3.1. Automobile Brake System Leakage Detection

Automotive production, a prominent sector within Industry 4.0, demands stringent quality standards [101]. In order for automobiles to operate effectively, it is necessary to have several essential fluids, including brake fluid, steering fluid, and radiator coolant. During the manufacturing process, these fluids are integrated into automotive systems through a specialized device known as "The Filling Machine." By incorporating filling machine into their production lines, automotive manufacturers collect detailed data on the filling process for each chassis. In this case study, we support the filling operator during the filling process.

3.1.1. Introduction and Problem Definition

During the filling process, the operator is responsible for handling the hoses and operating the filling machine. The operator closely monitors the pressure and fluid levels displayed on the machine's interface to determine the success of the filling operation. The filling process comprises sequential vacuuming and filling steps, which is called a cycle. Typical operational procedure for these steps are as follows:

Vacuuming: Prior to initiating the filling process, the brake or coolant system

is subjected to vacuuming. This step removes air from the system, facilitating the thorough distribution of the filling fluid. The desired outcome is for the air pressure to decrease to 3 millibars within 50-60 seconds. If the pressure fails to reach the required level, it indicates a leakage in the system, rendering it defective. Based on the severity of the leakage, the operator may choose to direct the automobile to the repair station for necessary fixes. After the initial vacuuming, the operator waits for 5 seconds, during which the pressure should not increase by more than 0.5 millibars. If needed, the operator addresses any issues with joints or bolts on the spot based on their expertise and then repeats the vacuuming procedure. Filling step starts when the operator is satisfied with the final vacuum pressure.

Filling: During the filling step, approximately 600-700 ml. of brake fluid are added. If any pressure fluctuations occur during the vacuuming process, filling is performed to visually check for leaks in the connections and bolts. The operator applies pressure to the joints and bolts to determine if the issue can be resolved at the current stations or if further repair is necessary.

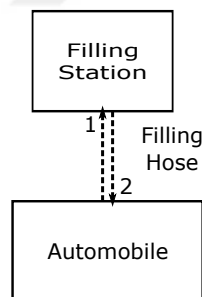


Figure 3.1: The Filling Process: Through the filling hose, (1) air is vacuumed off, (2) then fluid is pumped in.

The filling process is monitored by sensors, with one sensor measuring the pressure and the other monitoring the fluid level. In this investigation, we examined the readings from the filling machine sensors to identify any faults (specifically, leakage) in the brake fluid system. The goal was to assist the operator in making prompt decisions regarding the success or failure of the filling process. The system's sensor data and the operational data as listed in Table 3.1. The initial dataset has 12,151,666 readings for

51,250 unique car chassis. Average number of reading per successful cycle is 140.

Table 3.1: Features in the Filling Dataset.

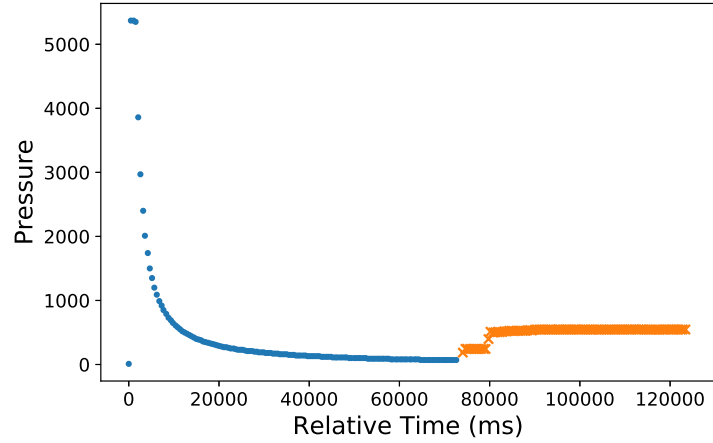
Operation	'vacuum' or 'fill'
Time	Timestamp for the sensor reading
Machine ID	Filling machine ID
Chassis	Chassis number
Vacuum pressure	Pressure reading during vacuuming
Fill amount	Volume of the filling fluid
Fill pressure	Pressure reading during filling

Figure 3.2 shows two example cycles from the dataset. In the figure, '.' marks *vacuum pressure*, and 'x' marks *fill pressure*. 3.2a shows a successful cycle of the processes. On the other hand, 3.2b shows unsuccessful filling attempts: the car has entered the process for more than once in approximately 17 hours period. In this case study, our aim is to detect any leakage before the given times in the filling process, thus increase operator efficiency.

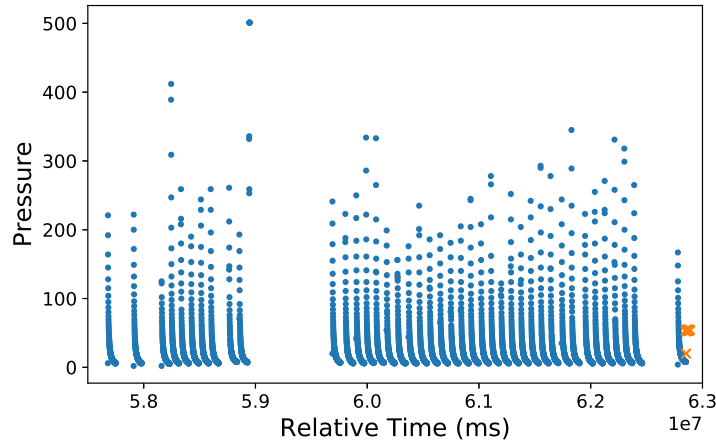
3.1.2. Methodology

The information obtained from the filling system is examined to anticipate potential leaks, determining the success or failure of the process. This prediction is treated as a classification task. The development of a classifier takes two distinct approaches: one involves extracting features and utilizing a machine learning-based classification algorithm, while the other involves directly employing a convolutional neural network for feature extraction during training.

The original dataset does not inherently involve any classification or regression tasks pertaining to the data or the filling machine. Initially, our objective was to create a classification task using this dataset. The raw dataset comprises 12,151,666 readings associated with 51,250 unique car chassis. Upon conducting a brief analysis, we discovered that the minimum time interval between two consecutive cycles is 2,000



(a) A successful filling cycle.



(b) Failing filling cycles.

Figure 3.2: Sample chassis readings from the filling dataset.

milliseconds. This information proved valuable in generating features from the time series data, specifically the variables "vacuum pressure," "fill amount," "fill pressure," and a corresponding label. Further analysis revealed that, on average, each chassis had around 140 readings for "vacuum pressure" and approximately 110 readings for both "fill amount" and "fill pressure." To condense the data, we serialized each cycle into 200 readings for "vacuum pressure," 130 readings for "fill amount," and 130 readings for "fill pressure," resulting in a total of 460 features for each cycle.

Every cycle is assigned a label based on the following criteria: If additional cycles are observed in the future for a specific car chassis, the current cycle is labeled as 1, indicating a leakage or failure. On the other hand, if no further cycles are observed, the

cycle is labeled as 0, indicating a successful operation. The resulting dataset consists of 53,254 samples, with 51,250 samples labeled as negative (96.23%) and 2,004 samples labeled as positive (3.77%).

3.1.3. Experiments

To tackle the classification problem, we employed various classical machine learning techniques such as the random forest classifier, gradient boosting classifier, and Gaussian process classifier. Additionally, we developed deep learning models using convolutional neural network (CNN) architectures. Brief description of ML methods are given in Section 2.3.2.

Grid search is used to calculate best set of parameters for random forest, and gradient boosting classifiers. The default model parameters and grid search suggested model parameters are show in 3.2.

Table 3.2: Hyper-parameters for RF and GB models.

Model	Defaults	Grid Search
Random Forest	max features: 'auto', estimators: 10, bootstrap: True, max depth: None	max features: 'sqrt', estimators: 500, bootstrap: False, max depth: 10
XGBoost	base score:0.5, learning rate:0.1, estimators:100	base score:0.05, learning rate:0.01, estimators:500
CatBoost	learning rate: 0.03, iterations: 1000	learning rate: 0.01, iterations: 2000

In the case of the GP classifier, there were no parameters to be tuned. However, we encountered difficulties running the classifier using the existing feature space. To address this issue, we employed "principle component analysis" (PCA) to reduce

the dimensions. Specifically, we experimented with PCA using different numbers of components, namely 5, 15, 17, and 29, in order to find an optimal reduced feature space. CNN models are hand-crafted with different number of filters and layers.

Models are trained using stratified 5-fold cross validation, and their average is reported as the evaluation score. Stratified cross validation honors positive samples by distributing them evenly over folds.

A commonly used evaluation metric for classification problems is accuracy. However, upon performing data preparation and cycle labeling, it became evident that only 2004 out of 53,254 samples (3.77%) were labeled as positive. This severe class imbalance poses a challenge. Relying solely on accuracy as an evaluation metric in such imbalanced datasets can be misleading. In our research, we not only reported accuracy but also considered alternative evaluation metrics for better assessment. We specifically looked into the "area under the curve" (AUC) metric [102], and the "Matthews correlation coefficient" (MCC) [74]. These metrics offer more reliable and informative measures of classification performance, particularly in the context of imbalanced datasets.

3.1.4. Results and Discussion

Evaluation results for classical ML and CNN models are shown in Table 3.3 and Table 3.4 respectively.

Gradient boosting methods outperform both Gaussian process classifiers, with a performance advantage of around 5-7%, and random forest classifiers, with a smaller advantage of 1-2%. In the comparison between XGBoost and CatBoost, CatBoost demonstrates a slightly superior performance, with a difference of less than 0.3%. This suggests that increasing the number of residual trees could potentially improve the classification results. The grid search analysis provides supporting evidence for the effectiveness of CatBoost and random forest classifiers, but not for XGBoost. While Gaussian process is a powerful technique, its efficiency and runtime performance are subject to debate. Due to its runtime and memory requirements of $O(n^3)$, Gaussian

Table 3.3: Random forest and gradient boosting results for leakage detection.

Method		ACC	AUC	MCC
Random Forest	defaults	0.9915	0.9354	0.8821
	grid search	0.9923	0.9476	0.8948
XGBoost	defaults	0.9925	0.9501	0.8965
	grid search	0.9928	0.9497	0.9004
CatBoost	defaults	0.9929	0.9522	0.9024
	grid search	0.9930	0.9522	0.9038
GPC w/ PCA	5 components	0.9861	0.9368	0.8201
	15 components	0.9876	0.9389	0.8372
	17 components	0.9883	0.9503	0.8490
	29 components	0.9893	0.9392	0.8571

process did not perform well for this particular dataset.

In this particular study, we conducted an analysis using data collected from sensors on filling machines to identify potential leakage in the brake fluid systems of automobiles during production. After performing initial preprocessing steps on the data, we applied classical machine learning methods to generate reliable predictions for leakage detection. Specifically, we employed the random forest classifier, gradient boosting classifier, and Gaussian process classifier algorithms for the binary classification task of determining success or failure. Our findings revealed that the gradient boosting methods outperformed the other two algorithms in terms of predictive accuracy.

Table 3.4: CNN architecture results for leakage detection.

Method	Architecture	ACC	AUC	MCC
Model1	30C45-S2-BN-30C35-S2-BN-Dx0.75-F150-F1	0.9915	0.9445	0.8834
Model2	30C45-S2-BN-30C35-S2-BN-Dx0.75-F150-F1	0.9927	0.9473	0.8983
Model3	30C45-S2-30C35-S2-BN-Dx0.75-F100-F1	0.9925	0.9474	0.8957
Model4	50C75-S2-BN-50C45-S2-BN-Dx0.75-F150-F1	0.9916	0.9386	0.8828
Model5	15C7-S2-BN-15C5-S2-BN-15C3-F100-F50-F1	0.9867	0.9504	0.8394

Notation:

$x\text{C}y$: Conv1D w/ x filters of size y ; $\text{S}z$, MaxPooling w/ pool size z ;

BN: Batch Normalization; $\text{D}x\text{d}$: Dropout w/ d ;

$\text{F}h$: Fully connected with h nodes.

Furthermore, in addition to traditional machine learning approaches, we also trained several convolutional neural network (CNN) models. Although previous literature has demonstrated successful applications of CNN models in analyzing time series data, our CNN model did not perform as good as the GB algorithm. The lack of quality in the training data, particularly in terms of the number of samples available, likely hindered our ability to effectively train the model and extract meaningful features.

3.2. Experimental Filtration System Remaining Useful Life Prediction

Filtration systems have widespread usage across various engineering disciplines, such as automotive, chemical, nuclear reactor, and process engineering. Additionally, they find application in numerous industrial sectors like food, petroleum, pharmaceuticals, metal production, and minerals [103]. The primary objective of filtration systems is to ensure the smooth operation of the overall process, making them crucial in maintaining process efficiency [104]. PHM Europe Conference organizers challenged the researchers to showcase their abilities on an experimental filtration system's condition monitoring data for PHME 2020 Conference. In this case study, we provide our solution, which carried us to the 3rd place in the challenge.

3.2.1. Introduction and Problem Statement

The presence of contaminants in liquids can lead to clogging in filtration systems. The PHME Conference Data Challenge 2020 investigates this phenomenon by utilizing an experimental setup designed to simulate varying degrees of filter clogging caused by contamination. The experimental rig consists of several components, including liquid tanks, one stirrer, one pump, one pulsation dampener, one filter, pressure and flow rate sensors, and a data acquisition system, as depicted in Figure 3.3. The suspension used in the experiments contains polyetheretherketone particles mixed with water at different concentrations. These particles come in three sizes: small ($45 - 53\mu m$), medium ($53 - 63\mu m$), and large ($63 - 75\mu m$). The dataset includes sensor readings obtained at a frequency of 10 Hz for experiments conducted with different particle sizes and concentrations. In this context, the filtration system is considered to be clogged when the pressure drop between the upstream and downstream measurements exceeds 20 psi.

The challenge dataset is composed of experiments that have been split into training and validation subsets. There is also a separate private test subset that is reserved for evaluating the submissions to the challenge. Concentration of particles in the experiments is denoted by the variable *Solid Ratio*, while the particle size is

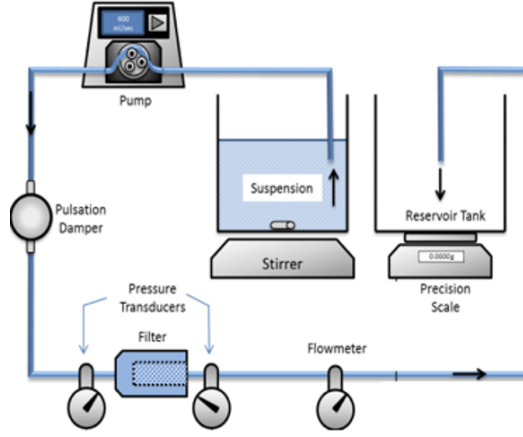


Figure 3.3: The experimental rig for PHME20 Data Challenge.

indicated by the variable *Particle Size* in the experiment metadata. Each combination of concentration and particle size is assigned a *Profile* in the dataset, which represents the operating conditions of the experiments. The training and validation subsets collectively consist of 32 experiments. Goal of the data challenge is to predict the remaining useful life of the training, validation, and test datasets at 10 seconds intervals.

3.2.2. Methodology

We have built a data processing workflow using Jupyter Notebook [105] to attack the Data Challenge. The workflow contains data preprocessing, model training, and evaluation phases as shown in Figure 3.4. We have executed preprocessing steps such as initial RUL assignment, operating conditions (profile) assignment, scaling etc. before the training phase.

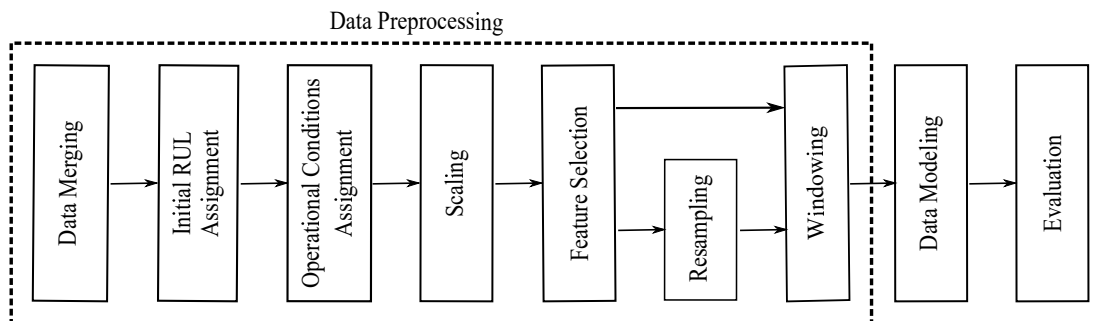


Figure 3.4: Data processing workflow for clog detection.

Data Merging: Initially, the dataset consisted of separate data files for each

experiment. To facilitate analysis, we merged these individual data files, resulting in two distinct datasets for training and validation purposes. As part of the merging process, we incorporated the experiment setups associated with each reading sample for subsequent processing. After this step, merged dataset includes the following information: *ExperimentID*, *ReadingID*, *Time*, three experiment setups, readings from three sensors, and the *Pressure Drop* variable.

RUL Assignment: RUL assignment refers to the process of assigning RUL labels to the samples in the dataset. Initially, the dataset does not include any ground truth RUL values for the experiments. However, since we were aware that the experiments were conducted until failure occurred, we were able to assign RUL values to each sample.

The two most commonly used strategies for RUL assignment are Linear RUL Assignment and Piecewise Linear RUL Assignment. In Linear RUL Assignment, the maximum RUL value corresponds to the length of the sensor reading sequence. The RUL value decreases by one with each subsequent reading until it reaches zero at the end. It's worth noting that the sensor readings may not end with a clogging failure, so negative RUL values were also assigned in these cases. On the other hand, Piecewise Linear RUL Assignment (PwL) suggests that the RUL remains constant until degradation occurs during the experiment. Once a fault is detected, the RUL starts to decrease. In our experimentation, we explored initial RUL values of 100, 125, and 150 to determine the most appropriate approach.

As part of the challenge, we have introduced a new RUL assignment strategy. This strategy involves identifying the faulty timestamp using a heuristic. We have observed that the *FlowRate* variable remains relatively constant with some added noise. Based on this observation, we aimed to pinpoint the timestamp at which this behavior changes, indicating the start of degradation. To achieve this, we utilized the *FlowRate* values between timestamps 400 and 1400, which were chosen via visual inspection. We estimated a line representing the nominal behavior using these values and calculated the intersection between the *FlowRate* and the line at the furthest timestamp. This intersection point is assumed to indicate the initiation of degradation, as illustrated in

Figure 3.5. Before and after this intersection point, we employed the linear RUL model for RUL assignment. The initial RUL was assigned from the start of the experiment until the estimated fault timestamp, while the remaining RUL was assigned from the fault timestamp until the end of the experiment. We have named this assignment strategy as RUL_{Fault} . A visualization of the various RUL assignment strategies, including the RUL_{Fault} strategy, is depicted in Figure 3.6.

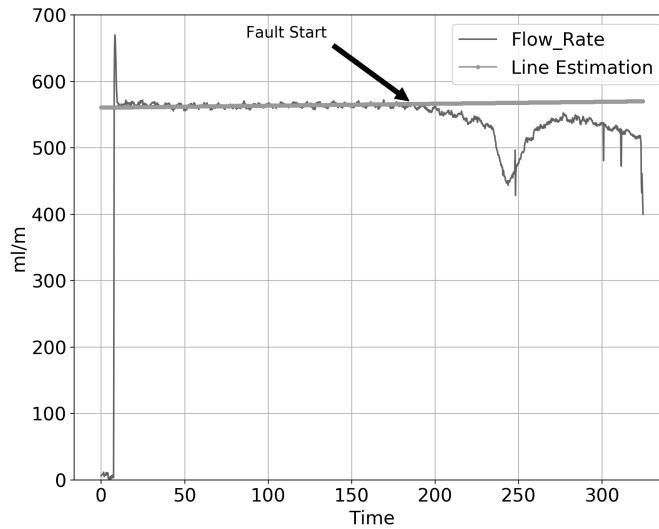


Figure 3.5: Fault estimation for PwL_{Fault} RUL assignment method.

Operational Conditions Assignment: The training and validation subsets contain a variable called *Profile* that describes the experiment setups. As these subsets have distinct profiles, we needed to establish a schema for assigning operational conditions. To accomplish this, we employed the K-Means clustering algorithm, utilizing the *SolidRatio* and *ParticleSize* features. This allowed us to assign operational conditions ($KmeansProfile$) to each sample in the dataset.

Scaling: Grouping the dataset by $KMeansProfile$, we performed scaling on the data using StandardScaler from Scikit-Learn Library [106]. First, the training data is scaled, and then the scaling parameters are applied to validation data.

