People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research
Mohamed Khider University - Biskra
Faculty of Exact Sciences and Sciences of Nature and Life
Computer Science Department

Order Number: :.....

THESIS

In Candidacy for the Degree of
DOCTOR 3rd CYCLE IN COMPUTER SCIENCE
Option : Artificial Intelligence

TITLE

Deep Learning for predictive maintenance

Presented by **Ikram REMADNA**

Defended on:

In front of the jury composed of:

Mr. Laid KAHLOUL	Professor at University of Biskra	President
Mr. Labib Sadek TERRISSA	Professor at University of Biskra	Supervisor
Mr. Salim BITAM	Professor at University of Biskra	Examiner
Mr. Houari TOUBAKH	Associate Professor at University Ouargla	Examiner
Mr. Mohamed SAYAH	Associate Professor at University Oran1	Examiner
Mr. Noureddine ZERHOUNI	Professor at ENSMM, Besançon, France	Guest

Academic year : **2022 – 2023**

Abstract

Recently, with the appearance of Industry 4.0 (I4.0), machine learning (ML) within artificial intelligence (AI), industrial Internet of things (IIoT) and cyber-physical system (CPS) have accelerated the development of a data-orientated applications such as predictive maintenance (PdM). PdM applied to asset-dependent industries has led to operational cost savings, productivity improvements and enhanced safety management capabilities. In addition, predictive maintenance strategies provide useful information concerning the source of the failure or malfunction, reducing unnecessary maintenance operations.

The concept of prognostics and health management (PHM) has appeared as a predictive maintenance process. PHM has become an unavoidable tendency in smart manufacturing to offer a reliable solution for handling industrial equipment's health status. This later requires efficient and effective system health monitoring methods, including processing and analysing massive machinery data to detect anomalies and perform diagnosis and prognosis. Prognostics is considered a key PHM process with capabilities for predicting future states, mainly based on predicting the residual lifetime during which a machine can perform its intended function, i.e., estimating the remaining useful life (RUL) of a system. The prognostic research domain is far from being mature, which is still new and explains the various challenges that must be addressed. Therefore, the work presented in this thesis will mainly focus on the prognostic of monitored machinery from an RUL estimation point of view using Deep Learning (DL) algorithms.

Capitalising on the recent success of the DL, this dissertation introduces methods and algorithms dedicated to predictive maintenance. We focused on improving the performance of aero-engine prognostic, particularly in estimating an accurate RUL using ensemble learning and deep learning. To this end, two contributions have been proposed, and the results obtained were validated by an extensive comparative analysis using public C-MAPSS turbofan engine benchmark datasets. The first contribution, for RUL predictions, we proposed two-hybrid methods based on the promising DL architectures to leverage the power of the multimodal and hybrid deep neural network in order to capture various information at different time intervals and ultimately achieve more accurate RUL predictions. The proposed end-to-end deep architectures jointly optimise the feature reduction and RUL prediction steps in a hierarchical manner, intending to achieve data representation in low dimensionality and minimal variable redundancy while preserving critical asset degradation information with minimal preprocessing effort. The second contribution, in a practical situation, RUL is usually affected by uncertainty. Therefore, we proposed an innovative RUL estimation strategy that assesses degrading machinery's health status (provides the probabilities of system failure in different time windows) and provides the prediction of RUL window.

Keywords: *Prognostics and Health Management (PHM)*, *Remaining useful life (RUL)*, *Predictive Maintenance (PdM)*, *C-MAPSS dataset*, *Ensemble learning*, *Deep learning*.

Résumé

A l'époque actuelle, le développement d'applications basées sur les données telles que la maintenance prédictive (PdM) traverse une évolution remarquable grâce à l'application de l'industrie 4.0 (I4.0), l'apprentissage automatique (ML), l'Internet industriel des objets (IIoT) et les systèmes cyber-physiques (CPS). L'application de la PdM dans le secteur industriel a permis de réduire les coûts opérationnels, d'améliorer la productivité et de renforcer les capacités de gestion de la sécurité. En outre, les stratégies de maintenance prédictive fournissent des informations utiles concernant la source de la panne ou du dysfonctionnement, réduisant ainsi les opérations de maintenance inutiles.

Le prognostics and health management (PHM) est un processus incontournable de maintenance prédictive car il offre une solution fiable de gestion de l'état de santé des équipements industriels. Il nécessite des méthodes efficaces de surveillance de l'état de santé des systèmes, notamment le traitement et l'analyse de données massives des machines pour détecter les anomalies et effectuer des diagnostics et des pronostics. Le pronostic est considéré comme un processus clé du PHM avec des capacités de prédiction des états futurs, principalement basé sur la prévision de la durée de vie résiduelle pendant laquelle une machine peut remplir sa fonction prévue, c'est-à-dire l'estimation de la durée de vie utile restante (RUL) d'un système. Le domaine de recherche sur le pronostic est loin d'être mature, ce qui est encore nouveau et explique les différents défis qui doivent être relevés. Par conséquent, le travail présenté dans cette thèse se concentrera principalement sur le pronostic des machines surveillées du point de vue de l'estimation du RUL en utilisant des algorithmes de l'apprentissage profond (AP).

Capitalisant sur le succès de ce dernier, cette thèse introduit des méthodes et des algorithmes dédiés à la maintenance prédictive. Nous nous sommes concentrés sur l'amélioration des performances du pronostic des moteurs aéronautiques, en particulier dans l'estimation d'un RUL précis en utilisant l'ensemble learning et apprentissage profond. Pour cela, deux contributions ont été proposées, ainsi que les résultats obtenus ont été validés par une analyse comparative approfondie en utilisant des données publiques de référence de turbomoteurs C-MAPSS. La première contribution consiste à proposer deux méthodes hybrides basées sur les architectures prometteuses de l'AP pour la prédiction de RUL, en tirant parti de la puissance du réseau neuronal profond multimodal et hybride pour capturer diverses informations à différents intervalles de temps et, par conséquent, obtenir des prédictions de RUL plus précises. Les architectures proposées optimisent conjointement les étapes de réduction des caractéristiques et de prédiction de RUL de manière hiérarchique, dans le but de réduire la dimensionnalité des données et la redondance des variables, tout en préservant les informations essentielles. En pratique, le RUL est généralement affecté par l'incertitude. Pour cette raison, dans la deuxième contribution, nous avons proposé une stratégie d'estimation du RUL qui évalue l'état de santé des machines dégradées (fournit les probabilités de défaillance du système dans différentes fenêtres temporelles) et fournit plutôt la prédiction d'une fenêtre de RUL.

الملخص

في الآونة الأخيرة ، مع ظهور الصناعة 4.0 (I4.0) ، أدى التعلم الآلي (ML) في الذكاء الاصطناعي (AI) ، وإنترنت الأشياء الصناعي (IIoT) والنظام السيبراني الفيزيائي (CPS) إلى تسريع تطوير البيانات الموجهة للتطبيقات مثل الصيانة التنبؤية (PdM) . أدى تطبيق PdM في الصناعات المعتمدة على الأصول إلى توفير التكاليف التشغيلية وتحسين الإنتاجية وتعزيز قدرات إدارة السلامة. بالإضافة إلى ذلك ، توفر استراتيجيات الصيانة التنبؤية معلومات مفيدة تتعلق بمصدر الفشل أو العطل ، مما يقلل من عمليات الصيانة غير الضرورية.

ظهر مفهوم التكهنات وإدارة الصحة (PHM) كعملية صيانة تنبؤية. أصبح PHM اتجاهًا لا مفر منه في التصنيع الذكي لتقديم حل موثوق به للتعامل مع الحالة الصحية للمعدات الصناعية. يتطلب هذا لاحقًا طرقًا فعالة لمراقبة صحة النظام ، بما في ذلك معالجة وتحليل بيانات الآلات الضخمة لاكتشاف الحالات الشاذة وإجراء التشخيص. تعتبر التكهنات عملية PHM رئيسية ذات إمكانيات للتنبؤ بالحالات المستقبلية ، تعتمد بشكل أساسي على التنبؤ بالعمر المتبقي الذي يمكن من خلاله الجهاز أداء وظيفته المقصودة ، أي تقدير العمر الإنتاجي المتبقي (RUL) للنظام. مجال البحث في التكهن لا يزال جديدًا ويشرح التحديات المختلفة التي يجب معالجتها. لذلك ، سيركز العمل المقدم في هذه الأطروحة بشكل أساسي على الإنذار للآلات المراقبة من وجهة نظر تقدير RUL باستخدام خوارزميات التعلم العميق (DL) .

بالاستفادة من نجاح DL ، تقدم هذه الأُطرُوحَة طرقًا وخوارزميات مخصصة للصيانة التنبؤية. ركزنا على تحسين أداء تنبؤات المحركات الهوائية ، خصوصا في تقدير RUL دقيقة باستخدام التعلم الجماعي والتعلم العميق. من أجل هذا، تم اقتراح مساهمتين ، وتم التحقق من صحة النتائج التي تم الحصول عليها من خلال تحليل مقارن مكثف باستخدام مجموعات البيانات المعيارية للمحرك المروحي C-MAPSS العامة. تتمثل المساهمة الأولى في اقتراح طريقتين هجنتين تستندان إلى معماريات DL الواعدة للتنبؤ RUL ، والاستفادة من قوة الشبكة العصبية العميقة متعددة الوسائط والهجينة لالتقاط معلومات متنوعة في فترات زمنية مختلفة ، وبالتالي الحصول على تنبؤات RUL أكثر دقة. تعمل البنى المقترحة بشكل مشترك على تحسين خطوات تقليل الميزات والتنبؤ RUL بطريقة هرمية ، بهدف تقليل أبعاد البيانات والتكرار المتغير ، مع الحفاظ على المعلومات الأساسية. في الممارسة العملية ، يتأثر RUL عادةً بعدم اليقين. لهذا السبب ، في المساهمة الثانية ، اقترحنا إستراتيجية تقدير RUL التي تقيم الحالة الصحية للآلات المتدهورة (توفر احتمالات فشل النظام في نوافذ زمنية مختلفة) وتوفر التنبؤ بنافذة RUL .

List of Publications

Journal Papers

1. Remadna, I., Terrissa, L. S., Al Masry, Z., and Zerhouni, N. (2022). RUL Prediction Using a Fusion of Attention-Based Convolutional Variational AutoEncoder and Ensemble Learning Classifier. IEEE Transactions on Reliability.

2. Remadna, I., Terrissa, S. L., Sayah, M., Ayad, S., and Zerhouni, N. (2022). Boosting RUL Prediction Using a Hybrid Deep CNN-BLSTM Architecture. Automatic Control and Computer Sciences, 56(4), 300-310.

3. Zroug, S., Remadna, I., Kahloul, L., Terrissa, S. L., and Benharzallah, S. (2022). Towards performance evaluation prediction in WSNs using artificial neural network multi-perceptron. Cluster Computing, 1-19.

4. Remadna, I., Terrissa, L. S., Ayad, S., and Zerhouni, N. (2021). RUL Estimation Enhancement using Hybrid Deep Learning Methods. International Journal of Prognostics and Health Management, 12(1).

5. Houfani, D., Slatnia, S., Kazar, O., Zerhouni, N., Saouli, H., and Remadna, I. (2020). Breast cancer classification using machine learning techniques: a comparative study. Medical Technologies Journal, 4(2), 535-544.

Conference Papers

1. Remadna, I., Terrissa, S. L., Zemouri, R., and Ayad, S. (2018, March). An overview on the deep learning based prognostic. In 2018 International Conference on Advanced Systems and Electric Technologies (IC_ASET) (pp. 196-200). IEEE.

2. Remadna, I., Terrissa, S. L., Zemouri, R., Ayad, S., and Zerhouni, N. (2019, July). Unsupervised feature reduction techniques with bidirectional GRU neural network for aircraft engine RUL estimation. In International Conference on Advanced Intelligent Systems for Sustainable Development (pp. 496-506). Springer, Cham.

3. Remadna, I., Terrissa, S. R., Ayad, L., Zemouri, S., and Zerhouni, N. Remaining Useful Life Estimation Using Principal Component Analysis with Bidirectional LSTM Neural Network. (2019, July). In International Conference on Control, Automation and Diagnosis (ICCAD).

4. Houfani, D., Slatnia, S., Kazar, O., Zerhouni, N., Saouli, H., and Remadna, I. A practical comparative study of machine learning algorithms for breast cancer diagnosis.(2019, January) In Conference: International Congress on Health Sciences and Medical TechnologiesAt: Tlemcen, Algeria.

5. Remadna, I., Terrissa, S. L., Zemouri, R., Ayad, S., and Zerhouni, N. (2020, May). Leveraging the Power of the Combination of CNN and Bi-directional Lstm Networks for Aircraft Engine RUL Estimation. In 2020 Prognostics and Health Management Conference (PHM-Besançon) (pp. 116-121). IEEE.

6. Zemouri, R., Al Masry, Z., Remadna, I., Terrissa, S. L., and Zerhouni, N. Hybrid architecture of deep convolutional variational auto-encoder for remaining useful life prediction. In Conference: Proceedings of the 30th European Safety and Reliability Conference and the 15th Probabilistic Safety Assessment and Management ConferenceAt: Italy.

7. Zroug, S., Remadna, I., Kahloul, L., Benharzallah, S., and Terrissa, S. L. (2021, July). Leveraging the power of machine learning for performance evaluation prediction in wireless sensor networks. In 2021 International Conference on Information Technology (ICIT) (pp. 864-869). IEEE.

8. Attache, S., Remadna, I., Terrissa, L. S., Maouche, I., and Zerhouni, N. (2022). IoT Based Prediction of Active and Passive Earth Pressure Coefficients Using Artificial Neural Networks. In International Conference on Networked Systems (pp. 252-262). Springer, Cham.

9. Zerari, A., Djedidi, O., Kahloul, L., Carlo, R., and Remadna, I. (2022). Paediatric Bone Age Assessment from Hand X-ray Using Deep Learning Approach. In International Conference on Computing Systems and Applications (pp. 373-383). Springer, Cham.

10. Attache, S., Terrissa, L. S., Remadna, I., Ayad, S., Madaci, F., and Zerhouni, N. (2022, May). Predicting COVID-19 Outbreak in Algeria Using Long Short-Term Memory Networks. In 2022 7th International Conference on Image and Signal Processing and their Applications (ISPA) (pp. 1-5). IEEE.

Chapter book

1. Remadna, I., Terrissa, S. L., Zemouri, R., Ayad, S., and Zerhouni, N. (2019, July). Unsupervised feature reduction techniques with bidirectional GRU neural network for aircraft engine RUL estimation. In International Conference on Advanced Intelligent Systems for Sustainable Development (pp. 496-506). Springer, Cham.

Poster

1. Remadna, I., Terrissa, S. L., and Ayad, S. Deep learning for predictive maintenance. Poster in LINFI Doctoral Day, 29 November 2017.

2. Remadna, I., Terrissa, S. L., and Ayad, S. Deep learning for RUL Estimation. Poster in LESIA Doctoral Day, 28-30 January 2018.

Acknowledgement

First and foremost, praises and thanks to **Allah**.

I would like to express my deep and sincere gratitude to my research supervisor **Pr.Labib Sadek Terrissa** for giving me the opportunity to do research and providing invaluable guidance throughout this research. His dynamism, vision, sincerity and motivation have deeply inspired me.

I would also like to thank deeply **Pr.Noureddine Zerhouni**, **Dr.Zaina Almasry**, **Dr.Ryad zemouri**, and **Dr.Soheyb Ayad** for their collaboration and support to complete this thesis successfully.

I also thank all the members of the jury **Pr. Laid Kahloul**, **Pr. Salim Bitam**, **Dr. Houari Toubakh** , and **Dr.Mohamed Sayah** for the time they spend to review this work, I am grateful for the attention they paid to my work.
Finally I thank all those who helped me in some way for the realizadtion of this work.

Ikram Remadna

Contents

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

AI	**A**rtificial **I**ntelligence
ANN	**A**rtificial **N**eural **N**etwork
ACVAE	**A**ttention **C**onvolutional **V**ariational **A**uto**E**ncoder
AdaBoost	**A**daptive **B**oosting
AHPS	**A**utomatic **Hyp**erparameters **S**election
AE	**A**uto-**E**ncoder
AM	**A**ttention **M**echanism
AUC	**A**rea **U**nder **C**urve
BDGRU	**B**i-**d**irectional **G**ated **R**ecurrent **U**nit
BO	**B**ayesian **O**ptimization
BDRNN	**B**i-**D**irectional **RNN**
BDLSTM	**B**i-**d**irectional **L**ong-**S**hort **T**erm **M**emory
BPTT	**B**ackpropagation **T**hrough **T**ime
CPS	**C**yber-**P**hysical **S**ystem
CBM	**C**ondition-**B**ased **M**aintenance
CNN	**C**onvolutional **N**eural **N**etwork
CAE	**C**onvolutional **A**uto-**E**ncoder
CM	**C**orrective **M**aintenance
DL	**D**eep **L**earning
DNNs	**D**eep **N**eural **N**etworks
DAE	**D**enoising **A**uto **E**ncoder
EOL	**E**nd **O**f **L**ife
FC	**F**ully **C**onnected
GP	**G**aussian **P**rocess
GAN	**G**enerative **A**dversarial **N**etwork
GRU	**G**ated **R**ecurrent **U**nit
GBM	**G**radient **B**oosting **M**achine
HPC	**H**igh **P**ressure **C**ompressor
HPT	**H**igh **P**ressure **T**urbine
I4.0	**I**ndustry **4.0**
IIoT	**I**ndustrial **I**nternet **o**f **T**hings
ISOMAP	**Iso**metric **Map**ping
KL	**K**ullback-**L**eibler
KNN	**K**-**N**earest **N**eighbor
KPCA	**K**ernel **PCA**
LSTM	**L**ong-**S**hort **T**erm **M**emory

LR	**L**inear **R**egression
LDA	**L**inear **D**iscriminant **A**nalysis
LPC	**L**ow **P**ressure **C**ompressor
LPT	**L**ow **P**ressure **T**urbine
ML	**M**achine **L**earning
MLP	**M**ulti-**L**ayer **P**erceptron
MTL	**M**ulti-**T**ask **L**earning
MSE	**M**ean **S**quared **E**rror
NB	**N**aïve **B**ayes
OEE	**O**verall **E**quipment **E**ffectiveness
PdM	**P**re**d**ictive **M**aintenance
PvM	**P**re**v**entive **M**aintenance
PHM	**P**rognostics and **H**ealth **M**anagement
PCA	**P**rincipal **C**omponent **A**nalysis
RUL	**R**emaining **U**seful **L**ife
ROC	**R**eceiver **O**peration **C**haracteristic
RF	**R**andom **F**orest
RMSprop	**R**oot **M**ean **S**quare **prop**agation
RTF	**R**un **T**o **F**ailure
ReLU	**Re**ctified **L**inear **U**nit
RMSE	**R**oot **M**ean **S**quare **E**rror
RNN	**R**ecurrent **N**eural **N**etwork
SVM	**S**upport **V**ector **M**achines
SAE	**S**parse **A**uto-**E**ncoder
SGB	**S**tochastic **G**radient **B**oosting
SMOTE	**S**ynthetic **M**inority **O**versampling **T**Echnique
tanh	**h**yperbolic **tan**gent
T-SNE	**T**-distributed **S**tochastic **N**eighbor **E**mbedding
VAE	**V**ariational **A**uto-**E**ncoder

Chapter I

INTRODUCTION

I.1 Context and Motivation

Every engineering asset or its critical machinery, during its service Lifetime, is prone to wear and tear. Its state continuously deteriorates over time until it reaches a breakdown level at which it is unable to perform its intended function, i.e., it fails. Furthermore, according to the Analyst firm Aberdeen Research, the unexpected failure of physical assets is estimated to result in an average loss of over US $260,000 per hour in industries such as mining, oil and gas, aerospace, and industrial manufacturing. Such industries are known as physical asset-dependent industries, as their earnings are heavily dependent upon their asset productivity [22]. However, the failure of assets can not only be a loss in terms of monetary costs resulting from poor machine reliability but also have significant negative consequences. In various cases, the malfunction of a machine can even lead to irreversible environmental damage or safety problems, such as a jet crash caused by engine failure, a rail accident caused by bearing failure, wasted crude oil and ocean pollution caused by mechanical failure on an offshore oil platform, and much more [4]. This points out the necessity of maintaining the most critical machinery before a breakdown can occur.

Availability of such machinery, prevention of unexpected breakdowns, maintainability, optimisation of service, and minimisation of life cycle costs/risks are of major concern for physical asset-dependent industries. Assets failure has prompted such industries to continuously monitor the dynamics and condition of physical assets equipped with multiple sensors that measure different aspects of the behaviour and operating conditions (e.g. ambient temperature, pressure, flying altitude, operating speed) of the assets [79, 96]. Prior to the sixties of the last century, maintenance was only performed after the asset failed. This fail-fix maintenance is called corrective maintenance. Due to the rapid development of the Internet of Things (IoT) and sensing technology with the increasing complexity and criticality of systems, industrials and researchers have shifted from unplanned corrective maintenance practices for non-critical machinery toward planned preventive maintenance (PM) (predict-prevent maintenance) [54, 71, 116]. More specifically, in PM, instead of waiting until the asset fails, maintenance activities are performed either according to a predefined schedule or by performing Condition-Based Maintenance (CBM) based on the current state of the machinery. Although CBM brought many benefits to industries, it still could not anticipate failure. Emphasising prognostics in the CBM framework was the initiative to the emergence of the Predictive Maintenance strategy (PdM), Prognostics and Health Management (PHM) concept, based on forecasting the evolution of degradation in the future and estimating the remaining useful life (RUL) of the asset [55].

PdM is the most cost-optimal maintenance strategy, achieving overall equipment effectiveness (OEE) exceeding 90% [41, 62, 113]. Concretely, PdM maximises an asset's working life and avoids costly breakdowns and lost production time caused by unplanned downtime. Currently, PdM is the state-of-the-art maintenance strategy, as it provides up-to-date diagnostics, probability of failure, and estimating Remaining Useful Life (RUL). The prediction of RUL is at the centre of PHM. RUL is

a crucial characteristic for prognostics, which is the number of times an asset can accomplish its intended task before requiring maintenance [55, 56, 121]. The maintenance decision within PdM could be based on different characteristics such as the remaining useful life (RUL), the reliability or the cost function. Therefore, the RUL form a key element of the PdM in decision-making and maintenance strategies developed to reduce the cost of maintenance. High-accuracy RUL prediction plays a critical role in the PdM for many fields, including manufacturing and various industrial cyber-physical systems. Furthermore, if the accurate mechanical asset RUL is known, manufacturing industries can plan future maintenance in advance and ensure a seamless repair and maintenance process. Since its inaccurate estimation can cause unexpected catastrophic failures.

The increase in complexity of modern physical assets necessitates an increasing number of sensing instruments to accurately measure an asset's behaviour and operating conditions during its life cycle. This increase in sensor measurements translates into large volumes of data representing a great opportunity for the maturity of data-driven PdM methods in industry and academia, mainly relying on machine learning-based computational methods (e.g. linear or non-linear regression) to deliver a more effective and efficient maintenance service. Numerous conventional machine learning methods have been used, including artificial neural network (ANN) [75, 89, 143, 166], support vector machines (SVM) [159], random forest (RF) [25], principal component analysis (PCA) [88] and so on to predict the RUL of various machines such as bearings, milling cutters, engines and drill pipe. However, these techniques are often challenging to describe the behaviour of such a complex engineered system mathematically, it requires the experience of the experts and prior knowledge of signal processing to manually select and extract meaningful features for real fault diagnosis and prognostics issues. Conventional frameworks cannot be updated in real-time and require a great deal of work dealing with large-scale data sets. Therefore, advanced computational methods, known as deep learning algorithms or deep neural networks (DNNs), are necessary to adaptively and automatically process highly non-linear and complex feature abstraction from raw asset data, eliminate the reliance on domain knowledge and manual feature engineering, and overcome the need for data preprocessing prior to ML-based analysis [133, 199]. In comparison, the deep learning algorithm makes it possible to integrate the PHM tasks such as feature extraction, feature selection, and classification/regression, into an end-to-end architecture and jointly optimise all the tasks in a hierarchical fashion. Nevertheless, applying DL techniques in the context of prognostics is still challenging, and the performance still needs improvement to make an accurate and effective maintenance decision. This thesis focuses primarily on developing, implementing and evaluating automatic and efficient data-driven methods for predicting the RUL and deals with some problems encountered with improving RUL estimation. These problems are addressed in the following section.

I.2 Purpose and Research Questions

Safety, reliability, availability, and maintainability are major concerns and indispensable in aeronautical systems that should be addressed due to the harsh working conditions and extended operating hours [33]. The turbofan engine, as an essential and critical component of aircraft, is a complex, highly sophisticated and precise part of the thermal asset, its unexpected breakdown is involved with 60% of airplane issues. Therefore, it is essential to accurately detect upcoming failures, avoiding catastrophic damage or sudden shutdowns that could result in economic and human losses [174]. According to statistics, aircraft engine maintenance costs are approximately 70% of their whole life cycle costs [59]. Therefore, predictive maintenance and monitoring are necessary to build a cost-effective maintenance strategy that detects upcoming degradation by predicting the RUL engines, preventing unplanned downtime and reducing maintenance costs. Therefore, we direct our research towards the estimation of the remaining useful life of a turbofan engine.

Indeed, accurate RUL estimations enable improved decision-making for operations and maintenance of such systems to approach zero downtime, which is a delicate process influenced by several unknowns challenges related to *data quality* and *model selection* that remain unaddressed. Raw asset sensor data (improper asset data representation) is rarely usable by data-driven methods, including ML algorithms, due to characteristics such as large number of sensor channels/variables or even redundant and unnecessary variables and variables with minimal variance (i.e. low information content) [50, 134]. These characteristics subsequently transform the data into lower quality with high dimensionality [12]. It is stated that the curse of dimensionality issue decreases uncertainty [45, 175]. Due to the change of operating conditions, during the continuous monitoring of the system's life cycle, gathered data with wide ranges of measurements is expected to contain discontinuity (Such as the fleets with multiple operating conditions containing discontinuous sensor measurements with wide ranges of parameter variations and multi-modal distributions [4]). Furthermore, in the case of fault diagnosis, faulty data tends to be relatively rare in most situations compared to healthy ones [78]. This latter is an imbalanced data problem that needs to be addressed. Unfortunately, ML algorithms generally perform poorly on lower data quality, which can yield high rates of inaccurate fault identification and/or time-of-failure (RUL) prediction results [96].

Most literary works have neither discussed nor explicitly justified the choice of suitable architecture to solve specific problems since the optimisation of their hyperparameters is based on a trial-and-error approach [2, 6, 7, 73, 97, 99, 131, 172, 188, 194, 195, 200]. However, this approach requires architectural engineering skills and domain expertise and is time-consuming, error-prone and tedious. Therefore, searching for an optimal and efficient architecture is still challenging and limited. Furthermore, some industries are still reluctant to adopt ML/DL algorithms, which cannot trust something they neither understand nor control due to the "black box" nature of ML/DL algorithms that impose a lack of understandability and transparency (i.e., their internal computation mechanisms are unexplainable) [133]. This

enables identifying the key research questions of data-driven prognostics related to the implementation of a prognostics model that will be addressed in this dissertation:

- *RQ1: How to choose an appropriate architecture? How can the prognostic model be improved to accurately and reliably predict RUL? What about combining a spatial feature extractor and a temporal feature extractor for improved prediction performance?*
- *RQ2: Is the performance of the prognostics method strongly influenced by the hyperparameter selection? How is the optimum value for the hyperparameters selected?*
- *RQ3: How can sufficient data quality be achieved? What is the impact of data quality on RUL estimation performance? How to deal with imbalanced data?*
- *RQ4: How to deal with complex multidimensional data? How can raw monitoring data be efficiently processed to obtain relevant features that explicitly and properly reflect failure progression? Can visualisation techniques be used to assess and analyse the quality of features and provide a minimum level of transparency?*
- *RQ5: How to discriminate degradation states and define the threshold setting of states (classes)? Does it take into consideration the opinion of the expert?*
- *RQ6: Does reducing the conflict zone (i.e., the boundary between classes) lead to fewer conflicting decisions for the classification? How can the conflict zone be identified and reduced?*
- *RQ7: How to validate the performances of a prognostics model?*

Thus, this dissertation aims to introduce methods and algorithms dedicated to enhancing the estimation of the RUL of a turbofan engine by addressing the challenges mentioned above. Novel RUL prediction approaches are proposed by applying advanced deep learning architectures that focus on feature reduction, intending to achieve data representation in low dimensionality and minimal variable redundancy while preserving critical asset information with minimal preprocessing effort. The main assumptions and original contributions are discussed in the following sections.

I.3 Assumptions

The contributions made in this thesis aim to enhanced data-driven prognostics by addressing the aforementioned issues. However, these contributions are valid for certain parameters defined by the following assumptions.

- It is assumed that degradation has already been detected.
- Identification of the number of degradation states and adjustment of threshold parameters under the expert assumption. In our study, three states of degradation are assumed, namely: degrading state, transition state and critical state. (As a result of the first assumption, the "steady state" was not considered.)

I.4 Contributions

In this dissertation, the main original contributions are achieved following a thorough literature review on different approaches for prognostics which are as follows:

- **Leveraging the Power of Multimodal and Hybrid Deep Neural Network Techniques for RUL Estimation Enhancement:** The first contribution aims at developing an efficient RUL prediction approach based on advanced end-to-end deep architectures that jointly optimise the feature extraction/reduction and RUL prediction steps in a hierarchical manner. The idea of the hybrid neural network method and the application of parallel multi-model emerged to leverage the power of different models instead of incorporating a single model such as CNN (Convolutional Neural Network), DNN or LSTM (Long-Short Term Memory), which was not previously exploited. It can capture various information at different time intervals and ultimately boost prognostic accuracy. Capitalising on the recent success of DL, two proposed hybrid methods based on promising architectures have been developed, including CNN to capture spatial features and BDGRU (Bi-directional Gated Recurrent Unit) to capture bi-directional temporal dependencies features. The first promising proposed hybrid model adopts CAE (Convolutional Auto-encoder) in an aero-engine prognostic problem to extract automatically useful features with high-level abstractions in Phase I. In Phase II, these CAE features serve as inputs to train the two temporal modeling tools simultaneously in a parallel manner (referred to as the BDLSTM path and BDGRU path) that can capture more robust features and eventually predict the RUL. Although CAE has been applied to different tasks, to the best of our knowledge, this is the first use of CAE in RUL's estimation problem for engine turbofan. For a comprehensive comparison, the second hybrid architecture proposed differently from what has been reported in the literature. It blends the CNN and BDGRU models simultaneously in parallel paths to capture local and temporal features directly from raw sensory data instead of just using CNN for feature extraction (referred to as CNN-BDGRU). The outputs from both paths (CNN and BDGRU) are concatenated to obtain the target RUL. Experiments results will shed light on the impact of combining deep neural network architectures, and the time complexity for both proposed methods is also discussed for comparison purposes.

- **Hybrid architecture of deep CVAE with an attention mechanism for RUL prediction:** The second contribution focuses on solving the problem of uncertainty in RUL prediction, as the prediction of a time interval can cover the whole distribution is more certain (i.e. it has a 100% probability of occurrence) rather than the prediction of a single deterministic RUL value. A new approach based on visual data analysis to predict when an in-service machine will fail is proposed. It consists of Attention Convolutional Variational AutoEncoder (ACVAE) in combination with a soft voting classifier, which was introduced as conducive to the predictive maintenance of aero engines. The starting point of

the proposed method was to automatically extract performance degradation features from multiple sensors using the Attention Convolutional Variational AutoEncoder method. ACVAE effectively integrates convolution calculation with an autoencoder to extract spatial information. Moreover, the power of the attention layer is in dynamically increasing the weights of the useful features in the ACVAE encoding phase to make the network pay attention to these vital features for RUL classes estimation. The primary objective of applying the ACVAE is to provide a more structured, disentanglement and lower-dimensional representation of the data that shows the best class distribution over a 2D latent space and demonstrates how well the ACVAE generalises. Besides that, the encoder, part of the ACVAE, is leveraged for data projection in a 2D visualisation latent space. The input vectors are encoded and displayed in this 2D space, which helps the expert visually analyse the spatial distribution of the training dataset. Under the expert assumption, three degradation classes are then defined according to two thresholds (α_1, α_2), namely: degrading state, transition state and critical state. The expert aims to determine the appropriate threshold setting by minimising the overlapping region between the degradation classes by analysing the spatial distribution. Following that, the RUL is predicted according to the latter degradation classes. The conflict zones are located near the boundaries between classes, which are identified when the classifiers give opposite responses for the same input data. Therefore, the soft voting classifier is used to reduce this conflict zone. It selects the highest probability class by combining the decisions of different classifiers using the probability classes average. Furthermore, Automatic Hyperparameters Selection (AHPS) is used to pick out the best configuration of hyperparameters for our hybrid architecture without being time-consuming and avoid error-prone. Experiments results will shed light on the impact of VAE-based architecture compared to the existing dimension reduction methods (PCA, ISOMAP "Isometric Mapping", and T-SNE "T-distributed stochastic neighbor embedding"), as well as the impact of hyperparameters selection and data quality on performance.

The utility of the proposed approaches is evaluated through the application of four dataset case studies of turbine engines (C-MAPSS NASA) from the PHM'08 challenge that has experienced varying operation conditions and a varying number of faults. In addition, a comparison with results from recent publications is also provided (Chapters IV and V).

I.5 Manuscript organisation

The rest of the thesis is organised as follows:

- Chapter II introduces basic definition of predictive maintenance and the Prognostics and Health Management (PHM) paradigm. It also discusses the benchmarking data sets for system-level prognostics.

- Chapter III provides a brief overview of the relevant theoretical foundations needed to understand deep learning and state data-driven prognostic challenges and issues.
- Chapter IV introduces the first contribution, which aims to develop an efficient RUL prediction approach by testing several deep learning techniques. We present the proposed deep end-to-end architectures that jointly optimise the feature reduction and RUL prediction steps in a hierarchical way, intending to achieve data representation in low dimensionality and minimal variable redundancy while preserving critical asset information with minimal preprocessing effort.
- Chapter V introduces the main contribution regarding developing an innovative RUL estimation strategy that simultaneously assesses the health status of degrading machinery and assigns them to estimate the RUL window.
- Chapter VI presents the general conclusion and discusses the future aims.

Chapter II

Predictive Maintenance and PHM

II.1 Introduction

Companies strive to be more sustainable and competitive, which must satisfy their customers with high-quality products delivered on time. Besides, successful operations in industries such as mining, aviation and industrial manufacturing rely heavily on the continuous operation of complex systems, where unexpected asset failure results in high maintenance costs, lost profits, reduced human safety and hazardous environmental impacts. Behind the scenes, a critical process is a maintenance, which plays a crucial role in improving a company's global competitiveness. As Mr Gross explains, "the success of the maintenance operation should be measured in uptime, not in the number of breakdowns repaired [58]". Therefore, industrials have shifted from unplanned corrective maintenance practices (fail-fix maintenance) toward planned preventive maintenance (PM) (predict-prevent maintenance).

This chapter briefly introduces the basic concepts related to our work. The chapter starts with an overview of Industry 4.0 and maintenance evolution in sections II.2 and II.3, respectively. Following that, an overview of PHM and the role of prognostics is given in II.4. Different benchmarking datasets for system-level prognostics are also summarised, and the occupancy dataset is detailed in section II.5. Finally, section II.6 gives a brief conclusion from what this chapter presents.

II.2 Towards the 4^{th} Industrial Revolution

The emerging of several digital technologies such as Big Data[127], cloud computing [182] or IoT [42] has characterized the rise of the 21^{st} century. These technologies have recently entered our daily life and industrial companies. Industrial companies have evolved rapidly over time through various industrial revolutions, from the steam engine to digitization technology. In this context, several initiatives have emerged to manage these technologies, such as smart manufacturing in the United States, internet+ in China, the industry of the future in France, and Industry 4.0 in Germany. The terminology «Industry 4.0» has gained ground internationally, which was first introduced at the Hanover Fair in 2011 [81].

Historically, the invention of steam engines kick-started the « Industry 1.0 » in the 18^{th} century, basically referred to as the mechanical revolution. It centred around the transition from purely manual production to machines through the use of steam and water as a source for power. The industrial revolution (I2.0) (Electrification) began in the 19^{th} century by discovering electricity and assembly line production. The third industrial revolution (I3.0) is also called the «Digital Revolution» and led to the change from mechanical and analogue systems to digital systems. This revolution focused on the developments of computers, microprocessors (memory-programmable controls), digital cellular phones, and Internet, therefore, automating an entire production process without human assistance. Today, we are witnessing the fourth industrial revolution, which also known as Industry 4.0 [90]. This evolution is derived from advances in digitization and data analysis disciplines to make

plants smarter and more efficient. It is the age of the Cyber-Physical Systems (CPS), Artificial Intelligence (AI), Machine Learning (ML), Internet of Things (IoT), Big Data and Data Mining, Internet of Service (IoS), and Cloud computing.

II.3 Evolution of maintenance

According to the standard [165], Maintenance is considered a set of all technical, administrative and management actions during the system's life cycle, which aims at maintaining or restoring it to a state that can perform the required function. The maintenance actions involve activities of control, verification, replacing and repair.

As mentioned above, manufacturing systems have evolved throughout four industrial revolutions. This industrial evolution has been accompanied by a change in the manufacturing functions such as the maintenance one. This function was evolved from Corrective Maintenance to Predictive Maintenance (See Figure II.1). Techniques for maintenance policies can be categorized into the following main classifications

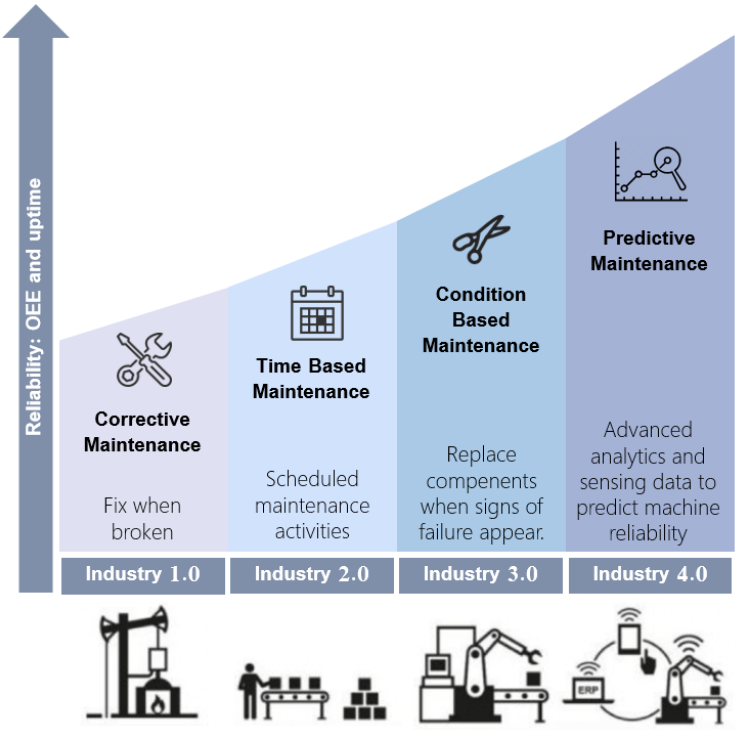


FIGURE II.1: The evolution of maintenance paradigm within the industrial revolutions[121].

- ***Corrective Maintenance (CM) :*** is also known as Run-to-Failure (R2F), reactive, unplanned or breakdown maintenance. It is the oldest and most straightforward manner of intervention, generally carried out only when a breakdown or failure has occurred and aims to re-establish the system to an operating state. This maintenance policy can only be appropriate if the time required for restoring, costs sustained for downtime and repair or failure consequences are

insignificant, and if no immediate safety risk is involved. In other words, such maintenance is suitable for non-critical assets [61]. This strategy leads to two types of intervention: Curative and Palliative, as shown in the characteristic graph (Figure II.2).

- *Palliative Maintenance*: has temporary repair actions intended to restore an asset to a specified state which does not necessarily imply its initial state (function provisional). This maintenance may relieve failure without addressing the causes that curative maintenance should follow.
- *Curative Maintenance*: has durable repair actions, definitively, intended to restore the system to its optimal performance (long-term restoration). This strategy can be decided either immediately following a failure or after palliative maintenance.

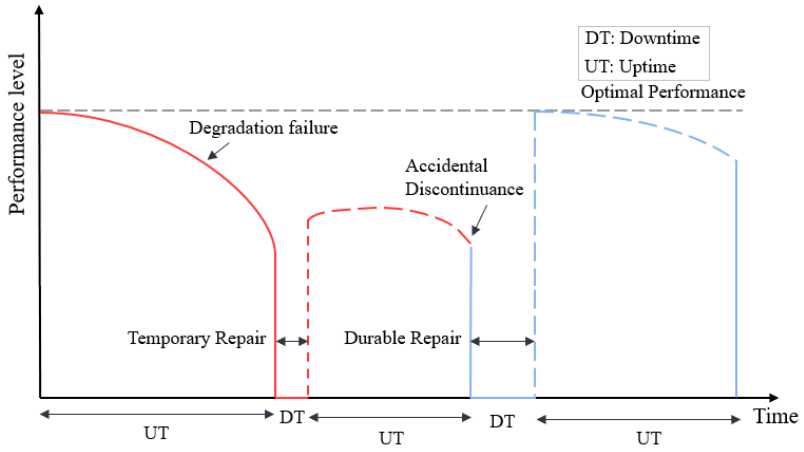


FIGURE II.2: Evolution of performance level in CM.

- ***Preventive Maintenance (PvM) :*** is a planned (or proactive) strategy for maintaining machines, introduced in the 1950s [168]. It is performed based on either predetermined intervals (usually time or event-based triggers) or according to prescribed criteria (following the analysis of the monitored evolution of significant parameters). This strategy aims to extend or sustain useful component life by reducing the probability of its failure during use, reducing the occurrence of unscheduled downtime, and eliminating the causes of serious accidents. Three main types of preventive maintenance are available for implementing an activity of failure avoidance:
  - ***Systematic Preventive Maintenance :*** according to a planned schedule, this strategy of maintenance is carried out periodically at regular and predefined intervals (time-based maintenance) or based on parameters of the utilisation of the machinery (hours of operation or kilometres operated) to ensure that the system retains a sufficient level of dependability, safety and performance. However, due to preventive action, unneeded maintenance actions are taken (over-maintenance), leading to periodic replacement of parts prematurely without prior check-in, increasing operating

costs and wasting resources [87]. As depicted in Figure II.3, no intervention can be taken before a predetermined intervention schedule (UT). Where UT: intervention period at fixed intervals, SPI: systematic preventive intervention.

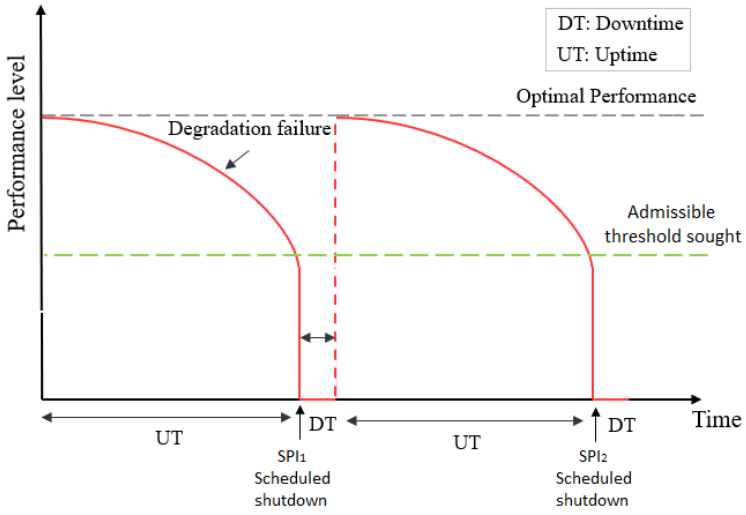


FIGURE II.3: Systematic preventive intervention.

- ***Condition-based Maintenance (CBM) :*** depends upon the actual condition of the component rather than on the operating time. It is intended to reduce the number of unnecessary scheduled time-based maintenance operations, which boots maintenance only when necessary. This method of maintenance is based on monitoring the performance of the asset and/or significant indicators [76, 184]. These later give the signalling that the component is deteriorating and the failure probability is increasing. The main function is to collect data with the industrial machines under operation (such as vibrations and temperature), and identify potential anomalies or degradation through historical analysis of similar equipment and knowledge acquired over time. CBM is based only on the current-state of the equipment and do not include prediction / forecasting of future states of the degrading equipment. Therefore, CBM usually cannot be planned in advance where the date of occurrence of the failure remains uncertain. As depicted in Figure II.4, intervention can be taken when certain indicators show signs of impending underperformance or failure. Where PCI: Condition-based maintenance intervention.
- ***Predictive Maintenance (PdM) :*** is more dynamic than CBM, where the maintenance can be scheduled and executed in prior. PdM is carried out following a forecast of the evolution of the condition of the assets over time (i.e., prognostics). It is based on the continuous monitoring of the asset, as like CBM. Furthermore, it utilizes the prediction tools based on the historical data (such as machine learning methods and statistical inference methods) to project the current state of the asset into the future in order to estimate the uptime before failure (Remaining Useful Life) [113].

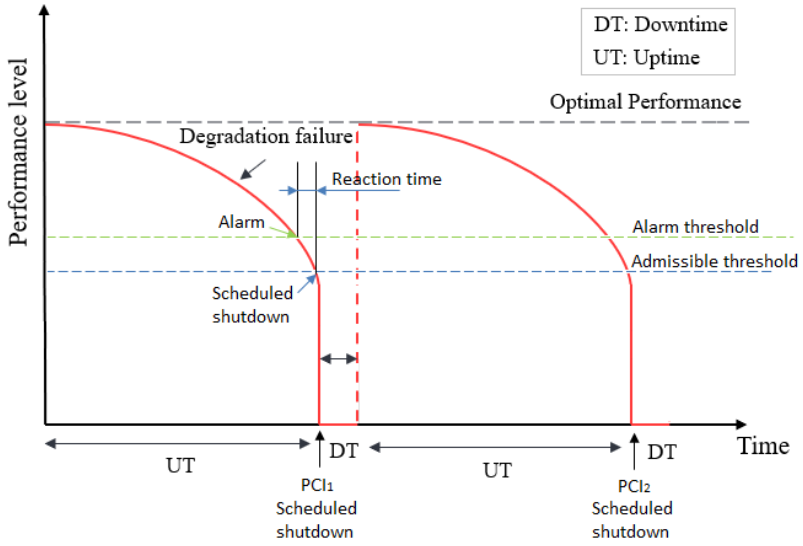


FIGURE II.4: Intervention of Condition-based Maintenance.

As shown in Figure II.5, estimation of remaining operating life enables the machinery to run till healthy state, and provides adequate/scheduled time to plan the required maintenance actions before a breakdown.

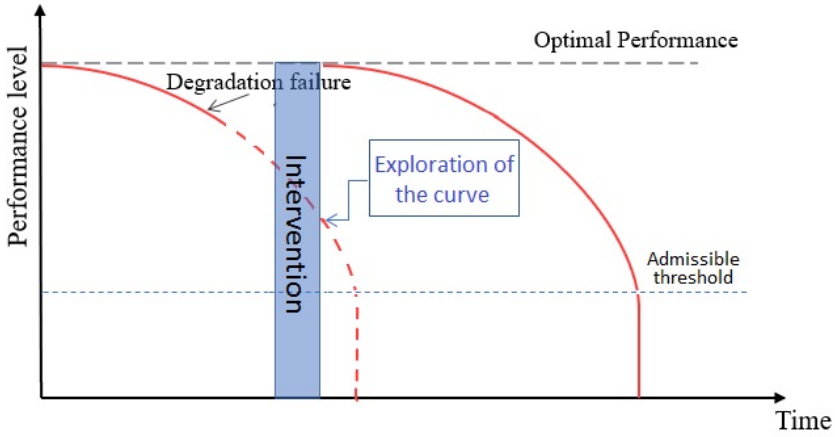


FIGURE II.5: Intervention of Predictive Maintenance.

It is required that any maintenance strategy ought to minimize equipment failure rates, must improve equipment condition, increase equipment operational life / availability, and reduce maintenance activities and costs. An overview of the maintenance classifications is shown in Figure II.1. PdM turned out to be one of the most promising strategies amongst other maintenance strategies that can achieve those characteristics [56]. PdM attracts the attention of the industries; hence, it has been applied in the era of I4.0 because it is capable of optimizing the use and management of assets [28].

II.4 PHM paradigm

Prognostics and Health Management (PHM) was introduced in the late 90s - early 2000s by American researchers and industrialists [55, 56]. It has since continued to progress and gain the attention of researchers and industrial companies over the few last decades. PHM appears as a complete and integrated approach to meet the expectations of modern industry in terms of availability, reliability and operational safety. Indeed, PHM aims to ensure the smooth functioning of critical machinery and to avoid undesirable events by managing the system based on its past, current and estimated future conditions.

Historically, PHM started as an improvement of CBM, which based on future estimated, current and past conditions. However, no consensual definition has yet been blurred in the literature. The Center for Advanced Life Cycle Engineering[1] defined PHM as «the mean to predict and protect the integrity of equipment and complex systems and avoid unanticipated operational problems leading to mission performance deficiencies, degradation and adverse effects to mission safety». In [121], the authors present PHM as «a set of tools that can be used in cascade or separately to monitor the health state of a system, predict its future evolution and/or optimize decisions».

PHM process mainly consists of three main aspects: (i) observation process that contains the data acquisition and processing; (ii) analysis process in which the system's conditions will be assessed, followed by diagnostics and prognostics, and (iii) action process that involves decision-making and applying through the human-machine interface. PHM framework are detailed in 7 steps from data acquisition to the Human Machine Interface (HMI) as described in the Open Standard Architecture of CBM (OSA/CBM) [56, 92], as shown in Figure II.6. The next items are overview of PHM modules:

II.4.1 Data Acquisition

After identifying the critical components and defining the physical quantities to be monitored, and selecting and location of sensors, this module is responsible for data gathering from traditional and smart sensors or transducers that provide initial monitoring information from machinery.

II.4.2 Data Processing

In most cases, collected data are not readily usable. Each PdM model has different requirements and these must be taken into consideration when choosing adequate preprocessing techniques to boost model performance. Some preprocessing techniques are briefly explained in the following:

- ***Data synchronization*** is used to gather signals sampled at different timestamps to create a time-series/cycle-based data that is easier to handle [170];

[1] «The Center for Advanced Life Cycle Engineering», https://calce.umd.edu/

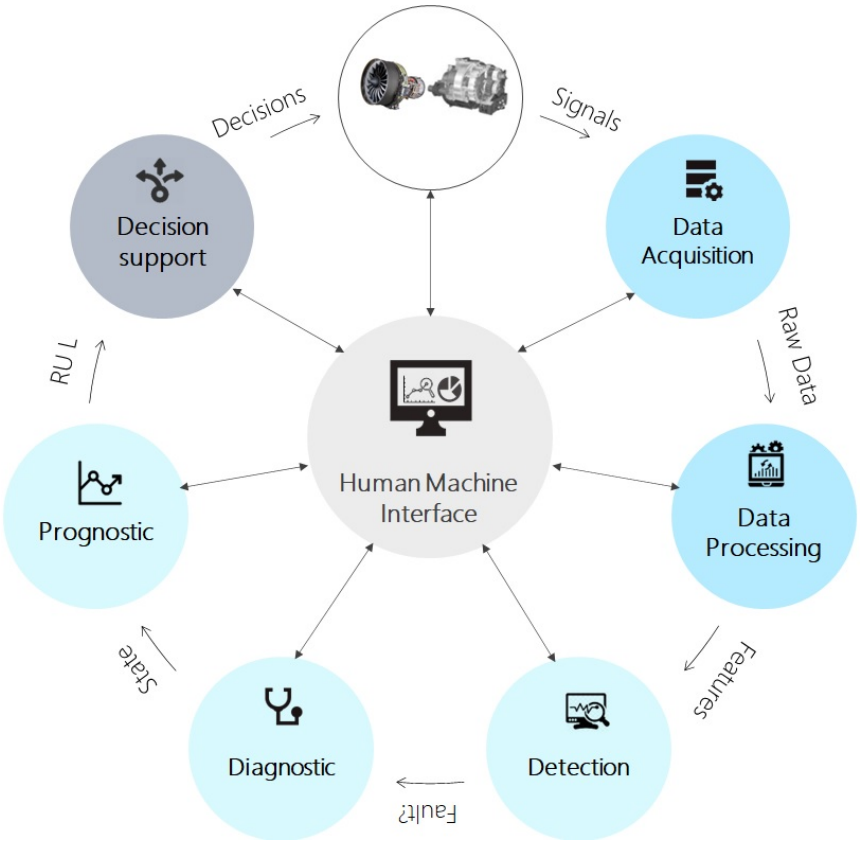


FIGURE II.6: PHM architecture (adapted from [76]).

- ***Data validation:*** basic sanity check, handle missing data, handle abnormal data value;
- ***Data cleaning*** removes or interpolates not available and missing values [37];
- ***Oversampling*** is applied for imbalance data handling to boost accuracy on commonly scarce failure data class or to deal with small datasets [43, 53];
- ***Segmentation*** splits data in chunks to analyse big datasets and enable parallelisation[80];
- ***Feature scaling*** like normalisation or standardisation scales all features to the same or similar space that enables comparisons [157];
- ***Data fusion*** is the process of merging data from multiple data stores. Careful integration can help to reduce and avoid redundancies and inconsistencies in the resulting data set.

Moreover, this layer also consists of extracting a relevant feature to be used as input for models in further stages. It can boost statistical and machine learning model performance, despite not being compulsory for deep learning models given these can extract new representative features that fit the problem automatically. ***Feature extraction*** refers to signal processing algorithms in time, frequency and time-frequency domains to transform raw measurement data into the informative signature of the behavior of the system. ***Feature selection*** algorithm removes

irrelevant and redundant features by selecting the optimal feature subset through filters, wrappers, or embedded methods [82]. Furthermore, ***Dimensionality reduction*** techniques such as principal component analysis (PCA), Linear Discriminant Analysis (LDA) and kernel principal component analysis (KPCA) have been widely adopted to generate a new subset (new space) of lower-dimensional features while retaining intuitive information of the original features.

II.4.3 Condition Monitoring

The features obtained from the data processing layer require to be combined and/or merged into a feature that describes the health of the system (i.e. a health indicator). The condition monitoring layer has responsibility for the construction of the health indicator (e.g. the degradation model), where the real-time data (extracted descriptors) are continuously compared with the previously constructed models of nominal behaviours in order to detect anomalies and trigger alarms according to previously defined thresholds based on criteria (performance, safety, etc.) established by the system operator [21, 55]. The condition monitoring layer is used to explain what was happening in a given situation. Estimating the current condition of the system helps the prognostics phase by determining the starting point along the continuum of condition.

II.4.4 Diagnostics

Once a failure has occurred, the diagnosis deals with the location and identification of the causes of anomalies or failures observed in the system (from the effects to the causes). It provides a more in-depth analysis to answer why this happened?

II.4.5 Prognostics

In engineering, prognostics is considered to be the process of monitoring the health status of an engineering asset and projecting the current health status of a degrading machine into the future in order to estimate the RUL (prediction of the lifetime at which it will no longer perform the required function). It provides a more in-depth analysis to answer what will happen? Different works in literature have formally defined the prognostics differently [154]. We provide four alternative definitions.

- the IOS defined prognostics as "prognostics is the estimation of time to failure and risk for one or more existing and future failure modes". — International Organization for Standardization [115].
- Goebel et al. stated that prognostics is ”the science of making prediction“ by estimating how long the system can still fulfill its purpose before failure occurrence [51].
- Hess et al. defined prognostics as "predictive diagnostics, which includes determining the remaining life or time span of proper operation of a component".

- Heng et al. described prognostics as the forecast of an equipment's condition to determine its future health state, the remaining time to failure, or the probability of reliable operations [65].

This module can be considered as "system lifetime prediction" which is responsible for estimating the remaining useful life before failure by modelling the evolution of degradation and fault progression. Various RUL prediction methods have been proposed, which can be categorized into three main approaches [54–56]: (1) physics model-based approach, (2) data-driven approach and (3) hybrid approach. The physics model-based approach uses the mathematical model that can be a set of differential or algebraic equations, which are very useful to predict RUL in cases where the failure data available are insufficient. This approach requires extensive physical background and knowledge. However, the data-driven approach used to model the degradation and estimate the RUL for the machine with enough failure data. The ease of collecting the monitoring data of many industrial systems has motivated many researchers to use data-driven models in estimating the RUL. Moreover, the hybrid approach integrates physics model-based and data-driven approaches to estimate its RUL.

The RUL is expressed by taking into account units that correspond to the fundamental measure of use of the overall system, such as cycles (i.e., the number of takeoffs) for commercial aircraft, hours of operation for aircraft engines, and kilometers or miles for vehicles. Whatever unit is to be selected, prognostics facilitates decision makers with useful information in managing upcoming maintenance and, therefore, may extend the service life of the machine. To exemplify the task of estimating the RUL, consider the left-hand side of Figure II.7 that represent machine degradation. RUL can be calculated between current time (tc) after degradation has been detected (tD), and the time at which predicted signal passes the failure threshold (FT) (assumed or precisely set by an expert), i.e., time to failure (tf). The FT does not necessarily indicate complete failure of the machinery, but a faulty state beyond which there is a risk of functionality loss, and end of life (EOL) [146]. Beside that, some confidence to the prediction is also constructed in order to indicate the degree of certainty of the RUL. The RUL can be described as follows [77].

$$RUL = tf - tc \tag{II.1}$$

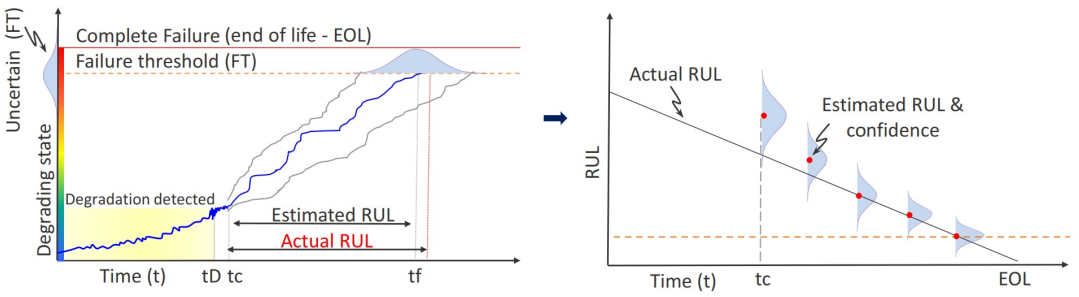


FIGURE II.7: Illustration of RUL estimates.

II.4.6 Decision support

Decision support is the process of choosing the optimal maintenance actions. Particularly, the decisions made in this layer, are based on the use of prognostics information such as the remaining useful life, degradation level, and failure probability.

II.4.7 Human-Machine Interface

The HMI is defined as a Graphical User Interface (GUI), which is used to visualize system health status, analyze data, and control the maintenance operations. It handles the interaction between the PHM system and the user.

II.5 Benchmarking datasets for System-Level Prognostics

To date, several prognostic datasets consisting of "run to failure" time series data have been published by institutions or companies such as FEMTO and PHM Society in an ongoing process. The NASA's Prognostics Center of Excellence (NASA – PCoE) collects and shares those prognostics data among the research community to develop prognostic algorithms, which are available at NASA data repository. Each available dataset is related to a component-level problem such as a turbofan engine, bearing, milling cutter, etc. Specifically, the distinct differences between three prognostic datasets related to the RUL prediction are pointed out in Table II.1. Pronostia-FEMTO Dataset and Milling Dataset have a few assets available for analysis. These datasets belong to the scope of predictive maintenance, which are described in the following:

- The *PHM08 dataset* [144] consists of run-to-failure data from sensors of a fleet of engines of the same type with multiple regimes. Each engine begins with different degrees of initial wear and manufacturing variation, which is unknown to users. It subsequently monitors its progress until an anomaly occurs, after which the engine reaches a failure state.
- The *Milling dataset* [1] includes acoustic emission, vibration and current sensor data gathered under different operating conditions for the objective of milling insert wear analysis.
- The *Pronostia-FEMTO dataset* [117] is a bearing monitoring dataset that contains run-tofailure and sudden failure data. The used sensors are thermocouples gathering temperature data and accelerometers that monitor vibrations in the horizontal and vertical axis.

II.5.1 Occupancy data set

The Commercial Modular Aero-Propulsion System Simulation (C-MAPSS) is a tool coded in the MATLAB-Simulink ® environment to simulate of a realistic high-bypass

TABLE II.1: Overview of three publicly available datasets.

	PHM08 Dataset [144]	**Milling Dataset [1]**	**Pronostia-FEMTO Dataset [117]**
Equipment	Turbofan Engine	Milling Cutter (Metal milling machine)	Bearing
Asset (Train:Test)	Fleet (218:218)	6 (3:3)	17 (6:11)
Operating conditions (Regime)	6	1	3
Fault conditions	HPC degradation, fan degradation	NA	inner race, outer race, rolling element
Data Type	Simulation (RTF)	Monitoring & Usage (RTF)	Testbed (RTF)
Sampling Interval	Operation Cycle- Flight	Operation Cycle- Cut	Every 10 min
Variable Type	Single Value (Parameter)	Single Value	Waveform
Device Used for Data Acquisition	Multi-sensor (or Operational and sensors) (or C-MAPSS tool–multiple sensors)	Accelerometer, Dynamometer, acoustic emission sensor (acoustic emission sensor, vibration sensor, current sensor)	PRONOSTIA platform (rotating speed sensor, force sensor, Speed sensor, temperature sensor,vibration sensors (accelerometers sensor))
PdM Data Description	Temperature, rotation speed, pressure	Vibration, force, temperature, federate, cutting depth	Vibration measurements data (vibration and temperature)
Collected signals	21	6	2
Variables	26	8	3

Notation:
Fleet– A large number of assets are considered.
RTF – Run-to-Failure data is provided.
NA - The criteria is neither clearly defined nor important to the problem.

and twin-spool commercial turbofan engine. Figure II.8 shows the rotating subcomponents of the turbofan engine model as defined in the C-MAPSS model documentation [47]. It consists of six main sub-components: fan, low-pressure compressor (LPC), high-pressure compressor (HPC), combustor or burner, high-pressure turbine (HPT), and low-pressure turbine (LPT), which can be affected by degradation in flow and efficiency. One of the first synthetic run-to-failure datasets collected from a turbofan engine simulation model was used for a prognostics data challenge at the PHM'08 conference, referred to as the PHM08 dataset, as mentioned in Table II.1. Another set was subsequently published with varying complexity levels, known as the C-MAPSS dataset. This challenge dataset provides degradation trajectories for four fleets of aircraft turbofan engines with unknown initial health states for users. The four fleets are presented into four sub-datasets in C-MAPSS, from #1 through

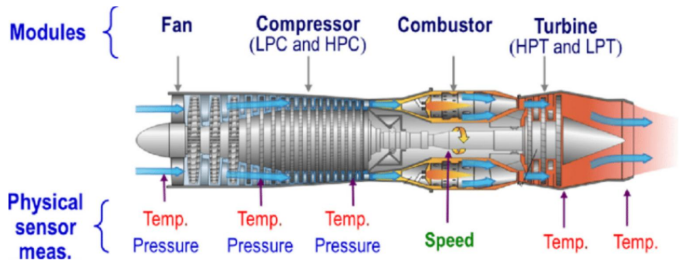


FIGURE II.8: Simplified scheme of the turbofan engine model.

#4 sets. The fundamental difference between these sub-datasets is attributed to the number of simultaneous fault modes and the operational conditions simulated in these experiments [144]. According to the description provided by [145], each sub-dataset consists of multidimensional noise-contaminated time series signals describing the degradation of the different engines within the same fleet by using three sensors to measure varying operating conditions and twenty-one sensors to measure the engines' lifecycle behaviour. Details of these sensors are provided in Table II.2.

TABLE II.2: Details of C-MPASS sensors.

Index	Symbol	Description	Unit
Operational Settings			
1	alt	Altitude	ft
2	Mach	Mach number	-
3	TRA	Throttle resolver angle	%
Sensor Measurements			
1	T2	Total temperature at fan inlet	$^{\circ}R$
2	T24	Total temperature at LPC outlet	$^{\circ}R$
3	T30	Total temperature at HPC outlet	$^{\circ}R$
4	T50	Total temperature at LPT outlet	$^{\circ}R$
5	P2	Pressure at fan inlet	psia
6	P15	Total pressure in bypass-duct	psia
7	P30	Total pressure at HPC outlet	psia
8	Nf	Physical fan speed	rpm
9	Nc	Physical core speed	rpm
10	epr	Engine pressure ratio (P50/P2)	-
11	Ps30	Static pressure at HPC outlet	psia
12	phi	Ratio of fuel flow to Ps30	pps/psi
13	NRf	Corrected fan speed	rpm
14	NRc	Corrected core speed	rpm
15	BPR	Bypass ratio	-
16	farB	Burner fuel-air ratio	-
17	htBleed	Bleed enthalpy	-
18	Nf_d	Demanded fan speed	rpm
19	$PCNfR_d$	Demanded corrected fan speed	rpm
20	W31	HPT coolant bleed	lbm/s
21	W32	LPT coolant bleed	lbm/s

Table II.3 provides a description of the datasets applied. Each fleet dataset is divided into subsets of training engine data and test engine data. All the engines

operate correctly at the beginning of each time series, but because some fault occurs during the series, they start to deteriorate. In the training set, the degradation of each engine grows in magnitude until failure occurs, while in the test set, the degradation ends sometime pre-failure. Also, it noted that the life cycle length of all engines vary. Each fleet dataset of engines is described as follows:

- The subsets **FD001** and **FD003** include 100 turbofan engines in both training and testing sets. As shown in Table II.3, these engines have experienced one operation condition and each has varying life cycles. The FD001 is characterized by deterioration of failure owing to a failed HPC, whereas the FD003 is characterized by deterioration of failure owing to a failed HPC and fan.
- he subsets **FD002** and **FD004** are more complicated than the other two subsets. The FD002 and FD004 include 260 and 248 training samples and 259 and 249 testing samples, respectively. These two sub-datasets comprise six different operational conditions. The turbofan engines have varying life cycles length. The FD002 is characterized by deterioration of failure owing to a failed HPC, while the FD004 is characterized by deterioration of failure owing to a failed HPC and fan.

TABLE II.3: Overview of the C-MAPSS dataset.

Data set	**FD001**	**FD002**	**FD003**	**FD004**
Train trajectories	100	260	100	248
Test trajectories	100	259	100	249
Operating conditions	1	6	1	6
Fault conditions	HPC[(1)]	HPC	HPC, fan[(2)]	HPC, fan
Maximum life span (Cycles)	362	378	525	543
Minimum life span	31	21	38	19
Average span in training set	206	213	247	246
Average span in test set	131	131	166	166

[(1)]HPC represents HPC degradation. [(2)]Fan represents fan degradation.