Feature Selection: We used the following features throughout our experiments: *FlowRate*, *UpstreamPressure*, *DownstreamPressure*, *PressureDrop*, *ParticleSize*, *SolidRatio*,

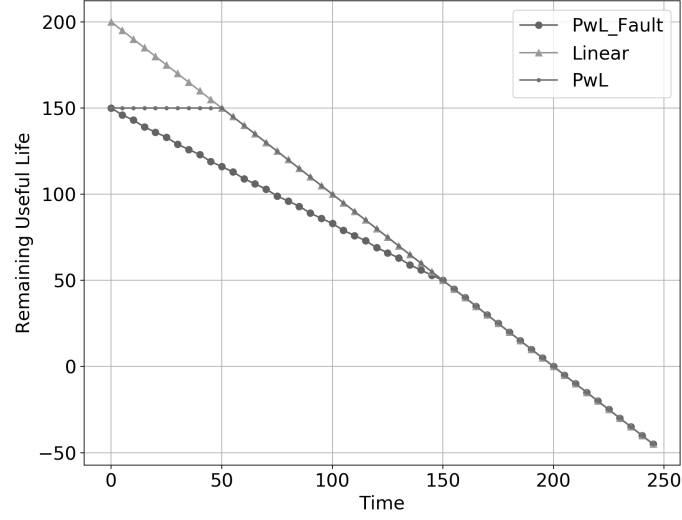


Figure 3.6: Linear, PwL, and *PwL_Fault* RUL assignment strategies.

and *KmeansProfile*.

Resampling: The runtime performance of GP is $O(n^3)$, where n is the number of samples in the dataset. In order to achieve reasonable training times, we resampled input data to d Hz, where $d \in \{1, 0.5, 0.33, 0.25, 0.2\}$, using $5/d$ strides. Initial results showed that $d = 0.2$ Hz gives reasonable training times. Resampling is used for GP only.

Windowing: During the model training phase, we incorporated window sizes of 10, 15, 20, 30, and 50 to create a contextual representation of the time series data. By applying these window sizes, we transformed the dataset into windowed segments, which were then used for model training.

3.2.3. Experiments

We have used RF, GB, and GP regression algorithms to estimate the useful life of the experimental filtration system. We have used and reported mean absolute error (MAE) in our experiments. Brief description of ML methods and evaluation metrics are given in Section 2.3.2.

Following the data preprocessing stage, we proceeded to train our data models using the transformed data. We utilized the model parameters listed in Table 3.5.

These specific parameter values for RF were obtained through a grid search procedure, whereas default values provided by the respective implementations were used for the other models.

Table 3.5: Model parameters for PHME20 models.

Algorithm	Model Parameters
RF	max features: 'auto', estimators: 100, bootstrap: True, max depth: 10, min samples leaf: 2, min samples split:2
CB	learning rate: 0.03, iterations: 1000
GP	Kernel: Rational Quadratic length scale: 1.0, alpha: 0.1

3.2.4. Results and Discussion

The Data Challenge requires the use of four models, depending on the percentage of data allocated for training. In our analysis, we specifically utilized the training and validation subsets separately, focusing solely on the required percentage of training experiments. For each case, we reported the Mean Absolute Error (MAE) scores. During our experimentation, we observed that the MAE score decreased as the initial RUL for Piecewise Linear RUL Assignment (PwL) decreased. Based on this observation, we made a decision to report PwL results only for the initial RUL value of 150. The results of our analysis are presented in Tables 3.6-3.9, respectively. In these tables, best results are given in **bold**.

Table 3.6: MAE scores using 25% of the training data.

		Window Size				
		10	15	20	30	50
RF	Linear	8.65	7.79	8.43	8.39	8.72
	PwL	5.98	6.46	6.07	6.06	6.48
	PwL_Fault	4.89	4.99	5.08	4.71	4.76
CB	Linear	8.27	7.82	7.17	7.01	8.14
	PwL	5.12	4.92	4.92	4.80	5.08
	PwL_Fault	4.63	4.05	3.96	3.76	3.79
GP	Linear	12.42	12.20	14.18	13.73	7.11
	PwL	11.01	9.69	9.38	5.88	2.19
	PwL_Fault	7.14	6.56	7.25	6.98	3.75

Table 3.7: MAE scores using 50% of the training data.

		Window Size				
		10	15	20	30	50
RF	Linear	7.81	7.07	7.97	8.10	8.12
	PwL	4.94	5.41	4.66	5.21	5.23
	PwL_Fault	4.21	4.37	4.36	4.30	4.30
CB	Linear	9.04	8.06	8.02	7.74	7.68
	PwL	4.64	4.63	4.61	4.47	4.54
	PwL_Fault	4.53	4.31	3.87	3.99	3.95
GP	Linear	10.75	9.71	9.28	8.94	6.08
	PwL	7.87	6.95	5.91	3.64	1.66
	PwL_Fault	5.77	5.54	5.14	4.29	2.96

Table 3.8: MAE scores using 75% of the training data.

		Window Size				
		10	15	20	30	50
RF	Linear	6.97	6.94	8.59	7.81	6.95
	PwL	4.56	4.63	6.16	5.20	4.86
	PwL_Fault	4.25	4.19	5.53	4.64	3.98
CB	Linear	8.72	7.61	9.31	8.63	8.79
	PwL	4.87	4.62	6.12	4.95	5.66
	PwL_Fault	4.41	4.45	5.48	4.74	4.56
GP	Linear	9.39	8.58	8.38	7.58	5.17
	PwL	6.61	5.35	4.68	2.76	1.02
	PwL_Fault	4.35	4.13	3.89	3.23	2.18

Table 3.9: MAE scores using full training data.

		Window Size				
		10	15	20	30	50
RF	Linear	7.45	7.12	8.07	7.52	7.38
	PwL	4.57	4.72	5.43	5.29	4.85
	PwL_Fault	4.52	4.22	4.88	4.39	4.20
CB	Linear	9.18	8.37	9.01	8.51	8.90
	PwL	4.79	4.62	5.55	4.69	5.53
	PwL_Fault	4.61	4.36	5.26	4.62	4.34
GP	Linear	9.32	9.07	8.56	8.00	5.77
	PwL	6.05	5.62	4.70	2.79	1.00
	PwL_Fault	4.30	3.86	3.72	3.12	2.10

Interestingly, it was observed that with an increase in the amount of training data, CatBoost achieved better results at smaller window sizes. Specifically, the best scores

were obtained using a window size of 15, as opposed to the previously used window size of 30 in earlier experiments. In contrast, Gaussian process (GP) consistently achieved the best results with a window size of 50 in all cases.

One requirement of the challenge was strict size limit, which imposed 6 MB size on the models. The size of the random forest (RF) model increases as the number of trees in the ensemble and the depth of these trees grow. In our experiments, it was found that four RF models alone occupied a significant space of 28 MBs. It is worth noting that the MAE scores for smaller models were up to three times worse than the reported scores in this study. It is important to mention that the Scikit-Learn implementation of GP stores both the training samples and the covariance matrix within the model. Consequently, the size of GP models increases with the number of samples present in the dataset. Although we used resampling of the dataset, four GP models are 14.3 MBs in total. CB models are about 1 MBs each, so we were able to submit our CB models.

Submissions to the data challenge are evaluated using public training and validation datasets, and a private test dataset. Our CB models, which ranked 3rd in the data challenge, scored 86.74 using the challenge’s penalty score.

In this section, we conducted an analysis as part of the PHME20 Data Challenge, which required predicting the RUL of an experimental filtration system under different experiment setups. In our study, we developed a data pipeline and compared the performance of random forest (RF), gradient boosting (CatBoost), and Gaussian process regression (GP) algorithms. Notably, the Gaussian process regressor consistently outperformed the other algorithms across all experiments. Additionally, we introduced a new RUL assignment strategy called *PwL_Fault*, which is based on a heuristic approach. *PwL_Fault* consistently outperformed the Linear RUL assignment strategy in all cases. Furthermore, while performing the best, GP does not yield any smaller model size.

3.3. Robotic Test Bench Fault Detection and Classification

Real-time condition monitoring of sensors and the recognition of abnormal patterns play a crucial role in the fault diagnosis of machine tools. Identifying the root

cause of the failure requires preprocessing, classification, and diagnosis of the data before proceeding to the next stage. The primary objectives of fault detection and diagnosis are to develop an effective indicator capable of identifying faulty process states and to take appropriate actions to prevent future failures. This indicator provides a prediction of the specific fault and forecasts the remaining operational time of the system based on the information obtained from condition monitoring. Competitors in the PHME21 data challenge are invited to demonstrate their skills on a real-world industrial robotic testbed in a manufacturing production line setup. In this case study, we provide our solution, which carried us to the 4th place in the challenge.

3.3.1. Introduction and Problem Definition

The objective is to examine the dataset provided by the Swiss Centre for Electronics and Microtechnology (CSEM) in order to identify faults within the robotic test-bench. The robotic test-bench in Figure 3.7 is comprised of various subsystems, including conveyor belt motors, an infrared camera, and robotic arms. These components are utilized in the continuous testing process of electronic components on a large-scale quality-control pipeline. The primary purpose of the test-bench is to assess the functionality of electrical fuses. It involves a 4-axis SCARA robot equipped with a vacuum gripper, which picks up electrical fuses from a feeder and transfers them to the fuse test-bench. At the test-bench, the fuse's electrical conductivity is checked, and if confirmed, the fuse is subjected to a 1.5-second interval of 200mA current to induce heating. The heating process is monitored using a thermal camera. Once testing is completed, the fuse is returned to the feeder via two conveyor belts.

The experimental dataset consists of 50 sensor readings that track the changes in various quantities of interest to monitor the real-time health condition of the machine. To reduce the data size, it is necessary to calculate statistical data points for each window and sensor. Furthermore, each signal is linked to a specific set of fields that characterize the features extracted from that signal through an automated data acquisition process. For every 10-second time window, the signals are described by the statistical features

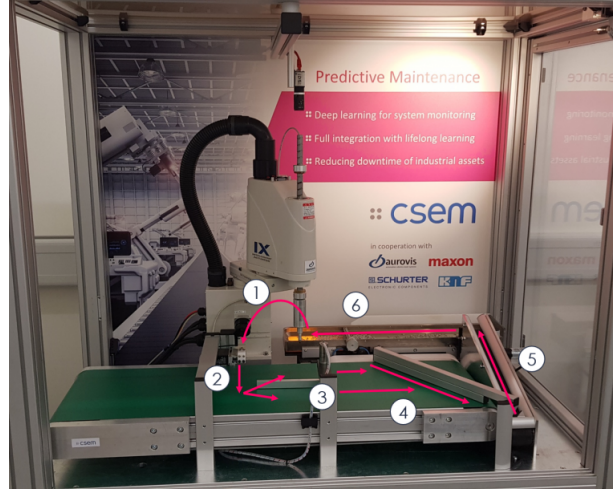


Figure 3.7: The Fuse Test Bench for PHME21 Data Challenge.

listed in Table 3.10. In the end, the dataset has 247 statistical features derived from 50 sensors.

Table 3.10: Statistical features derived from test-bench sensors.

Name	Description
Cnt	Number of samples in the time window.
Freq	Sampling frequency.
Max	Maximum value of the samples.
Min	Minimum value of the samples.
Std	Standard deviation of the samples.
Trend	Derivative-based trend of the samples.
value	Mean value of the samples.

An experiment on the test bench may run 1 to 3 hours. Experiments have been generated with a variety of seeded faults under controlled conditions. At the same time, some experimental data have been acquired under fault-free operating conditions. Data challenge dataset contains experiments divided into training and validation and model refinement subsets. In the evaluations of the submissions, a private test subset is used. Training and validation subsets contain 70 experiments, including fault-free experiments (Class 0) and 5 faulty classes (Classes 2, 3, 5, 7, and 9) in total. Model

refinement subsets contain 29 experiments, including fault-free experiments (Class 0) and 3 faulty classes (Classes 4, 11, and 12).

Fault-free experiments, which have Class 0, represent the behaviour of the machine during its normal operating regime. Unhealthy experiments which are characterized by anomalous behaviour, have been labeled with 8 different labels. Each fault is characterized by an anomalous behaviour of one or more signals.

The main objective of the challenge is identification and classification of the faults in unlabeled data. The methods how we handle challenge are given in the next section.

3.3.2. Methodology

We have built a data processing workflow, including data preprocessing, model training, and evaluation phases, to satisfy classification objective. The dataset is prepared for training and evaluation using the following steps:

Data Merging: The dataset consists of separate data files for each experiment. These individual data files were combined to create three distinct datasets: one for training and validation (consisting of 70 experiments) and two for model refinement (consisting of 29 experiments). To facilitate further processing, experiment setups were included for each reading sample. The merged training and validation subsets, as well as the model refinement dataset, contain readings from various sensors. The specific set of fields associated with each sensor were treated as features.

Imputation of Missing Values: In each experiment, the dataset exhibits instances of missing data. Since missing data can pose challenges in data analysis, we employed interpolation techniques to fill in the missing values. This approach helps to mitigate the potential issues associated with cases that contain missing values.

Data Scaling: Prior to data modeling, data scaling is conducted using the `RobustScaler` function from the Scikit-Learn Library [106]. During the scaling process, `RobustScaler` eliminates the median and scales the data based on the specified quantile range, typically set as the range between the 1st quartile and the 3rd quartile. This approach ensures that nominal data is smoothly transformed, while outliers are ap-

appropriately handled. The scaling procedure initially applies to the training data, and subsequently, the scaling parameters obtained are also applied to the validation data.

Feature Selection: For selecting the features to be utilized during model training, we employed the Leave One Feature Out Importance (LOFO-Importance) package [107]. This package evaluates the contribution of each feature to the learning task by iteratively excluding one feature at a time. The LOFO-Importance calculation considers the balanced accuracy metric, which effectively handles classification tasks on imbalanced datasets. Through this process, we identified the most relevant features for our model training.

Windowing: In order to provide a contextual representation for the time series data, we employed window sizes that are multiples of 5, ranging from 5 up to 50. This windowing technique divides the dataset into segments or windows of fixed length, capturing a specific temporal context. The windowed dataset was then utilized during the model training phase, allowing the models to learn from these contextualized segments of the time series data.

During the modeling stage, we employed tree-based ensemble learning algorithms, specifically gradient boosting, to develop our predictive models. Additionally, we utilized linear discriminant analysis as both a dimension reduction technique and a classifier. In addition to single classification models, we implemented Gaussian process models in two-phase classification processes. For the gradient boosting methods, we conducted hyper-parameter optimization using genetic algorithms. This approach allowed us to fine-tune the parameters of the gradient boosting models, optimizing their performance and ensuring the best possible results.

Our gradient boosting (GB) and linear discriminant analysis (LDA) models are constructed as pipelines, which consist of a scaler, a dimension reducer, and a classifier. Specifically, our two-phase classification model utilizes two separate pipelines for each phase. In the first phase, we employ eight one-vs-one classifiers, where each classifier is trained to distinguish between Class 0 and one of the faulty classes (represented by N, where N corresponds to each specific faulty class). This phase incorporates

Gaussian process (GP) modeling to predict the probability distribution for each possible classifier. By leveraging these predicted probability distributions, we gain insights into the likelihood of each class. In the second phase, we utilize LDA on the concatenation of these probability distributions. This enables us to make predictions regarding the true class labels by considering the combined information from the first phase. Overall, our pipeline-based approach involving GB, LDA, and the two-phase classification model enables us to effectively classify the data by leveraging the strengths of each technique and incorporating probabilistic predictions.

Since the dataset is imbalanced, evaluation of the model is performed using F1 score (F1), and Matthews correlation coefficient (MCC) [74] metrics in a 3-fold cross validation setup.

To evaluate and compare the performance of our models, we employed cross-validation scores. This allowed us to assess the effectiveness of the models across different subsets of the data. During the test experiments, we adopted a simple yet effective approach for final predictions. We considered the most frequently occurring fault label as the final classification label for each experiment. This decision rule helped us determine the predominant fault class based on the predictions generated by our models. By utilizing cross-validation and selecting the most common fault label for final predictions, we ensured robust performance evaluation and consistent classification results for the test experiments.

3.3.3. Experiments

For the implementation of our classification workflow, we utilized Scikit-Learn and various other Python libraries. During our analysis with the training data, we observed that certain experiments belonging to Class 0, specifically experiments 6, 23, 35, 49, 54, 56, 74, and 83, had a negative impact on the classification performance. Given that Class 0 had significantly more experiments (16 times more) compared to other classes, we made the decision to simply remove these specific runs from the dataset. This step was taken to mitigate the detrimental effect of these experiments and

ensure a more balanced and representative dataset for our classification tasks.

At the initial stage, we conducted the classification using the default settings of the classification methods. We employed a 3-fold cross-validation approach, applying it to various window sizes within the dataset. During the evaluation process, we monitored the performance of the models using evaluation metrics such as F1 score and MCC. Additionally, we analyzed the models' performance through the use of confusion matrices (CMs), which provided insights into the models' ability to correctly separate and classify different classes. The F1 score and MCC metrics provided numerical indications of the overall performance of the models, while the CMs offered a visual representation of the models' performance in terms of class separation. As an example, Figure 3.8 illustrates the CMs obtained from a sample run of the XGBoost (XGB) model. By assessing the models' performance through these evaluation metrics and CMs, we gained a comprehensive understanding of their effectiveness in accurately classifying the different fault classes within the dataset. The CMs suggest that the model fails to separate Class 4, 5 and 7, and that these classes are confused with each other and with Class 0.

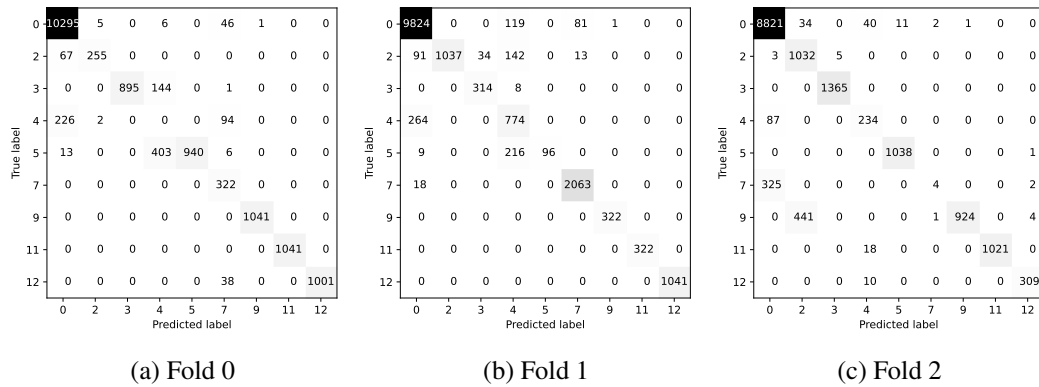


Figure 3.8: Confusion matrices per fold for fault classification.

In addition to the initial classification experiments, we conducted hyper-parameter optimization using Genetic Algorithms (GA) to further enhance the performance of the LightGBM (LGBM) and XGBoost (XGB) models. The GA optimization was employed to fine-tune the hyper-parameters of these models and improve their predictive

capabilities. The hyper-parameters for the GA evolutions were manually selected after several experimental runs. Table 3.11 presents the hyper-parameters used for the GA optimization process. These parameters were chosen carefully to facilitate effective optimization and identify the best model configuration. The hyper-parameters for the best-performing model were derived from these GA optimizations. This approach allowed us to identify the optimal combination of hyper-parameters, resulting in improved model performance and enhanced predictive accuracy.

Table 3.11: GA search parameters for hyper-parameter optimization.

GA Parameter	Value
Initial population size	30
Number of generations	4
Population size per generation	10
Mate probability	0.5
Mutation probability	0.5
Number of selected individuals per generation	3

3.3.4. Results and Discussion

The final results of the 3-fold cross-validation are summarized in Table 3.12. Notably, the two-phase classification approach (GP+LDA) yielded superior results when the window size was small (i.e., 5 or 10). This can be attributed to the classification power of Gaussian Process (GP), which effectively handles uncertainty and performs well with limited data. However, as the window size increased, other methods demonstrated better performance. While the gradient boosting (GB) implementations showed similar results across different window sizes, the XGBoost (XGB) model exhibited slightly higher MCC scores (approximately 0.01 to 0.02 points) compared to the LightGBM (LGBM) model. The linear discriminant analysis (LDA) scores fell between the scores of LGBM and XGB, indicating competitive performance. These

findings suggest that the choice of classification method should be tailored to the specific window size. GP+LDA may be more suitable for smaller window sizes, while GB models (especially XGB) are advantageous for larger window sizes. By considering these results, researchers can select the appropriate classification approach based on the window size and optimize the performance of the predictive maintenance models.

Table 3.12: Results of classification models for different window sizes.