Due to changing operating conditions, different sensor measurements corresponding to the operating parameters may be produced by the engine. It is expected that a dataset representing such a system with sequential observations and wide measurement ranges will contain discontinuities (discontinuity in C-MAPSS data), as shown in Figure II.9. The C-MAPSS datasets (FD001, FD002, FD003, FD004) are well-tested in the PdM domain. They are reliable examples of high-dimensional and discontinuous asset data due to multi-sensor measurements and the wide range of parameter variations during continuous sensor measurements.

Mathematically, the whole data can be defined as Eq.II.2. Considering that there are N machines of the same type, such as the turbofan engine, each engine contains $maxT_i, i \in \mathbb{N}$ run to machine end of life (cycles) collected by multiple sensors.

$$Dataset = \{(X^i, Y^i)\},\ i = 1, 2, 3, ..., N \tag{II.2}$$

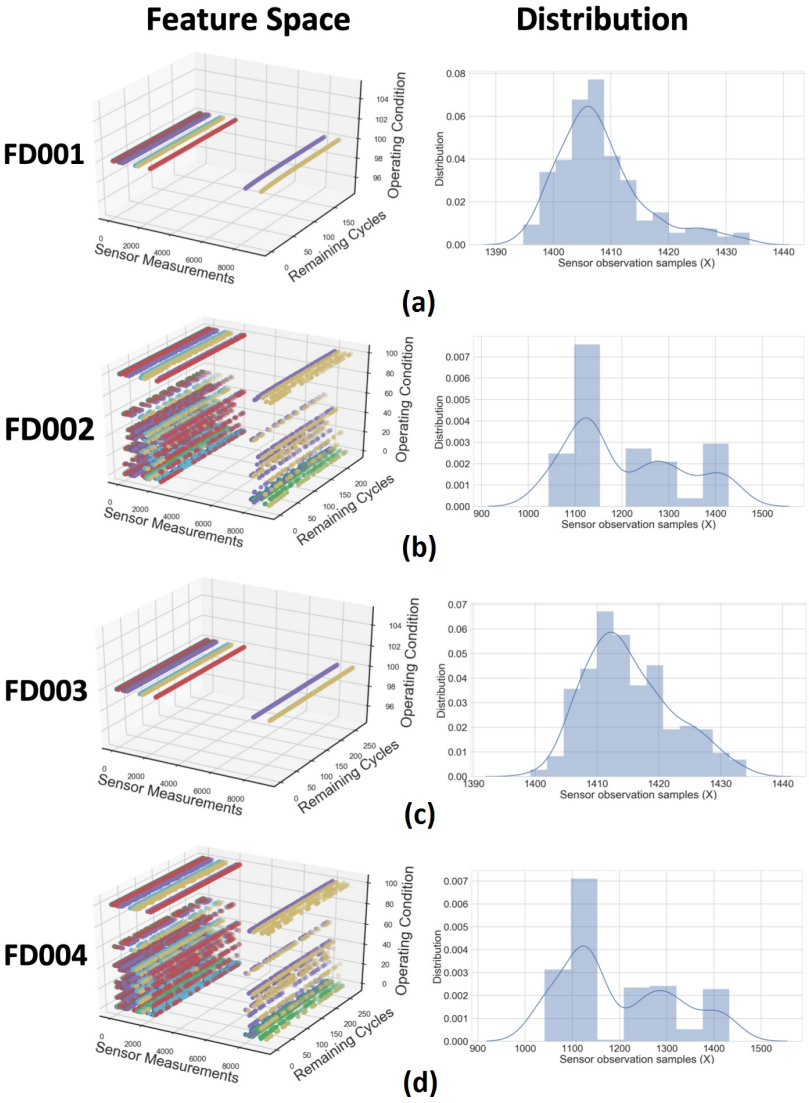


FIGURE II.9: A comparison between the feature space (sensor measurement, remaining cycles, operating condition) and corresponding feature distribution of the run-to-failure life cycle observation of CMAPSS FD001 (a), FD002 (b), FD003 (c) and FD004 (d).

Where X^i denotes the gathered sensor measurements matrix of an engine in which Y^i corresponds to the equipment operation cycles, as shown in Eq.II.3 and Eq.II.4, respectively.

$$X^i = [x_1, x_2, x_t, ..., x_{maxT_i}] \in R^{m \times maxT_i} \tag{II.3}$$

$$Y^i = [y_1, y_2, ..., y_{maxT_i}] \in R^{1 \times maxT_i} \tag{II.4}$$

where $maxT_i$ is the total operation cycles of the i-th engine and $x_t = [x_t^1, x_t^2, ..., x_t^m] \in R^{m \times 1}$ is an m-dimensional vector of sensor measurements at time t.
RUL can be calculated between the current time (y_t) after degradation has been detected and the failure time (T). The RUL can be described as follows [77].

$$RUL_t^i = \{maxT_i - y_t^i\},\ t = 1, 2, ..., maxT_i \tag{II.5}$$

II.6 Conclusion

Throughout history, maintenance strategies have evolved continuously to ensure asset safety, reliability, availability, and maintainability. Currently, the commonly applied strategies can be classified as corrective maintenance, systematic preventive maintenance, condition-based maintenance, and predictive maintenance. Predictive maintenance is the most cost-optimal maintenance strategy, achieving overall equipment effectiveness exceeding 90%. Concretely, it maximises an asset's working life and avoids costly breakdowns and lost production time caused by unplanned downtime. Currently, PdM is the state-of-the-art maintenance strategy, as it provides up-to-date diagnostics, probability of failure, and estimating Remaining Useful Life. The prediction of RUL is at the centre of PHM. RUL is a crucial characteristic for prognostics, which is the number of times an asset can accomplish its intended task before requiring maintenance.

This chapter introduced comprehensive basic concepts related to our thesis. The chaptethesis began with an overview of Industry 4.0 and maintenance evolution. Next, an overview of PHM and the role of the prognostics process is a crucial enabler to ensure the mission achievement of systems while reducing costs and risks. Finally, the chapter summarised the benchmarking data sets for system-level prognostics with the occupancy dataset. The following chapter briefly overviews the theoretical foundations to understand deep learning and state data-driven prognostic challenges and issues.

Chapter III

Prevailing ML and DL

III.1 Introduction

With the growing volume of industrial data and computational power, industry attention has turned to artificial intelligence and machine learning techniques in order to develop advanced data-driven frameworks for fault diagnosis and remaining life assessment. The direction of machine learning research has shifted to more complex models such as ensemble methods and deep learning, given their greater accuracy in processing larger datasets. This thesis chapter provides a brief overview of the relevant theoretical foundations needed to understand deep learning. We begin by defining what machine learning is and describing some learning scenarios in Section III.2. An overview of prevailing deep learning-based architectures as well as detailed regularization techniques for DL are provided in Section III.3, followed by deep Learning frameworks in Section III.4. Next, data-driven prognostic challenges and issues is also summarized III.5. Finally, section III.6 gives a brief conclusion from what this chapter presents.

III.2 Machine Learning Basics

Machine Learning (ML) was originally developed as a core subarea of Artificial Intelligence (AI) focused on developing computational learning methods. ML is also closely related to other fields, especially computational statistics, with which it often overlaps. The term "Machine Learning" was coined first by Arthur Lee Samuel. However, there is no universally accepted definition of ML. Different researchers have formally defined the term differently. We provide three alternative definitions.

- In Arthur L. Samuel's influential paper, ML is defined as "the field of study that gives computers the ability to learn without being explicitly programmed [138, 139]." — Arthur Lee Samuel, IBM scientist and AI pioneer, 1959.
- Tom M. Mitchell provided a widely quoted, more formal definition: "A computer program is said to learn from experience E with respect to some tasks T and performance measure P, if its performance on T, as measured by P, improves with experience E [112]." — Tom Michael Mitchell, ML Professor at Carnegie Mellon University, 1997.
- Ron Bekkerman et al. define ML as Machine learning that focuses on constructing algorithms for making predictions based on data without programming it to perform the task. The ML task aims to identify (to learn) a function $f : X \rightarrow Y$ that maps the input X (of data) into the output Y (of possible predictions). Function f is chosen from a specific function class, dependent on the type of learning algorithm used (Section III.2.1). [11] — Ron Bekkerman, CTO at Cherre, 2011.

III.2.1 Learning scenarios

Different machine learning scenarios can be classified mainly by the type of training datasets used as experience, among which the most common are briefly explained in the following sections, including (1) supervised learning, (2) unsupervised learning, (3) semi-supervised learning, (4) reinforcement learning, (5) Multitask learning, and (6) transfer learning [114].

III.2.1.1 Supervised learning

Typically, a supervised scenario involves learning a deterministic function $f(x, w)$ that maps inputs to outputs based on a labeled training set made up of input-output couples $(x, y) \in x \times y$. The aim is to reduce the magnitude of the prediction error ($loss$) as close to zero as possible and make predictions for unseen instances. There are two classic supervised learning tasks: (i) classification and (ii) regression tasks [11].

(i) **Classification:** the output domain is a finite and discrete set of possible outcomes (called classes, targets, labels, or categories), $Y \in \{C_1, ..., C_i\}$. The classification task can be a binary or multi-class classification depending on the number of output classes. Furthermore, other variants of the classification task predict the probability distribution over classes instead of categories. The learner model is called a classifier.

(ii) **Regression:** the output domain is the set of real numbers (continuous output values), $Y \in R$. Besides, the learner model is called a predictor.

In general, there are several supervised algorithms for learning the same task, such as Linear Regression (LR), Logistic Regression, Support Vector Machines (SVM), Neural Network (NN), K-Nearest Neighbours (KNN), Decision Tree (DT), and Naïve Bayes (NB). Though these single algorithmic models are generally successful, they might not be enough to achieve the highest potential accuracy with some problems. Ensemble methods [48] are meta-algorithms for minimizing generalization errors, both bias (boosting approach) and variance (bagging approach), and improving predictions (stacking approach) by combining several individual models (combining all weak learners or well-chosen strong and diverse learners), also called base-predictors or base-learners, into one predictive model. Ensemble learning techniques can be broadly classified into two categories: sequential ensemble and parallel ensemble. Both are depicted in Figure III.1 and are defined as follows:

(i) **Sequential Learning Methods (sequential ensemble) :** have the advantage of the dependence between the base learners where the base-predictors are trained sequentially. These methods are commonly known as **Boosting** methods, including Stochastic Gradient Boosting (SGB), Gradient Boosting Machine (GBM), Adaptive Boosting (AdaBoost), LightGBM, and Extreme Gradient Boosting (XgBoost).

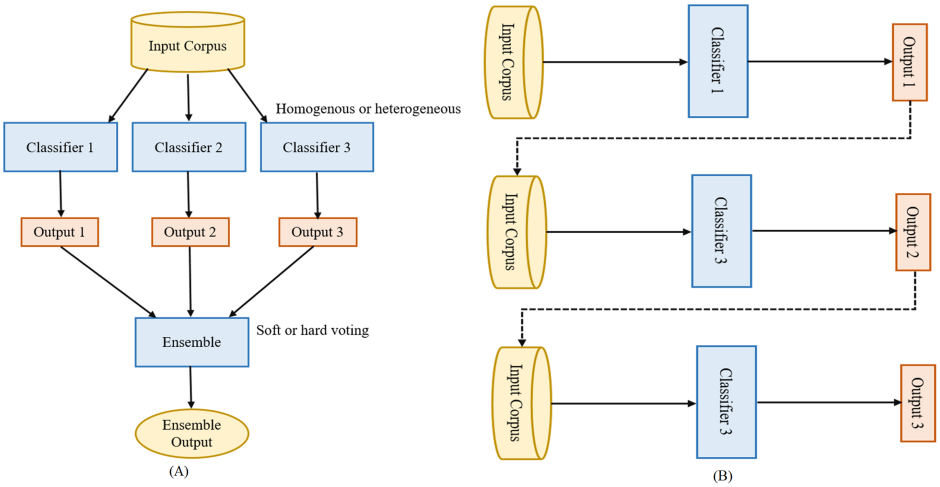


FIGURE III.1: *(A)* Parallel vs. *(B)* sequential ensemble learning.

- **Boosting:** is the process of building correlated homogeneous learners (are generated sequentially), each of which learns to fix the prediction errors of a prior model in the sequence of models and combines them following a deterministic strategy to achieve an intense learner with lower bias.

(ii) **Independent Learning Methods (parallel ensemble) :** the base-predictors are as uncorrelated as possible with each other and are trained in a parallel way. Parallel modeling has the advantages of simultaneous predictions, using several CPU cores to carry out the models simultaneously, and exploiting the characteristics of independence among them. The following discusses several methods that have been proposed to train a set of base-predictors independently.

- **Bagging (a name derived from bootstrap aggregating):** reduces the variance of an estimate by taking the mean of multiple homogeneous models (typically of the same type, with some minimum variations) [23, 52]. There are three basic steps to performing bagging. As a first step, generate multiple random, although overlapping, sub-samples of the training dataset (bootstrapping). The second step is to build and train multiple models independently on each bootstrapped sub-sample (parallel training). The third step takes all the predictions' averages (for regression) or majority votes (for classification) to make a final overall prediction (aggregation). One of the common ML algorithms that apply the bootstrap aggregating process is a *Random Forest*.
- **Stacked Generalization (Stacking):** is a method introduced by David H. Wolpert in 1992 [155, 179], in which a meta-model (a high-level model) is used to combine a set of heterogeneous or homogeneous models (lower-level models) learned in parallel using the k-fold cross-validation technique to achieve greater predictive accuracy. A significant variation of stacking is called **Blending**, which uses a one-holdout set instead of k-fold cross-validation.

- **Voting ensemble:** the same mechanism is used in bagging, which combines multiple heterogeneous models (typically of differing types) using simple statistics, such as voting or averaging. In the case of regression, this involves calculating the average of the predictions from the base-learners, expressed in Equation III.1. In the case of classification, this involves predicting the class with the most considerable sum of votes (Hard Voting, also known as majority voting, see Equation III.2) or with the most considerable summed probability from models (Soft Voting, see Equation III.3). A voting ensemble can offer a lower variance in the predictions made than individual models [24].

$$\hat{Y} = \frac{1}{n}\sum_{j=1}^{n} h_j(x) \tag{III.1}$$

$$\hat{Y} = C_{\underset{i}{argmax}} \sum_{j=1}^{n} h_j^i(x) \tag{III.2}$$

$$\hat{Y} = C_{\underset{i}{argmax}} \sum_{j=1}^{n} P_j^i \tag{III.3}$$

- **Bucket of Models:** is an ensemble method in which a model selection algorithm chooses the best model for each problem. In this technique, heterogeneous or homogeneous models are trained on the given training dataset. Finally, the model that best performs on the test set is chosen for future use. The most common approach used for model selection is cross-validation selection (sometimes called a "bake-off contest") [5].

III.2.1.2 Unsupervised learning

Unsupervised learning uses algorithms to analyze and cluster unlabeled datasets or discover hidden patterns without human intervention by investigating the similarities and differences in information. The most common unsupervised learning tasks are (i) clustering analysis, (ii) association rules, (iii) anomaly detection, and (iv) dimensionality reduction.

(i) **Clustering analysis:** involves discovering natural groupings (called clusters) for unlabeled data based on similarities or differences in information. Typically, data is partitioned into homogeneous clusters $\bigcup_{k=1}^{K} C_k$, with K being the set of cluster indices that share common characteristics. The most commonly used clustering algorithms are K-means and hierarchical clustering.

(ii) **Association Rule:** is a rule-based method for finding relationships or associations between variables in a dataset. The apriori algorithm is known to be used for association rule learning problems [10].

(iii) **Anomaly detection:** is the process of detecting an anomalous point or pattern in a dataset that does not conform to expected behaviour. These nonconforming points are called anomalies, outliers, or discordant observations [31].

(iv) **Dimensionality reduction:** refers to compressing data onto a lower-dimensional subspace or manifold while preserving as much of the variation in the dataset as possible. Various methods are available for dimensionality reduction, including Principal Component Analysis (PCA), Singular Value Decomposition (SVD), Isometric Mapping (ISOMAP), Kernel PCA (KPCA), t-Distributed Stochastic Neighbor Embedding (t-SNE), and Autoencoders (AE). It brings many advantages, including:

- Prevent the overfitting problem.
- Improve the ML performance through less misleading and redundant data.
- Mitigate the curse of dimensionality, theorised by Bellman.
- Reduce time and space complexity.
- Data visualisation and interpretation were made more accessible.

III.2.1.3 Semi-Supervised learning

Semi-supervised learning can be described as a hybrid approach between supervised and unsupervised learning. It is mainly relevant in scenarios where the dataset comprises a mixture of labeled and unlabeled data. Unlabeled data could be plentiful in the real world, such as in a medical image analysis task. Besides, annotating medical images individually is challenging due to the need for domain expertise. Therefore, some images are manually labelled, resulting in a substantial amount of unlabeled data. Consequently, in this case, semi-supervised learning is useful for labeling the unlabeled portion of the training set (known as "pseudo-labelling") in order to transform the unlabeled data to labeled data completely [167].

III.2.1.4 Reinforcement learning

The reinforcement learning algorithm (called an agent) is a trial-and-error approach that allows a model to continuously learn using a feedback loop between the system and its experiences. This type of learning is based on reward or penalty, and its ultimate goal is to use observations gathered from the interaction with the environment (known as an environment-driven approach) to take optimal actions that would maximize the reward or minimize the risk (by “rewarding” good behaviour and “punishing” bad behaviour) within a particular environment. Reinforcement learning algorithms have been used in sophisticated systems such as robotics and self-driving cars. The popular reinforcement learning models are Q-Learning, State-Action-Reward-State-Action (SARSA), Deep Q Network (DQN), and Deep Adversarial Networks. [20]

III.2.1.5 Multi-Task Learning

Multi-Task Learning (MTL) is a learning paradigm that has first been proposed by Caruana [27], in which individual models for performing potentially related tasks

are learned jointly to leverage such domain-specific information and commonalities across tasks. The simultaneous learning of multiple related or auxiliary tasks has been demonstrated, both theoretically and empirically, in enhancing individual task generalization. This form of learning is required in a wide range of real-world applications when various datasets are sufficiently similar or cannot be treated as independent, and also addresses the lack of training data issue.

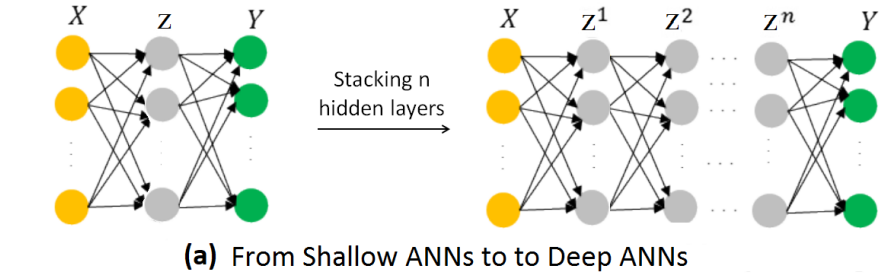
MTL is focused on a neural network architecture consisting of a shared network and a task-specific network. A shared network is usually achieved with either hard or soft parameter sharing of hidden layers [39]. Hard parameter sharing is typically applied by sharing the hidden layers between all tasks, while task-specific networks are made independent to separate the information of individual tasks. Under soft parameter sharing, each task has its own model with its own parameters. The distance between the model parameters of different tasks is added to the joint objective function. Though there is no explicit parameter sharing, there is an incentive for the task-specific models to have similar parameters.

III.2.1.6 Transfer Learning

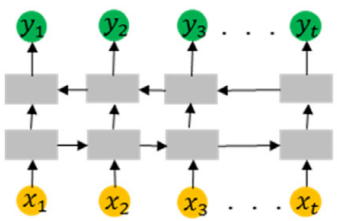
Transfer learning is an approach in which the knowledge learned by various pre-trained models on large benchmark datasets is transferred to other applications in order to tackle the undersized training dataset problem. In other words, the idea behind TL is to fine-tune pre-trained models for new tasks in the target domain. Hence, the new model can be initialized by transferred parameters instead of developing or training from scratch. This addresses the challenge of the enormous computational and storage resources required to develop deep learning models [123].

III.3 A brief overview of deep neural network architectures

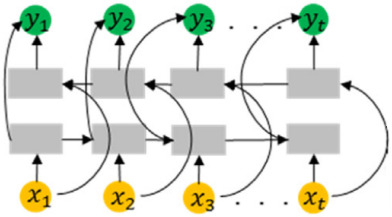
In the 1940s, feed-forward Artificial Neural Networks (ANNs) were inspired by the biological neural networks in human brains, which are composed of interconnected neurons. From a mathematical point of view, ANNs consist of a non-linear transformation $y = \sum_{j=1}^{n} f(w_j x_j)$ of the input x. A new term has been given to a more complex ANN called Deep Neural Networks (DNNs) [148, 191], consisting mainly of an input layer, multiple hidden layers, and an output layer, as depicted in Figure III.2. (a). The most distinct difference between the two terms, ANNs and DNNs, lies in the complexity of the network architecture. In ordinary ANN networks, a few layers are typically used, and each neuron is fully connected to all neurons in adjacent layers. In contrast, in DNNs learning architectures (deep architectures in NNs), higher complexity functions are used with more layers beyond shallow 1- or 2-layer networks, which construct deep hierarchical models that can automatically learn high-level representations of large-scale data and eliminates the reliance on domain knowledge and manual feature engineering . The simplest and uncomplicated deep architecture is the multi-layer perceptron (MLP) network shown in Figure III.2. (a) [133].



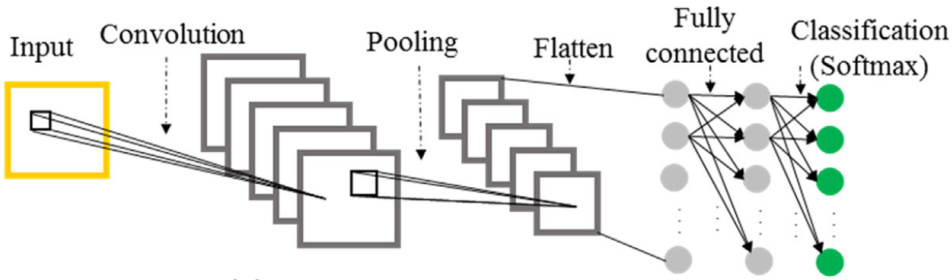
(a) From Shallow ANNs to to Deep ANNs



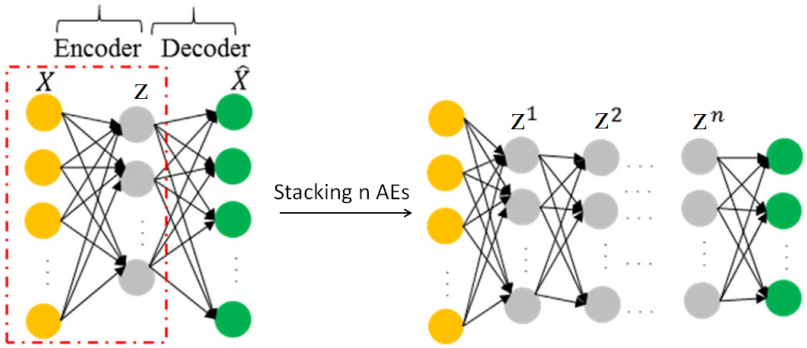
(b) Recurrent Neural Network (Unidirectional)



(c) Recurrent Neural Network (Bidirectional)

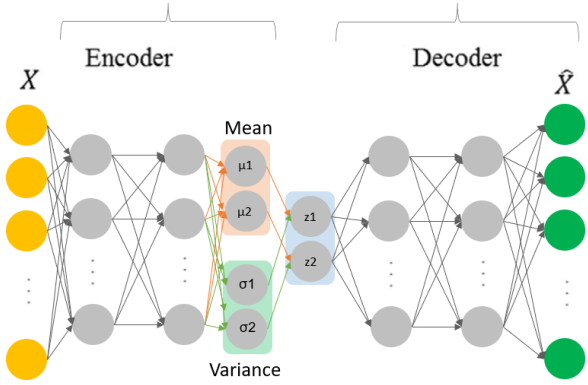


(d) Convolutional Neural Network



(e) Autoencoder (AE)

(f) Stacked Autoencoder



(g) Variational AutoEncoders (Generative models)

FIGURE III.2: Typical deep architectures.

Beginning in 2006, when the concept of Deep Learning (DL) was coined [29, 67], various DL algorithms have emerged and applied to a variety of data and have been embraced by various areas such as image analysis, natural language processing, health care, computer vision, machine health monitoring systems, and automatic speech recognition [14, 66, 132, 199]. This new area of research is growing rapidly, with new and advanced architectures being developed every few months. In fact, DL is not a new idea, which even dates back to the 1960s [29, 93, 118, 148]. The DL is gaining in popularity for the following reasons: **(i) Increasing Computing Power:** the rise in processing capabilities (like the GPU units), and the decreased hardware costs all significantly reduce the uptime of deep learning algorithms. **(ii) Increasing Data Size:** the huge amount of data can offset the complexity increase behind deep learning and improve its generalization capability. **(iii) Feature Engineering Power:** advanced DL algorithms have gained increasing interest from researchers due to their inherent ability to learn salient features from raw data automatically and can be considered an end-to-end framework without the need for features designed by human engineers. Deep learning approaches have also proven suitable and adequate for various application domains [118, 199].

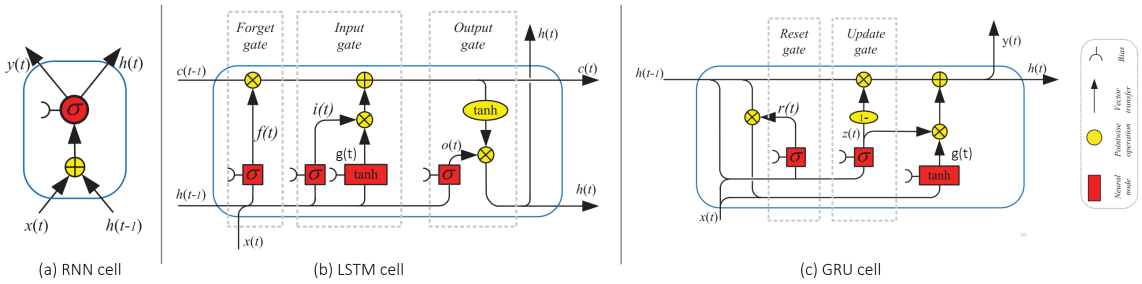
A brief introduction is provided only to the relevant deep learning algorithms applied in machine health monitoring. In the following subsection, three deep architectures, including recurrent neural networks (RNN), convolutional neural networks (CNN), and autoencoder (AE), and their corresponding variants are reviewed, respectively. In addition, a comparison is provided among these deep architectures.

III.3.1 Recurrent Neural Network

RNN [137] contains feedback loops to remember or save the state derived from previous inputs of the network, which are the most suitable for time series and sequential data, such as natural language and time-series data. Unlike the typical neuron, the RNN consists of a series of recurrent neurons that use backpropagation through a time algorithm [178] to propagate the network's error to previous time instances. However, the RNNs suffer from vanishing or exploding gradient issues throughout the training process in long-term periods [60]. Therefore, Long-Short Term Memory (LSTM) [70] and Gated Recurrent Unit (GRU) [36] were introduced to alleviate the issues mentioned above by incorporating gating functions into their state dynamics to extract dependencies on different time scales. LSTM or GRU is one of the main reasons behind the success of RNNs in recent years [137, 187]. The schematic diagram of RNN, LSTM, and GRU cells is depicted in Figure III.3. The simplest one is vanilla RNN, whose mathematical equation is given as follows:

$$h_t = \delta(Wx_t + Hh_{t-1} + b) \tag{III.4}$$

A hidden vector h is updated at time step t, where W and H represent matrices of transformation, and b is the bias vector. δ denote the nonlinear activation function such as sigmoid and tanh functions. As shown in Figure III.2.(b), RNNs can be stacked to create deeper networks by using the hidden state h as an input to the next



(a) RNN cell (b) LSTM cell (c) GRU cell

FIGURE III.3: Schematic diagram of RNN, LSTM, and GRU cells.

recurrent layer, described as:

$$h_t^l = \delta(W h_{t-1}^l + H h_t^{l-1}) \tag{III.5}$$

In addition, the bidirectional RNN structure [149] is an upgraded version of RNN that can process the sequence information in two directions, specifically, the forward and backward paths with two disconnected hidden layers, depicted in Figure III.2.(c). The following equations represent the function of the hidden layer and the $\rightarrow$ and $\leftarrow$ indicate forward and backward processes, respectively.

$$\overrightarrow{h_t} = \overrightarrow{H}(x_t, \overrightarrow{h_{t-1}}) \tag{III.6}$$

$$\overleftarrow{h_t} = \overleftarrow{H}(x_t, \overleftarrow{h_{t+1}}) \tag{III.7}$$

Afterwards, the final vector h_T is the concatenated vector of the outputs of the forward and backward processes as follows:

$$h_T = \overrightarrow{h_t} \oplus \overleftarrow{h_t} \tag{III.8}$$

As depicted in Figure III.3.(b), LSTMs change the structure of hidden units from "sigmoid" or "tanh" to memory cells, referred to as the long-term state $C(t)$ and the short-term state $H(t)$. Secondly, three control gates along the state path are inserted to regulate the cell states, namely the forget gate, the input gate, and the output gate. These gates control the flow of information to hidden neurons and preserve extracted features from previous timesteps [70]. The mathematical equations of LSTM are as follows:

$$f_t = \delta(W_f\, x_t + H_f\, h_{t-1} + b_f) \tag{III.9}$$

$$i_t = \delta(W_i\, x_t + H_i\, h_{t-1} + b_i) \tag{III.10}$$

$$o_t = \delta(W_o\, x_t + H_o\, h_{t-1} + b_o) \tag{III.11}$$

$$g_t = \tanh(W_g\, x_t + H_g\, h_{t-1} + b_g) \tag{III.12}$$

$$C_t = (f_t \otimes C_{t-1}) \oplus (i_t \otimes g_t) \tag{III.13}$$

$$h_t = \tanh(C_t \otimes o_t) \tag{III.14}$$

Where W*, H*, and b* denote the trainable weights and biases for each gate

indicated by *, respectively. Also, the δ is the sigmoid activation function. x_t is the current input, and i, f, o are the input, forget, and output gates, respectively. h_{t-1} denotes the previous iteration's hidden state, whereas h_t corresponds to the current hidden state. The cell states C_t, C_{t-1} are defined as the hidden states.

Similar to the LSTM unit, the GRU [36] has arisen as an upgraded LSTM structure network, which combines the forget and the input gates into a single update gate, and mixes cellular state and hidden state into one state. The number of gates in GRU is mainly reduced from 4 in LSTM to 2, called update and reset gates. The block diagram of a GRU cell is shown in Figure III.3.(c), and the equations to compute the GRU cell's state at each time step are as follows:

$$z_t = \delta(W_z\, x_t + H_z\, h_{t-1} + b_z) \tag{III.15}$$

$$r_t = \delta(W_r\, x_t + H_r\, h_{t-1} + b_r) \tag{III.16}$$

$$g_t = \tanh(W_g\, x_t + H_g\, (r_t \otimes h_{t-1}) + b_g) \tag{III.17}$$

$$h_t = z_t \otimes h_{t-1} \; + \; (1 - z_t) \otimes g_t \tag{III.18}$$

Where r and z are the rest and update gates, respectively.

Researchers and engineers usually attempt both GRU and LSTM to determine which one works better for their use case [137].

III.3.2 Convolutioanl Neural Network

CNN is a subtype of the deep discriminative architecture, which first came to the spotlight through the work of LeCuN in 1989 for processing gridlike topological data (images and time-series data) [93]. The success of CNNs has captured attention beyond academia. In the industry, companies such as Google, Microsoft, and Facebook have created active research groups to explore new CNN architectures [40]. The hierarchical structure of deep CNN gives it the ability to learn complex representations at different levels of abstraction. The critical difference between CNNs and shallow architectures is that CNNs benefit from parameter sharing, which allows the network to look for specific features at different positions [52].

A schematic of a typical CNN usually consists of three main neural layers (or so-called multi-building blocks), as depicted in Figure III.2.(d), whose convolutional layers alternate with pooling layers, eventually accompanied by some fully connected layers. In addition to different mapping functions, different regulatory units such as batch normalization and dropout are also incorporated to optimize CNN performance. The arrangement of CNN layers plays a crucial role in constructing new architectures and thus leads to improved performance [83]. We will briefly introduce and discuss the function and role of each layer.

A. **Convolutional layer:** utilizes various kernels to convolve the raw input data as well as the intermediate feature maps to generate invariant local features and several feature maps. The convolutional layer is usually linear and is followed by applying an activation function to produce a non-linear output [150].

The convolution operation is the essence of the CNN architecture, which can be performed on one-dimensional or two-dimensional data by utilizing a sliding convolutional filter that slides vertically and horizontally to capture useful features. The convolution operation can be defined as introduced in Equation III.19 [83].

$$f_l^k(p,q) = \sum_c \sum_{x,y} i_c(x,y).e_l^k(u,v) \tag{III.19}$$

where, (x,y) element of c^{th} channel of an input tensor i_c, which is element wise multiplied by $e_l^k(u,v)$ index of the k^{th} convolutional kernel k_l of the layer l^{th}. Whereas output feature-map of the k^{th} convolutional operation can be expressed as $F_l^k = [\, f_l^k(1,1), \ldots, f_l^k(p,q), \ldots, f_l^k(P,Q)]$, where P and Q total number of rows and columns of feature matrix, respectively.

There are three main characteristics of the convolution operation: sparse interactions, weight sharing and invariance representations [189]. These characteristics are described as follows:

- Weight sharing indicates that the same weight is used for more than one location, which leads to a decrease in the number of parameters.
- Sparse interactions refer to sparse connectivity or sparse weights that learn correlations among neighbouring pixels. They reduce the storage requirements of the parameters and the model's runtime, which require fewer operations for computing the output.
- The invariance representations mean that the output changes in the same way if the input changes.

B. **Pooling layer:** Pooling or down-sampling usually follows a convolutional layer, which can reduce the dimensions of feature maps and the number of parameters, which leads to better robustness against noise. Similar to convolutional layers, pooling layers also define shift-invariant features due to their computations taking neighbouring pixels into account. It sums up similar information in the neighborhood of the receptive field and outputs the dominant response within this local region. Equation III.20 shows the pooling operation in which Z_l^k represents the pooled feature-map of l^{th} layer for k^{th} input feature-map F_l^k, whereas $G_p(.)$ defines the type of pooling operation such as max, average, etc. [147].

$$Z_l^k = G_p(F_l^k) \tag{III.20}$$

C. **Dropout:** introduces regularization within the network, which will randomly remove some of their neurons or connections while a training process with a predefined probability of dropped [52].

D. **Fully-connected layer:** most architectures have final Fully-Connected (FC) layers that perform like a traditional neural network, commonly located after alternating the two mentioned layers. The FC layer maps the data to a 1D feature vector, which a classifier or predictor can use.

III.3.3 Auto-encoders

As a feed-forward neural network, an AE is also called an auto-associator, first introduced in the 1980s by Hinton and the PDP group [136]. It is designed for unsupervised learning and is used for efficient data compression, dimensionality reduction for data visualization, extracting useful features, and filtering out useless information [9, 15, 46, 68, 69, 186]. As shown in Figure III.2.(e), auto-encoder comprises two phases, including the encoder and the decoder, both of which are multilayered NNs parameterized with two weight vectors, ϕ and θ. Theoretically, θ should be the transpose of ϕ. It is trained to reproduce its inputs $x \in R^m$ where the dimension of x is denoted by $m \in N$. As a result, the output vectors have dimensions similar to the input vector, while fewer units are used in the latent space.

In the coding process, the input data x is transformed into a lower-dimensional latent representation z via a non-linear mapping, formulated as:

$$z = f_\phi(x) \tag{III.21}$$

Then, in the decoding process, the latent representation z is converted back into the original feature to obtain an approximation or reconstruction of the real data $\hat{x}$ as follows:

$$\hat{x} = h_\theta(z) \tag{III.22}$$

To reduce the reconstruction error between x and $\hat{x}$, the optimization parameters (ϕ, θ) are adopted and used Mean square errors (MSEs) in order to calculate the reconstruction's performance.

III.3.3.1 Variations of AEs

In the following, we briefly explain some of the variants of the proposed auto-encoder and briefly summarise their characteristics. The following variations are obtained by applying regularisation and modifying AE types.

- **Stacking Structure:** generally, a single layer cannot extract high-level representative features from raw data. The architecture of the deep auto-encoder (stacked auto-encoder, Figure III.2.(f)) has more than one hidden layer that helps to learn high-level representations, where the output of each hidden layer is connected to the input of the successive hidden layer.

- **Addition of Sparsity:** the addition of sparsity constraints on the hidden units targets captures sparse features from raw data. The sparsity of the representation can be achieved either by penalizing the hidden unit biases or by explicitly penalizing the output of hidden unit activations through the addition of the Kullback-Leibler (KL) divergence term to the cost function to completely prevent AE from learning the identity [91, 107, 161].

- **Addition of Denoising:** Instead of adding the penalty to the cost function, the denoising auto-encoder (DAE) deliberately adds noises to the training data, as

the AE trains with these corrupted data to reconstruct/denoise the clean input x from its corrupt sample x'. Adding the noise helps to learn the robust feature and to avoid the auto-encoders from learning the identity by copying the input to the output. The standard and most popular utilized noise is dropout noise/binary or gaussian noise, which sets a random fraction of the input features to zero [171].

- **Variational AutoEncoders (Generative models):** a VAE proposed by Kingma et al. has the same functions as the AE in the sense that it is composed of an encoder and a decoder Figures III.2.(g) and III.4. VAE becomes a popular generative model by combining bayesian inference and the efficiency of the NNs to obtain a nonlinear low-dimensional latent space ([140], [26], [95], [198]). The Bayesian inference is obtained by an additional layer used for sampling the latent vector z with a prior specified distribution $p(z)$, usually assumed to be a standard Gaussian $N(0, I)$, where I is the identity matrix. Standard Gaussian is not the only distribution used for latent variables in VAEs, but the choice depends on the type of data we are modelling. Such as the multivariate Gaussian distribution is used in the case of real-valued data and the Bernoulli distribution is applied in the case of binary data [84]. Each element z_i of the latent layer Z is obtained as follow:

$$z_i = \mu_i + \sigma_i.\epsilon \tag{III.23}$$

where μ_i and σ_i are the i^{th} components of the mean μ and standard deviation σ vectors, ϵ is a random variable following a standard Normal distribution ($\epsilon \sim N(0,1)$).

Unlike the AE which generates the latent vector z, the VAE generates vector of means μ_i and standard deviations σ_i. This allows to have more continuity in the latent space than the original AE. The VAE loss function given by the Equation III.24 has two terms. The first term L_{rec} is the reconstruction loss function (Equation III.25). Usually the negative expected log-likelihood (e.g., the cross-entropy function) is used ([72], [95], [173], [160], [26]) but the mean squared error can also be used [198]. The second term L_{KL} (Equation III.26) corresponds to the Kullback-Liebler (KL) divergence loss term that forces the generation of a latent vector with the specified Normal distribution ([84],[85]). The KL divergence is a theoretical measure of proximity between two densities q and p and it is noted by KL(q $\parallel$ p). The dissimilarities between these densities are asymmetric ($KL(q \parallel p) \neq KL(p \parallel q)$), non-negative and are minimized when $q(x) = p(x)\ \forall x$ [19]. Thus, the KL divergence term measures how close is the conditional distribution density q_ϕ(z | x) of the encoded latent vectors from the desired Normal distribution p(z). The value of KL is zero when two probability distributions are the same, which forces the encoder of VAE to learn the latent variables that follow a multivariate normal distribution over a k-dimensional latent space.

$$L = L_{rec} + L_{KL} \tag{III.24}$$

where

$$L_{rec} = -E_{q_\phi(z|x)}(log(p_\theta(x|z))), \tag{III.25}$$

$$L_{KL} = KL(q_\phi(z|x) \parallel p(z)) \tag{III.26}$$

with $p_\theta(x|z)$ is the conditional distribution density of the decoded latent vectors. When the VAE is trained, each function (i.e., the encoder and the decoder) can be used separately, either to reduce the space dimension by encoding the input data, or to generate synthetic samples by decoding new variables from the latent space (Figure III.4).

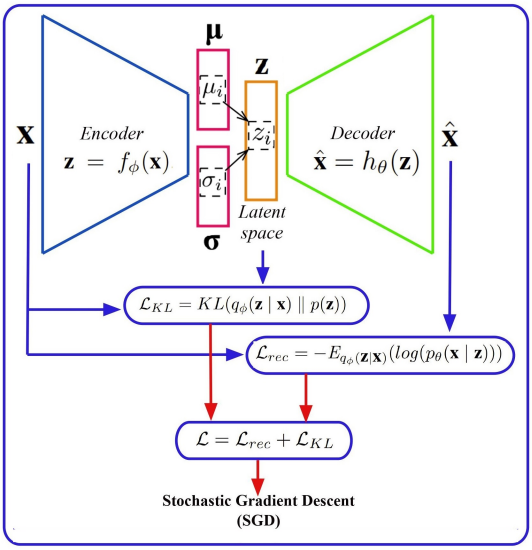


FIGURE III.4: The VAE loss function. The first term L_{rec} is the reconstruction loss function. The second term L_{KL} corresponds to the Kullback-Liebler divergence loss term that forces the generation of a latent vector with the specified Normal distribution.

III.3.4 Regularization for Deep Learning

Regularization is a set of techniques used to train more robust models that can prevent overfitting and improve the generalization of deep neural networks [52]. In this section, we will briefly describe the most popular and widely used regularization techniques that are intended to reduce their generalization errors, possibly at the expense of increased training errors.

- **Data augmentation:** is the process of artificially expanding the training dataset's size and increasing the data's diversity by generating synthetic (fake) samples. This technique is a data-space solution for any limited-data issue, as well as for overfitting issues and reducing the variance of the model by making it generalize better [153].

- **Early Stopping (ES):** is a very efficient hyperparameter selection algorithm (the number of training steps) and the most commonly used form of regularization in deep learning. Its idea is to find model parameters for the best validation error since its use requires the choice of a performance measure to monitor, a trigger to stop training, and a selection of the model weights to use. The training process can be stopped when the performance of the model on the validation set begins to degrade. The popularity of ES is due to its effectiveness and simplicity.

- **Parameter Sharing:** in specific models, such as the CNN, a significant advantage of parameter sharing over regularizing the parameters to be equal is that only a subset of the parameters (the unique set) needs to be stored in memory instead of learning a separate set of parameters at each location. This technique can lead to a significant reduction in the memory of the model [189].

- **Dropout:** provides a computationally inexpensive but powerful method to regularize models. The dropout layers will randomly removes some of their neurons while a training process with a predefined probability of dropped. Dropout may also be combined with other forms of regularization to yield further improvement [52].

- **Bagging and Other Ensemble Methods:** combining several models into an ensemble is an efficient way to expand the capacity of a model and reduce the bias (boosting method) or variance (bagging method). The latter has a regularization effect by reducing the generalization error [48].

- **Multi-Task Learning:** simultaneously handles multiply related tasks to improve generalization and reduce overfitting by leveraging the domain-specific and auxiliary information contained in the training signals of related tasks through shared representations [27].

We briefly summarize their disadvantages, and advantages of the three major classes in deep learning. Different properties listed in Table III.1.

In details, **Difficulty** indicates the hardness of model's structure and implementation, such as the function of layers ; **Memory usage** refers to the amount of memory consumption for training a deep neural network; **Time series** indicates the ability to learn a model of time series prediction problems; **Generalization** is used to refer the effectiveness of method in diverse media such as images, texts, and audio; **Unsupervised learning** indicates the capability to learn a model without pre-existing labels; **Feature learning** indicates that features can be learned automatically based on a data set; **Invariance** mentions whether the method has demonstrated its powerful for transformations such as rotation, scale and translation; **Parameter Sharing** mentions to the capability to share the weights by all neurons in a particular feature map; **The pros** and **cons** of the method are its advantages and disadvantages.

TABLE III.1: Comparisons between three classes of deep learning.

Deep Learning	**AE**	**RNN**	**CNN**
Difficulty	+++	++++	+++++
Memory usage	+++	++++	+++
Time series	No	Yes	Yes
Generalization	Yes	No	Yes
Unsupervised learning	Yes	No	No
Feature learning	Yes	Yes	Yes
Invariance	No	Yes	Yes
Parameter Sharing	No	No	Yes
Temporal Feature	No	Yes	Yes
How it works	-Dimensionality reduction in latent space keeping maximum input data variance. -Non-linear feature engineering and health index calculation.	-Model time-series and sequential data by propagating state information through time.	-Automatic feature extraction. -Univariate or multivariate convolutions of input. -Combined with pooling methods to reduce dimension.
Pros	-Useful information is retained, while Irrelevance is removed.	-Temporal dependencies are extracted in sequence data.	-Simple yet effective. -Takes advantage of neighbourhoods. -Minimized parameter number so less training time through weight sharing. -Can outperform LSTM.
Cons	-Requires high processing time. -Extracted features not specific for the task.	-Difficult to save the long-term dependence (vanishing gradient problem). -Need more resources.	-High computational complexity for high hierarchical model training.

Note: 'Yes' refers to the class has ability in the characteristic; otherwise, they will be marked by 'No'.

III.4 Deep Learning frameworks

With the great success of deep learning, there are many well-known companies and institutions, such as Google and University of Montreal, which have released DL frameworks. This section compares the widely popular open-source libraries for DL, namely TensorFlow, Keras, PyTorch, DeepLearning4J (DL4J), and Theano. We highlight these frameworks' essential standards and characteristics as shown in Table III.2, showing the supported platforms, programming languages, and each strong point. Table III.2 shows that each framework supports and is adequate for different pre-constructed DL algorithms. Then, the history of GoogleTrends[1] is depicted in Figure III.5. According to Google trend history results, we can conclude that the Keras library is prevalent within the users' community. Based on the characteristics of the frameworks set given in [177], some are more suitable for academic use, and some are more biased towards industry, as shown in Figure III.6.

III.5 Data-driven prognostic challenges and issues

The prognostic research field is still considered recent and immature, which is the reason for the various challenges and issues that have not yet been tackled. Therefore, it provides great opportunities for further research. This section highlights

[1]https://trends.google.com/trends/explore?q=TensorFlow,Theano,keras,PyTorch

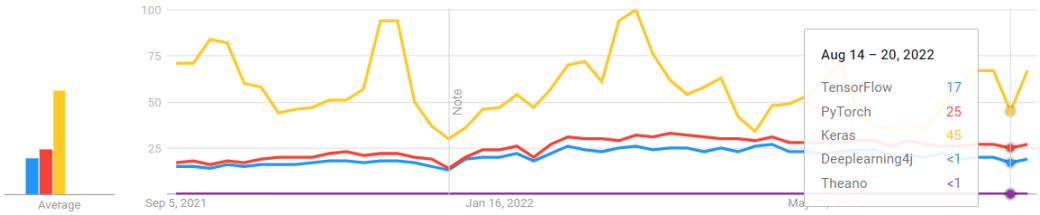


FIGURE III.5: Google Trends history.

TABLE III.2: Comprehensive comparison of DL frameworks.

	TensorFlow	**Keras**	**PyTorch**	**DL4J**	**Theano**
Creator	Google Brain team	François Chollet	Adam Paszke et al., Meta AI	Skymind engineering team	Y. Bengio, University of Montreal
API	Python, C++, Java, JavaScript	Python, R	Python, ONNX	Java, Scala, Clojure or Koltin	Python
License	Apache 2.0	MIT	BSD	Apache 2.0	BSD
Supports model	DNN,RNN,LSTM, CNN,Bi-LSTMs	DNN,RNN,LSTM, CNN,Bi-LSTMs	DNN,RNN,LSTM, CNN,Bi-LSTMs	DNN,RNN,LSTM, CNN	DNN,RNN, CNN
Written in	C++, Python	Python	Python,C	C++, Java	Python
Popularity	Very High Growing very fast	High Growing very fast	Medium Growing very fast	Medium-low Growing low	Medium-low Growing low
CUDA Support	Yes	Yes	Yes	Yes	Yes
Parallel Execution	Yes (Most fexible)	Yes (with the TensorFlow backend)	Yes	Yes	Yes (Not perfect)
Platform	Linux, OSX, Win	Linux, OSX, Win	Linux, OSX, Win, Andr.	Linux, OSX, Win, Andr. (Crossplatform)	Cross-platform
Stars in github	167k	56k	58.4k	12.6k	9.6k
Contributors	3.1k	1k	2.4k	61	351
Website	tensorflow.org	keras.io	pytorch.org	deeplearning4j.org	deeplearning.net/software/theano
Comment	1.The most popular library with complete functionality and several interface support. 2. Interfaces and documentation may be less explicit. 3.Debugging difficulties.	1. High-level API integrating with TensorFlow, and Theano. 2. Documentation is complete. 3. Easy to learn and easy to use. 4. Updates quickly.	1. Pre-trained models are available. However, no visualization tool available.	1. Speeds up training by integrating with Apache Hadoop and Spark. 2. Using Keras API bridges the gap between JVM languages and Python.	1.High level of flexibility. 2. No longer supported after release 1.0.0. 3.The debugging error message is quite tough to comprehend. 4. It is difficult to learn how to use it.

some of the significant challenges and issues related to data and model, grouped into two main aspects.

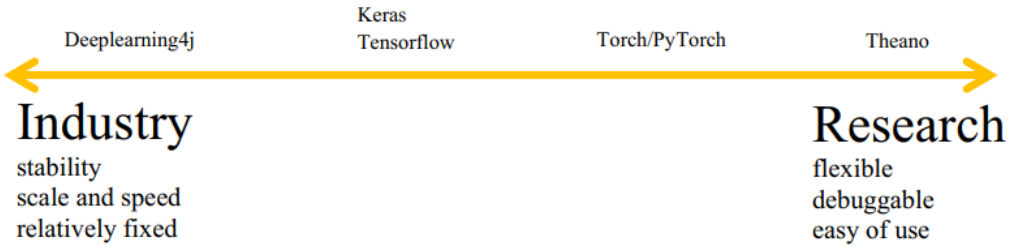


FIGURE III.6: Suitable frameworks for academic research vs industrial applications.

III.5.1 Data-related challenges

Many data-driven frameworks generally demand extensive historical event data, such as the time of failures. In real-world applications, industrial communities and asset owners disallow their components or systems to run to failure (RTF). Most of the time, these components are replaced or overhauled in the presence of the first fault. Therefore, the gathered data, referred to as "suspended data," characterizes the degradation evolution until the first fault without reaching the asset's failure (RTF data deficiency). As a result, the replacement/overhaul times are treated as failure times that conflict with the primary objective of the fault prognosis, leading the model to produce biased estimates (underestimation) of the time to failure. Furthermore, in the case of fault diagnosis, faulty data tends to be relatively rare in most situations compared to healthy ones. This latter is an imbalanced data problem that needs to be addressed.

In other situations, in industrial 4.0, the industry is moving toward the era of big industrial data due to the evolution of smart sensors and IoT technologies. In this context, big industrial data means more missing and incomplete data, noise and outlier data, or even redundant and unnecessary measures. This subsequently translates into lower data quality and high-dimensional data. It is stated that the curse of dimensionality issue decreases uncertainty [45]. Therefore, feature extraction, reduction, and selection are major challenges in the observation phase and are of significant importance and should be considered to enhance the results as well as computational efficiency. In addition, data compression could also assist in the analysis of the data to provide a better understanding of the information. These delicate processes select the most relevant features that would properly distinguish and separate the machine's health states.

III.5.2 Model-related challenges

Due to the exponential increase in computing power and the growth of industrial data, data-driven machine learning publication trends have turned towards more complex models, including ensemble methods and deep learning for three principal PHM tasks: fault detection, diagnostics, and prognostics. Deep learning methods have achieved excellent performance in real-world fault diagnosis and prognostics applications due to their inherent capability to overcome drawbacks of traditional

algorithms dependent on hand-designed features that require the experience of the experts and prior knowledge of signal processing. However, several challenges still hinder its widespread adoption and need to be developed and improved. Most literary works have not discussed nor justified explicitly the choice of suitable architecture to solve specific problems. Until now, most of the proposed architectures have been created manually by human experts, requiring architectural engineering skills and domain expertise. This implies a time-consuming, error-prone, and tedious trial-and-error process. Therefore, the search for an optimal and efficient architecture is still challenging. It is crucial to automate the model selection task in terms of hyperparameter and architecture selection for large-scale real industrial data to discover more efficient and complex architectures.

Many companies are still particularly reluctant to adopt DL techniques, as they cannot trust something they neither understand nor control. The difficulty resides in the "black box" nature of DL algorithms, which imposes a lack of understandability and transparency on the model, i.e., their internal computation mechanisms are unexplainable. Therefore, effective visualization techniques are necessary to provide a minimum level of transparency and to open the black box so that an expert can intervene to modify some parameters and facilitate the building and configuration of intrinsically DL architectures for complex machine health monitoring problems.

III.5.3 Synthesis: Toward enhanced data-driven prognostics

III.5.3.1 Issues to be addressed

The prognostic research domain is far from being mature, which is still new and explains the various challenges that remain unaddressed. Therefore, it offers many research perspectives. We realize that we cannot expect to face many challenges simultaneously. Hence, small cumulative contributions to the scientific development of new disciplines are needed. Hence, the work presented in this thesis will mainly focus on the prognostic of monitored machinery from RUL estimation point of view using DL models. According to the aforementioned challenges, the critical issues of data-driven prognostics to be addressed in this thesis can be pointed out as follows, as research questions.

- Although there are a large variety of proposed prognostic methods, the selection of architecture and the progress in building an effective and efficient prognostic approach is still challenging and limited. *RQ1: How to choose an appropriate architecture? How to improve the prognostic model to accurately and reliably predict the RUL? What about combining spatial feature extractor and temporal feature extractor for improved prediction performance?*
- *RQ2: How is it possible to attain sufficient data quality? What is the impact of data quality on RUL estimation performance? How to deal with imbalanced data?*
- *RQ3: Is the performance of the prognostics method strongly influenced by the hyperparameter selection? How is the optimum value for the hyperparameters selected?*

- *RQ4: How to obtain features that explicitly and properly reflect failure progression?* There is no way to ensure that the relevant features selected are the ones that can lead to better RUL estimates. *RQ5: Do visualization techniques evaluate and analyze the quality of the features and provide a minimum level of transparency?*
- *RQ6: How to discriminate degradation states and define threshold setting of states (classes)? Does it take into consideration the opinion of the expert?*
- *RQ7: Does reducing the conflict zone (i.e., the boundary between classes) lead to fewer conflicting decisions for the classification? How can the conflict zone be identified and reduced?*
- *RQ8: How to validate the performances of a prognostics model?*

The issues highlighted above further emphasize the need to improve the data-driven prognostic method. Therefore, the main assumptions, objective and contributions of this thesis are discussed in the following sections.

III.5.3.2 Assumptions

The contributions made in this thesis aims to enhanced data-driven prognostics by addressing the aforementioned issues. However, these contributions are valid for a certain parameters defined by the following assumptions.

- It is assumed that degradation has been already detected.
- Identification of the number of degradation states and adjustment of threshold parameters under expert assumption. In our study, three states of degradation are assumed, namely: degrading state, transition state and critical state. (As a result of the first assumption, the "steady state" was not considered.)

III.5.3.3 Objective and contributions

Dealing with the aforementioned issues and assumptions, the objective of this thesis is to develop and validate an enhanced data-driven prognostics approach that can estimate the RUL with an acceptable level of accuracy. Therefore, developments are focused on data-processing and prognostics modeling steps. According to that, the main contributions/chapters of this thesis are as follows.

- Provide a brief overview of benchmarking datasets for system-level prognostics (Chapter II) as well as the relevant theoretical foundations needed to understand deep learning and state data-driven prognostic challenges and issues (Chapter III).
- An end-to-end hybrid deep network architecture that jointly optimizes the feature reduction and RUL prediction steps in a hierarchical way. This approach handles the raw data with minimum preprocessing efforts and results in a lower-dimensional continuous representation with minimal variable redundancy that accurately describes asset life cycle degradation (Chapter IV).

- Development of an innovative strategy for RUL estimation that performs a simultaneous assess the health status of degrading machinery (provides the probabilities of system failure in different time windows) and assigns them to the RUL window estimation (Chapter V).
- To investigate the effectiveness of the proposed prognostic approaches, a case study on turbofan engine data (C-MAPSS NASA) from the PHM'08 challenge is presented, and a comparison with results from recent publications is also provided (Chapter IV and V).

III.6 Conclusion

With many successful applications of deep learning in areas such as computer vision and medical image analysis, deep learning continues to gain attention for representation learning, classification and prediction of time series in the field of PHM for fault diagnosis and RUL prediction of engineered systems. Deep learning has shown promising results in interpreting condition monitoring signals such as vibration, acoustic emission, and pressure due to its ability to extract complex representations from raw data. This chapter provided details on the relevant theoretical foundations needed to understand deep learning including existing DL-based architectures as well as identifying the critical issues and challenges for data-driven prognostics that need to be addressed in this thesis.

The following chapter aims to present the first contribution of this thesis, which leverages the power of multimodal and hybrid deep neural network architectures for RUL estimation enhancement. This work is constructing and selecting the appropriate DL model from a set of candidate models based on experimental trials once and once with a strictly theoretical background.

Chapter IV

Leveraging the Power of Multimodal and Hybrid Deep Neural Network Techniques for RUL Estimation Enhancement

IV.1 Introduction

Improving the availability and reliability of critical engineered assets while reducing untimely expenditures and safety risks is a critical challenge facing industrial and manufacturing systems. For this reason, prognostics is considered an essential PHM process with future capabilities, which mainly focuses on predicting the residual lifetime during which a machine can fulfil its intended function, i.e., estimating the RUL of a system. RUL is an important real-time performance indicator of operating systems under working conditions. Indeed, accurate RUL estimations enable improved decision-making for operations and maintenance of such systems with the goal of approaching zero downtime, which is a delicate process and is influenced by several unknowns related to the chosen model. The latter has inspired researchers to develop a variety of approaches to predict RUL. However, the choice of architecture and progress in building an effective and efficient prognostic approach is still challenging and limited. Therefore, while building a prediction model it is important tackle following issues. How is it possible to attain sufficient data quality? How to handle complex multidimensional data? How to choose an appropriate architecture? How to improve the prognostic model to accurately and reliably predict the RUL?

To account for such challenges, novel RUL prediction approaches are proposed by applying advanced deep learning architectures that focus on feature reduction, aiming to achieve a data representation with low dimensionality and minimal variable redundancy while maintaining the critical asset information needed for data-driven analysis. The principal contributions of this chapter are summed up as follows:

- An end-to-end deep network architecture that jointly optimizes the feature extraction/reduction and RUL prediction steps in a hierarchical way, which handles the raw data with minimum preprocessing efforts.
- Convolutional Auto-encoder integrates convolutional calculation with autoencoder to effectively extract spatial and useful degradation features.
- Capitalizing on the recent success of multi-model deep learning techniques, hybrid deep neural network approaches for RUL estimation are proposed.
- Multi-model assists to leverage the power of different models in a parallel to capture various information through different time intervals.
- Selection and construction of the appropriate DL model from a set of candidate models based on experimental trails once and once with a strictly theoretical background.

This chapter presents the first contribution in this thesis which aims to leverage the power of multimodal and hybrid deep neural network architectures for RUL estimation enhancement. The chapter's remainder is structured as follows: Section IV.2 reviews recent applications of deep learning models on the C-MAPSS dataset. Section IV.3 describes the proposed hybrid methods for RUL estimation of a turbofan

engine. The experimental results and discussions that demonstrate the effectiveness and superiority of the proposed hybrid models are considered in Section IV.4. Finally, we close the chapter with conclusion and future works in section IV.5.

IV.2 State-of-the-art Deep Learning Methods for Engines RUL Estimation

In recent years, the use of deep learning for representation learning, classification and prediction of time series in PHM has attracted increasing attention. The industrial dataset is possible to gather due to industrial IoT, which has promoted opportunities for industry and academia to leverage advanced data-driven techniques (DL-based architectures). Indeed, NASA's C-MAPSS turbofan time-to-failure dataset have been extensively analyzed with RUL estimation as a primary focus. In this section, we survey works that applied to the C-MAPSS dataset and have leveraged deep learning methods to tackle the task of RUL estimation. The selected works are presented in the following excerpts, which either applied the CNN, LSTM, or DNN using auto-encoders.

IV.2.1 CNN

Within the deep learning architecture, the first implementation of CNN for RUL estimation of aircraft engines was proposed by Babu et al., where the input data is segmented into sliding windows and afterward normalized. The CNN structure's ability to learn a higher-level abstract representation along with the multi-channel time series through its convolutional and average-pooling layers is shown. A linear regression layer is attached to the top layer to perform RUL predictions. The results showed the superiority and effectiveness of the CNN model over other machine learning models such as the Multilayer Perceptron (MLP), the Support Vector Machine (SVM), and the Relevance Vector Machine (RVM) [7]. In a similar study, Li et al. proposed a novel deep CNN-based approach for RUL forecasts of aircraft turbofan engines. The authors employ a time window strategy for data processing to improve feature extraction via deep CNN. The normalized sensor data are directly utilized as the model inputs. Besides, they use the dropout technique to prevent overfitting. This model achieves the most accurate estimation of RUL and the lowest Root Mean Square Error (RMSE) than 13 other data-driven methods. The authors also highlighted that optimum performance is achieved through 5 convolution layers and a time window length of 30 [99].

IV.2.2 RNN and its variants

The Recurrent Neural Network (RNN) retains internal memory to process sequential data. However, RNNs had the vanishing gradient problem arising in long sequence input, which cannot keep the previous information, except the latest one. To handle this issue, Zheng et al. suggested an engine RUL prediction method based on deep

LSTM, capable of capturing long-range dependencies of different time scales. The model consists of two LSTM layers combined with two Feed-forward Neural Network (FNN) layers and the output layer [200]. Results reveal that deep LSTM outperforms CNN presented by other researchers [7] based on RMSE. Similarly, Hsu et al. proposed an LSTM to address the RUL prediction problem for turbine engines, which can effectively to extract temporal dependencies from historical data [73]. Liao et al. have used LSTM, relying on the bootstrap procedure for uncertainty estimation of RUL. The bootstrap method is a good solution to obtain uncertainty prediction without any sensor data distribution [100]. The proposed approach achieved higher accuracy than CNN, and LSTM discussed by [7] and [200], respectively. Additionally, Yuan et al. proposed an LSTM to determine the fault location and estimate the RUL of the aero-engine in cases of complicated operations, hybrid failures and intense noises [188]. More recently, Wu et al. proposed a new deep LSTM approach for discovering the hidden long-term dependencies among sensor time-series signals to predict RUL. The grid search was also applied to tune the hyper-parameters, thereby obtaining the best network structure [181]. This method showed enhanced performance compared to other methods in the literature.

Another variant of LSTM used by Wang et al. is Bi-directional LSTMs that can learn the bi-directional temporal dependencies from sensor data for Aircraft Engine RUL estimation. It can capture long-range information in both futures (forward) and past (backward) contexts of the input sequence simultaneously [99]. In another study, a new bi-directional LSTM model was presented by Zhang et al. to identify the system degradation performance and subsequently predict RUL [195]. The proposed model consists of two BDLSTM layers and achieved promising results compared to LSTM, bi-directional RNN (BDRNN), MLP, and CNN reported by [7]. Another main variant of RNN that is recently utilized for RUL estimation and enhanced LSTM-model with few parameters, the Gated Recurrent Unit (GRU), is presented by Chen et al. [32]. The authors proposed a new approach for RUL estimation of a nonlinear degradation process, using Kernel PCA (KPCA) as the first phase for dimensionality reduction and nonlinear feature extraction. The second phase uses GRU to prevent the problem of long-term dependency and allows each recurrent unit to extract dependencies of different time scales adaptively.

IV.2.3 DNN using auto-encoders

In addition to the CNN and RNN architectures, AE is another main structure that is essentially a feature extractor for reducing data monitoring conditions performed in an unsupervised manner. Many studies have shown the leverage of using AE alongside another machine learning method for estimating the RUL of a turbofan engine [105, 158]. Song et al. proposed a new hybrid model integrating the advantages of AE and bidirectional LSTM to enhance the RUL's prediction accuracy [158]. The main idea is that the encoding part of AE (bottleneck) acts as input for the BDLSTM to produce the expected output. The results demonstrate that the combination of AE and BDLSTM outperformed the other methods, such as MLP, CNN,

LSTM, BDLSTM, and Autoencoder-LSTM. In this work, Ma et al. also proposed a novel end-to-end deep architecture based on a stacked sparse Autoencoder (SAE) and logistic regression. This study utilized the grid search procedure to optimize the hyper-parameters of the SAE model [105].

Inspired by these previous studies, the idea of the hybrid approach and applying a parallel multi-model emerged to leverage the power of different methods, which have a high potential to boost prognostic accuracy instead of incorporating only a single model such as CNN, DNN, or LSTM. Accordingly, capitalizing on the recent success of DL, this work presents a framework driven by an end-to-end ML system that introduces two new hybrid RUL prediction approaches to capture various information through different time intervals. Previous studies have shown the advantage of using AE alongside another machine learning method for estimating the RUL of the turbofan engine. Traditional AE uses a fully connected layer as reported in the literature, while the CAE model has promising feature extraction and dimensionality reduction capability through convolutional layers.

Aligning with Convolutional Auto-Encoders's power, the first promising proposed hybrid model adopts CAE in an aero-engine prognostic problem to extract automatically useful features with high-level abstractions. These CAE features serve as inputs to train the two temporal modeling tools simultaneously in a parallel manner (referred to as BDLSTM path and BDGRU path) that can capture more robust features and eventually predict the RUL. Although CAE has been applied to different tasks, to the best of our knowledge, this is the first use of CAE in RUL's estimation problem for engine turbofan.

For a comprehensive comparison, the second hybrid architecture proposed differently from what has been reported in the literature. It consists of CNN and BDGRU models simultaneously in parallel paths to capture local and temporal features directly from raw sensory data instead of just using CNN. The outputs from both paths (CNN and BDGRU) are concatenated to obtain the target RUL. The GRU has appeared as an enhanced LSTM model with few parameters for improving the training phase's speed and model performance. Besides, we used a BDGRU for the bi-directional temporal feature extraction and to prevent the long-term dependency problem. The superiority of the two proposed hybrid models is demonstrated using the public NASA's C-MAPSS dataset and compared with all its counterparts and the most robust results in the literature.

IV.3 Proposed Hybrid Deep Neural Network architectures

This section introduces the relevant hybrid deep learning approaches proposed in this contribution for the RUL prediction of an aircraft engine. Figure IV.1 describes the proposed approach for RUL estimation, which consists of two main stages. In the entirely offline training stage, the recorded historical data from sensors flow through the components of the training stage. Ultimately the degradation model for the RUL estimation is constructed based on deep learning methods. In the online prediction stage, the obtained current data is stored in the dataset and processed to obtain

normalized sequence data, where the trained model is applied to predict the RUL. Based on the RUL values, a maintenance action will be applied to the system at the exact scheduled moment.