Classifiers	ws=5		ws=10		ws=20		ws=30		ws=50	
	F1	MCC	F1	MCC	F1	MCC	F1	MCC	F1	MCC
LGBM	0.8	0.67	0.87	0.79	0.9	0.86	0.9	0.86	0.9	0.85
XGB	0.8	0.69	0.89	0.82	0.91	0.87	0.91	0.88	0.92	0.88
LDA	0.83	0.7	0.88	0.81	0.9	0.83	0.9	0.84	0.91	0.85
GP+LDA	0.89	0.82	0.9	0.84	0.87	0.77	0.49	0.43	0.54	0.41

The hyper-parameters and evaluation results are given in Table 3.13. Although GA optimization added 0.01 point to our previous results, we preferred to submit XGB model for window size 40.

Table 3.13: Model results after GA optimizations.

Classifier	Parameters	F1	MCC
LGBM	ws=49, learning rate=0.01, boosting type='gbdt', no of leaves=10, max depth=-1, no of estimators=300, min split gain=0, subsample=0.5, subsamplefreq=0, colsample=0.8, objective='multiclass'	0.91	0.88
XGB	ws=20, learning rate=0.001, booster='dart', max depth=40, no of estimators=750, min child weight=0.5, gamma=0, max leaves=10, subsample=0.8, colsample=0.8, objective='multi:softmax'	0.93	0.90

Our analysis for the PHME21 Data Challenge focused on detecting and classifying faults in a fuse test bench used for quality-control systems. To accomplish this, we developed a data pipeline and employed various classification algorithms, including gradient boosting with LGBM and XGB, linear discriminant analysis (LDA), and Gaussian process (GP). During the evaluation, we found that the two-phase Gaussian process classifier outperformed the other algorithms when working with smaller window sizes. However, as the window sizes increased, the performance of LGBM, XGB, and LDA classifications improved. Additionally, we conducted hyper-parameter optimization using genetic algorithms, resulting in a slight improvement of 0.01 points compared to our previous results. As a result of our efforts, our XGB-based solution achieved the 4th place ranking in the data challenge.

3.4. Summary

These preliminary studies serve as essential groundwork, providing valuable insights into the challenges and opportunities associated with each system. Through comprehensive analysis and experimentation, the chapter provides an overview of the preliminary studies conducted in the research thesis, emphasizing the significance of automobile brake system leakage detection, experimental filtration system remaining useful life prediction, and robotic test bench fault detection and classification. The subsequent sections delve into the methodologies employed, experimental setups, and results obtained, offering valuable insights that pave the way for the subsequent research chapters.

4. TURBOFAN ENGINE REMAINING USEFUL LIFE PREDICTION

This chapter presents our studies on NASA Turbofan Engine Degradation Simulation Dataset [91], aka C-MAPSS Dataset. The C-MAPSS dataset is widely recognized and utilized in the field of prognostics and predictive maintenance. The dataset plays a vital role in the development and evaluation of prognostic models and algorithms for its realistic representation, longitudinal data, multiple operating conditions, and standardized evaluation methods. In this chapter we introduce joint autoencoder-regressor (JAER) RUL prediction architecture in two versions, and possible improvements. We also introduce a fault detection approach for RUL labeling.

4.1. Introduction and Problem Definition

The C-MAPSS Dataset is a collection of run-to-failure data for turbofan engines, simulated using the C-MAPSS Software, a thermodynamic simulation model specifically designed for turbofan engines. This dataset is named after the software used for simulation. The simulation model for C-MAPSS Dataset encompasses 14 inputs and 21 outputs. The inputs consist of parameters such as fuel flow and 13 health parameters, which enable the simulation of faults and degradations in key rotating components of the turbofan engines. The 21 output parameters from the simulation capture the system response under different health states and operational conditions. The dataset is divided into four subsets: FD001, FD002, FD003, and FD004. Each subset corresponds to different operational conditions and health states. Within each subset, there are separate training and testing sets. These sets include 26 features recorded as time series data, including the unit number (EngineID), cycle time (Cycle), three operational parameters (Setting 1-Setting 3), and 21 output parameters (Sensor 1-Sensor 21). Although the training sets do not provide remaining useful life (RUL) labels, it is known that the data runs until failure. On the other hand, the test sets do not run until failure but do provide RUL labels. Further information about the C-MAPSS dataset can be found in Table 4.1.

The C-MAPSS dataset has become a standard benchmark for evaluating prognostic algorithms and techniques. Researchers can compare their models' performance against existing state-of-the-art methods, fostering collaboration, and advancement in the field. Our aim in this study is to develop a joint AE-Regression architecture for RUL prediction using the C-MAPSS dataset.

Table 4.1: C-MAPSS Dataset information.

		FD001	FD002	FD003	FD004
No of operational parameters		1	6	1	6
No of fault types		1	1	2	2
Train	No of engines	100	260	100	249
	Min no of cycles	128	128	145	128
	Max no of cycles	362	378	525	543
Test	No of engines	100	259	100	248
	Min no of cycles	31	21	38	19
	Max no of cycles	303	367	475	248

4.2. Proposed Framework

General framework for RUL prediction is given in 4.1. Similar to many machine learning-based methods, our approach follows a three-stage framework for RUL estimation. In the initial stage, we focus on data gathering and labeling. For data labeling, we introduce a novel empirical RUL assignment method that is specifically designed for this purpose. The second stage involves data preprocessing, where we apply normalization techniques to ensure the data is in a suitable format for further analysis and modeling. Lastly, in the third stage, we proceed with model training and evaluation. For this stage, we propose an end-to-end joint autoencoder-regressor model, which combines the capabilities of both an autoencoder and a regressor to achieve accurate RUL prediction. By following this framework, we aim to provide an effective and comprehensive solution for predicting the remaining useful life of the turbofan engines.

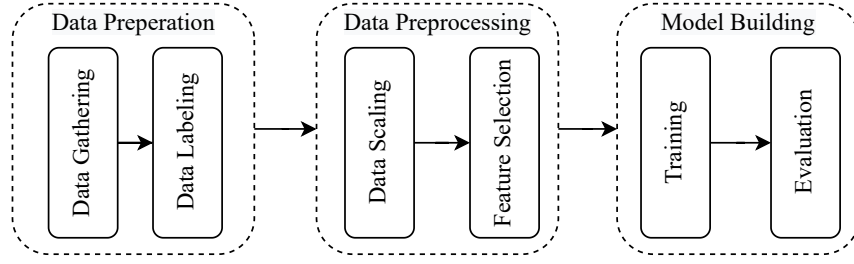


Figure 4.1: RUL prediction framework for C-MAPSS Dataset.

4.2.1. Data Labeling

In the initial phase of our workflow, we gather the required data and associated labels. The C-MAPSS dataset comprises sensor data in the form of time-series, while the labels represent indications of failure. It is crucial to convert these failure definitions into appropriate RUL labels. In the dataset, the engines fail at the last recorded measurement, and our goal is to determine the remaining useful lifetime of the system.

The most commonly used approaches for labeling Remaining Useful Life (RUL) are Linear RUL labeling (Linear) and Piecewise Linear RUL labeling (PwL) [98, 108–111]. In Linear RUL labeling, the maximum RUL value corresponds to the length of the sequence of sensor readings. The RUL value gradually decreases by one with each sensor reading until it reaches zero at the end. Specifically, for the i^{th} sensor reading, assuming all sensors are read at the same rate and time, the Linear RUL is defined as $RUL_i = C - i$, where C represents the number of samples until failure occurs. On the other hand, the Piecewise Linear RUL labeling (PwL) approach is defined by the formula presented in Equation 4.1. Here, RUL_{Max} is the empirically calculated maximum RUL value [109] obtained from the training dataset. Throughout this chapter, the labels obtained through Linear RUL labeling are referred to as Linear, while the labels obtained through PwL RUL labeling are denoted as $PwL\langle d \rangle$, where d represents RUL_{Max} for that particular label.

$$RUL_i = \begin{cases} RUL_{Max} & \text{if } i \geq RUL_{Max} \\ C - i & \text{otherwise} \end{cases} \quad (4.1)$$

Another approach to RUL assignment, known as fault detection-based RUL labeling, has gained attention in recent studies [112]. In this approach, the degradation or fault point of the n^{th} system, denoted as F_n , is predicted using anomaly detection techniques. Similar to the Piecewise Linear (PwL) approach, the RUL is assumed to be constant until the degradation point is reached. After the degradation point, the RUL decreases linearly over time according to Equation 4.2. This approach combines the benefits of fault detection techniques with a linear decrease in RUL to provide more accurate and dynamic RUL assignments.

$$RUL_i = \begin{cases} F_n & \text{if } i \geq F_n \\ C - i & \text{otherwise} \end{cases} \quad (4.2)$$

In this study, we propose an approach for RUL labeling based on fault detection, where the prediction of F_n is made through an empirical classification process. Our assumption is that by using a combination of normal samples from the beginning of the sequence and faulty samples from the end of the sequence, we can develop a classification model to determine whether a given sample is normal or faulty. While we may not have precise knowledge about which samples are normal or faulty, we can analyze the empirical evidence to establish a potential boundary. Specifically, we adopt an iterative approach where we assume that S_c samples (representing a certain percentage, denoted as $s\%$, of the data) from the start of the sequence are normal, while an equal number of samples from the end of the sequence are considered faulty. Our expectation is that training the model with this set of $2 \times S_c$ samples will provide the best interpretation of the remaining data (known as test data) and enable us to determine the position of F_n within this test data.

To determine the start of the fault point F_n , a fixed-size window is slid along the data starting from the first test sample, as illustrated in Figure 4.2. At each position of the window, we evaluate whether the trained model, based on empirical evidence, classifies the data on the left side of the window as non-faulty and the data on the right side as faulty. However, if the window is not positioned correctly, specifically to the

left of the fault location (as shown in the leftmost box in Figure 4.2), the classifier's response and the hypothesized labeling will not align. This sliding window approach allows us to identify the approximate location of the fault point by observing where the model's classification output aligns with the expected fault labeling. By iteratively moving the window and assessing the classifier's response, we can pinpoint the start of the fault point F_n and proceed with fault detection-based RUL labeling. In fact, while a perfect classifier for this window will give $[0,0,0,0,0,0,0,0]$ (assuming a window with 8 samples) instead of $[0,0,0,0,1,1,1,1]$. If the window is on the correct location, then the perfect classifier will result in $[0,0,0,0,1,1,1,1]$. Furthermore, when the window is on the right of the fault starting point, the perfect classifier will be $[1,1,1,1,1,1,1,1]$. In the context of a correct classifier, the divergence between the classifier's labeling outcomes and the anticipated results can be evaluated by employing MCC, as presented in Equation 2.5.

To determine the fault starting point using the empirical measure of the MCC, we propose an iterative approach involving the variation of two parameters: the classifier training sample size (S_c) and the test window location (α). This iterative process, outlined in Algorithm 1, aims to identify the optimal combination of S_c and α that yields the highest MCC score. By performing iterations on S_c and α , we generate a heatmap, represented by the `ScoreMatrix`, where the MCC scores are stored. The optimal S_c and α values are determined by identifying the maximum value within the heatmap's coordinates. To further refine the optimization, a suitable optimization algorithm such as a Gaussian Mixture Model [113] can be applied to calculate the mass center of the `ScoreMatrix` heatmap. This enables us to pinpoint the most favorable combination of S_c and α that maximizes the MCC score. The visualization of the `ScoreMatrix` heatmap and the resulting mass center is depicted in Figure 4.3, showcasing the effectiveness of the proposed optimization algorithm in identifying the fault starting point.

The construction of training and testing datasets in Algorithm 1 is visually represented in Figure 4.2. We make the assumption that normal samples, labeled as 0,

Algorithm 1 CalculateFn

```
1: function CALCULATEFN( $df, ws$ ) ▷  $df$ :data,  $ws$ :windows size
2:    $ScoreMatrix \leftarrow 0$ 
3:   for  $s = 5\%$  to  $40\%$  do
4:      $X_{train}, y_{train} \leftarrow CreateTrainingDataset(df, ws, s)$ 
5:      $Model \leftarrow Classifier().train(X_{train}, y_{train})$ 
6:     for  $\alpha = s + 1$  to  $100 - s$  do
7:        $X_{test}, y_{test} \leftarrow CreateTestingDataset(df, ws, s, \alpha)$ 
8:        $prediction \leftarrow Model.predict(X_{test})$ 
9:        $ScoreMatrix[s, \alpha] \leftarrow mcc\_score(y_{test}, prediction)$ 
10:    end for
11:  end for
12:   $F_n \leftarrow CalculateFnFromGMMComponents(ScoreMatrix)$ 
13:  return  $F_n$ 
13: end function
```

are located at the beginning of the sequence, while faulty samples, labeled as 1, are positioned at the end of the sequence. To create the training subset, we select a percentage, denoted as $s\%$, of the sequence from each side. The test subset is formed by using floating windows of size ws (determined through empirical selection) from the remaining data. In each iteration, a classifier is trained using the training subset. The trained classifier is then applied to predict the class for each window in the testing subset. Our assumption is that the window containing F_n will have the highest evaluation score.

Visualization of evaluation scores of the predictions in Algorithm 1 is given in Figure 4.3. The algorithm creates a heatmap as in (a). We calculate GMM components of evaluation scores (b), and take the component whose mean maximizes the evaluation score.

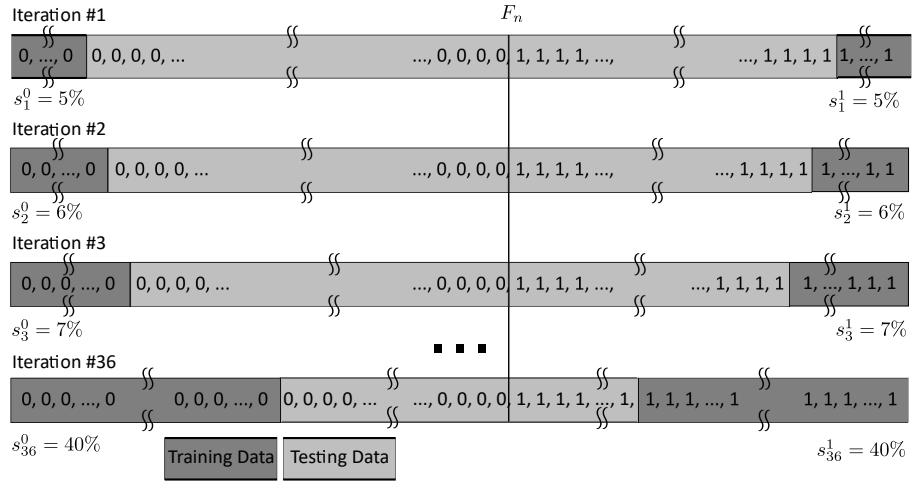


Figure 4.2: Visualization of training and testing data construction in Algorithm 1.

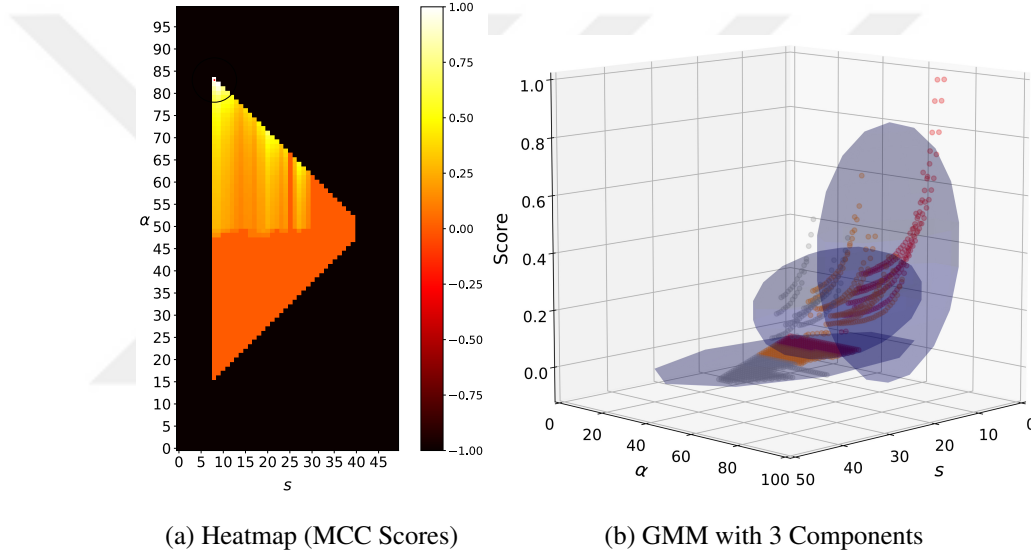


Figure 4.3: Visualization of Algorithm 1 score values.

4.2.2. Data Preprocessing

In the second stage of the framework, the dataset is prepared for model training and evaluation. This stage involves several steps, including data scaling, feature selection, and windowing. Initially, the dataset undergoes data scaling to ensure that all the features are on a comparable scale. Subsequently, exploratory analysis is conducted on the scaled datasets, revealing that not all features contribute equally to the task at hand. Some features exhibit a nearly constant value across the dataset, indicating a lack of informative content. Based on the analysis of 21 time series features, it has been observed that only a subset of sensors (specifically sensors 2 to 9, 11 to 15, 17,

and 20 to 21) contain valuable and informative data [114, 115]. Consequently, for the purpose of modeling the RUL predictor in this study, we solely utilize these selected features. By focusing on these particular sensors, we aim to capture the most relevant information and optimize the performance of the RUL prediction model.

When dealing with time-series data, deep networks tend to fit lower frequency functions first [116], which results over-fitting of the model to the signal. To exploit potential benefits of the time-series signal, we added non-linear components to the signals using non-linear transformations via harmonic functions as in Equation 4.3, where p is the original signal, $(\sin(.), \cos(.))$ non-linear components to be added to the signal, and i is the i^{th} harmonic. In this context non-linear transformations map the inputs to a higher dimensional space using high frequency functions increasing the variation in the appearance of a feature in the signal, which produce a better fit [117]. Along with the raw sensor data, we enhance the data by applying non-linear transformations, and adding new components produced by $i = 0$ and $i = 1$ in the equation.

$$\gamma(p) = (\sin(2^0 \pi p), \cos(2^0 \pi p), \dots, \sin(2^i \pi p), \cos(2^i \pi p)) \quad (4.3)$$

4.2.3. Training and Evaluation

This phase of the framework consists of three main steps: developing model architecture, model training, and model evaluation. Regarding the model architecture, we propose a joint autoencoder-regressor (JAER) model, designed to offer a flexible and adaptable structure. Two realizations of the architecture, and possible improvements are given in the following sections.

The architectural models are trained end-to-end. During the training step, model hyper-parameters are optimized using genetic algorithms, which is described in Section 2.3.2. RUL prediction performance on C-MAPSS dataset is the evaluated by the *ScoreFunction* [91] given in (Equation 4.4). In the equation; S is total penalty score, n is engine count in the subset, $d = \widehat{RUL} - RUL$, i.e. $d = RUL_{predicted} - RUL_{actual}$,

$a_1 = 13$, and $a_2 = 10$. In this equation, better RUL prediction performance yields lower scores.

$$S = \begin{cases} \sum_{i=1}^n e^{-\left(\frac{d}{a_1}\right)} & d < 0 \\ \sum_{i=1}^n e^{\left(\frac{d}{a_2}\right)} & d \geq 0 \end{cases} \quad (4.4)$$

Another common metrics used for C-MAPSS RUL prediction is Root Mean Square Error, which is defined in Equation 2.7.

4.3. Joint Autoencoder-Regressor Architecture: Base Model

General JAER architecture given in Figure 4.4 is flexible such that the autoencoder and the regressor can be realized in multiple ways. In this study we created two realizations of the achitecture, one base model with CNN autoencoder, called JAER-CNN, and another with InceptionTime-based autoencoder, called JAER-IT, respectively. In each version the regressor is implemented via LSTM layers.

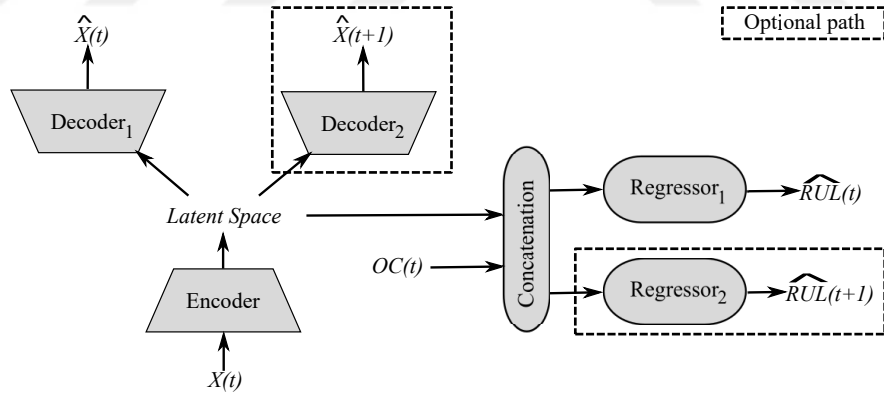


Figure 4.4: The proposed joint autoencoder-regression architecture for estimating RUL with time series data.