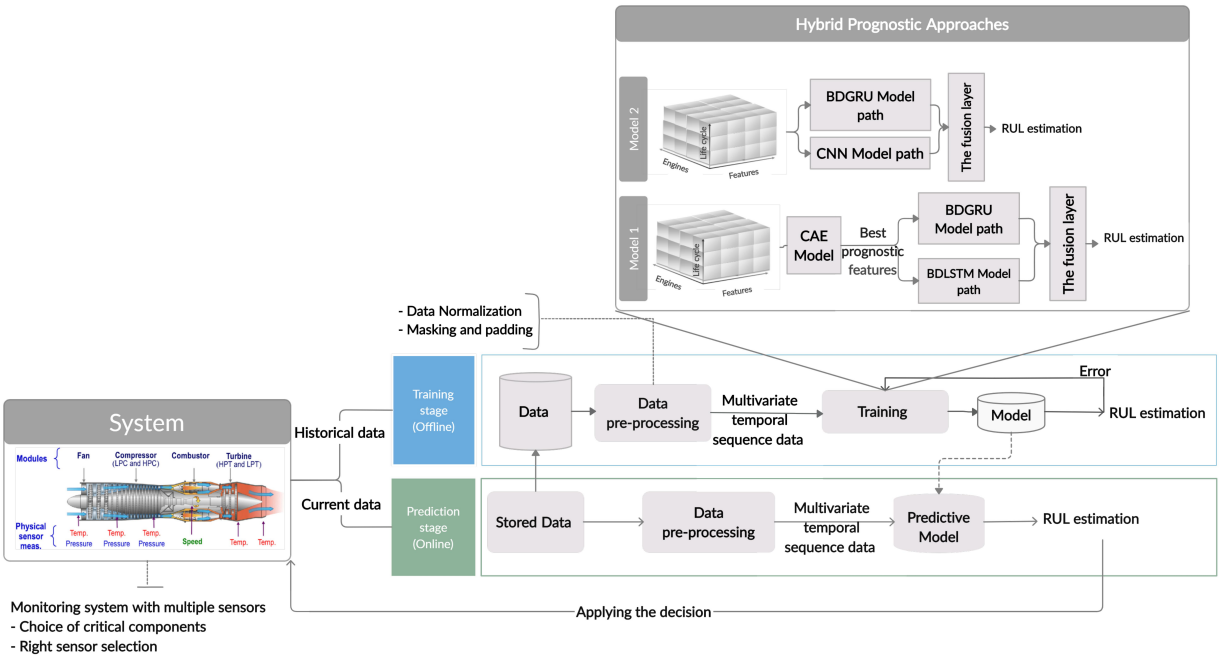


FIGURE IV.1: The proposed approach for RUL estimation.

Deep learning methods are proposed (φ) in this thesis to address the non-linearity function. Let X^i denote its input, and the observed RUL is as its output (formulated in Chapter II, Section II.5.1).

$$\check{RUL}^i = \varphi(x^i, RUL^i) \tag{IV.1}$$

To minimize the error between the predicted RUL value and the observed target RUL at time t.

$$Minimize : \{\check{RUL}_t^i, RUL_t^i\} \tag{IV.2}$$

IV.3.1 Convolutional Auto-encoder with BDGRU-BDLSTM Hybrid Model

As illustrated in Figure IV.2, our proposed hybrid model comprises two main stages: The first one is CAE, which is used to automatically extract performance degradation features while lowering the dimension of multiple sensors in an unsupervised manner. The second stage is the temporal modeling tool, which combines BDGRU and BDLSTM models simultaneously and in a parallel way to provide the RUL's estimation. The full details of the two main stages are described as follows.

A. **Stage 1: CAE module**

The CAE architecture consists of two parts, an encoder and a decoder, two symmetrical and reversed structures. The encoder network part comprises six convolution layers with the same filter size (10×1) and one max-pooling layer.

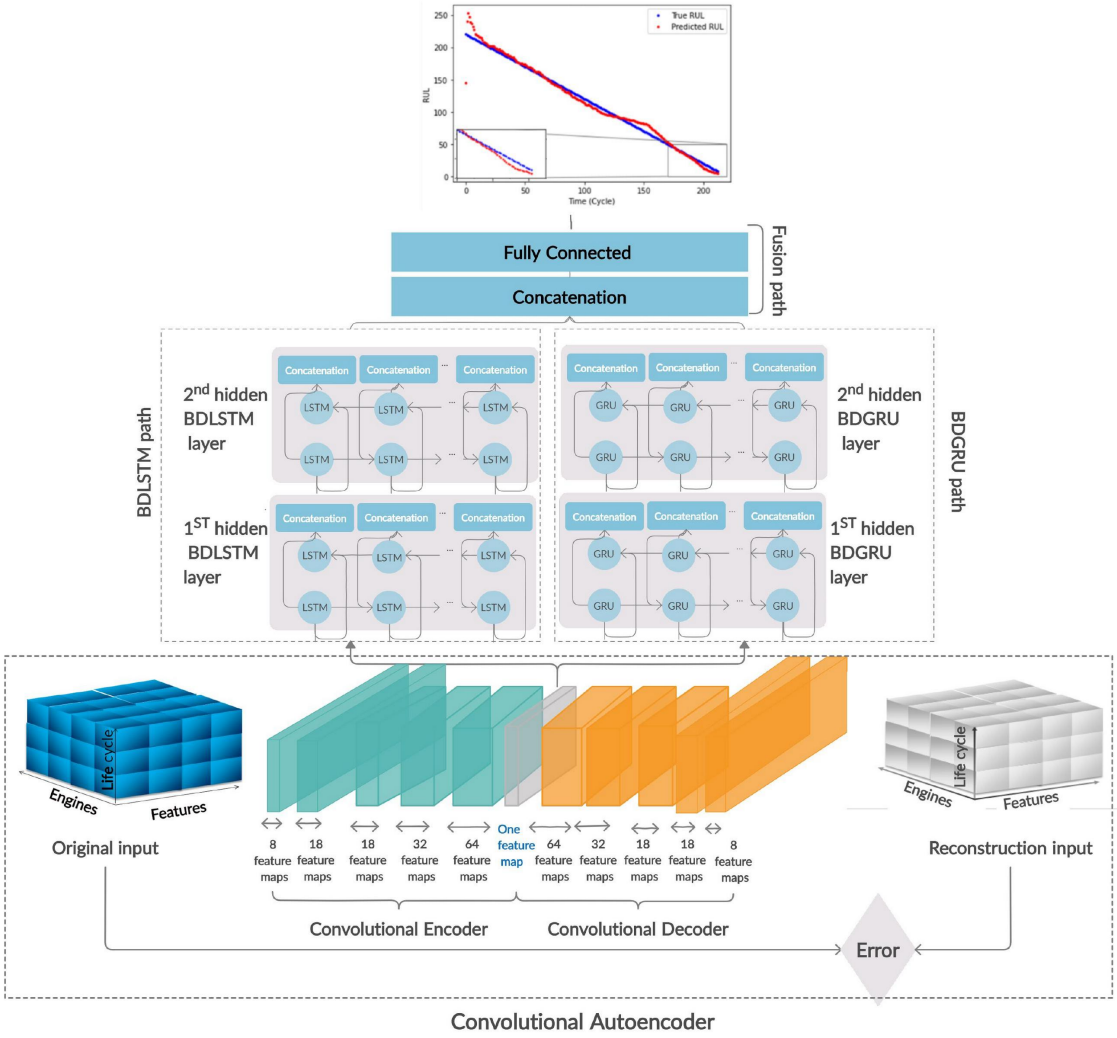


FIGURE IV.2: The first proposed hybrid model based on CAE with BDGRU-BDLSTM.

Precisely in this work, two pairs of convolution layers are stacked, where the number of filters is set to 8 and 18, followed by one Max-pooling layer with filter size (1×2). The third and fourth CNN layers consist of 32 and 64 filters, respectively. After every two pairs of convolution layers, a dropout layer is added to reduce overfitting and avoid repeatedly extracting identical features. The final convolution layer's filter is defined by one to obtain a unique feature map. All convolution layers used Rectified Linear Unit (ReLU) as the activation function. Furthermore, the zero-padding operation is used to maintain the feature map unaltered.

The operations of un-pooling and de-convolution are used in the decoder part of CAE to reconstruct the input instead of convolution and max-pooling operations used in the encoder part.

B. **Stage 2: BDGRU-BDLSTM hybrid module**

The useful features learned from CAE are used as input to the multi-model

structure that can maintain good generalization performance. The proposed multi-model is a temporal modeling tool, which is based on the combination of BDGRU and BDLSTM models simultaneously. This combination aims to obtain more robust features and eventually predict the RUL. Both paths (BDGRU and BDLSTM) share the same configuration, where two layers are stacked with 50 nodes. The layers use the hyperbolic tangent (tanh) as an activation function. A dropout technique is applied with a rate of 0.25 per layer. In the end, to estimate the RUL, the outputs from both paths are concatenated and will be inserted into a fully connected layer; this layer has one neuron and uses the exponential activation function.

IV.3.2 CNN-BDGRU Hybrid Model

As shown in Figure IV.3, the proposed hybrid model is based on the combination of CNN and BDGRU in a parallel manner for regression. The CNN path acts as a spatial feature extractor, while simultaneously, the BDGRU path is utilized for the bidirectional temporal feature extraction. Although there is no correlation between the two pathways, their outputs are concatenated to obtain the RUL's overall prediction.
Specifically, the BDGRU structure is designed to handle each sequence of data in two directions, through the GRU cells forward for prediction and backward direction for smoothing the prediction and relieving the noise impact. Below, we detail the structure of three major components of our hybrid model:

A. **The CNN path**

In our proposed model, CNN is exploited to capture spatial features by stacking the convolutional kernels. CNN is composed of five convolution layers, which have the same filter size (10×1). The first and second CNN layers consist of 18 filters, while the third and fourth layers contain the same number of 32 filters. The final convolution layer is used with a single filter to fuse all the previous feature maps to obtain a unique feature map. The ReLU is applied along with zero-padding for all convolution layers. In this way, a high-level representation is obtained for each raw collected feature.

B. **The BDGRU path**

The BDGRU is selected to learn the long-rang dependencies of features. Through this path, both forward direction and backward direction are computed in two separate GRUs independently. Their outcomes are fused and distributed to the next layer. Two BDGRU layers are stacked within the same configuration as the first proposed method used.
Besides, the BDGRU and GRU share the same cell architecture that allows for addressing the vanishing gradient problem. Furthermore, the hidden state of the BDGRU cell is expressed as follows:

$$h_T = (\overrightarrow{h_T} \bigotimes \overleftarrow{h_T}) \tag{IV.3}$$

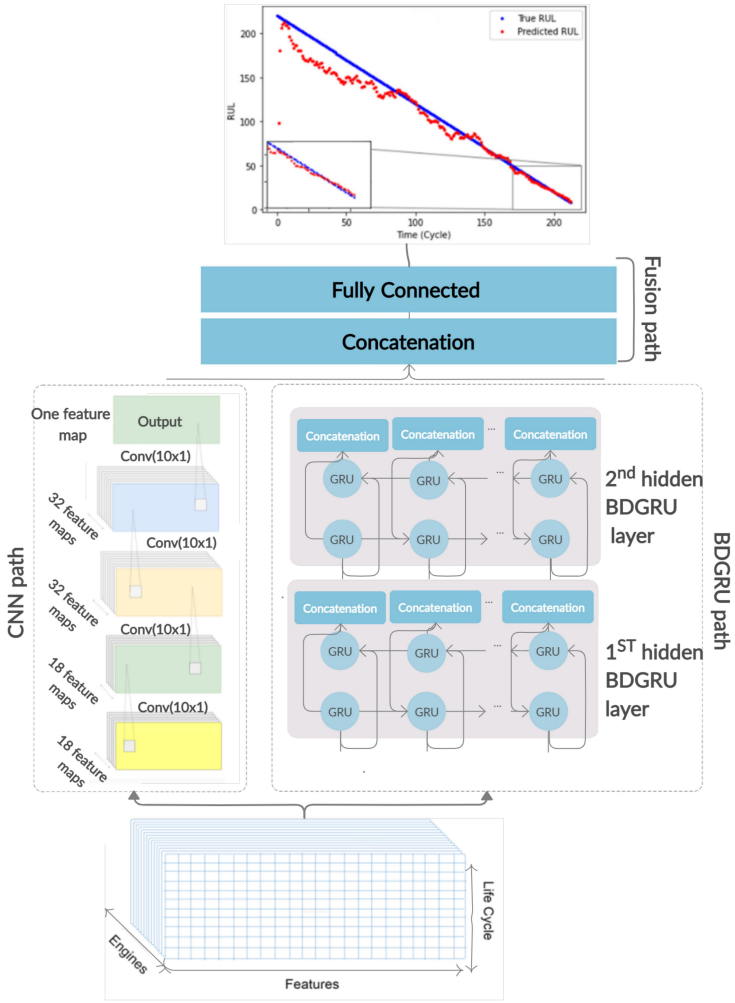


FIGURE IV.3: The second proposed hybrid model based on CNN and BDGRU.

Where $\rightarrow$ and $\leftarrow$ symbolize forward and backward process, respectively.

C. **The fusion path**

The final prediction at each time step is achieved by concatenating the outputs from both paths (CNN path and BDGRU path). This fusion layer has one neuron and uses the exponential activation function.

IV.4 Experiment study

In this section, the performance of the deep learning-driven prognosis approaches is evaluated on a prognostic benchmarking problem (C-MAPSS simulated turbofan engine dataset, Chapter II Section II.5.1). It starts with data processing which includes the data normalization, the masking, and the padding phase. Followed by evaluation metric and prediction procedure in Section IV.4.2 and IV.4.3, respectively.

Finally, the details of the results and comparisons with several architectures to provide a comprehensive examination of the proposed models are discussed in Section IV.4.4.

IV.4.1 Data pre-processing

The subsets FD001 and FD003 exhibit constant sensor measurements and three operational settings throughout the lifetime of the engine; which could not be useful for RUL estimation. However, all three operational sensor settings and all sensor measurements can provide useful information about the deterioration of a turbofan engine in FD002 and FD004. Consequently, unlike works [99], [105] [172] excluded the three operational settings and selected 14 sensors out of the 21 sensors. In the proposed methods, all three sensor settings and all sensor measurements are picked as input features for all sub-datasets. The goal is to avoid designing features manually by proposing flexible models of an End-to-End ML system using Deep Learning.

It is essential to prepare the data before training the models. Therefore, the data normalization, the masking, and the padding phase are used in this contribution.

IV.4.1.1 Data Normalization

According to the differentiation issue of feature range scales, several normalization methods have been proposed to ensure the same range scale of all features [125]. The Min-Max normalization given in Eq.IV.4 is used to map the raw features within the range of [0,1].

$$x_i' = \frac{x_i - \min(x_i)}{\max(x_i) - \min(x_i)} \tag{IV.4}$$

Where x_i is the time sequence of the i^{th} sensor measurements, Min and Max are the minimum and maximum values in x_i given its range, and the x_i' is the normalization input data. Figure IV.4 illustrates FD002 testing data before and after normalization.

IV.4.1.2 Masking and padding

The engines have varying length cycles; hence, the shorter sequences than the maximum length of cycles in the whole dataset are padded with zeros to obtain the same length. Consequently, mask zero is used in the training phase to record if the time step already exists or is just padding.

IV.4.2 Evaluation Metric

In this work, the RMSE and scoring function are used for evaluating the model's performance. The formulation of RMSE is defined in Eq.IV.5 to measure the effectiveness of the RUL prediction methods. The RMSE function penalizes both early (underestimate) and late (overestimate) predictions.

$$RMSE = \sqrt{(1/N)\sum_{i=1}^{N}(\Delta_i^2)} \tag{IV.5}$$

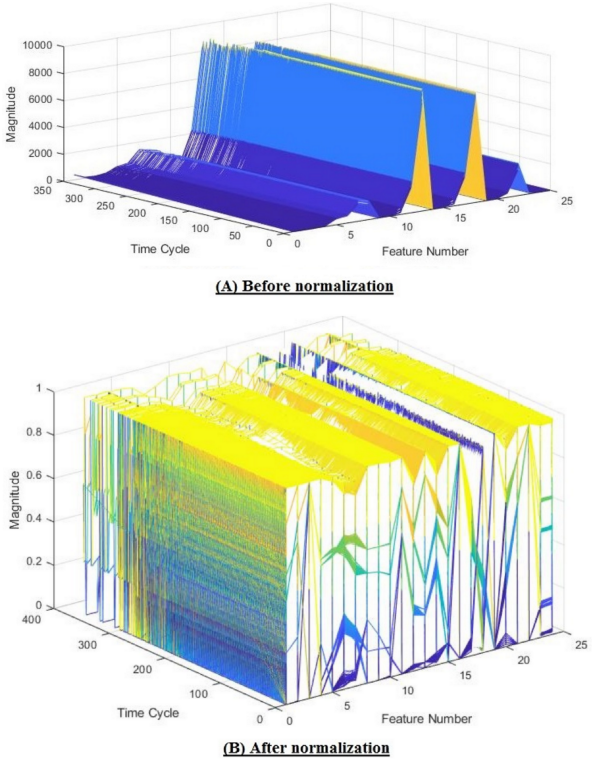


FIGURE IV.4: An illustration of FD002 testing data before and after normalization.

where N represents the total number of data samples. $\Delta_i = RUL'_i - RUL_i$, Δ_i is the error between the predicted RUL' value and the true RUL for the i^{th} test samples.
The scoring function adopted by the international conference on PHM data challenge is shown in Eq.IV.6

$$\begin{aligned} S &= \sum_{i=1}^{N} S_i \\ S_i &= \begin{cases} e^{-\Delta_i/13} - 1, & \text{if } \Delta_i < 0 \\ e^{\Delta_i/10} - 1, & \text{if } \Delta_i \geq 0 \end{cases} \end{aligned} \tag{IV.6}$$

This scoring function takes into account the impact of the maintenance costs, in which a higher penalty is imposed when the RUL is overestimated. Under this estimation, the maintenance will be scheduled after the appropriate time.

IV.4.3 Prediction procedure

For both proposed methods, first, the C-MAPSS sub-datasets are pre-processed where the data are normalized and padded. Next, the training sub-datasets are split into training and validation sets; 80% of the engines in sub-dataset are randomly selected for the training, while the remaining 20% are designated as validation set.

IV.4.3.1 CNN-BDGRU Training procedure

The flowchart of the proposed CNN-BDGRU is described in Figure IV.5. The hybrid model receives as inputs the normalized training set and the RUL values adopted as the target outcomes. The shape of the BDGRU input is a 3D tensor ([Min_Batch_size, Time_series_length, Feature_size]). On the other side, the shape of the CNN input is a 4D tensor ([Min_Batch_size, Time_series_length, Feature_size, 1]), where Min_Batch_size is the number of engines in mini-batch size equal to 128. Time_series _length equal to the maximum length of cycles in the whole dataset, and Feature_size equal to 24.

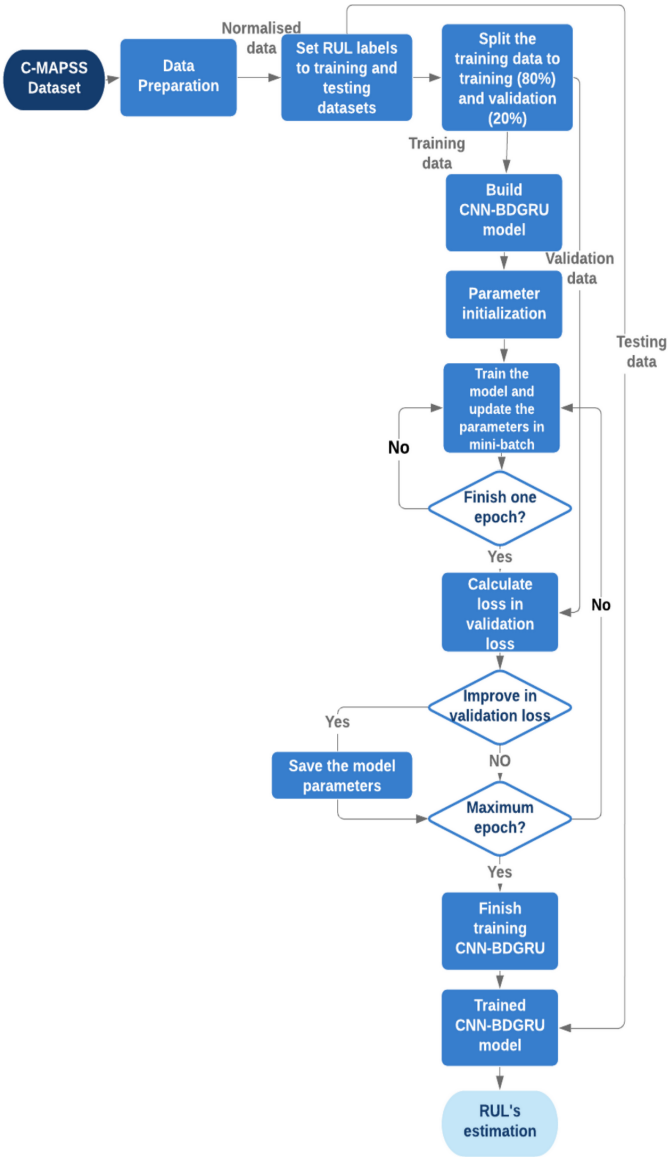


FIGURE IV.5: The flowchart of the proposed CNN-BDGRU model.

In the training process, a gradient based optimization algorithm adjusts the weights in the network based on the minimization of the objective function. Specifically, the Root Mean Square propagation (RMSprop) optimizer method is used to optimize the training model, with the learning rate set at 0.001 to achieve stable convergence

[135]. Besides, the Mean Square Error (MSE) serves as the loss function, which is expressed as,

$$MSE = (1/N)\sum_{i=1}^{N}(\Delta_i^2) \tag{IV.7}$$

Where $\Delta_i = RUL'_i - RUL_i$.

The maximum number of training epochs is 1000. For each training epoch, the samples are segmented into mini-batches. To avoid overfitting, the early stopping technique of regularization introduced. Its principal idea is that in the absence of the improvement in performance, the training process is discontinued.

Finally, the testing data samples are fed into the hybrid training model to estimate the RUL and obtaining the RMSE accuracy in the test set.

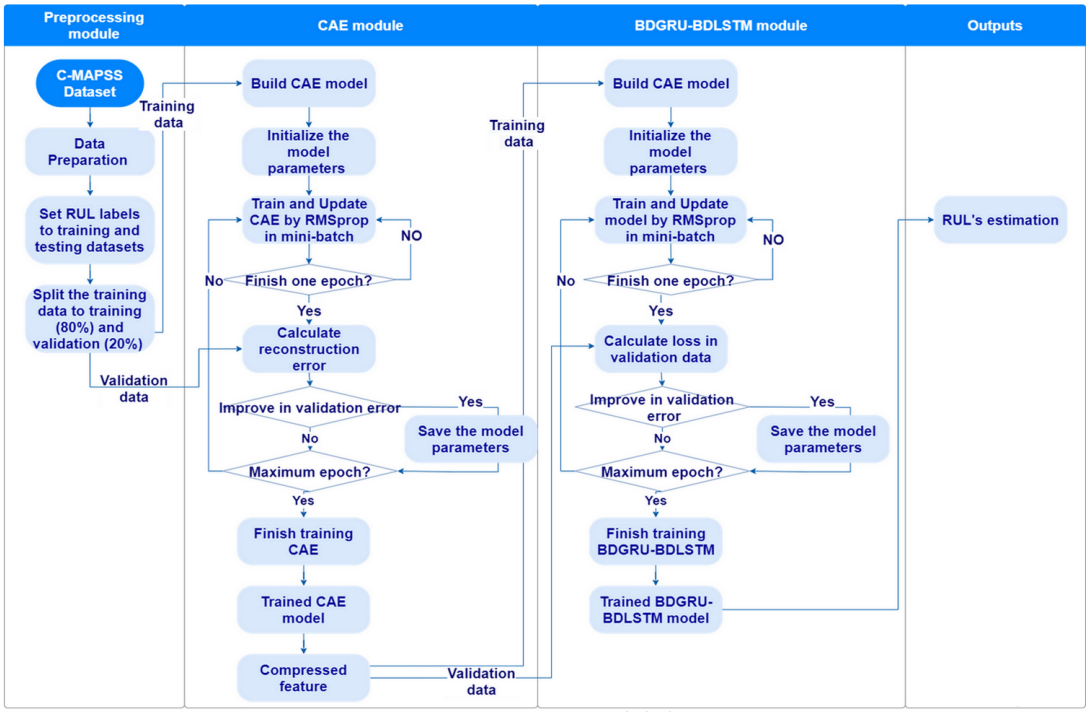


FIGURE IV.6: The flowchart of the proposed hybrid model based on CAE and BDGRU-BDLSTM.

IV.4.3.2 CAE with BDGRU-BDLSTM Training procedure

The process of training the proposed hybrid CAE with the BDGRU-BDLSTM model consists of two modules: a CAE model as the first phase and the BDGRU-BDLSTM method in the second phase. Both modules used the RMSprop optimizer and the MSE as a cost function. The whole process of training the proposed deep hybrid CAE with the BDGRU-BDLSTM model is summarized in Figure IV.6.

The whole CAE network is trained in an unsupervised manner that takes the normalized training set as input to reconstruct it; the encoder part represents the more robust deterioration features. The CAE network's weights updated iteratively during training through a gradient-based optimization algorithm based on the minimization of reconstruction errors (MSE), expressed in Eq.IV.7, where $\Delta_i = X'_i - X_i$,

Δ_i is the error between the reconstruction X' and the original input X for the i^{th} test samples. Besides, the samples are grouped into mini-batches for each training epoch, with a limit of 1000 training epochs.

After the CAE network's training process, the second step is to train the multi-model (BDGRU-BDLSTM) for the RUL estimation, where the encoder parameters are frozen during the multi-model training step. The extracted CAE features are fed to the multi-model (BDGRU-BDLSTM) as inputs, and the RUL values of the training set are used as the target outputs. Backpropagation through time (BPTT) is the training algorithm applied to update the weights for minimizing the error using RMSprop, with the learning rate set at 0.001. Furthermore, MSE is utilized as the loss function, which is expressed in Eq.IV.7, Where $\Delta_i = RUL'_i - RUL_i$. The total number of training epochs is 1000, with the application of the early stopping technique.

Finally, the convolutional encoder is used jointly with the BDGRU-BDLSTM model for the RUL estimation and obtained the RMSE accuracy on the test set in the operating phase.

IV.4.4 Results and Discussions

IV.4.4.1 Prediction performance

In this section, the obtained prediction results from applying each of the proposed hybrid models to the turbofan engines datasets are presented. The purpose of this approach is to make a thorough comparison of the different DL approaches for RUL predictions. The actual and predicted RUL values during the whole life-time of the two randomly selected engines out of several testing engine units across the four datasets (i.e., FD001-FD004) for both methods are depicted in Figure IV.7 - IV.10.

It is worth noting that the RUL prediction results for all engine units over the four sub-datasets are precisely predicted, especially for RUL's estimation at the last cycles of the engine unit is more reliable and closer to the true RUL than at the early cycles. Besides, it can be observed that when the RUL engines are large, the accuracy prediction is noticeably higher conversely to the smaller RUL engines (as shown in Figure IV.7 (b) engine 47). The reason is that when the engine degradation reaches failure, the fault features increase, and that can be extracted through the proposed methods and obtain better perdition results. The RUL's engine is linearly decreasing with time until the degradation engine samples are available. Moreover, accurate estimation of the late period in the engine life cycle plays a crucial role in enhancing operational reliability and system availability, maintaining workplace safety, and reducing maintenance costs.

According to Figure IV.11, we can easily observe from the distribution of box plots for experiments that the performance of the proposed hybrid model (CAE with BDGRU-BDLSTM) generally performs well on all four sub-datasets, in particular, FD002 and FD004 that are very complicated and the existing models typically fail to provide accurate prediction results for these sub-datasets. Hence, the CAE with the BDGRU-BDLSTM model achieves a good result on FD001 and FD003, the

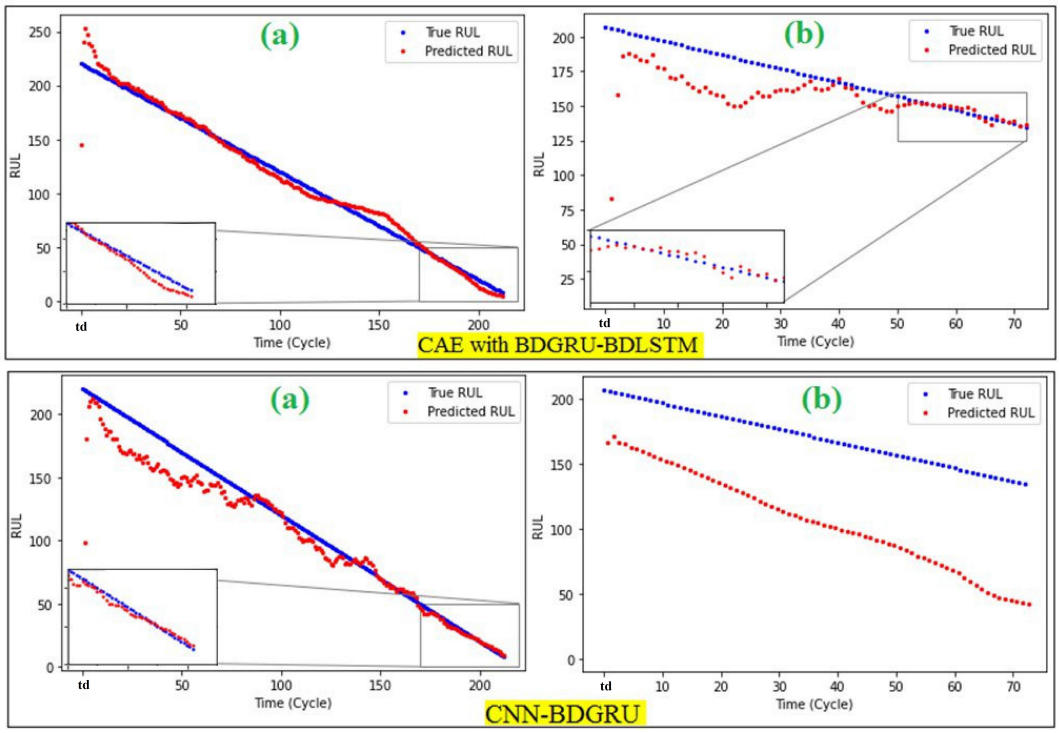


FIGURE IV.7: Predicted full life cycles of two testing engine units in FD001 dataset for both hybrid methods: (a) engine #81, and (b) engine #47.

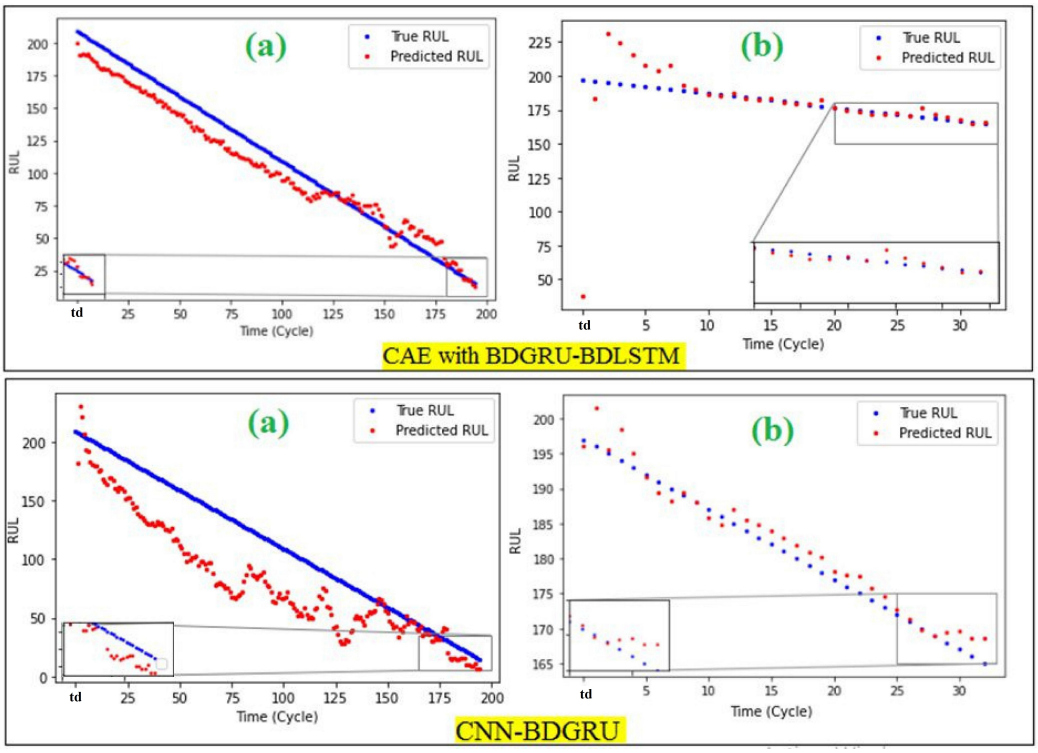


FIGURE IV.8: Predicted full life cycles of two testing engine units in FD002 dataset for both hybrid methods: (a) engine #45, and (b) engine #218.