In JAER-CNN version, three convolutional layers are stacked together to form the encoder, which is used to extract spatial information. Input to autoencoder is $X(t)$, the sensor values at time t . In the decoder part, three de-convolutional layers decodes $\hat{X}(t)$ and $\hat{X}(t+1)$, the signals at time t and $t+1$ respectively. Convolutional layers are wrapped in time distributed layer, which applies convolution operation to every slice of

the input. Batch normalization is performed between convolutional layers. Encoded sensor values for time t is concatenated to operational conditions $OC(t)$ at time t , before the RUL regression. The RUL regression layers contain two LSTM layers, which extracts temporal information from features extracted in encoder. Dense layers are used to generate $\widehat{RUL}(t)$ and $\widehat{RUL}(t+1)$ from temporal information. Using more than one output in neural networks (multitask learning) is expected to improve generalization [118].

4.3.1. Implementation

The RUL prediction framework, as proposed, has been implemented and applied to the Turbofan Engine Degradation Simulation Dataset [91]. In this section, we provide comprehensive details regarding the implementation of each stage within the framework, as well as the evaluation results obtained from analyzing the datasets.

The proposed JAER-CNN architecture has been applied to the FD001 and FD003 subsets for RUL prediction. In Section 4.2.3, a two-step approach was employed to evaluate and optimize the model architecture. In the first step, genetic algorithms were utilized to search for the optimal hyper-parameters of the model. During this optimization process, each model (represented as individuals in the population) was trained for 100 epochs. The root mean square error (RMSE) on the validation data was used as the fitness score for the genetic algorithms. The decision to limit the training epochs to 100 was based on experimental observations that indicated satisfactory convergence for individual models. This approach helped minimize the search time. However, it was also noted that some models could benefit from additional training epochs. Hence, in the second step, the top three models resulting from the first step were further trained for 1000 epochs, with early stopping employed. The genetic algorithms optimization parameters were manually selected after conducting several experimental runs. Details of the genes assigned to the individuals are presented in Table 4.2. These parameters were chosen carefully to ensure an effective optimization process for the JAER-CNN architecture.

Table 4.2: The JAER-CNN model hyper-parameters for GA optimization.

Model Parameters	Type	Range
General		
Window size	Random integer	5 – 30
Learning Rate	Random Real (log uniform)	$10^{-3} - 10^{-1}$
Convolutional Layers (Encoder)		
Number of Filters	Random integer	30 – 80
Kernel Size	Random from list	[3, 5, 7, 9, 11, 13]
Activation	Random from list	['relu', 'tanh', 'sigmoid', 'linear']
Apply Batch Norm.	Random boolean	[true, false]
Convolutional Layers (Decoder)		
Activation	Random from list	['relu', 'tanh', 'sigmoid', 'linear']
Dense Layer (Latent Space)		
Encoding Size	Random Integer	10 – 100
Activation	Random from list	['relu', 'tanh', 'sigmoid', 'linear']
LSTM Layers (RUL Generation)		
Number of LSTM Units	Random Integer	5 – 50
Fusion Layer		
Number of Nodes	Random Integer	5 – 100
Activation	Random from list	['relu', 'tanh', 'sigmoid', 'linear']
L1 Regularization	Random Real (log uniform)	$10^{-3} - 10^{-1}$

Joint autoencoder-regression provides a general architecture for RUL prediction. An example realization of the architecture is given in Figure 4.5. Hyper-parameters for the best of the models are also given in the figure.

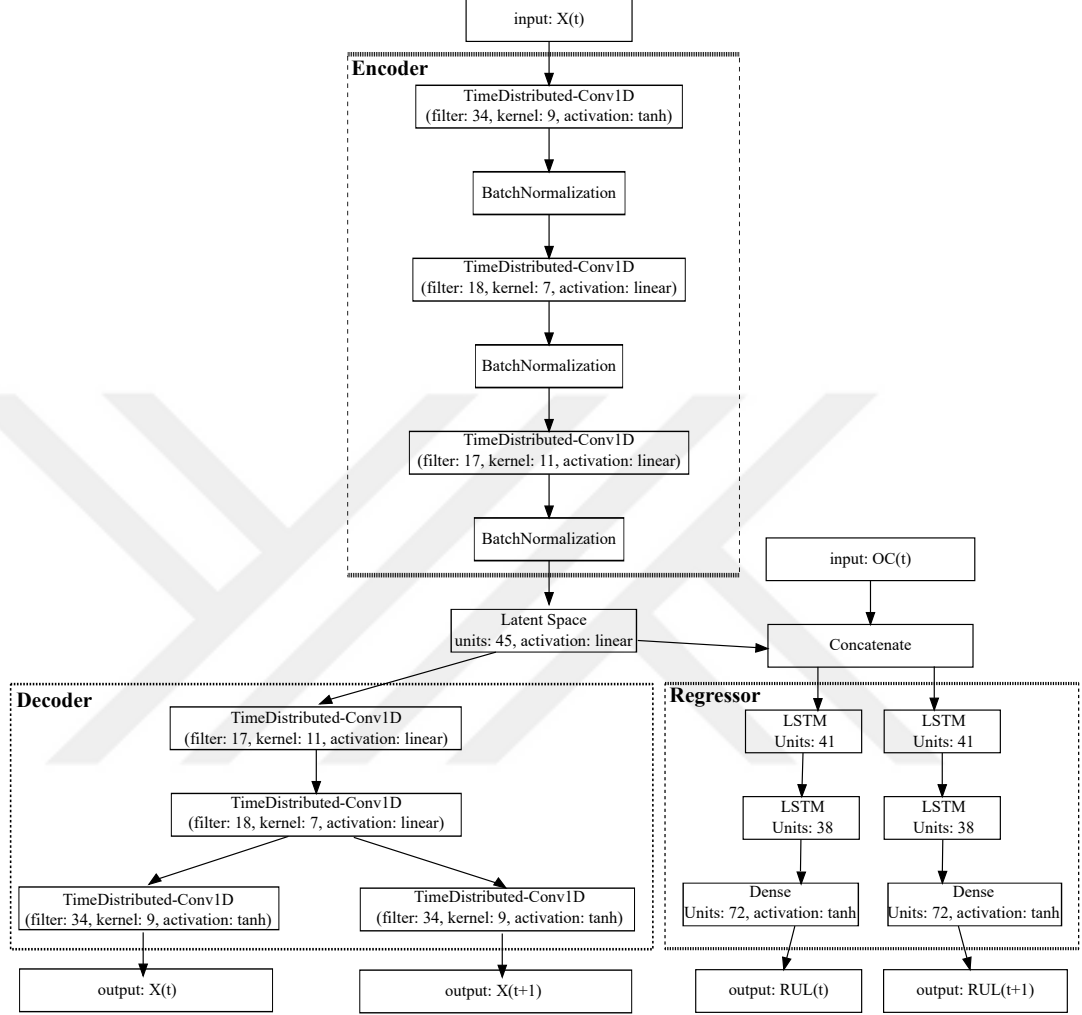


Figure 4.5: Architecture and hyper-parameters of the optimized model.

4.3.2. Results

The results obtained from experiments conducted on the C-MAPSS FD001 subset using the proposed joint autoencoder-regressor with non-linear transformations are presented in Table 4.3. Regarding the RUL labeling methods, it was observed that the Linear RUL labeling consistently yielded the worst root mean square errors (RMSEs) and scores. The scores for Linear labeling showed a gradual decline as the number of non-linear components increased. Comparing the PwL120 and PwL125 scores, they

were relatively close to each other (within $\pm 3\%$), and both outperformed the PwL130 scores. Additionally, the scores for both PwL120 and PwL125 decreased with the addition of non-linear components. On the other hand, the PwL130 score decreased only for the first two components ($i = 0$). Among the fault detection-based RUL labeling approaches, KGMM3 with raw sensor values demonstrated the best RMSEs and scores. Furthermore, the performance of PwLE and KGMM5 improved when two non-linear components ($i = 0$) were added, but this improvement was not observed for KGMM3. Overall, the findings indicate that the proposed joint autoencoder-regressor with non-linear transformations showed promising results, with fault detection-based RUL labeling methods, particularly KGMM3 with raw sensor values, exhibiting superior performance in terms of RMSEs and scores.

Table 4.3: JAER-CNN evaluation results on C-MAPSS FD001 data.

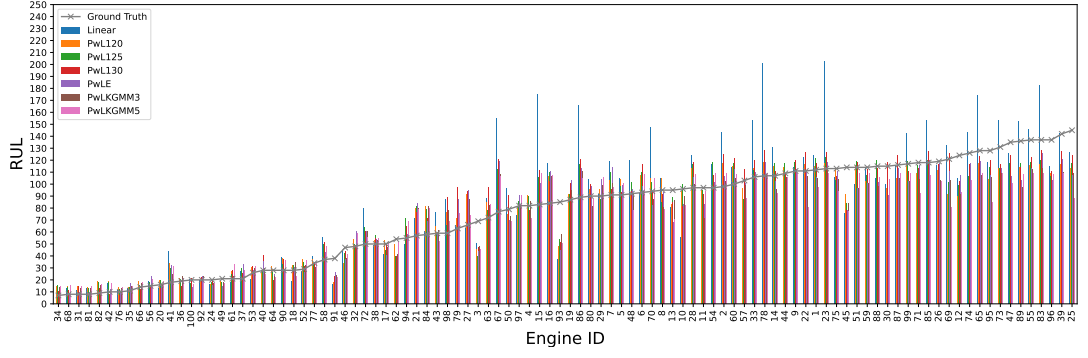
	Without Non-linear Transformations		With Non-linear Transformations (Eq. 4.3 with $i = 0$)		With Non-linear Transformations (Eq. 4.3 with $i = 1$)	
RUL	RMSE	Score	RMSE	Score	RMSE	Score
Linear	17.27	765	19.54	604	16.27	388
PwL120	14.41	291	<u>14.22</u>	272	<u>13.30</u>	262
PwL125	13.93	282	14.42	<u>279</u>	12.94	<u>263</u>
PwL130	<u>14.23</u>	366	14.19	309	14.74	372
PwLE	15.20	348	14.73	311	16.55	423
KGMM3	15.06	<u>286</u>	19.35	544	17.72	461
KGMM5	17.14	410	16.32	332	16.99	380

Figure 4.6 provides a visual representation of the RUL performance on the test engines. To enhance readability, the test engines are sorted on the horizontal axis based on the ground truth RUL values. The analysis includes three different cases, considering the presence or absence of non-linear transformations on the inputs. It is

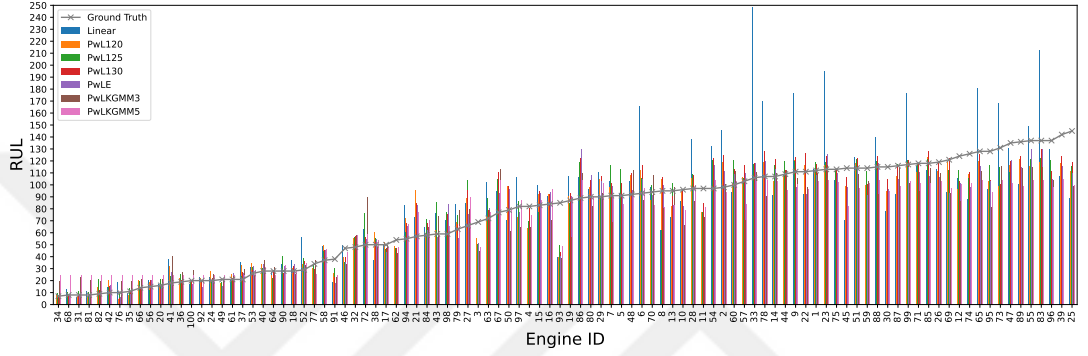
observed that the inclusion of non-linear transformations leads to improved prediction results, with the PwL120 approach demonstrating better performance. It is worth noting that when the estimates deviate significantly from the actual values, this discrepancy is observed consistently across all methods. This suggests potential issues with the accuracy of the ground truth labels used for evaluation. Additionally, Figure 4.7 illustrates the temporal RUL estimations for a typical test run (specifically Engine 100). The estimations generally exhibit closer proximity to the assigned RUL values. While the incorporation of non-linearization tends to enhance the average estimation performance, there are instances where it may surpass the performance of the linear case. Overall, the findings from these visualizations highlight the impact of non-linear transformations on improving RUL estimations, particularly in the case of the PwL120 approach. However, variations and individual cases should be carefully considered to ensure a comprehensive evaluation of the prediction performance.

Results of proposed model on C-MAPSS FD003 subset is given in 4.4. Contrary to previous findings, the inclusion of non-linear components in the data did not lead to an improvement in the model's performance. Our suspicion is that while these non-linear transformations mapped the raw inputs to a higher dimensional space and increased the signal's variability, they also amplified the noise variations. As a result, the model's ability to extract meaningful features was disrupted. When comparing the RMSE and Score values for the labels PwL120, PwL125, and PwL130, they exhibited similar performance, which was significantly better than the results obtained from KGMM3 and KGMM5. In general, the PwL RUL labeling method outperformed other labeling methods, with PwL120 achieving the highest Score among all the labels.

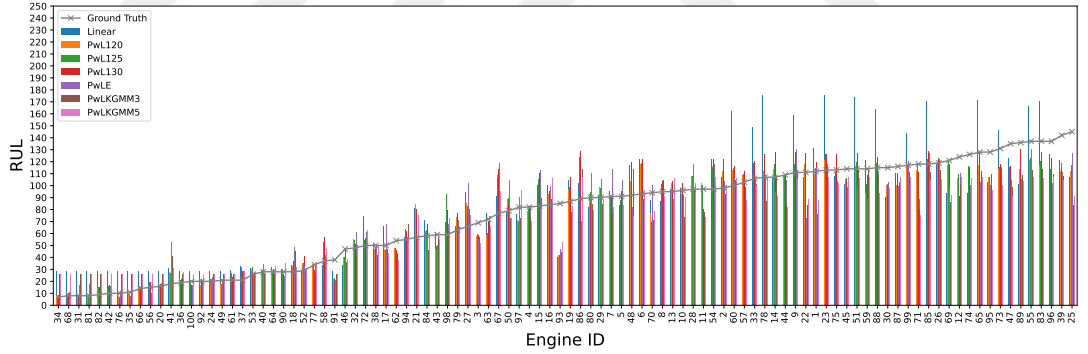
Prior to conducting our analysis, we had a hypothesis that Linear RUL labeling would yield the poorest results, and this was indeed validated for the FD001 dataset. However, it is noteworthy that the performance of Linear RUL labeling significantly improved by 50% when non-linear components were incorporated into the process. Conversely, our expectation was that fault detection-based RUL labeling would exhibit superior performance. We think that noise in the data lower down GMM's performance



(a) Without Non-linear Transformations.



(b) With Non-linear Transformations (Eq. 4.3 with $i = 0$).



(c) With Non-linear Transformations (Eq. 4.3 with $i = 1$).

Figure 4.6: RUL prediction performance of algorithms on C-MAPS FD001 test data.

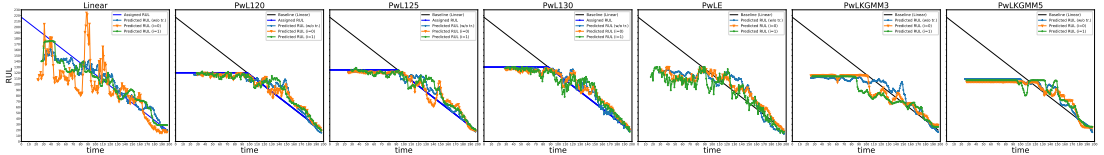


Figure 4.7: Plots showing typical results of RUL prediction of algorithms.

for fault start prediction in 1. It seems that piecewise linear labeling suppressed the noise in the data by setting a RUL_{Max} value to the RUL labels. We expected that the

Table 4.4: JAER-CNN evaluation results on C-MAPSS FD003 data.

	Without Non-linear Transformations		With Non-linear Transformations (Eq. 4.3 with $i = 0$)		With Non-linear Transformations (Eq. 4.3 with $i = 1$)	
RUL	RMSE	Score	RMSE	Score	RMSE	Score
Linear	15.39	383	13.86	317	15.85	483
PwL120	14.36	278	15.31	368	15.87	370
PwL125	<u>14.09</u>	296	14.88	381	15.19	<u>351</u>
PwL130	13.96	<u>290</u>	<u>14.46</u>	<u>328</u>	<u>15.46</u>	320
KGMM3	15.30	347	15.72	406	16.03	419
KGMM5	18.81	489	19.54	632	21.66	779

model performance for FD003 to be lower than it was for FD001, as FD003 had two faults occurring simultaneously. Simultaneous faults seem to affect the performance of the fault detection, which in turn affects RUL prediction performance with KGMM3 and KGMM5 RUL labeling on FD003 subset.

In Table 4.5, we present a comparison of our results with recent studies that have utilized deep learning methods on the C-MAPSS dataset. The **boldface** and underline formatting highlight the two best results for each column. Ellefsen et al. [119] achieved the best RMSE and Score on the FD001 subset. Their final model consisted of six layers, which were optimized through a hyper-parameter search. For a comprehensive overview of their work on the C-MAPSS subsets, readers can refer to [120], and we have included their results in the table for completeness. Wen et al. [121] reported the k-fold ensemble results of residual CNN for various numbers of residual blocks. Their reported Score pertains to subsets suitable for a k-fold cross-validation setup, and it remains unclear whether they utilized the official test data for these subsets. Al-Dulaimi et al. [122] employed features extracted from parallel CNN and LSTM layers. Their seven-layer network architecture demonstrated comparable results on the FD001 subset. Our proposed architecture, which combines a CNN-based autoencoder and

LSTM layers in a sequential manner, consists of nine layers and achieves performances comparable to state-of-the-art methods. By including our results in the comparison, it is evident that our proposed architecture, incorporating both CNN-based autoencoder and LSTM layers, delivers competitive performances on the C-MAPSS dataset, aligning with the achievements of existing cutting-edge approaches.

Table 4.5: Comparison of JAER-CNN model with some recent deep learning work on C-MAPSS.

	Year	FD001		FD003		Architecture
		RMSE	Score	RMSE	Score	
Ellefsen et al. [119]	2019	12.08	186	-	-	CNN + LSTM
Ellefsen et al. [120]	2019	12.56	<u>231</u>	12.10	251	RBM + LSTM
Al-Dulaimi et al. [122]	2019	13.02	245	11.22	288	Hybrid DNN
Mo et al. [123]	2020	<u>12.19</u>	259	12.85	343	Multiheaded CNN+LSTM
Jiang et al. [124]	2020	12.75	281	<u>11.35</u>	<u>278</u>	Bidirectional LSTM+CNN
Chaoub et al. [125]	2021	13.26	284	13.11	352	MLP+LSTM+MLP
Li et al. [126]	2022	12.65	278	12.78	307	Multihead CNN + Discriminator networks
JAER-CNN	2023	13.30	262	14.36	<u>278</u>	Joint Autoencoder Regressor

4.3.3. Discussions

Evaluation of the proposed joint autoencoder - regressor architecture along with RUL assignment methods on C-MAPSS shall be interpreted in three aspects: the impact of the joint autoencoder-regressor models, the influence of various RUL labeling methods, and the consequences of applying non-linear transformations to the original data. Here are the general observations we made regarding the three aspects of the results:

- Our proposed architecture demonstrates effective RUL prediction capabilities across different systems and scenarios.
- Both the model architecture and the RUL assignment method play crucial roles

in achieving successful RUL prediction.

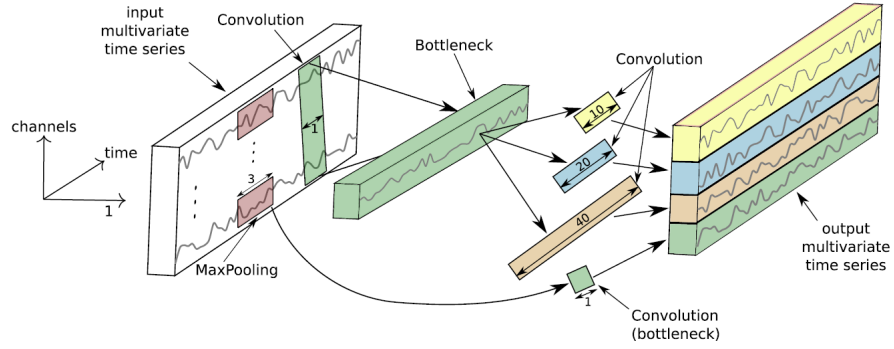
- While the linear degradation assumption and linear RUL labeling provide an initial advantage in the RUL prediction process, it is important to note that they may not always yield the best performance in all cases. When feasible, incorporating a heuristic approach to the degradation process for fault detection can be beneficial.
- The piecewise linear RUL labeling method performs well, indicating that assigning a fixed value for the maximum RUL (RUL_{Max}) for the entire dataset is as effective as fine-tuning it per system.
- The addition of non-linear components to the existing signals has the potential to enhance the model's performance.

4.4. InceptionTime-Based Model

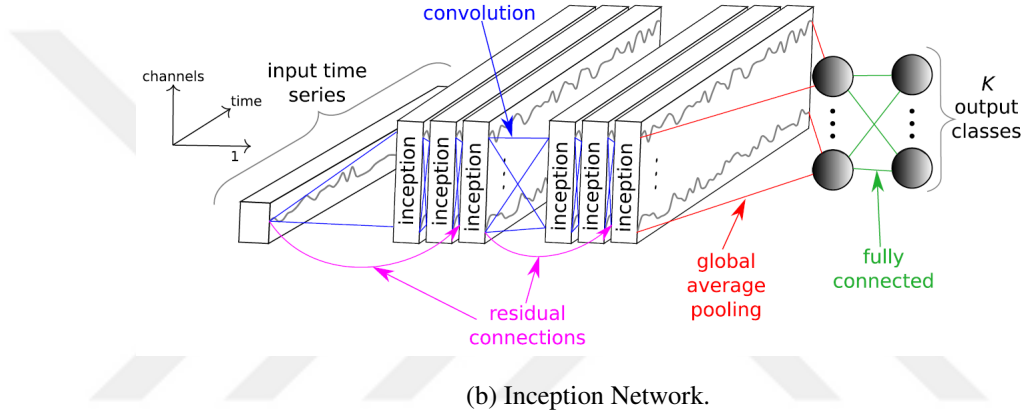
Image classification is one of the main tracks of deep learning research. Szegedy et al. in [127] proposed the Inception-v1 model for image classification, which introduced Inception modules for spatial information extraction and dimension reduction. Later in [128] they improved the model with Inception-ResNet by adding residual connections, enhancing performance and convergence speed. InceptionTime [129] is an adaptation of inception blocks and residual connections of the Inception-ResNet model to time series classification problems. Second version of the JAER model uses InceptionTime networks in the autoencoder, thus this version is called JAER-T.

The focal point of InceptionTime revolves around the Inception module (Figure 4.8a). This module utilizes 1D convolutional layers with different filter lengths in parallel to identify patterns of varying lengths. The outputs of these convolutional layers are concatenated to form the output of the Inception module. Multiple Inception modules are stacked together to form an Inception network, as depicted in Figure 4.8b. Additionally, residual connections are introduced for each of the the third Inception module in the stack. The network's depth increases along with longer filter sizes, enabling the model to capture larger patterns. The number of Inception modules in

the network stack and the lengths of the filters in the 1D convolutional layers depend greatly on the dataset and the specific domain.



(a) Inception Module.



(b) Inception Network.

Figure 4.8: Inception Module and Inception Network.

4.4.1. Implementation

In JAER-IT architecture, we use decreasing number of filters in each Inception module to support latent space generation in the encoder. This version of the architecture is applied to FD001 subsets for RUL prediction. As before, GA optimization is used for this model. Genes for individuals are given in 4.6. Genetic algorithms optimization parameters are selected manually after a few experimental runs. The process of optimizing the models commenced with an initial population size of 50. It then proceeded with an additional five generations, each consisting of 50 individuals. Throughout this stage of model training, a total of 300 models were trained, and the best performances achieved were documented.

Table 4.6: The JAER-IT model hyper-parameters for GA optimization.

Model Parameters	Type	Range
General		
Window size	Random integer	5 – 30
Learning Rate	Random Real (log uniform)	$10^{-3} - 10^{-1}$
Autoencoder Model		
Number of 1D Conv. in Inception Block	Random integer	2 – 4
Apply Batch Norm.	Random boolean	[true, false]
Number of Blocks in Inception Layer	Random integer	1 – 3
Number of Inception Layers	Random integer	1 – 3
Number of Filters for Inception Layers	Random integer	60 – 100
Dense Layer (Latent Space)		
Encoding Size	Random Integer	5 – 200
Activation	Random from list	['relu', 'tanh', 'sigmoid', 'linear']
LSTM Layers (RUL Generation)		
Number of LSTM Layers	Random Integer	1 – 3
Number of LSTM Units	Random Integer	10 – 160
Fusion Layer		
Number of Fusion Layers	Random Integer	1 – 3
Number of Nodes	Random Integer	10 – 200
Activation	Random from list	['relu', 'tanh', 'sigmoid', 'linear']

4.4.2. Results and Discussion

Results for proposed JAER-IT model and non-linear transformations on C-MAPSS FD001 subset is given in Table 4.7 and performance improvements for the model is given in Table 4.8. For this version of the model, linear RUL labeling gave the worst of the RMSEs and Scores. PwL120 and PwL125 also took first and second places while improving their RMSEs (upto 8.4%) and Scores (upto 18.2%) in the results table. Other labeling methods also improved their performance upto 26%. In terms of RUL prediction performance, we conclude that JAER-IT experiment results were better than JAER-CNN, especially for raw sensor data, and JAER-IT rendered the use of non-linear transformations unnecessary in most of the experiments. On the other hand, unlike in JEAR-CNN, adding new features using non-linear transformation (see (4.3)) did not improve JAER-IT RUL prediction performance. For cases with (4.3) ($i = 1$), which added two components, there were three negative improvements. These components produced $4 \times d$ new features, where d is the dimension of the raw data. 77-layers JAER-IT model extracted better features on raw data than it did with *engineered* data. We suspect that, while mapping the raw inputs to a higher dimensional space and increasing the variation in the signal, non-linear transformation also increased the variations of the noise, which in turn disrupted the JAER-IT model's feature extraction capability. JAER-CNN model, which has been modest with nine layers, tolerated this noise well.

Prior to conducting our analysis, we hypothesized that Linear RUL labeling would yield the least favorable results, and our findings confirmed this expectation. However, it is worth noting that the performance of Linear RUL labeling showed a significant improvement of 50% when non-linear components were incorporated into the JAER-CNN model. Additionally, the utilization of JAER-IT further enhanced the performance of Linear RUL labeling by an additional 42.9%. Conversely, we initially anticipated that fault detection-based RUL labeling would outperform general piece-wise RUL labeling. However, it appears that the presence of noise in the data may have hindered the performance of the Gaussian Mixture Model (GMM) in predicting the fault start point in Algorithm (1). On the other hand, the piece-wise linear labeling approach

Table 4.7: JAER-IT evaluation results on C-MAPSS FD001 data.

	Without Non-linear Transform		Non-linear Transform (with $i = 0$)		Non-linear Transform (with $i = 1$)	
RUL	RMSE	Score	RMSE	Score	RMSE	Score
Linear	17.05	437	16.5	570	18.37	576
PwL120	<u>13.2</u>	238	<u>13.69</u>	256	<u>13.24</u>	<u>259</u>
PwL125	13.05	<u>243</u>	13.29	270	13.56	250
PwL130	13.58	292	14.13	304	12.85	270
PwLE	13.99	257	14.25	<u>267</u>	14.63	290
KGMM3	14.56	265	14.9	<u>267</u>	15.35	314
KGMM5	15.71	308	15.18	304	15.76	309

Table 4.8: Performance improvements on JAER-IT with C-MAPSS FD001 data.

	Without Non-linear Transform		Non-linear Transform (with $i = 0$)		Non-linear Transform (with $i = 1$)	
RUL	RMSE	Score	RMSE	Score	RMSE	Score
Linear	1.3	42.9	15.6	5.6	-12.9	-48.5
PwL120	8.4	18.2	3.7	5.9	0.5	1.1
PwL125	6.3	13.8	7.8	3.2	-4.8	4.9
PwL130	4.6	20.2	0.4	1.6	12.8	27.4
PwLE	8.0	26.1	3.3	14.1	11.6	31.4
KGMM3	3.3	7.3	23.0	50.9	13.4	31.9
KGMM5	8.3	24.9	7.0	8.4	7.2	18.7

demonstrated the ability to mitigate the impact of noise in the data by assigning a specific *RUL_{Max}* value to the RUL labels. Overall, our observations indicate that while Linear RUL labeling exhibited improved performance with the inclusion of

non-linear components, the performance of fault detection-based RUL labeling was influenced by the presence of noise in the data. The piece-wise linear labeling method effectively suppressed the noise by assigning appropriate RUL_{Max} values to the RUL labels.

4.5. Joint Autoencoder-Regressor Architecture: Possible Improvements

Further studies on JAER-IT model includes modifications to the base JAER architecture, such as addition of the Gaussian noise, and pruning the models.

The JAER-IT model was further studied by modifying the base architecture and adding Gaussian noise. The best performing models from the previous stage (see Section 4.4) were then used to create a new population with added Gaussian noise. The model was trained for five more generations variance values of $\sigma^2 = 0.1$ and $\sigma^2 = 0.01$. The best performing model from each generation was reported. For the next stage, the top 50 models selected from the previous stage were pruned by 10%, and the best performance was reported from the resulting models.

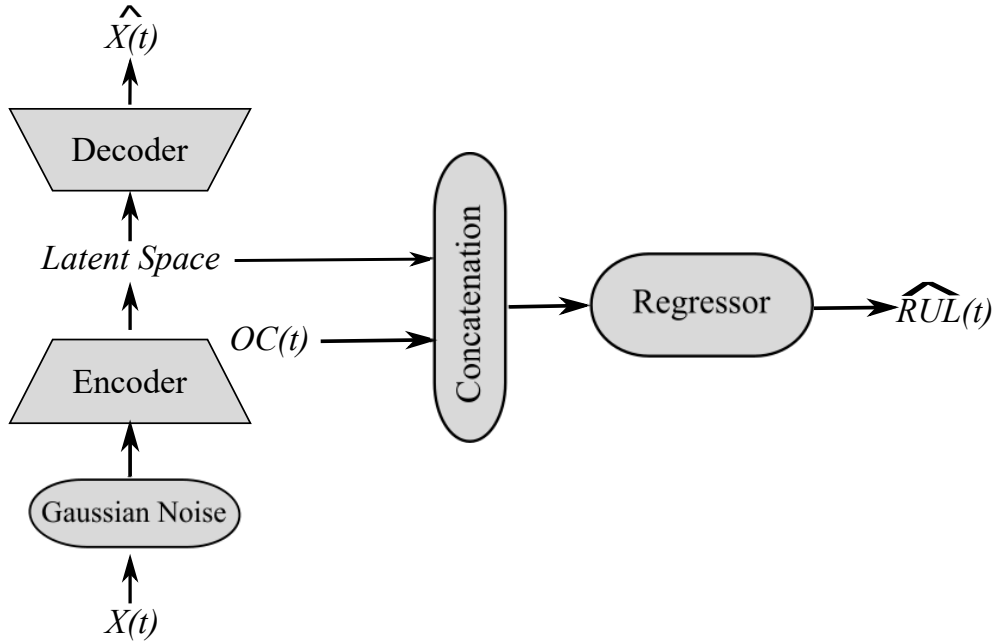


Figure 4.9: JAER-IT architecture with Gaussian noise layer.

Result for second and third stages are given in Table 4.9 and Table 4.10 respec-

tively.

Table 4.9: Results of the base JAER-IT models and Gaussian noise added models.

	Base ($\sigma^2 = 0$)		$\sigma^2 = 0.1$		$\sigma^2 = 0.01$	
	RMSE	Score	RMSE	Score	RMSE	Score
Linear	17.05	437	14.87	373	16.12	376
PwL120	13.20	238	12.92	212	12.75	206
PwL125	13.05	243	12.90	225	13.06	228
PwL130	13.58	292	13.37	280	13.41	260

Table 4.10: Results of the pruned models.

	$\sigma^2 = 0.1$		$\sigma^2 = 0.01$	
	RMSE	Score	RMSE	Score
Linear	15.80	499	16.62	499
PwL120	13.58	241	14.25	280
PwL125	14.80	351	15.04	349
PwL130	13.90	291	13.95	298

When evaluating the scores in the first stage, it is observed that the combined autoencoder-regression architecture provides the best score for PwL120. In this stage, the calculated RMSE value for PwL120 and PwL125 is 0.15, which corresponds to 1%. In the second stage, the obtained score values show that the added Gaussian noise improves performance between 3.9% and 14.5%. The average improvement is higher for a variance value of $\sigma^2 = 0.01$. Consistent with the score values, there is an improvement in the RMSE values ranging from 1.1% to 12%. Only one RMSE value shows a negligible increase of 0.7%. In the final stage, it is observed that the network pruning process decreases the performance. All the results obtained in this stage are lower than the achievements in the first stage. It can be concluded that network pruning does not yield good results when used in conjunction with noise addition.

The outcomes of the investigation, as well as the identification of the most outstanding studies utilizing deep learning architectures, which were recently published, are presented in Table 4.11. The table demonstrates that the model based on InceptionTime exhibits competitive results. The inclusion of Gaussian noise leads to a performance improvement of 13.4%, bringing the results closer to the established state of the C-MAPSS dataset.

Table 4.11: Comparison of JAER architectures with some recent deep learning work on C-MAPSS.

	Year	RMSE	Score	Architecture
Ellefsen et al. [119]	2019	12.08	186	CNN + LSTM RUL: Per engine
Al-Dulaimi et al. [122]	2019	13.02	245	Hybrid DNN RUL: PwL125
Mo et al. [123]	2020	12.19	259	Multiheaded CNN+LSTM RUL: PwL125
Jiang et al. [124]	2020	12.75	281	Bidirectional LSTM+CNN RUL: PwL125
Chaoub et al. [125]	2021	13.26	284	MLP+LSTM+MLP RUL: PwL130
Li et al. [126]	2022	12.65	278	Multihead CNN + Discriminator networks
JAER-CNN	2023	13.30	262	JAER /w CNN RUL: PwL120
JAER-IT	2023	13.20	238	JAER /w InceptionTime RUL: PwL120
JAER-IT-GN	2023	12.75	206	JAER /w InceptionTime RUL: PwL120 Gaussian Noise: $\sigma^2 = 0.01$

It has been observed that when the effectiveness of the InceptionTime network in feature extraction is combined with the effectiveness of LSTM in time series prediction, the architecture produces competitive results in the literature. The addition of Gaussian noise layer to the architecture reduces overfitting in the model and improves its performance. The enhanced architecture yields results that are close to the known state of the technique in the literature.

4.6. Summary

In this study, we introduce a comprehensive end-to-end joint autoencoder-regressor architecture for data-driven remaining useful life (RUL) prediction. The architecture consists of two versions: JAER-CNN, which utilizes a convolutional neural network (CNN) based autoencoder, and JAER-IT, which employs an InceptionTime-based autoencoder. Both versions leverage long-short term memory (LSTM) based regressors to predict the RUL. To improve the regression problem, the operating conditions of the process are incorporated into the autoencoder's latent space. The model hyperparameters are optimized using genetic algorithms, ensuring the best configuration for the architecture. The proposed architecture is validated on the C-MAPSS Dataset benchmark, and the results demonstrate its capability to predict RUL for various systems and scenarios. Notably, the JAER-IT model exhibits competitive performance when compared to recent research on the C-MAPSS dataset. Additionally, we propose a novel fault detection-based RUL labeling method, which adopts an iterative classification approach to detect the initiation of system degradation. This method proves effective in identifying the start of degradation in such systems, contributing to more accurate RUL predictions.

5. SHIP ENGINE ROOM FAULT CLASSIFICATION

The main propulsion system of a ship relies on a complex arrangement of components, including the shaft, propellers, and auxiliary equipment. In the case of merchant ships, marine diesel engines are commonly used to provide power for navigation. These engines consist of various intricate parts such as cylinders, pistons, valves, nozzles, and a turbocharger system. Any faults within these engines can have significant repercussions, impacting the overall system performance and hindering the ship's mission. Hence, it is crucial to continuously monitor the engine and auxiliary systems during operation and analyze diagnostic data to assess their condition. Fault diagnostics play a pivotal role in ensuring safety and product quality in modern industries. The primary objectives of fault diagnostics encompass detecting faults, identifying their specific nature, and conducting comprehensive fault analysis. Researchers have been actively investigating fault diagnosis techniques for several decades [130], recognizing its critical importance in maintaining system integrity and preventing potential failures.

This study is out attempt to adapt joint autoencoder-regression architecture to fault classification for ship engine failures. This section will introduce a new ship engine room dataset, with various initial start conditions, and two faults injected separately. We proposed JAEC architecture on out previous JAER architectures. Also we attempt to predict NO_x and SO_x gases emissions from the engine, during normal and faulty working conditions.

5.1. Introduction

This study focuses on the analysis of monitoring data obtained from a crude oil tanker's main engine in two distinct operating conditions: full ahead loaded and full ahead unloaded. The data used for analysis is sourced from a highly realistic full-mission engine room simulator provided by Kongsberg, an international technology group that supplies high-technology systems to customers in maritime and other areas. The dataset encompasses over 60 sensor measurements, including both normal working

conditions and specific malfunction scenarios. The malfunction scenarios specifically involve the diesel engine and consist of two distinct cases: injection valve nozzle wear and injection valve nozzle clogging. Each run of the failure scenarios concludes with the occurrence of one of these malfunctions, if present. By analyzing this dataset, the study aims to achieve fault detection and identification for the main engine, enabling a comprehensive understanding of the engine's performance under different operating conditions and potential malfunction scenarios.

As part of our research on the dataset, we formulated a classification problem to detect the presence of defined malfunctions and identify the specific type of malfunction. To address this problem, we developed a joint autoencoder-classifier model comprising a convolutional autoencoder and a long-short term memory (LSTM) regressor. This joint architecture enabled us to train an end-to-end multi-target deep neural network using the multivariate time series data provided in the dataset. The autoencoder component of the model extracted meaningful features from the input data, while the LSTM regressor utilized these extracted features to predict the target variables. Furthermore, we integrated the operating conditions of the process into the model by incorporating them as additional parameters in the latent space of the autoencoder. This incorporation aimed to enhance the model's understanding of the data and improve its performance in detecting and identifying malfunctions. To identify the most informative sensors for the classification task, we conducted an analysis using the best-performing models. This analysis allowed us to investigate the sensors that significantly influenced the classification performance, providing valuable insights into the key indicators for detecting and diagnosing malfunctions in the main engine system.

In addition to collecting sensor data, we also recorded the amounts of released NO_x and SO_x emissions during the data collection process. As part of our initial study, we extended our analysis to include the estimation of Nitrogen oxides (NO_x) and Sulphur oxides (SO_x) gas emissions. Specifically, we developed several gradient boosting based regressors to estimate these emissions for different ship loads and various types of faults. By incorporating this aspect into our research, we aimed to gain insights

into the environmental impact and emission levels associated with different operational conditions and fault scenarios in the ship's main engine system.

5.2. Kongsberg Engine Room Simulator Dataset

The Kongsberg K-Sim is a widely recognized ship engine room simulator that offers a high level of accuracy and realism. Among its various configurations, one notable setup is the ERS MAN B&W 5L90MC VLCC L11-V (referred to as MC90-V), which specifically simulates a very large crude carrier equipped with a MAN B&W slow speed turbocharged diesel engine as its propulsion unit. This engine model is based on actual engine data, ensuring that the simulator's dynamic behavior closely resembles the response of a real engine. Additionally, the K-Sim simulator encompasses features such as a control room operator station, panels, bridge, and steering panels. Furthermore, K-Sim offers additional applications like Neptune for classroom training and TLDS for engine room monitoring, providing a comprehensive suite of tools for maritime training and monitoring purposes.

We utilized the Neptune application within the Kongsberg K-Sim simulator to create *exercises* that involved multiple runs of the core simulator. Through Neptune, we defined various scenarios by specifying the initial conditions for the MC90-V dataset. These initial conditions encompassed factors such as the ship's load and speed, sea water temperature, and sea conditions. In total, we created 18 distinct scenario initializations to capture a range of operating conditions, as outlined in Table 5.1. Additionally, we employed the TLDS (Engine Room Monitoring Data System) functionality to record simulator variables as engine room monitoring data, enabling us to gather comprehensive data for our analysis.

The simulator offers a vast range of approximately 1500 malfunctions that can be injected into the exercises. However, for the purpose of our research, we focused specifically on two malfunctions occurring in the 1st cylinder of the engine: Cylinder 1 injection valve nozzle wear (referred to as M2503) and Cylinder 1 injection valve nozzle clogged (referred to as M2508). These particular malfunctions were chosen

Table 5.1: Initial Conditions for MC90-V Dataset.