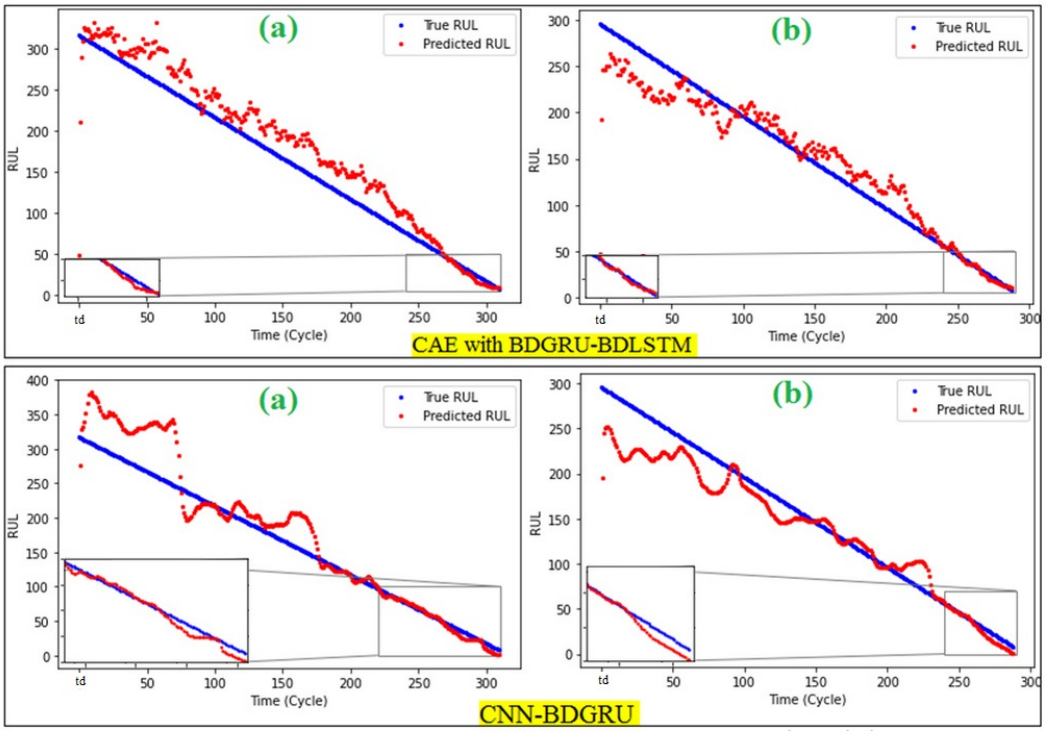


FIGURE IV.9: Predicted full life cycles of two testing engine units in FD003 dataset for both hybrid methods: (a) engine #39, and (b) engine #99.

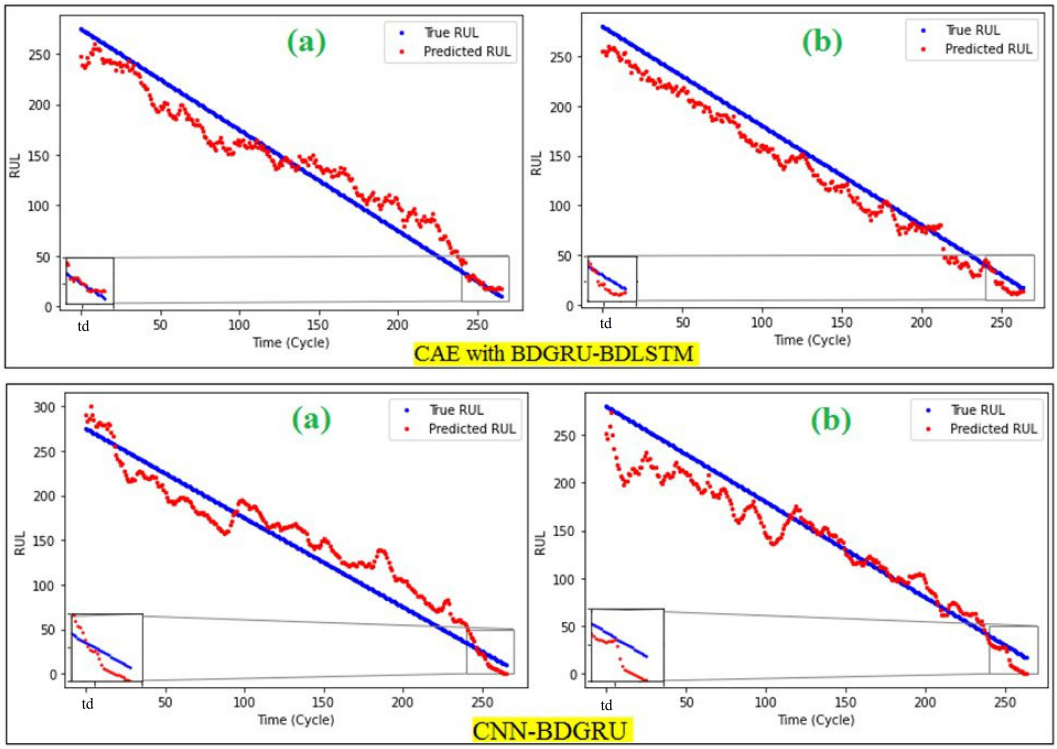


FIGURE IV.10: Predicted full life cycles of two testing engine units in FD004 dataset for both hybrid methods: (a) engine #40, and (b) engine #216.

simplest sub-datasets. Table IV.1 shows the results of both proposed hybrid models in terms of the values of RMSE and score, where IMP is the improvement of the proposed CAE with BDGRU-BDLSTM model over the CNN-BDGRU model. It is defined as IMP= (1-(CAE with BDGRU-BDLSTM / CNN-BDGRU)) $\times$ 100. From the IMP values, we can observe that the CAE with BDGRU-BDLSTM hybrid model consistently obtains RMSE values lower than the CNN-BDGRU model, which has improved the performance in term of RMSE to 14.208%, 6.83%, 3.967%, and 5.537% for group FD001, FD002, FD003 and FD004, respectively.

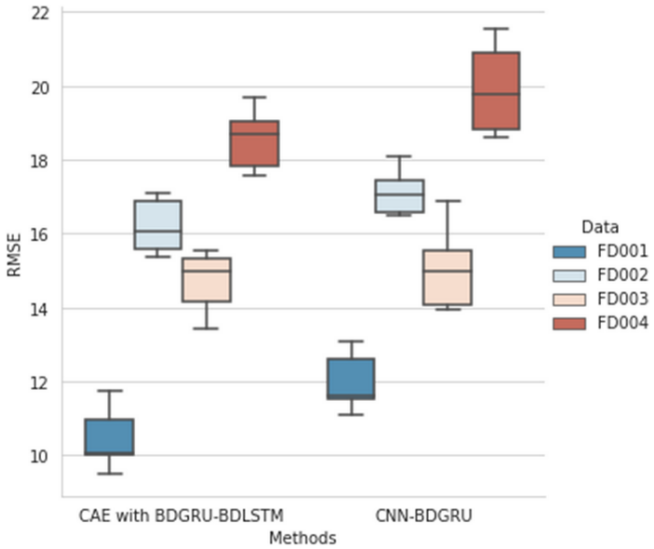


FIGURE IV.11: Box plot of the RMSE for both proposed hybrid models over the NASA turbofan engines datasets.

TABLE IV.1: RMSE and score values of the proposed methods on C-MAPSS dataset

Methods		**FD001**	**FD002**	**FD003**	**FD004**
CAE with BDGRU-BDLSTM	RMSE	9.51	15.35	13.41	17.57
	Score	213	1274	350	1528.18
CNN-BDGRU	RMSE	11.085	16.476	13.964	18.60
	Score	245	1198.42	387	1592
IMP	RMSE	14.208%	6.83%	3.967%	5.537%
	Score	13.06%	-6.3%	9.56%	4.19%

In term of Score values, the proposed hybrid model (CAE with BDGRU-BDLSTM) achieved the lowest Score than the CNN-BDGRU model on FD001, FD003 and FD004, while on FD002, it was a slightly higher Score (worst results). The IMP in term of Score values is around 13.06%, 9.56%, 4.19% for group FD001, FD003 and FD004, respectively.

IV.4.4.2 Computational Cost Analysis

The time complexity for both proposed methods (CAE with BDGRU-BDLSTM, CNN-BDGRU) is discussed in this section. The complexity of the pooling and the Fully Connected layers (FC) takes 5-10 % of the overall computational time [63]. Therefore, their complexities are not involved in the total time complexity of the proposed models.

The complexity of the CNN layers is calculated as: $\sum_{l=1}^{d} n_{l-1}\ s_l^2\ n_l\ m_l^2$, where l is the index of a convolutional layer, d is the number of convolutional layers, n_l is the number of filters in the l-th layer, n_{l-1} is the number of input channels of the l-th layer, s_l is the spatial size of the filter, and m_l is the spatial size of the output feature map.

Considering that LSTM is local in both space and time, which means that for each time step LSTM's storage complexity does not depend on the input sequence length [70]. We conclude that LSTM's complexity per time step and weight is estimated just as O(1). Therefore, the overall complexity of all LSTM layers per time step is equal to: $\sum_{i=1}^{d}\ W_i$, where W is the number of weights, i is the index of a LSTM layer, d is the number of LSTM layers. The time complexity of GRU and FC is similar to an LSTM. While the BDLSTM or BDGRU's runtime complexity is increased by twice ($\sum_{i=1}^{d}\ 2\ W_i$). Furthermore, W equals KH + KCS + HI + CSI for LSTM or GRU architecture. On the other hand, for FC, W equals IH + HK. Where I is the number of inputs units, K is the number of outputs, H is the number of hidden units, C is the number of memory cell blocks, S is the size of the memory cell blocks.

(A) **Computational Complexity of CNN-BDGRU architecture**
The CNN-BDGRU complexity per time step can be calculated as the sum of the complexities of the convolutional layers and the BDGRU layers.

$$O(\sum_{l=1}^{5} n_{l-1}\ s_l^2\ n_l\ m_l^2\ +\ \sum_{i=1}^{2} 2\ W_i) \tag{IV.8}$$

For all the training processes with a function of the input length (x) and epochs (e), the total time complexity is equal to:

$$O([\sum_{l=1}^{5} n_{l-1}\ s_l^2\ n_l\ m_l^2\ +\ \sum_{i=1}^{2} 2\ W_i]\ x\ e) \tag{IV.9}$$

(B) **Computational Complexity of CAE with BDGRU-BDLSTM**
To determine the time complexity of the proposed method that integrates the CAE with BDGRU-BDLSTM, we need to compute the time complexity of convolutional layers, the BDLSTM layers, as well as BDGRU layers. Therefore, its overall time complexity is estimated as Eq.IV.10, as a function of the input length for all the training process.

$$O([(\sum_{l=1}^{10} n_{l-1}\ s_l^2\ n_l\ m_l^2\ )x\ e]\ +[(\sum_{i=1}^{2} 2\ W_i\ +\ \sum_{i=1}^{2} 2\ W_i')\ x'\ e']) \tag{IV.10}$$

We can conclude that the computation time of CNN-BDGRU model is less than the second proposed model (CAE with BDGRU-BDLSTM) in terms of time complexity.

IV.4.4.3 Compared with other approaches

Various prognostic popular methods are performed for comparison purposes, including DNN, RNN, LSTM, GRU, and CNN (as shown in Figure IV.12). We tried different structures for these methods, and we picked the best ones as follows:

1) **DNN :** contains two hidden layers, which have 50 neurons in each hidden layer and, ultimately, one neuron is attached for RUL estimation.

2) **RNN :** consists of two recurrent layers with hidden units of 50 nodes. Dropout is employed with a rate of 0.25 in each RNN layer.

3) **LSTM :** is implemented with a similar configuration to the RNN method to extract the long-term dependencies.

4) **GRU :** We use the same configuration of LSTM for the GRU structure for comparison purposes.

5) **CNN :** Five convolution layers are stacked with the same configuration of the CNN path of the proposed CNN-BDGRU method. At the end of this method, one neuron is attached for RUL estimation.

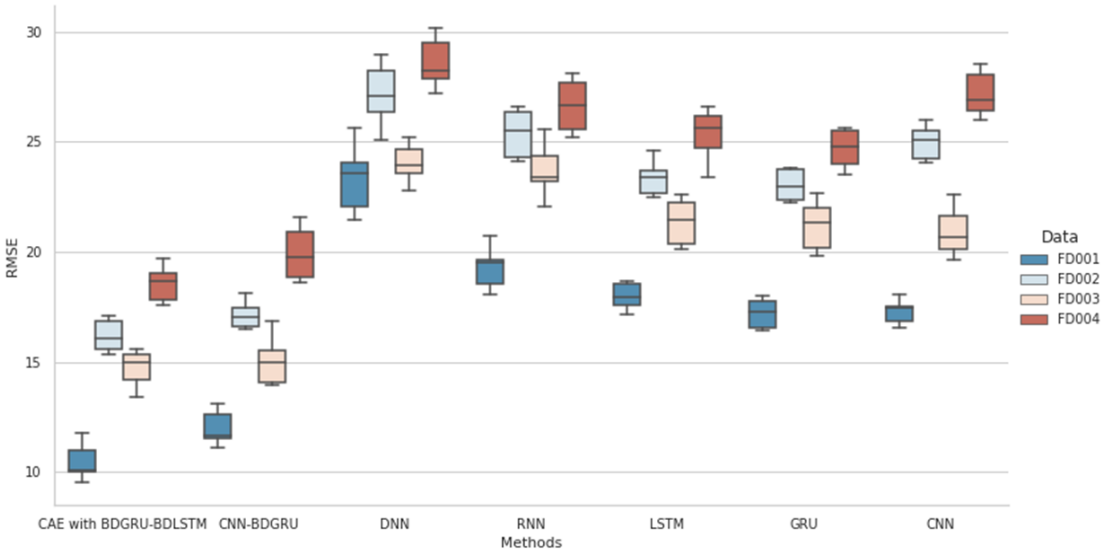


FIGURE IV.12: Box plot of the RMSE for both proposed hybrid models compared to the other architectures on the NASA turbofan engines datasets.

The average performance of DNN, RNN, LSTM, GRU and CNN on each C-MAPSS sub-datasets have been reported in Figure IV.12 as a RMSE box plot. Among all methods, DNN and RNN performed worse (higher RMSE) than the remaining

methods on all four sub-datasets. CNN achieved slightly lower RMSE values than other comparing methods on single operating condition datasets, i.e. FD001 and FD003. On the other hand, GRU and LSTM achieved a lower RMSE than other comparing methods on multiple operation condition datasets, i.e. FD002 and FD004. These results demonstrate powerful of our proposed models, which achieve observable lower average RMSE values (better) in all subsets than other architectures. Besides, the obtained performance from FD002 and FD004 is slightly lower RMSE prediction accuracy, and the reason is that these sub-datasets are more complicated than the FD001 and FD003.

To analyze the results in more detail and to demonstrate the powerful of the proposed CAE as an advanced feature extractor method, the three different features are introduced for comparison purposes. The first kind of features is only raw data with normalization, the second features constructed from the PCA method, and the last features created from the proposed CAE method.

For PCA, the principal components explaining 99% of the data variance were chosen as most appropriate in this study; the original features are reduced to 15 principal components. Considering that $X_p \subset X_m$, where m is the number of original features, and p is the number of principal components, with p<m. The curve of the cumulative sum of variance with the principal components for FD003 using the PCA method is shown in Figure IV.13.

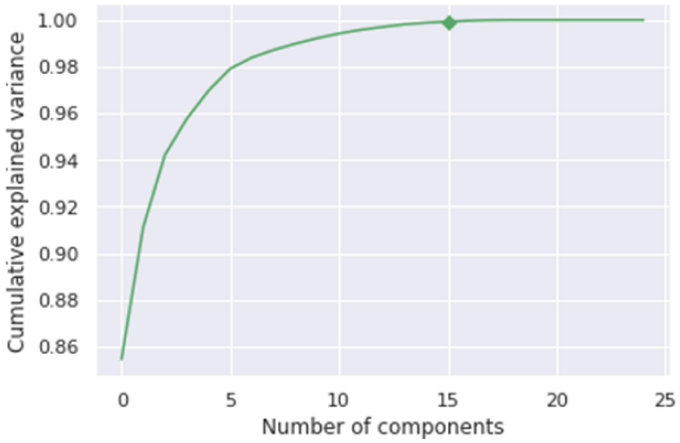


FIGURE IV.13: The curve of the cumulative sum of variance with N° of principal components of the C-MAPSS sub-dataset FD003.

Figure IV.14 shows the distribution box plots of the RMSE testing of multi-model BDGRU-BDLSTM with different features. The proposed BDGRU-BDLSTM multi-model is based on the combination of BDGRU and BDLSTM models simultaneously and in a parallel manner to predict the RUL. "None" indicates that the normalized raw data were used as input. Overall, when trained BDGRU-BDLSTM model on the normalized raw data showed the worst performance over the C-MAPSS datasets. Interestingly, the performance of BDGRU-BDLSTM improved with feature extraction methods. Among the three feature extraction methods, the CAE method can learn robust features than the remaining methods, which gives the best and minimum RMSE values with BDGRU-BDLSTM layers in all sub-datasets.

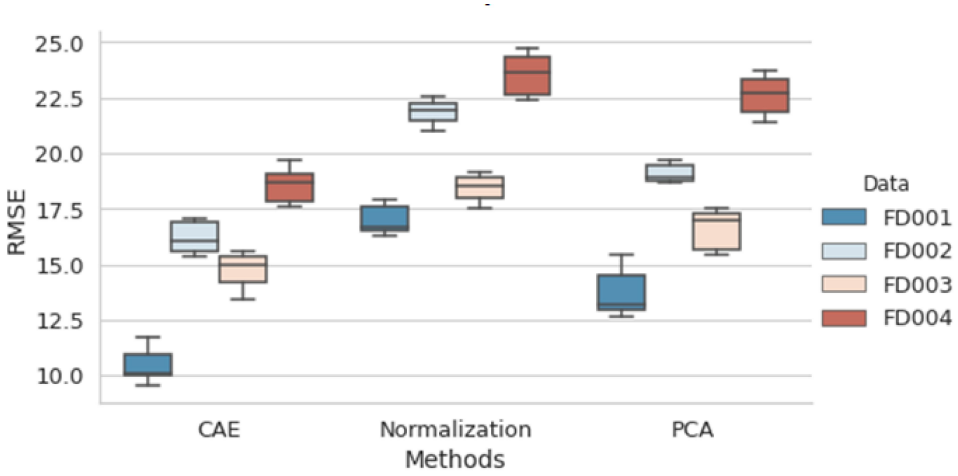


FIGURE IV.14: Box plot of the RMSE for BDGRU-BDLSTM method with different features on C-MAPSS dataset.

IV.4.4.4 Effect analysis

To demonstrate the effectiveness of multiple-model DL techniques, we present the effect of combining two DL methods CNN and BDGRU in sequential versus parallel, shown in Figure IV.15. The comparison result is quantified using RMSE of RUL prediction, and we can notice that the combination of CNN-BDGRU in parallel pathways achieved promising results compared to CNN-BDGRU in a sequential way.

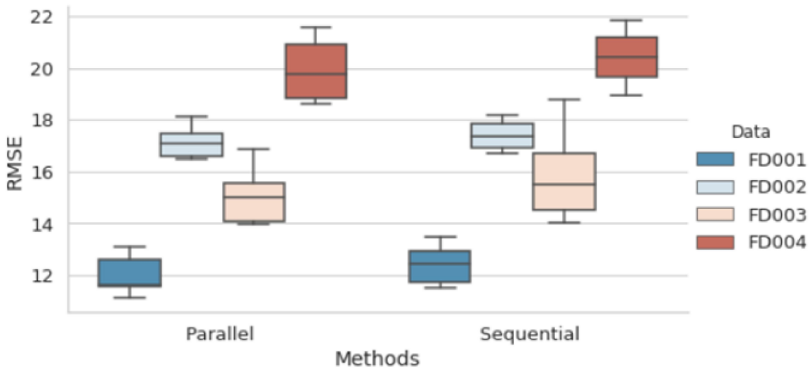


FIGURE IV.15: Box plot of the RMSE for CNN-BDGRU in sequential versus parallel.

To verify the validity of the CAE with BDGRU-BDLSTM structure, three experiments are conducted for comparison purposes. In the experiments, we merge CAE once with BDGRU and once with multi-models BDGRU-BDLSTM. According to Figure IV.16, we can observe that the combination of CAE with the multi-model BDGRU-BDLSTM has achieved good results on all C-MAPSS sub-datasets.

IV.4.4.5 Comparison with the latest works

Many scholar research has been reported on all sub-datasets C-MAPSS and used in more than 60 publications. Recent studies on the C-MAPSS dataset have been taken into account for comparison to show powerful of the proposed models. Table IV.2

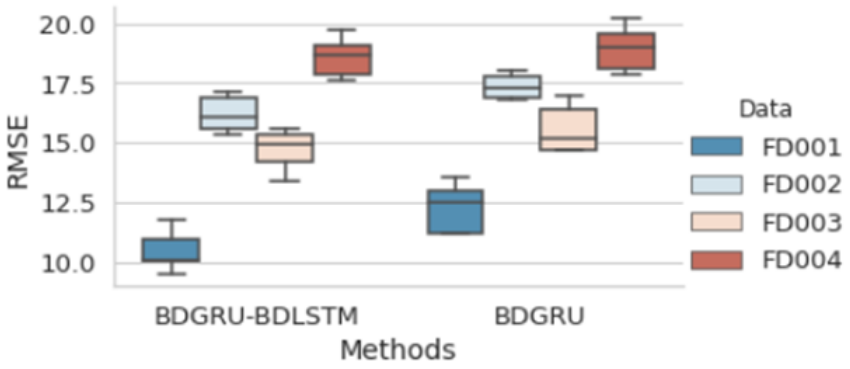


FIGURE IV.16: Box plot of the RMSE for CNN-BDGRU in sequential versus parallel.

recapitulates the results of recent studies of the advanced DL methods on the RUL estimation problem extended to all C-MAPSS sub-datasets.

TABLE IV.2: Performance comparison with the recent DL methods for RUL estimation on the C-MAPSS Dataset.

		CAE with BDGRU-BDLSTM	**CNN-BDGRU**	**CNN [7]**	**LSTM [200]**	**CNN [99]**	**BDLSTM [172]**	**BDLSTM [195]**	**AE-BDLSTM [158]**	**LSTM [181]**
FD001	RMSE	**9.51**	**11.085**	18.44	16.14	12.61	13.65	15.42	13.63	18.33
	Score	**213**	**245**	1.2867×10^3	3.38×10^2	273.7	2.95×10^2	/	2.61×10^2	655
FD002	RMSE	**15.35**	**16.476**	30.29	24.49	22.36	23.18	/	/	/
	Score	**1274**	**1198.42**	1.3570×10^4	4.45×10^3	10412	4.13×10^3	/	/	/
FD003	RMSE	13.41	13.964	19.81	16.18	**12.64**	**13.74**	/	/	19.78
	Score	350	387	1.5962×10^3	8.52×10^2	**284.1**	**317**	/	/	828
FD004	RMSE	**17.57**	**18.60**	29.15	28.17	23.31	24.86	/	/	/
	Score	**1528.18**	**1592**	7.8864×10^3	5.55×10^3	12466	5.43×10^3	/	/	/

Table IV.2 shows that the proposed hybrid methods have achieved promising results compared to the recent studies on all C-MAPSS sub-datasets quantified using the RMSE and score metrics. Especially for the complicated datasets FD002 and FD004, the Score and RMSE prediction accuracy obtained from both methods higher than the existing methods. Except on the sub-dataset FD003, the DCNN method [99], and Deep Bidirectional LSTM [172] are slightly higher Score and RMSE (worse) than both our proposed hybrid methods. However, our proposed methods used all three sensor settings and all sensor measurements as input without manually designing features, unlike works [105], [99], and [172] that picked 14 sensors data and excluded the three operational settings. Amongst these recent studies, it is worth mentioning that our proposed method that used CAE is the first attempt to adopt CAE in aero-engine prognostic problem in order to extract useful features that serve as inputs for the two separate and parallel pathways (referred to as BDLSTM path and BDGRU path) to obtain more robust features. Furthermore, the reason why our methods are proposed is exhibited superior performance among the most existing methods and for capitalizing on the recent success of multiple-model deep learning techniques and aligning with the power and the success of Convolutional Auto-Encoders to extract automatically useful features with high-level abstractions from complex data.

We want to point out that the CAE with DBGRU-BDLSTM results are better than our CNN-BDGRU results about 14.208%, 6.83%, 3.967%, and 5.537% in terms of the RMSE values, for FD001, FD002, FD003, and FD004, respectively. However, the time cost of the CAE with BDGRU-BDLSTM is higher than the CNN-BDGRU according to the time complexity metric.

IV.5 Conclusion

This chapter presented our first contribution that consist of developing an efficient RUL prediction approach by testing several deep learning techniques. The proposed deep end-to-end architectures could be considered as a first step to optimise the feature reduction and subsequently RUL prediction as second step, in a hierarchical way, with the aim of achieving data representation in low dimensionality and minimal variable redundancy while preserving critical asset information with minimal preprocessing effort. Besides, the idea of hybrid methods emerged to leverage the power of various methods, and ultimately enhancing and obtaining a more accurate prediction. Firstly, we proposed a CAE with a temporal modeling tool that combines BDGRU and BDLSTM models in a parallel manner for degradation features extraction and RUL prediction. We found that the CAE is more suited for data extraction and reduction rather than conventional approaches. Secondly, a hybrid architecture consisting concurrently of CNN and BDGRU models is developed and applied to capture local and temporal features for the RUL estimation.

In this work, it was clearly observed that the obtained results show significant improvements in the RUL prediction compared with previous similar works. In the following chapter, we focus on predict an interval of time that covers all the distribution rather than providing only a single deterministic predicted value of the RUL that has a zero probability of being accurate. Besides, we seek to solve such issues related to data quality as well as model selection task in terms of hyperparameter and architecture selection. Furthermore, further research is required to speed up training time.

Chapter V

RUL Prediction using a fusion of Attention-based Convolutional Variational AutoEncoder and Ensemble Learning Classifier

V.1 Introduction

Most studies have treated the prognostics as a regression problem, providing only a single deterministic predicted value of the RUL (estimating the exact time of failure), which has a zero probability of being accurate. Whereas predicting an interval of time that covers all the distribution is more certain (i.e., has a 100% probability of occurrence) [21]. Therefore, this chapter introduces an innovative approach based on visual data analysis that provides system failure probabilities in different time windows. In constructing this new RUL estimation approach, we faced several challenges related to data quality and model selection, which need to be developed and improved.

The data quality challenges include: How can adequate data quality be achieved? What is the impact of data quality on RUL estimation performance? How to deal with imbalanced data? How do features that explicitly and properly reflect failure progression be obtained? There is no way to ensure that the relevant features selected are the ones that can lead to better RUL estimates. Do visualization techniques evaluate and analyze the quality of the features and provide a minimum level of transparency? How to discriminate degradation states and define threshold setting of states (classes)? Does it take into consideration the opinion of the expert? **The model selection challenges include:** How is a suitable architecture picked? How to improve the prognostic model to accurately and reliably predict the RUL? Is the performance of the prognostic method strongly influenced by the hyper-parameter selection? How is the optimum value for hyperparameters selected? Does reducing the conflict zone (i.e., the boundary between classes) lead to fewer conflicting classification decisions? How can the conflict zone be identified and reduced?

The major contribution of this work is a new methodology that addresses the aforementioned challenges and which can be summarized as follows:

- Leveraging the power of the VAE architecture in the disentanglement of latent space.
- Attention Convolutional Variational Autoencode (ACVAE) integrates convolutional calculation with autoencoder to effectively extract spatial features.
- Attention mechanism is embedded to make the network pay attention to the useful features.
- ACVAE is more aimed at improving dimensionality reduction capability and achieving better spatial distribution and overall visualization.
- The approach aims to predict the probability that the machine will fail within different time windows (three degradation classes).
- The task of defining the three degradation classes takes into consideration the opinion of an expert.
- Ensemble learning is applied using the voting classifier to predict the class labels by averaging the class probabilities in order to reduce the conflict zone.

- Automatic Hyperparameters Selection (AHPS) is used to pick out the best configuration of hyperparameters for our hybrid architecture.

This chapter presents the main contribution, which aims to develop an innovative RUL estimation strategy that simultaneously assesses the health status of degrading machinery (provides the probabilities of system failure in different time windows) and assigns them to the RUL window estimation. The remainder of this chapter is organized as follows. An overview of the related work as well as of the research gap, is given in section V.2. Section V.3 provides the problem formulation and the description of the proposed RUL estimation method. In section V.4, the analysis of the results is given. Finally, we close the chapter with a conclusion and future works in section V.5.

V.2 Related work

This section analyzes recent research by focusing on four main challenges: architecture selection, dimensionality reduction and visual explanation techniques, attention mechanism, and model hyperparameter optimization that can improve prognostic performance.

V.2.1 Data-driven methods for RUL estimation

The industrial dataset is possible to gather due to industrial IoT, which has promoted opportunities for industry and academia to leverage advanced data-driven techniques. Indeed, NASA's C-MAPSS turbofan time-to-failure data set has been extensively analyzed with RUL estimation as a primary focus. Many previous data-driven methods for machine status monitoring and RUL estimation, including two main methods, conventional machine learning and deep learning, have been applied and seen great success recently [86, 129]. Some studies [74, 75, 89, 143, 166] have used the promising Artificial Neural Network (ANN) approach to predict the RUL of various machines, such as bearings, milling cutters, engines, and drill pipes. In order to keep the optimum set of features, [88] applied Principal Component Analysis (PCA) as the first phase, with ANN as the second phase, to predict the RUL of the roller ball bearings. In [128], the authors also proposed a method to predict RUL based on neuro-fuzzy. However, this conventional ML cannot address sequential data. Besides, the biggest limitation of the PCA approach lies in its linear projection.

Recently, burgeoning DL approaches have been widely applied in various research for prognostic and diagnostic tasks, known for their ability to process highly non-linear and varied data in their raw form without any human intervention. Within the deep learning architecture, recurrent neural networks can mainly handle temporal data analysis, which prompted researchers to applied it to the industrial PHM process. Some researchers [64, 108, 109] have proposed RNN-based methods for the prognostic issue. However, RNNs has a vanishing gradient or exploding problem arising in long sequence input, which means it cannot keep the previous information except the latest one. In order to handle this issue, Long-Short Term Memory is

the upgraded variant of RNN in which different gating mechanisms are proposed. [188] proposed an LSTM to determine the fault location and estimate the RUL of the aero engine. More recently, several works [6, 73, 119, 162, 181, 200] have suggested LSTM-based approaches for RUL estimation, showing the efficacy of performing LSTM over RNN. As an improvement, another variant of LSTM was used by [172] is Bi-directional LSTM (BLSTM), which can learn the bi-directional temporal dependencies from sensor data for aircraft engine RUL estimation. Therefore, it can simultaneously capture long-range information in the input sequence's future (forward) and past (backward) contexts. Moreover, a new BLSTM model was presented by [195] for identifying the system degradation performance and subsequently predicting RUL. However, recurrent networks increase computational burdens.

Although convolutional neural network are one of the most dominant methods for image processing [14, 101, 130, 142, 163], CNNs have also been explored for RUL prediction by [7, 99] on the multi-channel time series. CNN architectures are designed to extract features through weight-sharing filters and show a noticeable improvement in prediction accuracy. Hybrid deep neural network models have also been reported in the literature [2, 97, 131] to leverage the power of different DL methods, which integrate CNN and LSTM models simultaneously to extract temporal and spatial features.

V.2.2 Dimensionality Reduction and Visual Explanation techniques

Massive and large-dimensional data often contain uninformative or redundant features, making data analysis difficult and increasing the processing time. Besides, reduced data visualisation is crucial to understanding better how data is distributed and interpreting and analysing classifier performance. Some classical data visualisation and dimensionality reduction methods such as PCA, Isometric Mapping (ISOMAP), and T-distributed Stochastic Neighbor Embedding (T-SNE) have been used and reported in academic research [3, 106, 124]. Additionally, several studies have shown the advantage of using Auto-Encoder (AE) to decrease the data dimension and automatically extract the performance degradation features from multiple sensors, which is suitable for enhancing predictive accuracy and reducing the model's complexity. In [158], the authors proposed a new hybrid model integrating the advantages of AE and BLSTM to enhance the RUL's prediction accuracy. A similar study was suggested by [105] for RUL estimation based on a stacked Sparse Auto-Encoder (SAE) and Logistic Regression (LR).

Most generative applications deal with image processing, as in [72], where a VAE was also trained to generate face images with much clearer and more natural noses, eyes, teeth, and hair textures, as well as good backgrounds. The ML algorithms' black-box nature imposes the industry's unwillingness to adopt them. More recently, in nonlinear process monitoring, the deep Variational Autoencoder method has been successfully applied to address both the curse of dimensionality and the scarcity of interpretability and transparency by projecting the high-dimensional process data into a lower-dimensional space ([30, 35, 95, 110, 111, 140, 173, 176, 192–194,

197, 198]). This latter is argued upon and supported by [52]; the ability to map a lower-dimensional space could increase a model's generalization capabilities. In [192–194], the authors propose a VAE architecture as a 2D-Visualisation tool of latent space to understand how data is distributed. This latter can help to get a better idea of how the model interprets the data.