Condition	Possible Values
Ship Speed/Load	FAL: Full Ahead Loaded FAU: Full Ahead Unloaded
Sea Water Temperature	20°C 25°C 28°C
Sea Condition (Beauf)	0 4 6

as they align with the objectives of our study and provide relevant insights into the behavior of the engine under faulty conditions.

In our study, we conducted a total of 2862 runs, taking into account 18 different initial conditions and 3 malfunction states. For each initial condition, we divided the runs into a training set consisting of 35 runs and a testing set consisting of 18 runs. Each run comprised a collection of 1000 to 1400 data samples recorded at a frequency of 1 Hz until the occurrence of a failure. The monitoring data consisted of over 60 variables, which can be categorized into two groups: *real sensors* and *simulated sensors*. In this preliminary analysis, we focused specifically on the data captured by the real sensors. For this study we restrict ourselves to ship load *Full Ahead Loaded* and *Full Ahead Unloaded*, sea water temperature 20°C and Sea Condition 0.

5.3. Problem Definition

The initial stage in the diagnosis process involves detecting faults within a monitored system, which is essential for identifying the root cause. The primary objectives of fault detection and diagnosis are to identify significant indicators that can be utilized for predicting the health of a system and taking appropriate actions to prevent future failures. These key indicators enable the prediction of the fault class before the system

becomes inoperative.

The MC90-V Dataset presents unique challenges, particularly in terms of *fault identification* and *classification*. The main goal is to determine the state of the engine, whether it is operating normally or experiencing a fault, and to predict the specific fault type if one is present. The fault-free exercise runs, labeled as M0000, represent the engine's behavior during its normal operating condition. During these runs, the engine operates smoothly without any issues. Faults related to Cylinder 1 are intentionally introduced by the simulator with a random delay. These faults are labeled as M2503 and M2508. Two objectives of this study are to detect and classify faults in the test data and to identify the signals that contribute the most to the prediction process.

The main engines of ships operate on diesel fuel, which generates various gases during combustion. With the increasing number of ships traveling each day, their emissions have become a global concern. The two primary pollutants emitted by ships are Nitrogen oxides (NO_x) and Sulphur oxides (SO_x) gases. These emissions have a significant impact on the ozone layer within the troposphere of the Earth's atmosphere, leading to the greenhouse effect and global warming. Another objective of this study is to estimate the emissions of NO_x and SO_x resulting from the combustion process.

5.4. Methods and Techniques

The primary objective of this study was to develop a fault classifier capable of distinguishing between M0000, M2503, and M2508. To achieve this, we followed the workflow depicted in Figure 5.1, which consists of data preprocessing, model training, and evaluation phases. The data preprocessing step was performed to prepare the data for training and evaluation processes.

In the data preprocessing stage, we performed several actions to prepare the data for analysis. Firstly, we merged the simulation data, which was originally distributed across approximately 23,000 files, into a single dataset. Next, we addressed missing values in the dataset by applying a *backfill* imputation method to fill in the gaps. To ensure consistency and comparability across different features, we also applied data

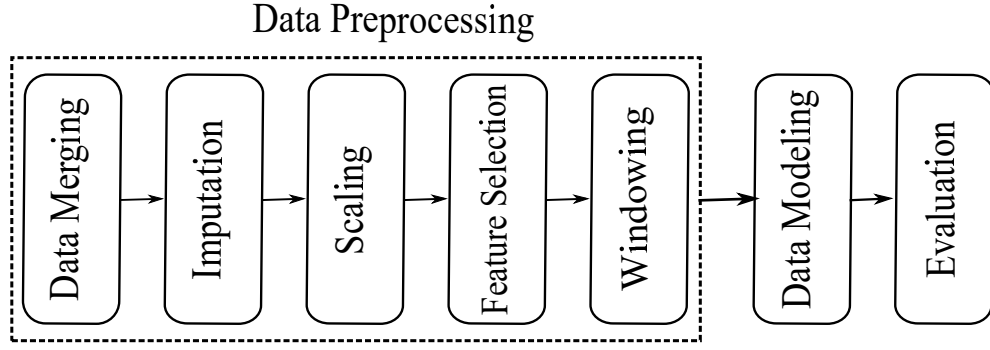


Figure 5.1: General data processing workflow for fault classification.

normalization and scaling techniques. Additionally, feature selection was conducted to identify the most relevant variables for our analysis. Finally, we applied a windowing technique to create contextual segments of the time series data, allowing us to capture the temporal patterns in the data effectively.

The data modeling and optimization stage of our study involves the utilization of a joint autoencoder-classifier (JAEC) architecture. This architecture incorporates a convolutional neural network (CNN)-based autoencoder, as depicted in Figure 5.2. In this architecture, the input consists of sensor values $X(t)$ at time t , which are fed into the autoencoder. The decoder component of the autoencoder generates reconstructed signals $\hat{X}(t)$ at time t . The encoded sensor values in the latent space are then concatenated with the corresponding operational conditions $OC(t)$ at time t . This combined data is passed through the classification path, which includes two long-short term memory (LSTM) layers and dense layers that generate predicted classes $\widehat{Class}(t)$. In the JAEC-CNN variant, the convolutional layers in the autoencoder are wrapped in time distributed layers, enabling the convolution operation to be applied to each temporal slice of the input. Furthermore, the number of filters in the CNN layers decreases in the encoder part of the architecture to support the generation of the latent space. Batch normalization is performed between the convolutional layers to improve the stability and performance of the model.

In order to estimate NOx and SOx emissions, we propose the use of gradient boosting, and hyper-parameters for the models are optimized using genetic algorithms as described in Section 2.3.2. Accuracy (ACC) and F1 score (F1) are used as evaluation

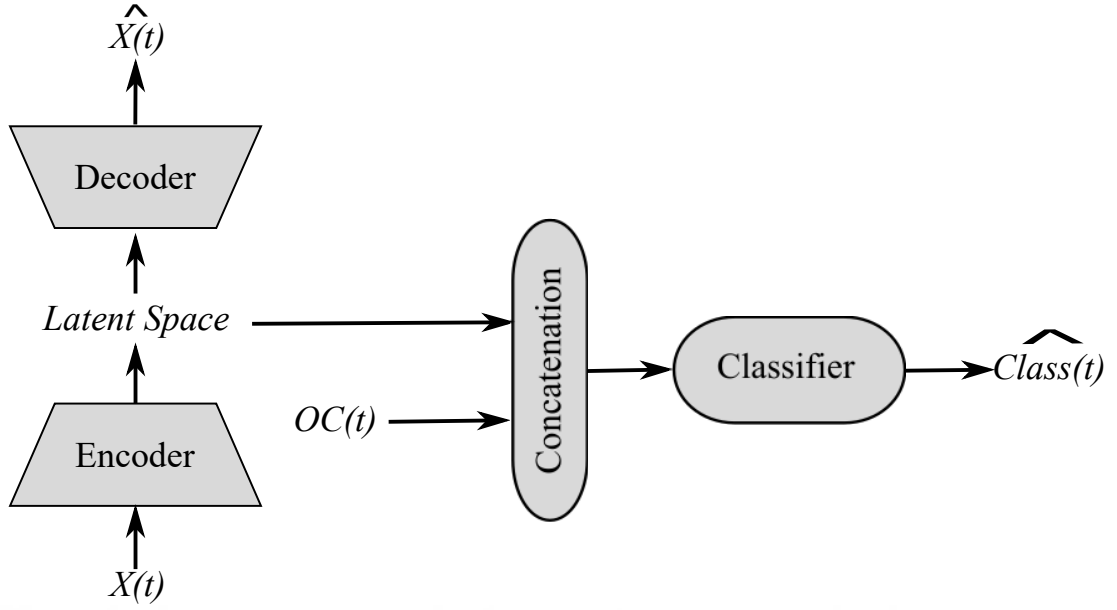


Figure 5.2: Proposed joint autoencoder classifier architecture.

metrics as described in Equation 2.1 and Equation 2.4, respectively.

5.5. Experiments and Results

The proposed framework for fault classification and emission estimation on the MC90-V dataset was implemented using the Scikit-Learn [106] and Keras [131] libraries. In this section, we provide specific details about the implementation process for each task and present the evaluation results obtained.

5.5.1. Fault Classification

In the data preprocessing step, the initial dataset consisting of individual data files for each exercise run was processed to prepare it for training and evaluation. These individual files were merged to create two separate subsets: the training set and the test set. The dataset also contained missing data, which were addressed by performing interpolation to fill in the missing values and ensure the integrity of the data. To normalize the data and bring it to a standard scale, Z-score normalization was applied, resulting in a mean of zero and a standard deviation of one. This normalization process was performed on the training data, and the scaling parameters obtained were then applied to the test data to maintain consistency. During exploratory data analysis,

it was discovered that some features in the dataset had zero variance, indicating that they did not provide useful information for training. Therefore, these features were removed from the dataset before proceeding with the model training phase. Additionally, windowing was employed to create a contextual framework for the time series data. Window sizes ranging from 5 to 100 were utilized to capture the temporal dependencies within the data. The rearranged and windowed data were then used for the subsequent model training process.

The model training and optimization of the JAEC (joint autoencoder-classifier) model was conducted using the DEAP library, which is a Genetic Algorithms (GA) based optimization framework [132]. The GA optimization process involved tuning the hyper-parameters of the model to find the best configuration. The hyper-parameters used in the GA evolutions were manually selected after a series of experimental runs. These parameters were chosen to guide the optimization process and improve the performance of the model. During the training and evaluation process, different percentages of the available training and test data were used. The data was divided into regions, starting from the beginning of each run. This division into regions allowed for a better understanding of how the fault developed over time within each run. By leveraging the GA optimization framework and carefully selecting the hyper-parameters, we were able to train and optimize the JAEC model to achieve improved performance and accuracy in fault classification.

5.5.2. Emission Estimation

The estimation of NO_x and SO_x emissions followed a similar approach to the fault classification task, with some variations. The preprocessing steps, including data merging, missing value imputation, and data normalization, were applied as before. However, windowing was not performed for this task. To estimate the NO_x and SO_x emissions, we utilized the XGBoost library [133], which is a popular gradient boosting framework. Hyper-parameter optimization was also carried out to fine-tune the model's performance and achieve accurate estimations of the gas emissions. An interesting

aspect of this task was the utilization of the fault label, which provided additional information for investigating the relationship between the fault occurrences and the gas emissions. This allowed for a deeper understanding of how faults in the engine system impact the emissions of NO_x and SO_x gases. Overall, by adapting the fault classification framework and employing the XGBoost library, we successfully estimated the NO_x and SO_x emissions in relation to different fault scenarios, contributing to a better understanding of the emissions patterns in the ship engine room.

5.5.3. Results and Discussion

Results for the optimized fault classification model can be found in Table 5.2 for various percentages of the training data and the full test data. The accuracy (ACC) and F1 metrics show similar performance, which is expected given the balanced nature of the dataset for each fault class. The confusion matrix in Figure 5.3 illustrates how the accuracy increases with the addition of new training data. The slight decrease in accuracy observed when transitioning from 60% to 80% of the training data can be attributed to the random initializations of the models. With a smaller amount of data, there was a higher degree of confusion between the M0000 and M2508 fault classes. However, as the percentage of training data increased, this confusion decreased significantly from 21.44% to 2.7%. Similar improvements were observed for other instances of confusion between fault classes. These findings demonstrate the effectiveness of using an optimized model for fault classification, as the model's performance improves with an increasing amount of training data, resulting in reduced confusion between fault classes.

The experimentation with different percentages of the training and test data provided us with the scores presented in Table 5.3. The highest scores were achieved when an equal percentage of train and test data was used, indicated by the diagonal entries in the table. When new test samples were added to the left of the diagonal, the scores increased. This can be attributed to the fact that these new samples were coming from known regions, as they were part of the training data. On the other hand,

Table 5.2: Classification results for different training percentages.

	Test (100%)	
Train (%)	ACC	F1
20	61.78	60.56
40	82.34	81.32
60	91.78	91.78
80	91.14	91.12
100	93.61	93.63

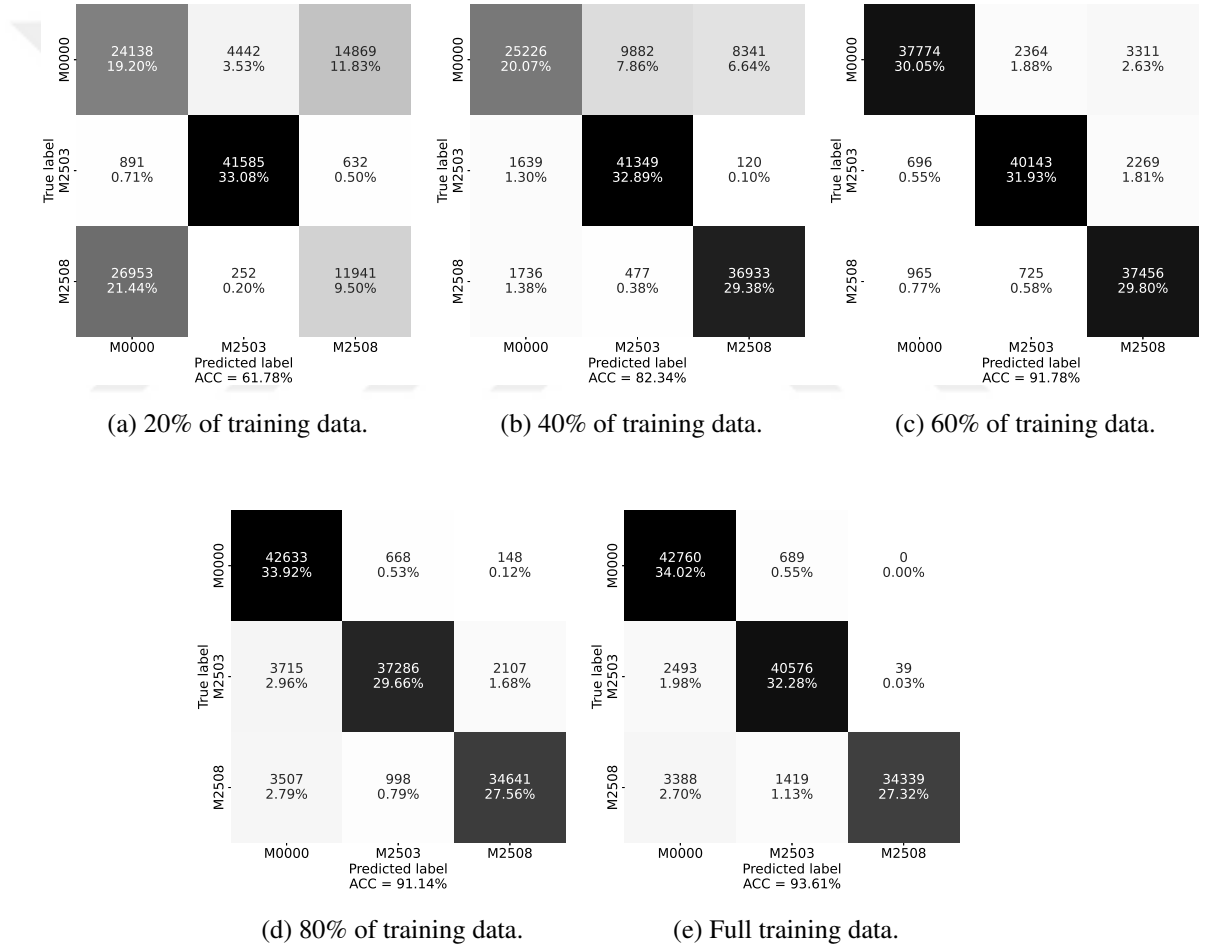


Figure 5.3: Confusion matrices for varying amount of training data.

when new test samples were added to the right of the diagonal, the scores decreased. This is because these new samples were coming from unknown regions that were not represented in the training data. It is worth noting the exception observed in the first

row of the data in Table 5.3, where the training data was 20% and the test samples were increased from 20% to 40%. In this case, we suspect that the adjustments made to the designated initial conditions at the very beginning of the exercise caused fluctuations in the sensor data. However, as the number of samples increased from a similar region where the fault had not fully developed yet, the effect of these fluctuations was mitigated. Overall, these observations highlight the importance of using a representative training data set that covers a wide range of fault conditions to achieve optimal classification performance.

Table 5.3: Classification results for varying amount of train and test data.

Test(%)	20		40		60		80		100	
Train(%)	ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1
20	79.77	79.83	85.39	85.39	73.22	72.88	65.35	64.78	61.78	60.56
40	78.87	79.13	90.55	90.65	90.42	90.45	85.67	85.25	82.34	81.32
60	79.15	79.12	90.68	90.68	94.00	94.00	93.70	93.69	91.78	91.78
80	73.98	74.01	88.37	88.58	92.51	92.63	94.48	94.55	91.14	91.12
100	73.17	73.23	88.01	88.25	92.28	92.41	94.31	94.39	93.61	93.63

Using the best classification model, we conducted an analysis to evaluate the impact of each feature on the accuracy of the model. Initially, we calculated a base score for the model using the full training and test data. We then proceeded to iteratively assess the contribution of each feature by introducing noise to a specific feature and recalculating the score. The difference between the base score and the new score provided an indication of the contribution of each feature. By sorting these contributions, we obtained a rank for each feature. The results of this analysis, displayed in Figure 5.4, revealed that the top-ranking features were directly associated with the combustion process in Cylinder 1, including various temperatures and emissions. Notably, the removal of features with a negative contribution led to an increase in accuracy from 93.61% to 95.94%. This represented a 2.5% improvement in the overall classification performance, which aligned with our expectations. These findings underscore the significance of the identified features in accurately classifying faults, particularly those

related to the combustion process in Cylinder 1. Furthermore, the removal of features with a negative impact enhanced the overall performance of the classification model.

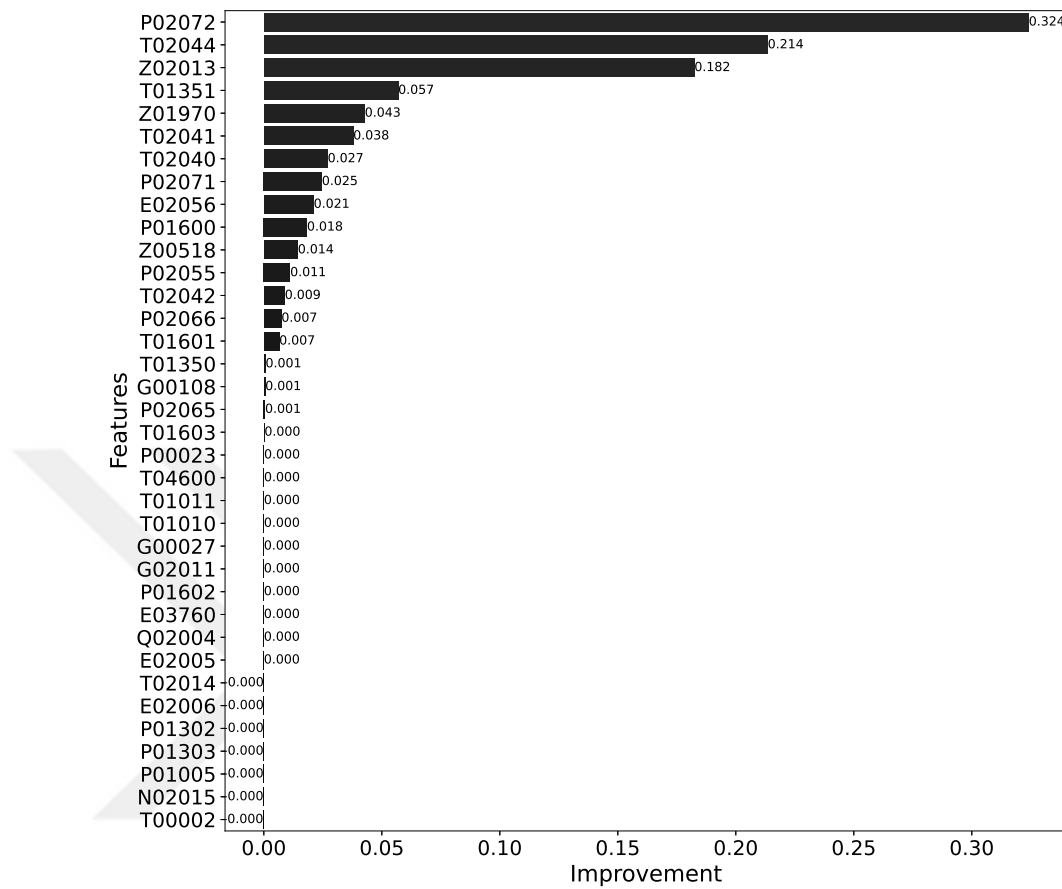


Figure 5.4: Contribution of features to the fault classification task.

The estimation results for NO_x and SO_x gases are presented in Table 5.4 and Table 5.5 respectively. As anticipated, the emissions were the lowest when the ship engine operated without any faults (M0000). The smaller FAU and FAL scores indicated that having knowledge of the ship load contributed to more accurate emissions estimation. Introducing different estimators for ship loads led to a reduction of up to 50% in the MAE score for the full dataset. Comparing the gas emissions of M2503 and M2508, it was observed that the MAE and RMSE scores for M2508 were higher than those for M2503. This discrepancy may suggest complications in the burning process during the development of the M2508 fault, leading to higher variance in emissions. To provide further context, the mean and standard deviation of NO_x and SO_x emissions for the

dataset are provided in Table 5.6 and Table 5.7 respectively. It is noteworthy that our MAE and RMSE scores are comparable to the respective standard deviations, indicating that our estimations align well with the variability in the dataset. These findings highlight the effectiveness of our approach in estimating NO_x and SO_x emissions in relation to different fault scenarios and ship loads. The results demonstrate the importance of considering fault conditions and accurate ship load estimations in achieving more precise emissions estimations.

Table 5.4: NO_x estimations for different faults and ship loads.

\Ship Load	ALL Conditions		FAL		FAU	
Data in use\	MAE	RMSE	MAE	RMSE	MAE	RMSE
FULL Data	0.0213	0.0478	0.0094	0.0337	0.0092	0.0276
M0000	0.0021	0.0030	0.0017	0.0023	0.0018	0.0028
M2503	0.0079	0.0115	0.0066	0.0093	0.0056	0.0076
M2508	0.0164	0.0554	0.0151	0.0575	0.0208	0.0562

Table 5.5: SO_x estimations for different faults and ship loads.

\Ship Load	ALL Conditions		FAL		FAU	
Data in use\	MAE	RMSE	MAE	RMSE	MAE	RMSE
FULL Data	0.0053	0.0161	0.0031	0.0084	0.0027	0.0111
M0000	0.0005	0.0009	0.0004	0.0006	0.0005	0.0007
M2503	0.0068	0.0098	0.0053	0.0075	0.0028	0.0038
M2508	0.0039	0.0188	0.0030	0.0114	0.0041	0.0201

5.6. Summary

In this study, we created a new dataset using MC90-V Engine Room Simulator. The dataset provides unique challenges in terms of predictive maintenance:

- Health index generation: Predicting the health state of the machinery, i.e. Cylinder

Table 5.6: Mean and standard deviations of NOx emissions in the dataset for different faults and ship loads.

\Ship Load	ALL Conditions		FAL		FAU	
Data in use\	MEAN	STD	MEAN	STD	MEAN	STD
FULL Data	14.1854	1.0699	15.0664	0.5027	13.2622	0.6433
M0000	13.9247	1.0893	15.0275	0.0207	12.8496	0.0390
M2503	14.6247	1.0358	15.4461	0.5228	13.7989	0.7186
M2508	13.9818	0.9141	14.7040	0.4593	13.1176	0.4605

Table 5.7: Mean and standard deviations of SOx emissions in the dataset for different faults and ship loads.

\Ship Load	ALL Conditions		FAL		FAU	
Data in use\	MEAN	STD	MEAN	STD	MEAN	STD
FULL Data	13.0589	0.4050	13.1646	0.4764	12.9478	0.2718
M0000	12.8619	0.0584	12.9208	0.0041	12.8043	0.0039
M2503	13.4252	0.5029	13.6387	0.5666	13.2101	0.3037
M2508	12.8650	0.1027	12.9179	0.0637	12.8013	0.1046

1.

- Fault identification and classification: Identifying the state of the machinery, whether it is operating normal, and predicting the fault type if it is not.
- Remaining useful life prediction (RUL): Predicting time to failure for the machinery.
- Failure prediction: Predicting whether the machinery will fail in a given period of time.

Our primary objective was to identify and classify faults, specifically focusing on Cylinder 1 injection valve nozzle wear and Cylinder 1 injection valve nozzle clogging. To achieve classification, we employed an end-to-end trained joint autoencoder classifier model. The optimized models achieved an accuracy score of 93.61%. Through further

exploration of feature contributions and mitigating negative effects, we were able to improve the model accuracy to 95.94%.

Additionally, we investigated NO_x and SO_x emission estimations for various faults and ship loads. Our findings indicated that with knowledge of the ship load, working conditions, and engine health state, we could achieve up to a 50% improvement in estimations using the complete dataset. We also validated our assumption that M0000 runs resulted in lower emissions.

We believe that the joint autoencoder-classifier model will be valuable for other time series estimation tasks in different domains, particularly where operating conditions play a significant role.

6. CONCLUSIONS AND FUTURE WORK

6.1. Conclusions

This research focused on the development of deep learning-based predictive maintenance methods within the context of Industry 4.0. The research aimed to address several research questions, including leveraging publicly available datasets for maximum value in prognostics, identifying suitable deep learning techniques for prognostic research, handling limitations of scarce failure data, and effectively incorporating operating conditions into deep learning architectures for improved prognostic performance.

To achieve these objectives, a comprehensive deep learning framework was developed. This framework encompassed fault classification and remaining useful life (RUL) prediction, utilizing techniques such as autoencoders, LSTM (Long Short-Term Memory), and genetic algorithms. One of the main contributions of this research was the introduction of an end-to-end joint autoencoder-regressor architecture for data-driven RUL prediction. Two versions of the model, JAER-CNN and JAER-IT, were proposed, incorporating convolutional neural networks and InceptionTime-based autoencoders, respectively. Both versions employed LSTM-based regressors and integrated the operating conditions of the process into the autoencoder's latent space, enhancing the regression problem. Genetic algorithms were used to optimize the model's hyper-parameters. The architecture was validated on the C-MAPSS Dataset benchmark, demonstrating its capability to predict RUL for different systems and scenarios. Notably, the proposed JAER-IT model exhibited competitive performance when compared to recent research on the C-MAPSS dataset. Joint autoencoder architecture is also used as a classification framework on a ship engine room dataset to identify and classify faults. To achieve classification, we employed an end-to-end trained joint autoencoder classifier (JAEC) model.

A novel fault detection-based RUL labeling method was also introduced, utilizing an iterative classification approach to detect the start of degradation in such systems. Additionally, a new dataset was developed, providing valuable resources for health

index generation, fault identification and classification, remaining useful life prediction, and failure prediction.

The findings of this research highlight the effectiveness of the proposed deep learning-based joint autoencoder architecture for fault detection and RUL prediction. Furthermore, the novel fault detection-based RUL labeling method addresses the challenge of limited labeled data in industrial systems. These contributions contribute to the advancement of predictive maintenance practices in Industry 4.0.

It is important to note that deep learning techniques require substantial amounts of labeled data, which may not always be available in industrial systems. The proposed fault detection algorithm addresses this challenge by assisting the framework in accurately labeling the data.

In conclusion, this research significantly contributes to the field of predictive maintenance by introducing an effective deep learning framework for fault detection and RUL prediction. The results demonstrate the capability of the proposed architecture to predict RUL in different systems and scenarios, while the fault detection-based RUL labeling method provides an innovative approach to overcome limitations in labeled data. The development of a new dataset further supports future research in health index generation, fault identification and classification, remaining useful life prediction, and failure prediction. These findings have practical implications for optimizing maintenance strategies, improving equipment reliability, and reducing operational costs in various industrial applications.

6.2. Future Work

It is recommended that future research continues to explore interpretability and explainability of deep learning models in the context of predictive maintenance. Additionally, further investigations can focus on expanding the application areas and datasets, considering emerging technologies and advancements in the field. Collaboration between researchers, industry professionals, and domain experts will be vital in advancing the field of deep learning-based predictive maintenance in Industry 4.0.

REFERENCES

- [1] Hornby A. S., (2004), *Oxford Advanced Learner's Dictionary*, 6th Edition, Oxford University Press.
- [2] Holmberg K., Adgar A., Arnaiz A., Jantunen E., Mascolo J., and Mekid S., (2010), *E-maintenance*, Springer.
- [3] "Maintaining virtual system uptime in today's transforming it infrastructure." (2016), <https://www.stratus.com/assets/aberdeen-maintaining-virtual-systems-uptime.pdf>, Access date: July 4, 2023.
- [4] "Downtime costs auto industry \$22k/minute - survey." (2006), <https://news.thomasnet.com/companystory/downtime-costs-auto-industry-22k-minute-survey-481017>, Access date: July 4, 2023.
- [5] Scheffer C. and Girdhar P., (2004), *Practical Machinery Vibration Analysis and Predictive Maintenance*, Elsevier Science.
- [6] Huang C.-G., Huang H.-Z., and Li Y.-F., (2019), "A bidirectional lstm prognostics method under multiple operational conditions," *IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS*, 66 (11), 8792-8802.
- [7] Zhang B., Zhang S., and Li W., (2019), "Bearing performance degradation assessment using long short-term memory recurrent network," *COMPUTERS IN INDUSTRY*, 106, 14-29.
- [8] Halil Y. and Karadayi M., (2014), "Titreşim Analizi ile Pompalarda Arıza Tesbiti ve Kestirimci Bakım İçin Örnek Bir Çalışma," *Tesisat Mühendisliği*, 22 (140), 36-51.
- [9] Di Fiore E., Ferraro A., Galli A., Moscato V., and Sperli G., (2022), "An anomalous sound detection methodology for predictive maintenance," *EXPERT SYSTEMS WITH APPLICATIONS*, 209.
- [10] Liu R., Yang B., and Hauptmann A. G., (2020), "Simultaneous bearing fault recognition and remaining useful life prediction using joint-loss convolutional neural network," *IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS*, 16 (1), 87-96.
- [11] Luo B., Wang H., Liu H., Li B., and Peng F., (2019), "Early fault detection of machine tools based on deep learning and dynamic identification," *IEEE*

TRANSACTIONS ON INDUSTRIAL ELECTRONICS, 66 (1), 509-518.

- [12] Ma M. and Mao Z., (2021), “Deep-convolution-based lstm network for remaining useful life prediction,” *IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS*, 17 (3), 1658-1667.
- [13] Deutsch J. and He D., (2018), “Using deep learning-based approach to predict remaining useful life of rotating components,” *IEEE TRANSACTIONS ON SYSTEMS MAN CYBERNETICS-SYSTEMS*, 48 (1), 11-20.
- [14] Li X., Zhang W., and Ding Q., (2019), “Deep learning-based remaining useful life estimation of bearings using multi-scale feature extraction,” *RELIABILITY ENGINEERING & SYSTEM SAFETY*, 182, 208-218.
- [15] Yang B., Liu R., and Zio E., (2019), “Remaining useful life prediction based on a double-convolutional neural network architecture,” *IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS*, 66 (12), 9521-9530.
- [16] Zhang Y., Xiong R., He H., and Pecht M. G., (2018), “Long short-term memory recurrent neural network for remaining useful life prediction of lithium-ion batteries,” *IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY*, 67 (7), 5695-5705.
- [17] Li H., Zhao W., Zhang Y., and Zio E., (2020), “Remaining useful life prediction using multi-scale deep convolutional neural network,” *APPLIED SOFT COMPUTING*, 89.
- [18] He B., Liu L., and Zhang D., (2021), “Digital twin-driven remaining useful life prediction for gear performance degradation: A review,” *JOURNAL OF COMPUTING AND INFORMATION SCIENCE IN ENGINEERING*, 21 (3).
- [19] Wang B., Lei Y., Li N., and Wang W., (2021), “Multiscale convolutional attention network for predicting remaining useful life of machinery,” *IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS*, 68 (8), 7496-7504.
- [20] Maschler B. and Weyrich M., (2021), “Deep transfer learning for industrial automation: A review and discussion of new techniques for data-driven machine learning,” *IEEE INDUSTRIAL ELECTRONICS MAGAZINE*, 15 (2), 65-75.
- [21] Basora L., Olive X., and Dubot T., (2019), “Recent advances in anomaly detection methods applied to aviation,” *AEROSPACE*, 6 (11).
- [22] Zonta T., da Costa C. A., da Rosa Righi R., de Lima M. J., da Trindade E. S.,

- and Li G. P., (2020), "Predictive maintenance in the industry 4.0: A systematic literature review," *Computers & Industrial Engineering*, 150, 106889.
<https://www.sciencedirect.com/science/article/pii/S0360835220305787>.
- [23] Wang B., Lei Y., Li N., and Yan T., (2019), "Deep separable convolutional network for remaining useful life prediction of machinery," *MECHANICAL SYSTEMS AND SIGNAL PROCESSING*, 134.
- [24] Sayyad S., Kumar S., Bongale A., Kamat P., Patil S., and Kotecha K., (2021), "Data-Driven Remaining Useful Life Estimation for Milling Process: Sensors, Algorithms, Datasets, and Future Directions," *IEEE Access*, 9, 110255–110286.
- [25] Bengtsson M. and Lundström G., (2018), "On the importance of combining "the new" with "the old" – one important prerequisite for maintenance in industry 4.0," *Procedia Manufacturing*, 25, 118-125.
- [26] Lei Y., Li N., Guo L., Li N., Yan T., and Lin J., (2018), "Machinery health prognostics: A systematic review from data acquisition to RUL prediction," *Mechanical Systems and Signal Processing*, 104, 799–834.
- [27] Mao W., He J., and Zuo M. J., (2020), "Predicting remaining useful life of rolling bearings based on deep feature representation and transfer learning," *IEEE TRANSACTIONS ON INSTRUMENTATION AND MEASUREMENT*, 69 (4), 1594-1608.
- [28] Althubaiti A., Elasha F., and Teixeira J., (2022), "Fault diagnosis and health management of bearings in rotating equipment based on vibration analysis – a review," *Journal of Vibroengineering*, 24 (1), 46-74. cited By 10.
- [29] Ji D., Wang C., Li J., and Dong H., (2021), "A review: data driven-based fault diagnosis and rul prediction of petroleum machinery and equipment," *SYSTEMS SCIENCE & CONTROL ENGINEERING*, 9 (1), 724-747.
- [30] Kumar A., Gandhi C. P., Zhou Y., Kumar R., and Xiang J., (2020), "Latest developments in gear defect diagnosis and prognosis: A review," *MEASUREMENT*, 158.
- [31] Sun C., Ma M., Zhao Z., Tian S., Yan R., and Chen X., (2019), "Deep transfer learning based on sparse autoencoder for remaining useful life prediction of tool in manufacturing," *IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS*, 15 (4), 2416-2425.

- [32] Wang B., Lei Y., Yan T., Li N., and Guo L., (2020), "Recurrent convolutional neural network: A new framework for remaining useful life prediction of machinery," *NEUROCOMPUTING*, 379, 117-129.
- [33] Shen S., Sadoughi M., Li M., Wang Z., and Hu C., (2020), "Deep convolutional neural networks with ensemble learning and transfer learning for capacity estimation of lithium-ion batteries," *APPLIED ENERGY*, 260.
- [34] Ren L., Zhao L., Hong S., Zhao S., Wang H., and Zhang L., (2018), "Remaining useful life prediction for lithium-ion battery: A deep learning approach," *IEEE ACCESS*, 6, 50587-50598.
- [35] Shen S., Sadoughi M., Chen X., Hong M., and Hu C., (2019), "A deep learning method for online capacity estimation of lithium-ion batteries," *JOURNAL OF ENERGY STORAGE*, 25.
- [36] Luo K., Chen X., Zheng H., and Shi Z., (2022), "A review of deep learning approach to predicting the state of health and state of charge of lithium-ion batteries," *JOURNAL OF ENERGY CHEMISTRY*, 74, 159-173.
- [37] Ming W., Sun P., Zhang Z., Qiu W., Du J., Li X., Zhang Y., Zhang G., Liu K., Wang Y., and Guo X., (2023), "A systematic review of machine learning methods applied to fuel cells in performance evaluation, durability prediction, and application monitoring," *INTERNATIONAL JOURNAL OF HYDROGEN ENERGY*, 48 (13), 5197-5228.
- [38] De Santo A., Galli A., Gravina M., Moscato V., and Sperli G., (2022), "Deep learning for hdd health assessment: An application based on lstm," *IEEE TRANSACTIONS ON COMPUTERS*, 71 (1), 69-80.
- [39] Chen Z., Wu M., Zhao R., Guretno F., Yan R., and Li X., (2021), "Machine remaining useful life prediction via an attention-based deep learning approach," *IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS*, 68 (3), 2521-2531.
- [40] Zhang J., Jiang Y., Wu S., Li X., Luo H., and Yin S., (2022), "Prediction of remaining useful life based on bidirectional gated recurrent unit with temporal self-attention mechanism," *RELIABILITY ENGINEERING & SYSTEM SAFETY*, 221.
- [41] Li X., Zhang W., Ma H., Luo Z., and Li X., (2020), "Data alignments in machinery remaining useful life prediction using deep adversarial neural networks," *KNOWLEDGE-BASED SYSTEMS*, 197.

- [42] Zhao R., Wang D., Yan R., Mao K., Shen F., and Wang J., (2018), "Machine health monitoring using local feature-based gated recurrent unit networks," *IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS*, 65 (2), 1539-1548.
- [43] Huang M., Liu Z., and Tao Y., (2020), "Mechanical fault diagnosis and prediction in iot based on multi-source sensing data fusion," *Simulation Modelling Practice and Theory*, 102. cited By 55.
- [44] Souza R. M., Nascimento E. G. S., Miranda U. A., Silva W. J. D., and Lepikson H. A., (2021), "Deep learning for diagnosis and classification of faults in industrial rotating machinery," *COMPUTERS & INDUSTRIAL ENGINEERING*, 153.
- [45] Zhu J., Chen N., and Shen C., (2020), "A new deep transfer learning method for bearing fault diagnosis under different working conditions," *IEEE SENSORS JOURNAL*, 20 (15), 8394-8402.
- [46] Zhu J., Chen N., and Peng W., (2019), "Estimation of bearing remaining useful life based on multiscale convolutional neural network," *IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS*, 66 (4), 3208-3216.
- [47] Chen Y., Peng G., Zhu Z., and Li S., (2020), "A novel deep learning method based on attention mechanism for bearing remaining useful life prediction," *APPLIED SOFT COMPUTING*, 86.
- [48] Ren L., Sun Y., Cui J., and Zhang L., (2018), "Bearing remaining useful life prediction based on deep autoencoder and deep neural networks," *JOURNAL OF MANUFACTURING SYSTEMS*, 48, 71-77.
- [49] Peng W., Ye Z.-S., and Chen N., (2020), "Bayesian deep-learning-based health prognostics toward prognostics uncertainty," *IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS*, 67 (3), 2283-2293.
- [50] Li T., Zhao Z., Sun C., Yan R., and Chen X., (2021), "Hierarchical attention graph convolutional network to fuse multi-sensor signals for remaining useful life prediction," *RELIABILITY ENGINEERING & SYSTEM SAFETY*, 215.
- [51] Ma Y., Wu L., Guan Y., and Peng Z., (2020), "The capacity estimation and cycle life prediction of lithium-ion batteries using a new broad extreme learning machine approach," *JOURNAL OF POWER SOURCES*, 476.
- [52] Li W., Sengupta N., Dechent P., Howey D., Annaswamy A., and Sauer D. U.,

- (2021), “One-shot battery degradation trajectory prediction with deep learning,” *JOURNAL OF POWER SOURCES*, 506.
- [53] Zhang W., Li X., and Li X., (2020), “Deep learning-based prognostic approach for lithium-ion batteries with adaptive time-series prediction and on-line validation,” *MEASUREMENT*, 164.
- [54] Khumprom P. and Yodo N., (2019), “A data-driven predictive prognostic model for lithium-ion batteries based on a deep learning algorithm,” *ENERGIES*, 12 (4).
- [55] Hong J., Lee D., Jeong E.-R., and Yi Y., (2020), “Towards the swift prediction of the remaining useful life of lithium-ion batteries with end-to-end deep learning,” *APPLIED ENERGY*, 278.
- [56] Li X., Zhang W., Ma H., Luo Z., and Li X., (2022), “Degradation alignment in remaining useful life prediction using deep cycle-consistent learning,” *IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS*, 33 (10), 5480-5491.
- [57] Nguyen K. T. P. and Medjaher K., (2019), “A new dynamic predictive maintenance framework using deep learning for failure prognostics,” *RELIABILITY ENGINEERING & SYSTEM SAFETY*, 188, 251-262.
- [58] Li X., Ding Q., and Sun J.-Q., (2018), “Remaining useful life estimation in prognostics using deep convolution neural networks,” *RELIABILITY ENGINEERING & SYSTEM SAFETY*, 172, 1-11.
- [59] Ellefsen A. L., Bjorlykhaug E., Aesoy V., Ushakov S., and Zhang H., (2019), “Remaining useful life predictions for turbofan engine degradation using semi-supervised deep architecture,” *RELIABILITY ENGINEERING & SYSTEM SAFETY*, 183, 240-251.
- [60] Zhang J., Wang P., Yan R., and Gao R. X., (2018), “Long short-term memory for machine remaining life prediction,” *JOURNAL OF MANUFACTURING SYSTEMS*, 48 (C, SI), 78-86.
- [61] Wu J., Hu K., Cheng Y., Zhu H., Shao X., and Wang Y., (2020), “Data-driven remaining useful life prediction via multiple sensor signals and deep long short-term memory neural network,” *ISA TRANSACTIONS*, 97, 241-250.
- [62] Yu W., Kim I. Y., and Mechefske C., (2019), “Remaining useful life estimation using a bidirectional recurrent neural network based autoencoder scheme,”

- [63] da Costa P. R. d. O., Akcay A., Zhang Y., and Kaymak U., (2020), “Remaining useful lifetime prediction via deep domain adaptation,” *RELIABILITY ENGINEERING & SYSTEM SAFETY*, 195.
- [64] Al-Dulaimi A., Zabihi S., Asif A., and Mohammadi A., (2019), “A multimodal and hybrid deep neural network model for remaining useful life estimation,” *COMPUTERS IN INDUSTRY*, 108, 186-196.
- [65] Tsui K.-L., Chen N., Zhou Q., Hai Y., and Wang W., (2015), “Prognostics and health management: A review on data driven approaches,” *Mathematical Problems in Engineering*, 2015, 1-17.
- [66] Khan S. and Yairi T., (2018), “A review on the application of deep learning in system health management,” *Mechanical Systems and Signal Processing*, 107, 241–265.
- [67] Jolliffe I., (2002), *Principal Component Analysis*, Springer New York, NY.
- [68] Tharwat A., Gaber T., Ibrahim A., and Hassanien A. E., (2017), “Linear discriminant analysis: A detailed tutorial,” *Ai Communications*, 30, 169-190,.
- [69] Ren L., Sun Y., Wang H., and Zhang L., (2018), “Prediction of bearing remaining useful life with deep convolution neural network,” *IEEE ACCESS*, 6, 13041-13049.
- [70] Breiman L., (2001), “Random forests,” *Mach. Learn.*, 45 (1), 5–32.
- [71] Friedman J. H., (2001), “Greedy function approximation: A gradient boosting machine,” *The Annals of Statistics*, 29 (5), 1189–1232.
- [72] Rasmussen C. E. and Williams C. K. I., (2005), *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*, The MIT Press.
- [73] Back T., Fogel D. B., and Michalewicz Z., (2000), *Evolutionary Computation I: Basic Algorithms and Operators*, 1st Edition, IOP Publishing Ltd.
- [74] Matthews B., (1975), “Comparison of the predicted and observed secondary structure of t4 phage lysozyme,” *Biochimica et Biophysica Acta (BBA) - Protein Structure*, 405 (2), 442–451.
- [75] “Case western reserve university bearing data center.” (2018),

<https://engineering.case.edu/bearingdatacenter>, Access date: July 4, 2023.