Attention Mechanisms (AM) have recently been widely applied in neural network architectures. It has been proved successful in natural language processing for machine translation tasks ([8, 104]), fault detection [102], RUL estimation tasks [34, 38, 164], and various computer vision tasks such as facial expression recognition [98] and fruit classification [183]. The attention mechanism can make the neural network allocate more attention to useful features. [38] used a soft attention mechanism to provide visualisation of the learned attention weights at each RUL prediction step to gain its interpretability besides retaining the predictive power of LSTM networks. [102] employed a convolutional autoencoder with AM to enhance the local features of samples.

V.2.3 Model's Hyperparameters optimization

Deep learning models are full of hyperparameters in terms of architecture and training parameters (such as the number or type of layers and the learning rate). Their optimization by most of the reviewed papers is based on a trial-and-error approach [2, 6, 7, 73, 97, 99, 131, 172, 188, 194, 195, 200]. However, this approach can be time-consuming and error-prone due to a lack of understanding of the impacts of parameters. In order to overcome these challenges, automatic hyperparameter selection has been proposed, such as grid search [17], random search [16], and Bayesian optimization [152, 156]. They typically carry out the research by discretizing the hyperparameter space. The grid search is the most applied strategy that tests all possible combinations (exhaustive searching) [38, 105, 158, 181]. Although this approach can theoretically obtain the optimal global parameters, it is extremely computationally expensive and suffers from the curse of dimensionality. The reason is that the number of combinations grows exponentially with the number of hyperparameters. Compared with grid search, random search eliminates the need for an exhaustive search of all possible combinations by picking them randomly. Bayesian optimization is an efficient hyperparameter tuning method for complex DNN methods ([18, 80, 180]). Its principle is to pick parameter combinations in a well-thought-out way based on a probabilistic model. This probabilistic model uses previous evaluations to obtain the posterior predictive distribution using the Bayesian formula. Therefore, we aim to apply Bayesian Optimization (BO) based on Gaussian Process (GP) to reduce the time spent on hyperparameter tuning, which disregards certain areas of the parameter space that are unlikely to yield the best results.

V.2.4 Research Gaps and Contribution

As summarised in Table V.1, many contributions are proposed for the RUL prediction of turbofan engines using various DL architectures. Nevertheless, applying DL

TABLE V.1: Previous work comparison.

Works	Data analysis Task	Architecture	Attention Mechanism	Dimensionality Reduction	Visual Explanation	AHPS	Stacking Ensemble ML
[7]	Regression	CNN	✗	✓	✗	Trial-and-error	✗
[188]	Regression	LSTM	✗	✗	✗	Trial-and-error	✗
[200]	Regression	Deep LSTM	✗	✗	✗	Trial-and-error	✗
[6]	Classification	LSTM	✗	✗	✗	Trial-and-error	✗
[73]	Regression	LSTM	✗	✗	✗	Trial-and-error	✗
[172]	Regression	BLSTM	✗	✗	✗	Trial-and-error	✗
[195]	Regression	BLSTM	✗	✗	✗	Trial-and-error	✗
[99]	Regression	CNN + FFNN	✗	✗	✗	Trial-and-error	✗
[158]	Regression	AE + BLSTM + FFNN	✗	✓	✗	Grid search	✗
[105]	Regression	SAE + LR	✗	✓	✗	Grid search	✗
[97]	Regression	LSTM + CNN	✗	✓	✗	Trial-and-error	✗
[2]	Regression	LSTM + CNN + FFNN	✗	✓	✗	Trial-and-error	✗
[38]	Regression	LSTM + FFNN	✓	✗	✓	Grid search	✗
[119]	Classification	LSTM	✗	✗	✗	Trial-and-error	✗
[181]	Regression	DLSTM	✗	✗	✗	Grid search	✗
[194]	Classification	CVAE + MLP	✗	✓	✓	Trial-and-error	✗
Our proposed approach	Classification	CVAE + Stacking classifiers	✓	✓	✓	Bayesian optimization	✓

(✓): verified **(✗)**: not verified
(Regression): predict a single deterministic value of the RUL **(Classification)**: predict an interval of time

techniques in the context of prognostics is still challenging, in addition to the fact that RUL prediction is frequently affected by uncertainty in a practical context and may cause problems. Inspired by these previous studies, this work focuses on six main points: architecture selection, Dimensionality Reduction, Visual Explanation techniques, Attention Mechanism, Model's Hyperparameters optimisation, and ensemble learning method that can improve prognostics performance. By analysing Table V.1, a novel approach for predicting RUL based on visual data analysis is proposed. Attention Convolutional Variational AutoEncoder is used to extract performance degradation features from multiple sensors automatically. ACVAE effectively integrates convolution calculation with an autoencoder to extract spatial information.

Moreover, the attention layer is embedded between the encoder and decoder, which is used to dynamically increase the weights of the useful features in the encoding phase to make the network pay attention to these vital features for RUL class estimation. The primary objective of applying the ACVAE is to provide a more structured, disentanglement and lower-dimensional representation of the data that shows the best class distribution over a 2D latent space and demonstrates how well the ACVAE generalises. The encoder, part of the ACVAE, is leveraged for data projection in a 2D visualisation latent space. The input vectors are encoded and displayed in this 2D space, which helps the expert visually analyse the spatial distribution of the training dataset. Three degradation classes are then defined according to two thresholds (α_1, α_2). The expert aims to determine the appropriate threshold setting by minimising the overlapping region between the degradation classes by analysing the spatial distribution. Following that, the RUL is predicted according to the latter degradation classes.

V.3 Methodology

V.3.1 Problem Formulation

To address the uncertainty in RUL, we propose to use $\varphi(.)$ for dimensionality reduction capabilities, which provides a better latent space distribution Z. Let X^i denote its input (formulated in Chapter II, Section II.5.1), sequential sensor measurement. Z^i is a latent representation generated by the encoder function $Z^i = f_\varphi(X^i)$, whereas $\widehat{X}^i$ is an approximation or reconstruction of the real data X^i (see Eq. (V.1)).

$$Z^i, \widehat{X}^i = \varphi(X^i) \tag{V.1}$$

The error between the X^i and reconstruction $\widehat{X}^i$ is minimized as follows:

$$Minimize : \{\widehat{X}^i, X^i\} \tag{V.2}$$

Thus, the RUL values are divided into three RUL degradation classes Y_{class} according to two thresholds (α_1, α_2) that are defined based on the spatial distribution analyse with the expert.

In order to address the non-linearity function, the ensemble ML method is proposed (θ) for RUL class estimation. Let Z^i symbolise its input, and the observed Y^i_{class} symbolise its output (formulated in Chapter II, Section II.5.1). The predicted RUL classes are given as follows.

$$\widehat{Y}^i_{class} = \theta(Z^i, Y^i_{class}) \tag{V.3}$$

The prediction accuracy between the predicted RUL classes $\widehat{Y}^i_{class}$ and the observed Y_{class} is maximized as follows:

$$Maximize : \{\widehat{Y}^i_{class}, Y^i_{class}\} \tag{V.4}$$

V.3.2 Remaining useful life estimation based on CVAE with attention mechanism

The RUL class estimation methodology is shown in Figure V.1. The VAE approach can generate new data (through continuity) as well as it has been demonstrated as a promising tool for dimensionality reduction in the context of machinery fault diagnosis [126, 140]. However, VAE must be fully explored for fault diagnosis and prognosis. In this work, we propose ACVAE which is a VAE based architecture for predicting RUL classes. It is composed of an encoder and a decoder, which are two symmetrical and reversed structures. Both the encoder and the decoder have two convolutional layers. We utilized the same padding and a 6×1 kernel for convolutional layers in the encoder. The stride was 1×1 for the first convolutional layer and 2×2 for the second. The convolutional layer's output is expressed as follows:

$$C_i = f(\sum X \odot w_i + b_i) \tag{V.5}$$

Where f is the activation function, $\odot$ is the convolution operation, w_i and b_i represent the weight parameter and bias of i^{th} convolutional kernel, respectively. The final convolution will generate an output $H^c = \{h_1, h_2, ..., h_{\frac{W_L}{2}}\}$ where $H^c \in R^{\frac{m \times W_L}{2}}$. Given input data of sequence length W_L with m number of features (sensor variables).

Moreover, the attention layer is also embedded in the encoding part, which is used to dynamically increase the weights of the useful features to make the network pay attention to these vital features [104, 185]. The attention computational is given as follows:
The attention weights:

$$\alpha_i = softmax(W_a \cdot h_i) \tag{V.6}$$

The context vector:

$$c_i = \sum_i \alpha_i \cdot h_i \tag{V.7}$$

The attention vector:

$$a_i = tanh(W_c[c_i; h_i]) \tag{V.8}$$

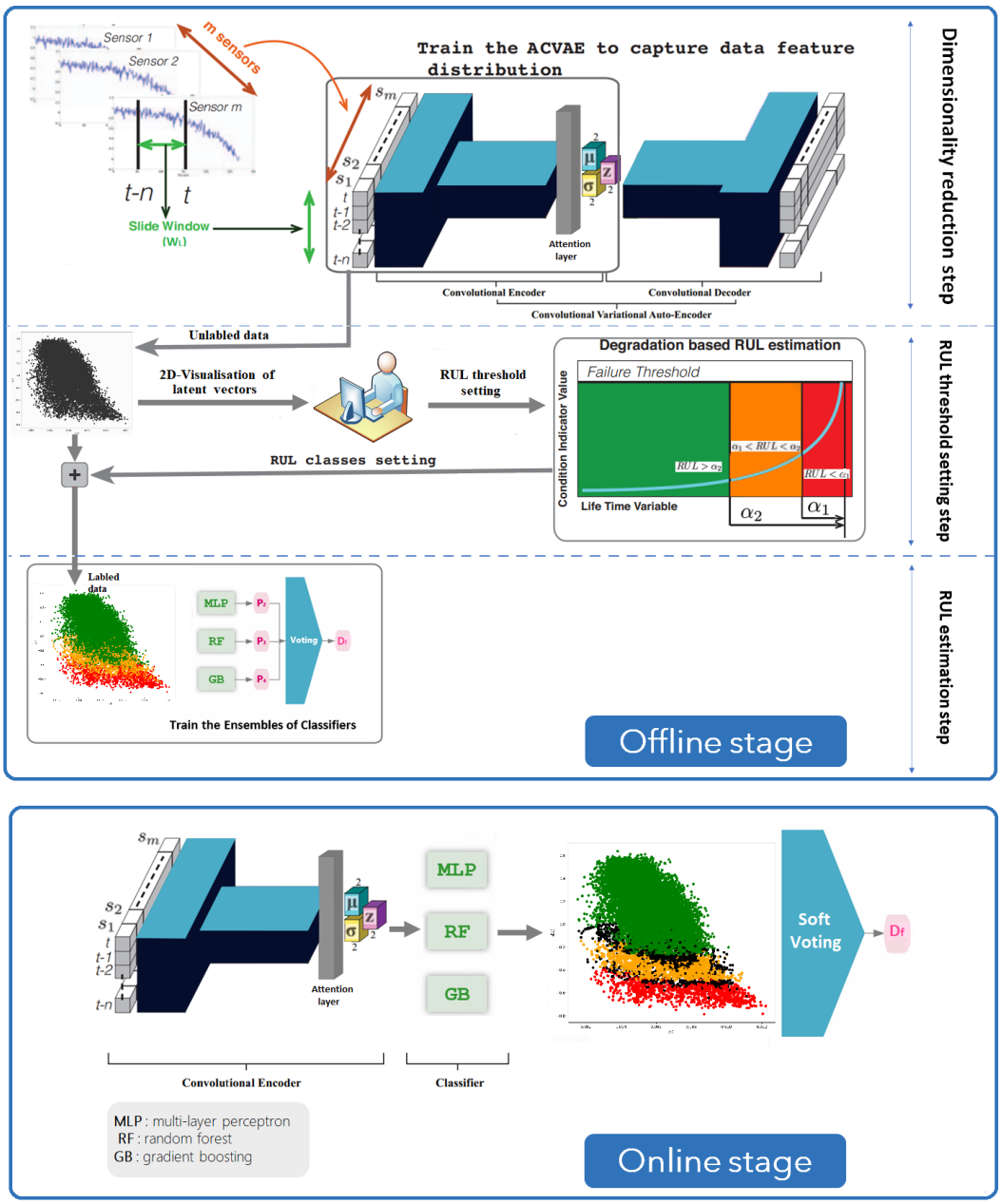


FIGURE V.1: A framework of RUL estimation using the soft voting classifier with MLP (Multi-layer perceptron), RF (Random forest), and GB (Gradient Boosting). D_f is the final class label.

Where h_i is the features extracted by the encoder.

The latent two-dimensional space is represented by two 2D-layers for the encoder, the mean and the standard deviation layers (i.e., μ and σ). One 2D-sampling layer (**Z**) for the decoder.

The first step was to train the whole ACVAE architecture for reconstructing the input vector using the deconvolutional operation (decoder part), as shown in Eq. (V.9).

$$D_i = f(\sum Z \otimes \bar{w}_i + \bar{b}_i) \tag{V.9}$$

$\otimes$ is the deconvolution operation, $\bar{w}_i$ and $\bar{b}_i$ represent the weight parameter and bias of i^{th} deconvolutional kernel, respectively.

Training the ACVAE does not need the label information of the input data. The whole ACVAE is trained to attain a coupling optimization of both the reconstruction and disentanglement quality (See Algorithm 1). The training loss of ACVAE is defined as the sum of the reconstruction loss and the similarity loss (See Eq. (III.24)). The used reconstruction loss is the Mean Squared Error (MSE), which measures how close the decoder output is to the original input, as expressed in Eq. (V.10).

$$L_{rec} = 1/N \sum_{i=1}^{N} (X_i - X_i')^2 \tag{V.10}$$

The similarity loss is the KL divergence between the latent space distribution and the standard Gaussian (zero mean and unit variance), which regularizes the distribution of the latent space (given in Eq. (V.11)).

$$L_{KL} = -1/2 \sum_{i=1}^{N} \sum_{j=1}^{k} (1 + log(\sigma_{ij}^2) - \sigma_{ij}^2 - \mu_{ij}^2) \tag{V.11}$$

Where N is the number of samples, X_i denotes the real data, X_i' is its reconstruction, and k is the size of the latent vectors.

Algorithm 1 ACVAE training algorithm.

Input: Sliding window training data $X_{train} = \{(x^i)\},\ i = 1, 2, 3, ..., N$
Output: Probabilistic encoder f_φ, Probabilistic decoder g_θ

$\varphi, \theta \leftarrow$ Initialize parameters randomly
repeat
  for i=1 to N **do**
    L samples from $\epsilon \sim N(0, 1)$
    $z^{(i,l)} = f_\varphi(\epsilon^i, x^i)$ ▷ *Compute z_i via reparametrization trick*
  end for
  $E = 1/_L \sum_{l=1}^{L} (\log\, p_\theta(x^i|z^{(i,l)})) - \sum_{i=1}^{N} -KL(q_\varphi(z|x^i)||\, p_\theta(z))$ ▷ *Compute the loss*
  $\varphi, \theta \leftarrow$ Update parameters using gradients of E
until convergence of parameters φ, θ

In this work, we aim to predict the probability that the machine will fail within different time windows. To do this, the expert faced a challenging problem: how to select the most appropriate thresholds α_1 and α_2 that are used to label the data (three degradation classes)? Three degradation classes are then defined according to two thresholds (α_1, α_2) as follows:

- Degradation class 0 (Deg 0): $RUL > \alpha_2$,
- Degradation class 1 (Deg 1): $\alpha_1 < RUL \leq \alpha_2$,
- Degradation class 2 (Deg 2): $RUL \leq \alpha_1$.

When the training process of the ACVAE is successfully done, the encoder part is then used as a 2D-Visualisation tool by a human expert in order to analyse the spatial distribution of the data set that has been separated into three distinct degradation

classes. The expert has made assumptions about picking out the values of thresholds α_1 and α_2, experimenting with different thresholds values ($\alpha_1 = \{10, 20\}, \alpha_2 = \{20, 30, 70, 90\}$). Each couple ($\alpha_1$, α_2) will generate a particular overlapping situation between the degradation classes, which is easily visualised and examined by the expert in the 2D-latent space. By analysing the spatial distribution of different couples (α_1, α_2), the expert tries to choose the appropriate threshold setting that minimises the overlapping region between the degradation classes. Indeed, the appropriate thresholds give thinner conflict areas (the boundaries) between classes, thus fewer instances (Z_i points) within boundaries, which gives fewer conflicting decisions for the classification.

The second step is to train the ensemble of classifiers for the RUL class estimation. The encoder parameters obtained by the previous step are frozen during the classifier training step. Algorithm 2 represents the active learning classifier that is used to construct a high-performance classifier by starting learning with a small training set. Via the incremental learning process, the misclassified points (uncertainty points) from $X^{2D}_{validation}$ are actively added into the training set X^{2D}_{train} based on a threshold of probability (As shown in [94, 120, 151, 201]).

Finally, in the operating stage, the convolutional encoder is used jointly with soft voting-based ensemble classifiers to estimate the degradation of RUL classes (as shown in Figure V.1).

Algorithm 2 Train the classifier.

Input: Sliding window training data $X_{train}, Y_{train} = \{(x^i, y^i)\},\ i = 1, 2, 3, ..., N$
Output: Trained classifier β

Train the ACVAE on X_{train} ▷ Algorithm 1
$X^{2D}_{train} = f_\varphi(X_{train})$ ▷ Use the encoder to extract the 2D-latent space
$X^{2D}_{train}, X^{2D}_{validation}, Y_{train}, Y_{validation}$= split($X^{2D}_{train}, Y_{train}$)
Randomly Initialize β
for i=1 to Q **do** ▷ Number of query
    Train the classifier β on X^{2D}_{train}
    Validate the classifier with $X^{2D}_{validation}$
    Sort misclassified examples by error score $X^{2D}_{misclas}$
    $X^{2D}_{train} = X^{2D}_{train} \cup X^{2D}_{misclas}$ ▷ Add the misclassified examples to train
    $X^{2D}_{validation} = X^{2D}_{validation} - X^{2D}_{misclas}$ ▷ Remove misclassified from validation set
end for

V.4 Results analysis

For the purpose of evaluating multiple experimental results, the various performance measures are first highlighted in Section V.4.1, followed by the data preprocessing section V.4.2, including feature selection, data normalisation, sliding window, and data rebalancing. Finally, the details of the results and comparisons with several architectures to comprehensively examine the proposed model are discussed in Section V.4.3, including data preprocessing parameters analysis, visualisation of latent vectors and identification the conflict zone, and performance analysis. In this

work, we only used the failure set FD001 of the C-MAPSS dataset from the NASA repository to validate the proposed method.

V.4.1 Evaluation metrics

To fairly evaluate the proposed model's performance on the test dataset, a range of various performance metrics is adopted (Eq. (V.12) - (V.17)). These metrics involve accuracy, precision, sensitivity, specificity, F-score, G-mean, Receiver Operation Characteristic (ROC), and Area Under Curve (AUC) [57]. The below formulae metrics could be assessed with $|TP|$ the number of the true positive, $|TN|$ the number of the true negative, $|FP|$ the number of the false positive (i.e. a false alarm), and $|FN|$ the number of the false negative (i.e. a missed alarm).

- **Accuracy:** refers to the ratio of the total correct predictions. It is formulated as:

$$Accuracy = \frac{|TP| + |TN|}{|TP| + |FP| + |FN| + |TN|} \tag{V.12}$$

- **Precision:** expresses the ratio of correctly predicted positive instances. Formally, it can be expressed as:

$$Precision_{macro} = \frac{1}{n}\sum_{i=1}^{n}\frac{|TP_i|}{|TP_i| + |FP_i|} \tag{V.13}$$

- **Recall or Sensitivity:** measures how much a classifier can recognize positive instances correctly identified by the classifier. It is computed using the following equation:

$$Sensitivity_{macro} = \frac{1}{n}\sum_{i=1}^{n}\frac{|TP_i|}{|TP_i| + |FN_i|} \tag{V.14}$$

- **Specificity:** calculates how much a classifier can recognize negative instances correctly identified by the classifier. The equation gives it:

$$Specificity_{macro} = \frac{1}{n}\sum_{i=1}^{n}\frac{|TN_i|}{|TN_i| + |FP_i|} \tag{V.15}$$

- **F1-score:** to maximize both precision and recall, the F1-score metric is the harmonic mean of the precision and recall. This combination reaches its highest possible value at 1, indicating perfect precision and recall, and its lowest possible value at 0, if either the precision or the recall is zero. It can be formulated as:

$$F1 - score = 2 \times \frac{Precision \times Sensitivity}{Precision + Sensitivity} \tag{V.16}$$

- **G-Mean:** measures the trade-off between sensitivity (true positive rate) and specificity (true negative rate) by the following:

$$G - Mean = \sqrt{Sensitivity \times Specificity} \tag{V.17}$$

Only when both sensitivity and specificity are high can the G-mean attain its maximum, which indicates a better classifier.

- **ROC curve:** is a graphical plot showing the performance of a classification model at all classification thresholds. This curve plots two parameters: True Positive Rate and False Positive Rate.
- **AUC:** is the fraction of the total area under the ROC curve. AUC provides a single value for assessing the performance of the classifier and an examination of the classifier's stability and consistency.

V.4.2 Data preprocessing

Before applying the proposed model, it is essential to prepare heterogeneous data adequately. The specific steps are described as follows:

V.4.2.1 Feature selection

In the FD001 dataset, there are three operating indicators, and 21 different aircraft engine sensors plotted on a histogram to observe the variations throughout the whole lifecycle of engines (See Figure V.2). The sensors 1, 5, 6, 10, 16, 18, and 19 exhibit constant values throughout the engine, which cannot provide relevant degradation information to accomplish the task and only increases the training time of neural networks. Therefore, 14 of the 21 sensors were selected, with indices of 2, 3, 4, 7, 8, 9, 11, 12, 13, 14, 15, 17, 20, and 21. Besides, the three operational settings are removed because these datasets are exposed to a single operating condition. Besides that, the same features were confirmed using the prognosability measure, also called consistency or failure consistency (the equation is given in V.18). It assigns a numerical value that measures the variability of condition indicators at failure on a scale of zero to one. A high-ranking feature more accurately monitors the degradation process and is consequently more appropriate for training the RUL prediction model, as shown in Figure V.3.

$$\text{Prognosability} = \exp\left(-\frac{\text{std}_j\left(x_j\left(N_j\right)\right)}{\text{mean}_j\left|x_j(1) - x_j\left(N_j\right)\right|}\right), j = 1, \ldots, M \tag{V.18}$$

V.4.2.2 Data Normalization

The sensor measurements have a varied range of values. Therefore, many differentiation issue normalization methods guarantee the same range scale of all sensor measurements [157]. This work uses the Min-Max normalization given in Eq. (V.19) to map the selected features within the range of [0,1].

$$x'_i = \frac{x_i - \min(x_i)}{\max(x_i) - \min(x_i)}. \tag{V.19}$$

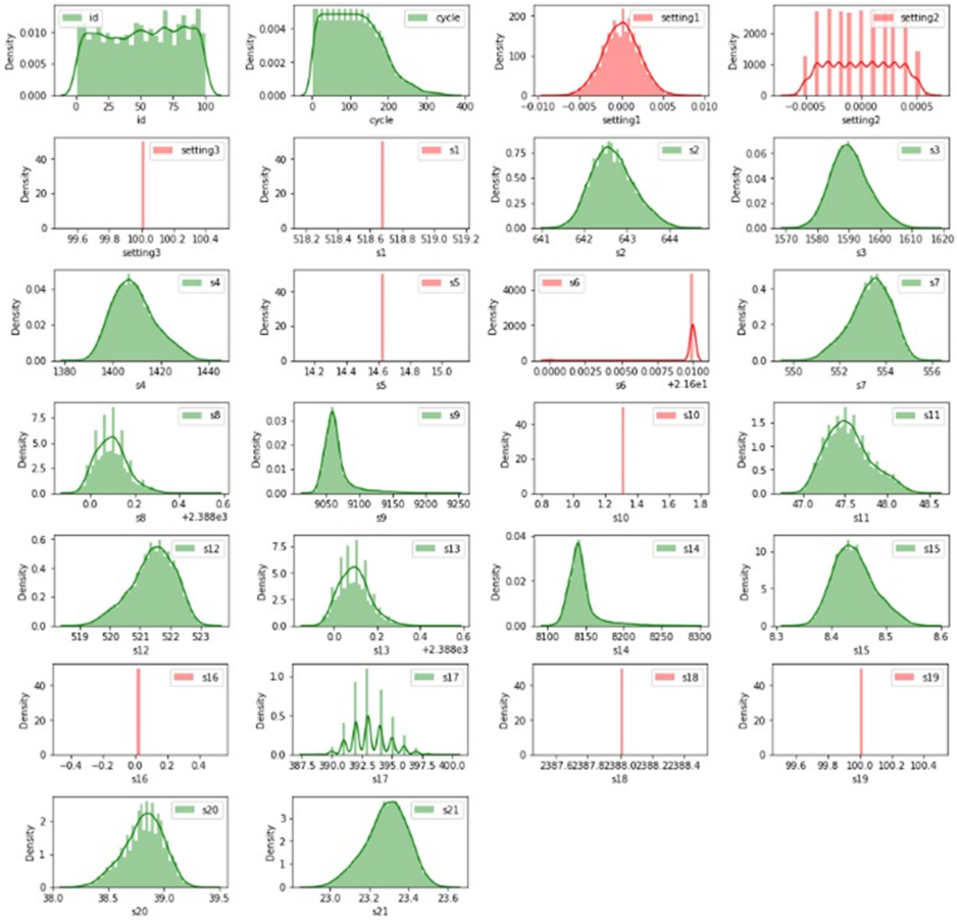


FIGURE V.2: FD001 Data Analysis.

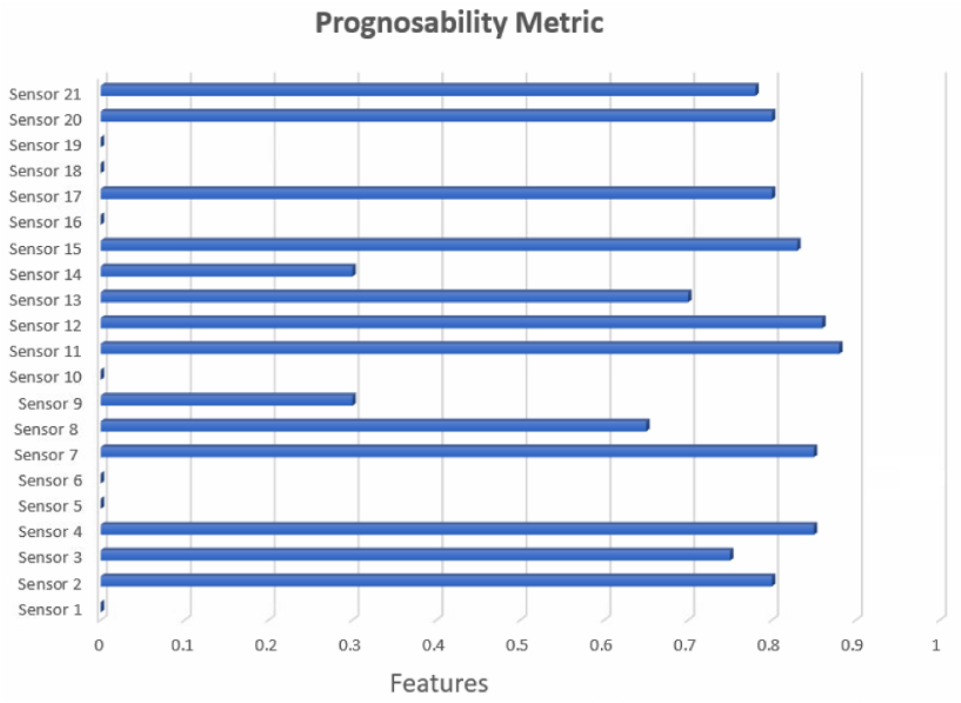


FIGURE V.3: Feature selection using the prognosability metric.

V.4.2.3 Sliding Window

The multivariate time-series sensor signals can provide more degradation information, which leads to an accurate prediction. Therefore, the Sliding Window (SW) method segments the data samples into a sliding time window along the engine life span (as shown in Figure V.4).

The extracted data by SW each time is a 2D matrix $W_L \times W_f$, W_L as the length of the sliding time window, and W_f as a number of the selected prognosis feature. Moreover, the SW is moved with only one data point. Thus, the number of sliding time windows generated from data is $\sum_{i=1}^{n} maxT_i - W_L$, where $maxT_i$ is the engine lifespan and n number of engines.

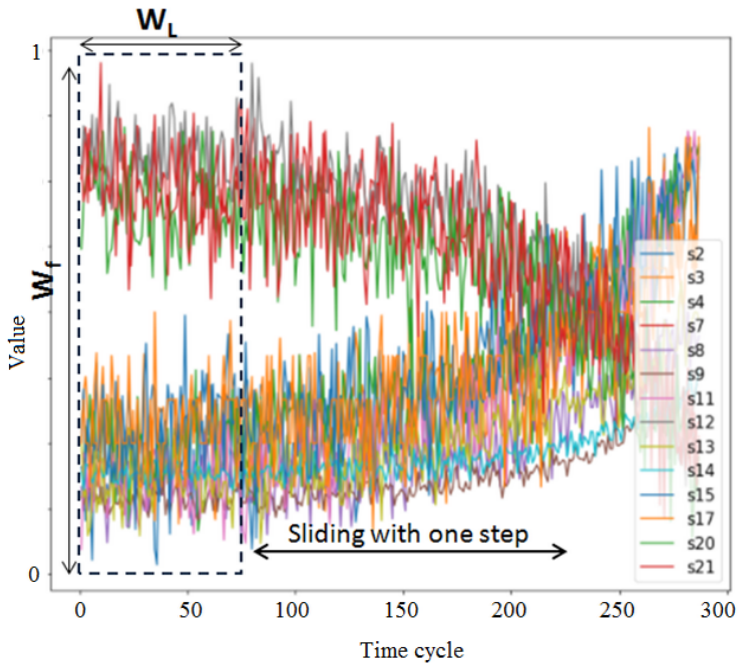


FIGURE V.4: One training sample using the sliding window method.

V.4.2.4 Data rebalancing

The problem of class imbalance is faced when one of the classes is underrepresented over others. It is challenging to train classifiers on imbalanced data, as they become biased towards a set of classes. A widely applied approach to dealing with imbalance is resampling, either using different algorithms from undersampling (removing some majority-class data points) or over-sampling (adding more minority-class data points). The under-sampling can cause the wastage of important information. On the other hand, the Synthetic Minority Oversampling TEchnique (SMOTE) is one of the dominant oversampling methods in literature [13, 44]. Consequently, in this work, the SMOTE-based K-Nearest Neighbors (KNN) method is implemented to handle the class imbalance problem.

Figure V.5 depicts the class distribution bar charts, where the blue bar refers to real data showing that the data used was highly imbalanced. The red bar indicates

synthetic instances in minority classes that follow the original distribution utilizing the SMOTE-KNN algorithm. Instead of excessively increasing the number of synthetic instances in the minority classes (equal to the majority class), we experimented with different oversampling thresholds and picked the best ones (see Figure V.5).

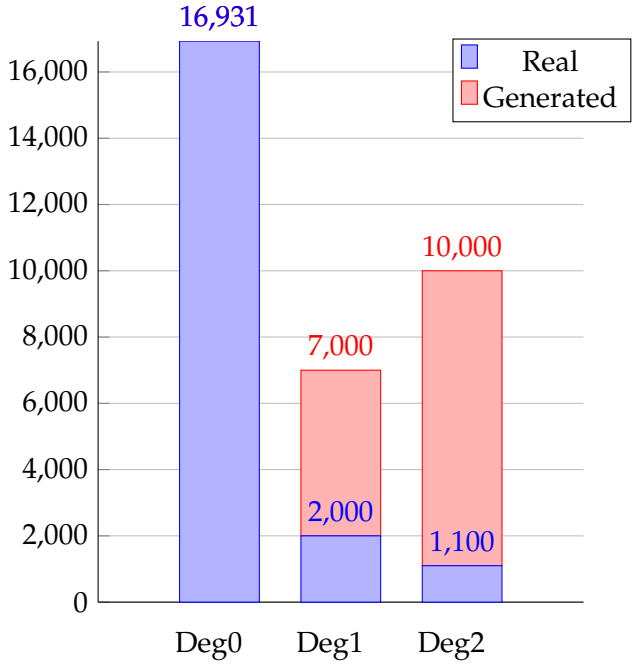


FIGURE V.5: Data rebalancing using a SMOTE-based KNN on training set FD001.

V.4.3 Results

V.4.3.1 Data Pre-processing Parameters Analysis

Several sliding window sizes (W_L) were tested and evaluated according to different degradation thresholds α_1 and α_2, which are the most sensitive pre-processing parameters. Table V.2 reflects the performance of our proposed approach obtained for each test through evaluation measures, with different W_L and thresholds (α_1, α_2). In our experiments, we considered values of W_L in $\{6, 16, 26\}$, taking into account the minimum engine life span available in the test dataset. The values of thresholds (α_1, α_2) are in {(10,20), (10,30), (10,70), (10,90), (20,30), (20,70), (20,90)}.

Table V.2 shows that the proposed method with a sliding window equal to six timesteps ($W_L = 6$) yields the best performing scores overall. We should point out that if the thresholds are α_1=10 and α_2=20, the classifier maximizes the recall (Sensitivity) with low precision in the $W_L = 16$ compared with $W_L = 6$. The model with $W_L = 16$ gives us a 58%, 77%, 97%, 61%, and 97% for precision, recall, specificity, F1-Score, and accuracy accordingly. On the other hand, the model has achieved 71%, 73%, 96%, 63%, and 98% for precision, recall, specificity, F1-Score, and accuracy, respectively, with $W_L = 6$. The high recall and low precision indicate that most of the

faults are correctly recognized, but there are a lot of false alarms (as in $W_L = 16$). Contrarily, a low recall and high precision appear that some faults are missed, but those real faults are flagged, and there are no false alarms (as in $W_L = 6$). In addition, The classifier with a high specificity shows fewer false alarms (as in $W_L = 16$).