- [76] Qiu H., Lee J., Lin J., and Yu G., (2006), "Wavelet filter-based weak signature detection method and its application on rolling element bearing prognostics," *Journal of Sound and Vibration*, 289 (4), 1066-1090.
- [77] "Machinery fault database (mafaulda) - multivariate time-series acquired by sensors on a spectraquest's machinery fault simulator (mfs)." (2018), <https://www02.smt.ufrj.br/offshore/mfs/>, Access date: July 4, 2023.
- [78] Nectoux P., Gouriveau R., Medjaher K., Ramasso E., Chebel-Morello B., Zerhouni N., and Varnier C., (2012), "PRONOSTIA : An experimental platform for bearings accelerated degradation tests," In: *IEEE International Conference on Prognostics and Health Management*, 1-8, Denver, Colorado, USA, Jun.
- [79] Cheng C., Ma G., Zhang Y., Sun M., Teng F., Ding H., and Yuan Y., (2020), "A deep learning-based remaining useful life prediction approach for bearings," *IEEE-ASME TRANSACTIONS ON MECHATRONICS*, 25 (3), 1243-1254.
- [80] Dempsey P., Handschuh R., and Afjeh A., (2002), "Spiral bevel gear damage detection using decision fusion analysis," In: *Proceedings of the Fifth International Conference on Information Fusion*, 1, 94-100, Annapolis, MD, USA, July.
- [81] Wang B., Lei Y., Li N., and Li N., (2020), "A hybrid prognostics approach for estimating remaining useful life of rolling element bearings," *IEEE Transactions on Reliability*, 69 (1), 401-412.
- [82] Zhang W., Li X., Ma H., Luo Z., and Li X., (2021), "Transfer learning using deep representation regularization in remaining useful life prediction across operating conditions," *RELIABILITY ENGINEERING & SYSTEM SAFETY*, 211.
- [83] Ding Y., Jia M., and Cao Y., (2021), "Remaining useful life estimation under multiple operating conditions via deep subdomain adaptation," *IEEE TRANSACTIONS ON INSTRUMENTATION AND MEASUREMENT*, 70.
- [84] Cheng Y., Hu K., Wu J., Zhu H., and Shao X., (2021), "A convolutional neural network based degradation indicator construction and health prognosis using bidirectional long short-term memory network for rolling bearings," *ADVANCED ENGINEERING INFORMATICS*, 48.
- [85] "Milling data set." (2007), <https://www.nasa.gov/content/prognostics-center-of->

excellence-data-set-repository, Access date: July 4, 2023.

- [86] “Phm society conference data challenge 2010.” (2010), https://phmsociety.org/phm_competition/2010-phm-society-conference-data-challenge/, Access date: July 4, 2023.
- [87] “Backblaze dataset.” (2023), <https://www.backblaze.com/b2/hard-drive-test-data.html>, Access date: July 4, 2023.
- [88] Severson K. A., Attia P. M., Jin N., Perkins N., Jiang B., Yang Z., Chen M. H., Aykol M., Herring P. K., Fraggadakis D., Bazant M. Z., Harris S. J., Chueh W. C., and Braatz R. D., (2019), “Data-driven prediction of battery cycle life before capacity degradation,” *Nature Energy*, 4 (5), 383–391. <https://doi.org/10.1038/s41560-019-0356-8>.
- [89] “Battery data set.” (2007), <https://www.nasa.gov/content/prognostics-center-of-excellence-data-set-repository>, Access date: July 4, 2023.
- [90] “Rwth battery dataset.” (2021), <https://git.rwth-aachen.de/isea/battery-degradation-trajectory-prediction>, Access date: July 4, 2023.
- [91] “Turbofan Engine Degradation Simulation Data Set,” (2008), <https://www.nasa.gov/content/prognostics-center-of-excellence-data-set-repository>, Access date: July 4, 2023.
- [92] Zio E., (2022), “Prognostics and health management (phm): Where are we and where do we (need to) go in theory and practice,” *Reliability Engineering & System Safety*, 218, 108119.
- [93] Liu H., Liu Z., Jia W., and Lin X., (2021), “Remaining useful life prediction using a novel feature-attention-based end-to-end approach,” *IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS*, 17 (2), 1197-1207.
- [94] Miao H., Li B., Sun C., and Liu J., (2019), “Joint learning of degradation assessment and rul prediction for aeroengines via dual-task deep lstm networks,” *IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS*, 15 (9), 5023-5032.
- [95] Hu C.-H., Pei H., Si X.-S., Du D.-B., Pang Z.-N., and Wang X., (2020), “A prognostic model based on dbn and diffusion process for degrading bearing,” *IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS*, 67 (10), 8767-8777.

- [96] Ding Y., Jia M., Miao Q., and Huang P., (2021), “Remaining useful life estimation using deep metric transfer learning for kernel regression,” *Reliability Engineering and System Safety*, 212. cited By 54.
- [97] Xiang S., Qin Y., Luo J., Pu H., and Tang B., (2021), “Multicellular lstm-based deep learning model for aero-engine remaining useful life prediction,” *RELIABILITY ENGINEERING & SYSTEM SAFETY*, 216.
- [98] Zhang A., Wang H., Li S., Cui Y., Liu Z., Yang G., and Hu J., (2018), “Transfer learning with deep recurrent neural networks for remaining useful life estimation,” *APPLIED SCIENCES-BASEL*, 8 (12).
- [99] Li X., Xu Y., Li N., Yang B., and Lei Y., (2023), “Remaining useful life prediction with partial sensor malfunctions using deep adversarial networks,” *IEEE/CAA Journal of Automatica Sinica*, 10 (1), 121-134. cited By 13.
- [100] Ragab M., Chen Z., Wu M., Foo C. S., Kwoh C. K., Yan R., and Li X., (2021), “Contrastive adversarial domain adaptation for machine remaining useful life prediction,” *IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS*, 17 (8), 5239-5249.
- [101] Davies S., (2007), “Automotive special report - a holistic approach to quality,” *Manufacturing Engineer*, 86 (1), 8-13.
- [102] Bradley A. P., (1997), “The use of the area under the roc curve in the evaluation of machine learning algorithms,” *Pattern Recogn.*, 30 (7), 1145–1159.
- [103] Sparks T., (2012), *Solid-Liquid Filtration: A User's Guide to Minimizing Cost and Environmental Impact, Maximizing Quality and Productivity*, Elsevier Science. Butterworth-Heinemann/ICHEME series.
- [104] Skaf Z., Eker O. F., and Jennions I. K., (2015), “A simple state-based prognostic model for filter clogging,” *Procedia CIRP*, 38, 177 - 182. Proceedings of the 4th International Conference on Through-life Engineering Services.
- [105] Kluyver T., Ragan-Kelley B., Pérez F., Granger B., Bussonnier M., Frederic J., Kelley K., Hamrick J., Grout J., Corlay S., Ivanov P., Avila D., Abdalla S., and Willing C., (2016), “Jupyter notebooks – a publishing format for reproducible computational workflows,” In: Loizides F. and Schmidt B., editors, *Positioning and Power in Academic Publishing: Players, Agents and Agendas*.
- [106] Pedregosa F., Varoquaux G., Gramfort A., Michel V., Thirion B., Grisel O.,

- Blondel M., Prettenhofer P., Weiss R., Dubourg V., Vanderplas J., Passos A., Cournapeau D., Brucher M., Perrot M., and Duchesnay E., (2011), “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, 12, 2825–2830.
- [107] “Lofo (leave one feature out) importance,” (2018), <https://github.com/aerdem4/lofo-importance>, Access date: July 4, 2023.
- [108] Li X., Ding Q., and Sun J. Q., (2018), “Remaining useful life estimation in prognostics using deep convolution neural networks,” *Reliability Engineering and System Safety*, 172 (11), 1–11.
- [109] Heimes F. O., (2008), “Recurrent neural networks for remaining useful life estimation,” In: *International Conference on Prognostics and Health Management*, 1-6, Denver, CO, USA, Oct.
- [110] Zhao Z., Bin Liang, Wang X., and Lu W., (2017), “Remaining useful life prediction of aircraft engine based on degradation pattern learning,” *Reliability Engineering and System Safety*, 164 (457), 74–83.
- [111] Malhotra P., Tv V., Ramakrishnan A., Anand G., Vig L., Agarwal P., and Shroff G., (2016), “Multi-sensor prognostics using an unsupervised health index based on lstm encoder-decoder,” *1st SIGKDD Workshop on Machine Learning for Prognostics and Health Management*, abs/1608.06154.
- [112] Ellefsen A., Bjorlykhaug E., Æsøy V., and Zhang H., (2019), “An unsupervised reconstruction-based fault detection algorithm for maritime components,” *IEEE Access*, PP, 1-1.
- [113] Reynolds D., (2015), “Gaussian mixture models,” In: Li S. Z. and Jain A. K., editors, *Encyclopedia of Biometrics*, 2nd Edition, Boston, MA: Springer US.
- [114] Zhang C., Lim P., Qin A. K., and Tan K. C., (2017), “Multiobjective Deep Belief Networks Ensemble for Remaining Useful Life Estimation in Prognostics,” *IEEE Transactions on Neural Networks and Learning Systems*, 28 (10), 2306–2318.
- [115] Li Q., Gao Z., Tang D., and Li B., (2016), “Remaining useful life estimation for deteriorating systems with time-varying operational conditions and condition-specific failure zones,” *Chinese Journal of Aeronautics*, 29 (3), 662–674.

- [116] Rahaman N., Baratin A., Arpit D., Draxler F., Lin M., Hamprecht F., Bengio Y., and Courville A., (2019), “On the spectral bias of neural networks,” In: *Proceedings of the 36th International Conference on Machine Learning* (Chaudhuri K. and Salakhutdinov R., editors), 5301–5310, Long Beach, CA, USA, Jun.
- [117] Mildenhall B., Srinivasan P. P., Tancik M., Barron J. T., Ramamoorthi R., and Ng R., (2020), “Nerf: Representing scenes as neural radiance fields for view synthesis,” In: *Computer Vision – ECCV 2020* (Vedaldi A., Bischof H., Brox T., and Frahm J.-M., editors), 405–421, Glasgow, Scotland, UK, Aug.
- [118] Caruana R., (1997), “Multitask Learning,” *Machine Learning*, 28.
- [119] Ellefsen A. L., Ushakov S., Aesoy V., and Zhang H., (2019), “Validation of Data-Driven Labeling Approaches Using a Novel Deep Network Structure for Remaining Useful Life Predictions,” *IEEE Access*, 7, 71563–71575.
- [120] Listou Ellefsen A., Bjørlykhaug E., Aesoy V., Ushakov S., and Zhang H., (2019), “Remaining useful life predictions for turbofan engine degradation using semi-supervised deep architecture,” *Reliability Engineering and System Safety*, 183 (6), 240–251.
- [121] Wen L., Dong Y., and Gao L., (2019), “A new ensemble residual convolutional neural network for remaining useful life estimation,” *Mathematical Biosciences and Engineering*, 16 (2), 862–880.
- [122] Al-Dulaimi A., Zabihi S., Asif A., and Mohammadi A., (2019), “A multimodal and hybrid deep neural network model for Remaining Useful Life estimation,” *Computers in Industry*, 108, 186–196.
- [123] Mo H., Lucca F., Malacarne J., and Iacca G., (2020), “Multi-Head CNN-LSTM with Prediction Error Analysis for Remaining Useful Life Prediction,” *Conference of Open Innovation Association, FRUCT*, 2020-September, 164–171.
- [124] Jiang Y., Lyu Y., Wang Y., and Wan P., (2020), “Fusion network combined with bidirectional LSTM network and multiscale CNN for useful life estimation LSTM network and multiscale CNN for useful life estimation,” In: *12th International Conference on Advanced Computational Intelligence*, 620–627, Dali, Yunmnan, China, Mar.
- [125] Chaoub A., Voisin A., Cerisara C., and Iung B., (2021), “Learning representations with end-to-end models for improved remaining useful life

- prognostics,” *CoRR*, abs/2104.05049.
- [126] Li X., Xu Y., Li N., Yang B., and Lei Y., (2023), “Remaining useful life prediction with partial sensor malfunctions using deep adversarial networks,” *IEEE/CAA Journal of Automatica Sinica*, 10 (1), 121-134.
 - [127] Szegedy C., Liu W., Jia Y., Sermanet P., Reed S., Anguelov D., Erhan D., Vanhoucke V., and Rabinovich A., (2015), “Going deeper with convolutions,” In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, June, 1–9, Boston, MA, USA, June.
 - [128] Szegedy C., Ioffe S., Vanhoucke V., and Alemi A. A., (2017), “Inception-v4, inception-ResNet and the impact of residual connections on learning,” In: *Proceedings of the 31st AAAI Conference on Artificial Intelligence, AAAI’17*, San Francisco, California, USA, Feb.
 - [129] Ismail Fawaz H., Lucas B., Forestier G., Pelletier C., Schmidt D. F., Weber J., Webb G. I., Idoumghar L., Muller P. A., and Petitjean F., (2020), “InceptionTime: Finding AlexNet for time series classification,” *Data Mining and Knowledge Discovery*, 34 (6), 1936–1962.
 - [130] Heo S. and Lee J. H., (2018), “Fault detection and classification using artificial neural networks,” *IFAC-PapersOnLine*, 51 (18), 470-475. 10th IFAC Symposium on Advanced Control of Chemical Processes ADCHEM 2018.
 - [131] “Keras deep learning for humans,” (2015), <https://keras.io>, Access date: July 4, 2023.
 - [132] Fortin F.-A., De Rainville F.-M., Gardner M.-A., Parizeau M., and Gagné C., (2012), “DEAP: Evolutionary algorithms made easy,” *Journal of Machine Learning Research*, 13, 2171–2175.
 - [133] Chen T. and Guestrin C., (2016), “Xgboost,” In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD’16*, San Francisco, California, USA, Aug.

BIOGRAPHY

Kürşat İNCE is a computer engineer with a passion for cutting-edge technologies and a notable background in the field. He completed his undergraduate and graduate studies at Bilkent University, earning degrees in Computer Engineering and Information Science in 1996 and Computer Engineering in 1999, respectively. Kürşat's research area focuses on Deep Learning-Based Predictive Maintenance Methods, aiming to explore the potential of deep learning techniques for fault classification and remaining useful life prediction. Throughout his career, Kürşat has played pivotal roles in the development of technology in Türkiye. He was instrumental in creating the first Turkish firewall, HAVELSAN Interceptor Firewall. He excelled as a software engineer in various military projects and later led the Software Development Environment Standardization Project, enhancing software development processes at HAVELSAN. Currently, Kürşat serves as the R&D Coordinator at HAVELSAN, driving innovation and research efforts. He also contributes to the data science community as an organizer of the Data Istanbul Meetup Group. Kürşat İnce's academic achievements and diverse professional experiences have earned him recognition and respect in the field. His dedication to advancing Deep Learning-Based Predictive Maintenance Methods reflects his commitment to pushing the boundaries of technology and contributing to the field's growth.

APPENDICES

A. Publications within the Scope of the Thesis Study

İnce K. and Genç Y. "Data Analysis for Automobile Brake Fluid Fill Process Leakage Detection using Machine Learning Methods," Innovations in Intelligent Systems and Applications Conference (ASYU), Izmir, Turkey, 2019, pp. 1-5,
doi: 10.1109/ASYU48272.2019.8946399

İnce, K., Sirkeci, E., and Genç, Y. "Remaining Useful Life Prediction for Experimental Filtration System: A Data Challenge." PHM Society European Conference, 5(1).
doi: 10.36001/phme.2020.v5i1.1317

İnce, K. and Genç, Y. "Remaining Useful Life Estimation on Turbofan Engines Using Joint Autoencoder-Regression," 28th Signal Processing and Communications Applications Conference (SIU), Gaziantep, Turkey, 2020, pp. 1-4,
doi: 10.1109/SIU49456.2020.9302365.

İnce, K., Ceylan, U., Erdoğan, N. N., Sirkeci, E., and Genç, Y. "Fault Detection and Classification for Robotic Test-bench" PHM Society European Conference, 6(1), 7.
doi: 10.36001/phme.2021.v6i1.3040

İnce K., Ceylan U., and Y. Genç, "Remaining Useful Life Prediction on C-MAPSS Dataset via Joint Autoencoder-Regression Architecture," 2022 30th Signal Processing and Communications Applications Conference (SIU), Safranbolu, Turkey, 2022, pp. 1-4, doi: 10.1109/SIU55565.2022.9864796.

İnce, K., Koçak, G., and Genç, Y. "Joint Autoencoder-Classifer Model for Malfunction Identification and Classification on Marine Diesel Engine Diagnostics Data" PHM Society European Conference, 7(1), 211–218.
doi: 10.36001/phme.2022.v7i1.3335

K. İnce and Y. Genç, "Joint autoencoder-regressor deep neural network for remaining useful life prediction," Engineering Science and Technology, an International Journal,

B. Systematic Review Queries

Systematic review process started with querying the databases for predictive maintenance and deep learning. Full set of query terms are given below.

Scopus:

```
(TITLE-ABS-KEY("predictive maintenance")  
OR TITLE-ABS-KEY("condition based maintenance")  
OR TITLE-ABS-KEY("remaining useful life"))  
AND TITLE-ABS-KEY("deep learning")  
AND (PUBYEAR > 2018))
```

Web of Science:

```
TS= (("predictive maintenance"  
OR "remaining useful life"  
OR "condition based maintenance")  
AND "deep learning"))
```

IEEE Xplore:

```
(( "All Metadata": "predictive maintenance"  
OR Search\_All: "remaining useful life"  
OR Search\_All: "condition based maintenance")  
AND ( "All Metadata": "deep learning"))
```

Science Direct:

```
("predictive maintenance"  
OR "remaining useful life"  
OR "condition based maintenance")  
AND "deep learning")
```