The critical goal is to get the best classifier that maximizes both precision and recall (Best F1-score) with a modest specificity. It is also observed that the result obtained by the sliding window W_L=6 showed up a higher accuracy with these thresholds (α_1 = 10, α_2 = 20) and (α_1 = 10, α_2 = 30). On the other hand, based on the F1-score, a better estimation is obtained by the sliding window W_L=6 with α_1 = 10 and α_2 = 30. Note that the sliding window W_L=26 obtains the worst result with the thresholds α_1 = 10 and α_2 = 30.

TABLE V.2: Results of our approach on test set FD001 vs. (W_L) and (α_1,α_2).

		Precision	Sensitivity	Specificity	F1-Score	AUC	Accuracy	G-mean
W_L=6	$\alpha_1 = 10, \alpha_2 = 20$	**0.71**	0.73	0.96	0.63	**0.99**	**0.98**	0.84
	$\alpha_1 = 10, \alpha_2 = 30$	**0.71**	**0.83**	0.91	**0.76**	**0.99**	**0.98**	**0.87**
	$\alpha_1 = 10, \alpha_2 = 70$	0.62	0.79	0.84	0.67	0.93	0.89	0.82
	$\alpha_1 = 10, \alpha_2 = 90$	0.61	0.79	0.81	0.66	0.88	0.83	0.80
	$\alpha_1 = 20, \alpha_2 = 30$	0.69	0.73	0.95	0.64	0.98	0.96	0.84
	$\alpha_1 = 20, \alpha_2 = 70$	0.67	0.77	0.84	0.71	0.92	0.87	0.81
	$\alpha_1 = 20, \alpha_2 = 90$	0.64	0.75	0.81	0.68	0.88	0.80	0.78
W_L=16	$\alpha_1 = 10, \alpha_2 = 20$	0.58	0.77	**0.97**	0.61	**0.99**	0.97	0.86
	$\alpha_1 = 10, \alpha_2 = 30$	0.60	0.76	0.93	0.66	0.98	0.96	0.84
	$\alpha_1 = 10, \alpha_2 = 70$	0.61	0.78	0.85	0.65	0.91	0.89	0.81
	$\alpha_1 = 10, \alpha_2 = 90$	0.57	0.77	0.8	0.61	0.87	0.82	0.79
	$\alpha_1 = 20, \alpha_2 = 30$	0.62	0.72	0.93	0.62	0.97	0.94	0.82
	$\alpha_1 = 20, \alpha_2 = 70$	0.67	0.72	0.85	0.69	0.91	0.87	0.78
	$\alpha_1 = 20, \alpha_2 = 90$	0.62	0.73	0.80	0.66	0.86	0.77	0.77
W_L=26	$\alpha_1 = 10, \alpha_2 = 20$	0.47	0.64	0.94	0.50	0.98	0.97	0.78
	$\alpha_1 = 10, \alpha_2 = 30$	0.55	0.72	0.91	0.61	0.97	0.96	0.81
	$\alpha_1 = 10, \alpha_2 = 70$	0.61	0.79	0.85	0.64	0.92	0.89	0.82
	$\alpha_1 = 10, \alpha_2 = 90$	0.59	0.76	0.82	0.61	0.89	0.83	0.8
	$\alpha_1 = 20, \alpha_2 = 30$	0.60	0.70	0.91	0.60	0.96	0.93	0.8
	$\alpha_1 = 20, \alpha_2 = 70$	0.68	0.71	0.85	0.70	0.90	0.88	0.78
	$\alpha_1 = 20, \alpha_2 = 90$	0.66	0.72	0.82	0.68	0.87	0.82	0.77

The confusion matrices are depicted in Figure V.6, where the ordinate shows the reference label while the abscissa represents the predicted one. We mention that classifier fails to predict the last class (Deg 2) when the thresholds of α_1 and α_2 are too near. Even with the SW size variation, the predicted degradation 2 is almost nonexistent for $\alpha_1 = 10$ and $\alpha_2 = 20$. Furthermore, the performance of the RUL prediction is improved with the increase of the gap between the two degradation thresholds. The good results are obtained by the slide window W_L = 6 with $\alpha_1 = 10$, $\alpha_2 = 30$ and W_L = 26 with $\alpha_1 = 10$, $\alpha_2 = 70$. For W_L = 26 with $\alpha_1 = 10$, $\alpha_2 = 70$, the proportion of the true positive classification for the degradation classes 0, 1 and 2 are respectively 95%, 60% and 82%. On other hand, for W_L = 6 with $\alpha_1 = 10$, $\alpha_2 = 30$,

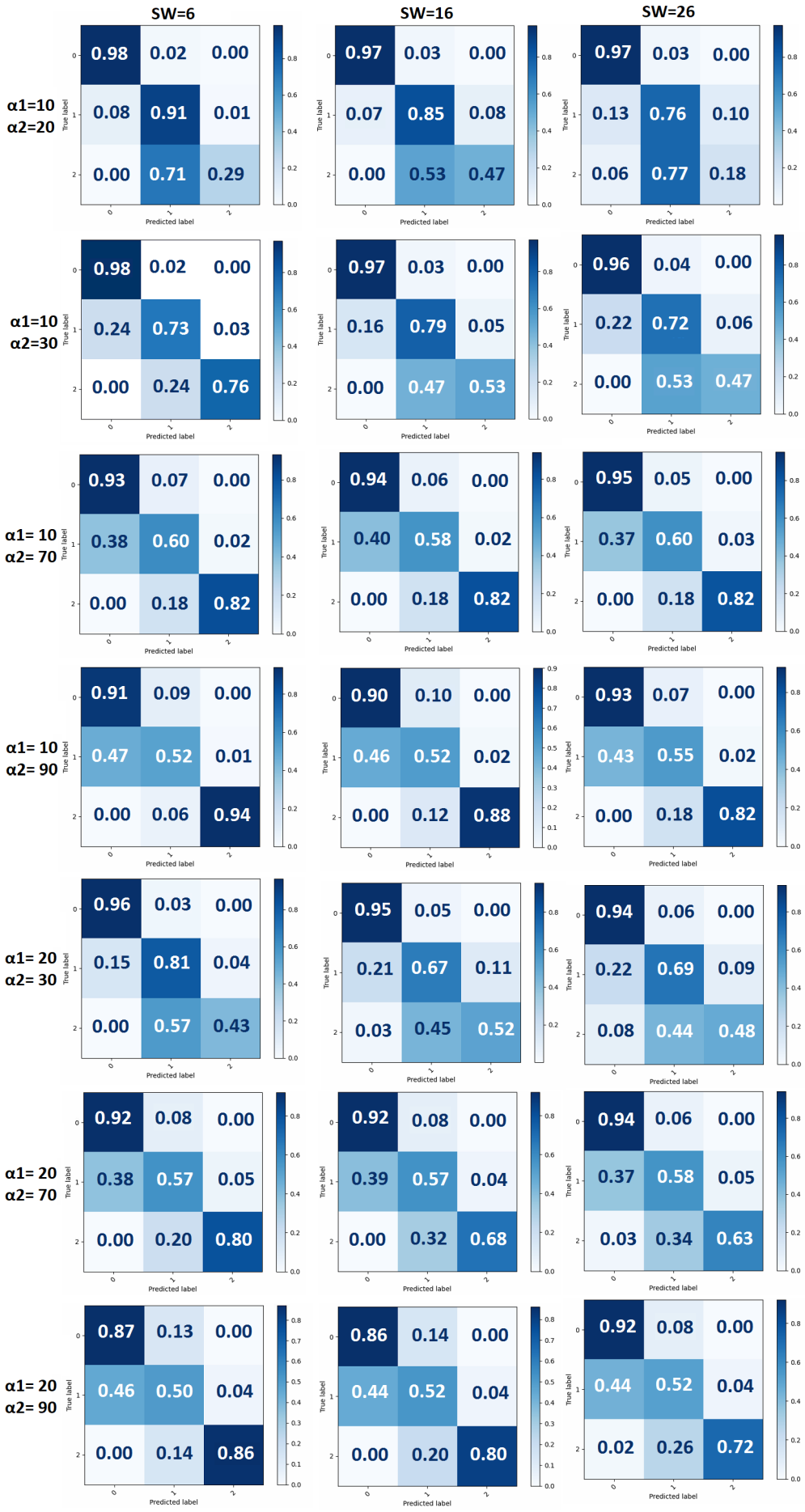


FIGURE V.6: Confusion matrices of test set FD001 vs. (W_L) and (α_1,α_2).

the proportion of the true positive classification for the degradation classes 0, 1 and 2 are respectively 98%, 73% and 76%.

Based on the results, one remarks that our approach is entirely adapted for RUL estimation. To be more precise, all the false positive predictions belong to the less critical degradation class. The following performances can be observed in the confusion matrix generated by the sliding window W_L= 6 and (α_1 = 10, α_2 = 30):

- **Deg 0:** 98% are correctly classified as degradation 0, and 2% are incorrectly classified as degradation 1.
- **Deg 1:** 73% are correctly classified as degradation 1, and 24%, 3% are incorrectly classified as degradation 0, degradation 2, respectively.
- **Deg 2:** 76% are correctly classified as degradation 2, and 24% are incorrectly classified as degradation 1.

Table V.3 gives the detail of the ACVE architecture used. The optimal ACVE parameters have been selected using AHPS based on Bayesian Optimization with a Gaussian Process model [169], which is handled by the Keras-tuner library [122], as described in Algorithm 3. The acquisition function used is Upper Confidence Bound (UCB), for more details can be found [196]. The hyperparameter range detailed in Table V.4, along with the values selected by BO. These most sensitive hyperparameters are chosen due to their highest impact on performance.

TABLE V.3: The Proposed Hybrid deep convolutional variational auto-encoder architectures with attention mechanism.

Layer	Type	Neurons	Kernels
	`Encoder`		
0	Input vector	$W_L \times 14 \times 1$	-
1	Convolution	$W_L \times 14 \times 64$	6×1
2	Convolution	$3 \times 7 \times 128$	6×1
3	Convolution	$3 \times 7 \times 1$	6×1
4	Reshape	3×7	-
5	Attention vector	128	-
6	Mean layer	2	-
7	Standard deviation layer	2	-
	`Decoder`		
0	Sampling layer	2	-
1	Deconvolution	$3 \times 7 \times 128$	6×1
2	Deconvolution	$W_L \times 14 \times 64$	6×1
3	Output vector	$W_L \times 14 \times 1$	-
	`Ensemble learning`		
0	Sampling layer	2	-
1	Gradient Boosting	Estimators=500	-
2	Random Forest	Estimators=32	-
3	Multi-layer Perceptron	10	-
4	Output	3	-

Algorithm 3 Hyperparameter Tuning using Bayesian Optimization with Gaussian Processes

Required: D : *Hyperparameters combination*, **u :** *Acquisition function*
Initial Settings: Randomly Initialize D
for $n = 1$ *to* T **do** ▷ T represents the maximum trial run.
Find x_t by minimizing u over GP :
$x_t = argmin_x\ u(x|D_{1:t-1})$
Evaluate the objective function y_t = f(x_t).
Augment the observation set $D = D \cup (x_t, y_t)$, update the posterior of function f.
end for
return Choosing the best hyperparameters combination

TABLE V.4: Description of the hyperparameters, their range and the selected values.

Name	Range	Selected
N° of convolutional layer	`Min=1, Max=3, Step=1`	2
N° of filters per layer	`Min=8, Max=256, Step=8`	(64,128)
Filter size	`Min=1, Max=16, Step=1`	6×1
Learning rate	`Min=1e-4, Max=0.5, Sampling=LOG`	0.001
Batch size	`Min=32, Max=1024, Step=32`	128
Activation function	`relu, tanh, sigmoid, softplus, softsign, selu, elu`	elu
Optimizer	`Adam, Adadelta, Adamax, SGD, RMSprop, Adagrad, Nadam, Ftrl`	RMSprop

V.4.3.2 Visualisation of latent vectors and identification of the conflict zone

To quantitatively assess the proposed method's performance, we have compared the visualisation performance of the ACVAE method with three state-of-the-art dimension reduction methods, including PCA, ISOMAP, and T-SNE. Figure V.7 displays the 2D-space distribution of six different dimensionality reduction methods. As a recall, going from the green to the red colour point means a decrease in the health state of the engine machine (colours indicate the three degradation classes). In looking at Figure V.7, it can be deduced that ACVAE seems to be able to cluster the dataset more effectively according to RUL degradation reasoning. It can also be argued that the ACVAE resulted in a more compacted spatial distribution compared to the covered areas by other methods. The horizontal and vertical axis extends from approximately 0.002 to 0.012 and -0.8 to 0.6, respectively.

Indeed, the best average accuracy was gained by ACVAE and ACAE, around 98%. The worst result obtained by the T-SNE method is approximately 78% accuracy, as shown in Table V.5. The quantitative results showed that the precision of degradation classes 1 and 2 are almost non-existent with ISOMAP, PCA, and T-SNE. Besides, the T-SNE method shows a low precision of 0%, a sensitivity of 2%, a F1-Score of 1%, and an AUC of 42% for degradation class 2, making degradation class 2 unpredictable. The performance of CVAE is slightly improved when the attention mechanism is added across all measures except the sensitivity, with 8%, 1%, 6%,

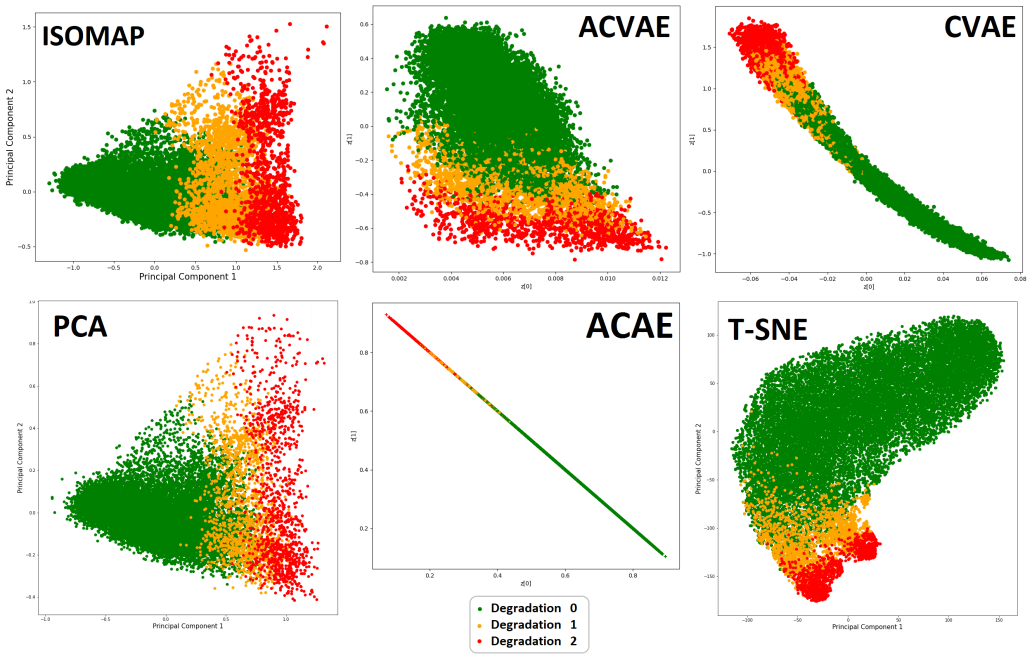


FIGURE V.7: The 2D-visualisation using five dimensionality reduction methods: ISOMAP, ACVAE, CVAE, PCA, ACAE and T-SNE.

TABLE V.5: Comparison of the proposed ACVAE with other methods: ISOMAP, PCA, TSNE, ACAE and CVAE based on test set FD001.

		Precision	Sensitivity	Specificity	F1-Score	AUC	Accuracy	G-mean
CVAE With AM	Deg 0	0.99	0.98	0.77	0.99	0.98		
	Deg 1	0.55	0.73	0.984	0.62	0.98		
	Deg 2	0.59	0.76	1.00	0.67	1.00		
	Mean	**0.71**	**0.83**	**0.917**	**0.76**	**0.99**	**0.98**	**0.87**
CVAE Without AM	Deg 0	0.99	0.98	0.743	0.99	0.982		
	Deg 1	0.49	0.66	0.981	0.56	0.978		
	Deg 2	0.42	0.88	0.998	0.57	0.998		
	Mean	**0.63**	**0.84**	**0.907**	**0.70**	**0.986**	**0.97**	**0.87**
CAE With AM	Deg 0	0.99	0.99	0.617	0.99	0.978		
	Deg 1	0.60	0.55	0.99	0.57	0.975		
	Deg 2	0.40	0.71	0.998	0.51	0.999		
	Mean	**0.66**	**0.75**	**0.868**	**0.69**	**0.98**	**0.98**	**0.81**
ISOMAP	Deg 0	1.00	0.94	0.95	0.97	0.985		
	Deg 1	0.21	0.61	0.942	0.31	0.932		
	Deg 2	0.12	1.00	0.99	0.21	0.998		
	Mean	**0.44**	**0.85**	**0.960**	**0.50**	**0.971**	**0.93**	**0.90**
PCA	Deg 0	1.00	0.95	0.958	0.97	0.99		
	Deg 1	0.12	0.44	0.95	0.19	0.92		
	Deg 2	0.11	1.00	0.99	0.19	0.998		
	Mean	**0.41**	**0.80**	**0.966**	**0.45**	**0.969**	**0.94**	**0.87**
T-SNE	Deg 0	1.00	0.79	0.92	0.88	0.88		
	Deg 1	0.00	0.02	0.92	0.01	0.42		
	Deg 2	0.01	1.00	0.85	0.02	0.93		
	Mean	**0.34**	**0.60**	**0.896**	**0.30**	**0.743**	**0.78**	**0.73**

0.4%, 1%, for Precision, Specificity, F1-Score, AUC, and Accuracy, respectively. The CVAE with AM gave us the highest performance compared to the other dimensionality reduction methods.

Besides, we can observe that our proposed VAE-based architecture outperforms an attention convolutional autoencoder (AE-based architecture), as these results prove its effectiveness in extracting useful performance degradation features. It is also interesting to see the difference between the distribution and the clustering of degradation classes in both VAE-based architecture (ACVAE) and AE-based architecture (ACAE). Our approach seems to be able to map degradation features into a less disentangled latent space (as shown in Figure V.7). These results hypothesise that the standard autoencoder would suffice in the case of dissimilar data classification. Therefore, the power of the VAE architecture comes with similar-looking data (degradation features) that usually overlap in some areas where AE fails to disentangle. According to these results, we can visually perceive them by analyzing the clusters obtained by the worst and best methods, concentrated in different areas. T-SNE distribution seems to cover more area with the horizontal and vertical axis extending from approximately -100 to 150 and -150 to 100, respectively.

The 2D-latent space representation of three training engine units is presented to visually appreciate the effect of our proposed ACVAE method on the spatial distribution (See Figure V.8). We can approximately see that spatial distribution seems to be the RUL engine degradation, where the RUL's engine is linearly decreasing with time until the degradation engine reaches failure.

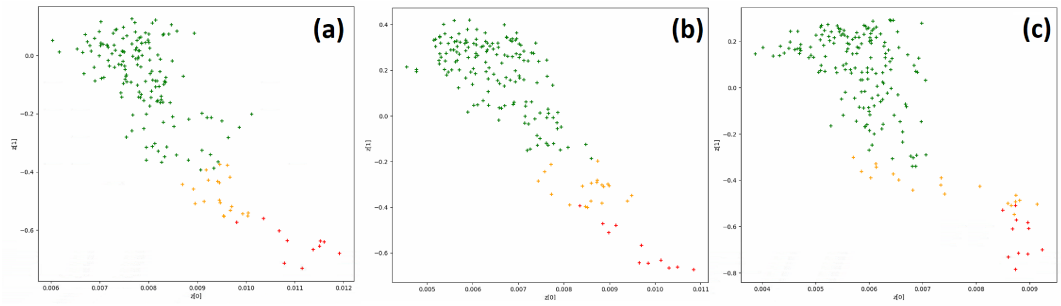


FIGURE V.8: The 2D-latent representation of three training engine units in FD001 using our proposed hybrid architecture: (a) engine #1, (b) engine #6, and (c) engine #100.

Figure V.9 (a) shows the conflict zone obtained by the learning process described by Algorithm 4. This conflict area is represented by black points when the classifiers gave opposite responses for the same input data I. These are considered samples with uncertain classification in the boundary between classes. As indicated in Algorithm 4, several classifiers β_j were trained with the 2D-latent space X^{2D}_{train}. If two classifiers for each input sample I of the X^{2D}_{train} give two opposite responses, β_j and β_k with j $\neq$ k, a sample I is regarded as part of the conflict zone (uncertain samples). This conflict area is reduced by applying a soft voting classifier that combines the decisions of different classifiers by averaging the class probabilities (Figure V.9 (b)). The soft voting classifier selects the class with the highest average probability.

Algorithm 4 Identification and reduction of the Conflict Zone

Input: Sliding window training data $X_{train} = \{(x^i, y^i)\},\ i = 1, 2, 3, ..., N$
Output: Conflict Zone

Step1: Identification of the conflict zone
Train C classifiers β_j on X^{2D}_{train} as follow :
for j=1 to T **do**

    Train the classifier β_j on X^{2D}_{train} ▷ Algorithm 2

    Save the parameters of β_j
end for

for Each input sample I of the X^{2D}_{train} **do**
    for Each two classifier β_j and β_k with j $\neq$ k **do**
        if $\psi_j(I) \neq \psi_k(I)$ **then** ▷ ψ_j is the output class obtained by the classifier β_j

            The I is considered to be part of the conflict zone
        end if
    end for
end for

Step2: Reduction of the conflict zone
for Each input sample I of the X^{2D}_{train} **do** ▷ *Soft Voting in Ensemble Learning*
    $P_f = Average_{i=1}^{classes} \sum_{j=1}^{C} P_{ij}$ ▷ P_{ij} *is probability of each target variable*
    $D_f = argmax\ P_f$
end for

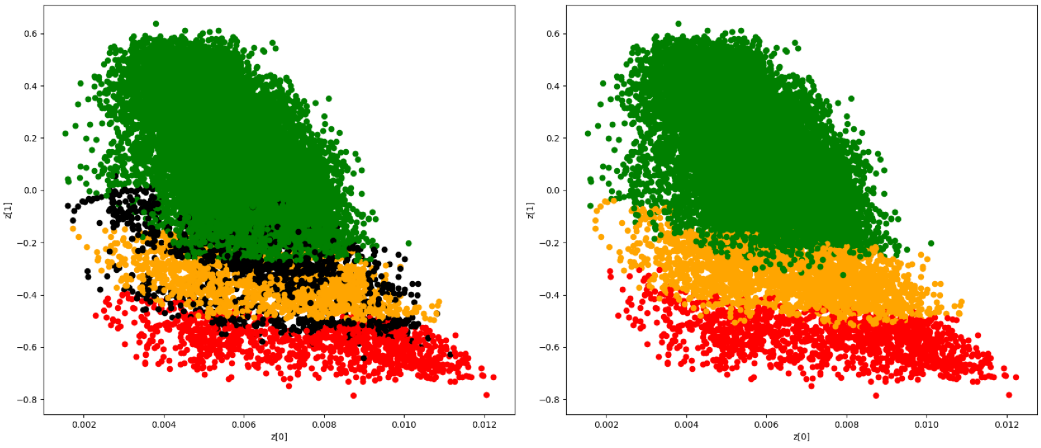


FIGURE V.9: Identification and Reduction of the conflict zone: (a) 2D-visualisation for the conflict zone where black points indicate the samples with uncertain classification (b) 2D-visualisation of the soft voting classifier results.

V.4.3.3 Performance analysis

In this section, a comparison of the soft voting classifier with other powerful existing models has been conducted using different evaluation metrics. Table V.6 depicts the

performance of the different ML classifiers on the FD001 subset, with and without oversampling method. It can be concluded from Table V.6 that all classifiers with and without oversampling method show a high accuracy, that is, 98%. Therefore, the performance of classifiers has to be investigated on other measures, such as Precision, Sensitivity, Specificity, F1-Score, and G-mean.

TABLE V.6: Comparison of the soft voting classifier with other powerful ML models, with and without SMOTE.

		Precision	Sensitivity	Specificity	F1-Score	AUC	Accuracy	G-mean
With SMOTE								
LR	Deg 0	0.99	0.98	0.77	0.99	0.98		
	Deg 1	0.51	0.69	0.98	0.59	0.98		
	Deg 2	0.37	0.76	1.00	0.50	1.00		
	Mean	0.63	0.81	0.916	0.69	0.986	**0.98**	0.86
GD	Deg 0	0.99	0.99	0.75	0.99	0.978		
	Deg 1	0.54	0.67	0.985	0.60	0.977		
	Deg 2	0.42	0.82	1.00	0.56	0.999		
	Mean	0.65	**0.83**	0.91	0.72	0.985	**0.98**	0.867
MLP	Deg 0	0.99	0.98	0.76	0.99	0.98		
	Deg 1	0.51	0.69	0.98	0.59	0.98		
	Deg 2	0.44	0.82	1.00	0.57	1.00		
	Mean	0.65	**0.83**	0.91	0.72	0.987	**0.98**	**0.87**
RF	Deg 0	0.99	0.99	0.707	0.99	0.98		
	Deg 1	0.53	0.63	0.985	0.58	0.97		
	Deg 2	0.41	0.88	0.998	0.56	1.00		
	Mean	0.64	**0.83**	0.896	0.71	0.98	**0.98**	0.86
KNN	Deg 0	0.99	0.98	0.74	0.99	0.96		
	Deg 1	0.51	0.67	0.98	0.58	0.956		
	Deg 2	0.44	0.82	1.00	0.57	0.97		
	Mean	0.65	**0.83**	0.91	0.71	0.96	**0.98**	0.866
Voting Classifier	Deg 0	0.99	0.98	0.77	0.99	0.98		
	Deg 1	0.55	0.73	0.984	0.62	0.98		
	Deg 2	0.59	0.76	1.00	0.67	1.00		
	Mean	0.71	**0.83**	**0.917**	**0.76**	**0.99**	**0.98**	**0.87**
Without SMOTE								
LR	Deg 0	0.99	1.00	0.57	0.99	0.9836		
	Deg 1	0.71	0.53	0.99	0.61	0.981		
	Deg 2	0.64	0.41	1.00	0.50	0.998		
	Mean	0.78	0.65	0.85	0.70	0.987	**0.98**	0.74
GB	Deg 0	0.99	1.00	0.52	0.99	0.9776		
	Deg 1	0.70	0.50	1.00	0.58	0.975		
	Deg 2	1.00	0.24	1.00	0.38	0.999		
	Mean	**0.90**	0.58	0.839	0.65	0.98	**0.98**	0.695
MLP	Deg 0	0.99	0.99	0.57	0.99	0.983		
	Deg 1	0.69	0.54	0.99	0.61	0.981		
	Deg 2	0.80	0.47	1.00	0.59	0.999		

TABLE V.6: **Continued.**

	Mean	0.83	0.67	0.85	0.73	0.987	**0.98**	0.755
RF	Deg 0	0.99	0.99	0.54	0.99	0.959		
	Deg 1	0.62	0.51	0.99	0.56	0.9558		
	Deg 2	0.75	0.35	1.00	0.48	0.997		
	Mean	0.79	0.62	0.84	0.68	0.971	**0.98**	0.72
KNN	Deg 0	0.99	0.99	0.542	0.99	0.949		
	Deg 1	0.65	0.51	0.992	0.57	0.945		
	Deg 2	0.83	0.29	0.999	0.43	0.998		
	Mean	0.82	0.60	0.84	0.67	0.964	**0.98**	0.7124
Voting Classifier	Deg 0	0.99	0.99	0.57	0.99	0.9828		
	Deg 1	0.70	0.55	0.99	0.61	0.981		
	Deg 2	1.00	0.41	1.0	0.58	0.999		
	Mean	**0.90**	0.65	0.85	0.73	0.987	**0.98**	0.745

Table V.6 indicate a trade-off between precision and sensitivity (Recall). Our model cannot simultaneously have high sensitivity and high precision, either in adding synthetic samples or not (although we do aim for high precision and high recall value). When applying the oversampling SMOTE method (adding synthetic samples in both Deg 1 and 2), the performance of classifiers is slightly improved with 18%, 6.7%, 3%, and 12.5% for sensitivity, specificity, F1-Score, and G-mean, respectively, except for the Precision (SMOTE has a negative effect in both degradation class 1 and 2 Precision rate). There is a cost associated with getting lower sensitivity or precision. Ideally, in this case, we aim to avoid machine failure situations, reduce the probability of unscheduled downtime (due to the high cost), and eliminate the causes of serious accidents (safety). Therefore, we choose to maximize sensitivity rather than precision, where the classifier can catch many faults (predicting failure) but end up with many false alarms. In other words, we can endure if a non-failure is flagged as a failure, but a failure should not be labelled as a non-failure. Additionally, we compared the obtained results using F1-score that conveys both precision and sensitivity into one coherent metric, where it can be concluded that our proposed method with SMOTE showed up a higher F1-score rate than without SMOTE. We can clearly observe from Table V.6 that the method of combining classifiers (soft voting classifier) has achieved maximum precision and F1-Score value of 71% and 76%, respectively, compared to other prevailing ML algorithms.

As depicted in Figure V.10, the true positive classification obtained without oversampling SMOTE gave us the lowest accuracy, which means that the soft voting classifier fails to predict the least critical degradation classes (Deg 1 and 2). Figure V.11 represents the ROC curve of our proposed approach ACVAE with a soft voting classifier over sliding windows W_L=6 and α_1 = 10, α_2 = 30. This curve plots the true-positive rate (sensitivity) against the false-positive rate (specificity), which assesses the AUC of the degradation classes. In [119], the authors used the LSTM approach for temporal features extraction and predicted the probability that the equipment will fail within a prespecified time window. It can be seen that the sensitivity of Deg

1 (true positive classification of Deg 1) is improved by increasing the gap between the two thresholds (α_1, α_2). Contrarily, the sensitivity of Deg 0 is decreasing. We mention that this approach fails to predict Deg 1 when the thresholds α_1 and α_2 are too near.

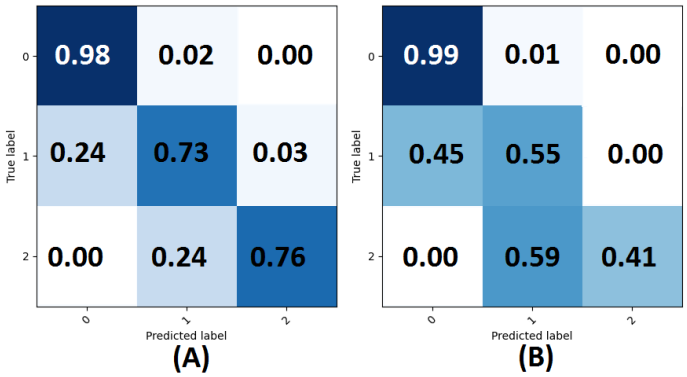


FIGURE V.10: The confusion matrices were obtained on the test set FD001 for the soft voting classifier, (A) with SMOTE and (B) without SMOTE.

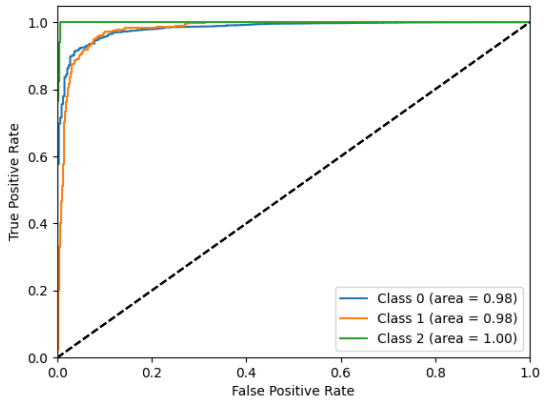


FIGURE V.11: ROC cure of our proposed approach over sliding windows W_L=6 and α_1 = 10, α_2 = 30.

We also compared the obtained results from the proposed approach with two published works that have used the same subset for the same objective (See Table V.7). In [194] and [119], authors have used the confusion matrix as the only performance evaluation. Table V.7 represents the proportion of the true positive classification obtained on the test sets FD001, as well as the mean sensitivity metric. Note that Deg 0 refers to low degradation, Deg 1 indicates medium degradation, and Deg 2 points out to high degradation. The latter triggers various maintenance interventions. As it can be seen from the results, our proposed method delivers better results with different alpha values than the existing methods, as it showed up a higher sensitivity rate (as shown in Table V.7).

As shown in Table V.8, when the machine belongs to the state Deg 0, the mean confusion probability that the machine belongs to Deg 2 is not negligible. The worst case is 3.06%, where $\alpha_2 = 20$. From a maintenance viewpoint, all the false positive predictions belong to high critical degradation (such as the Deg 1 belongs to Deg

TABLE V.7: Comparison with previous work.

		Probability confusion matrix			
		$\alpha_1 = 10, \alpha_2 = 20$	$\alpha_1 = 10, \alpha_2 = 30$	$\alpha_1 = 10, \alpha_2 = 70$	$\alpha_1 = 10, \alpha_2 = 90$
[119]	Deg 0	94.25	88.74	**66.67**	46.19
	Deg 1	0.39	14.45	**67.46**	77.33
	Deg 2	99.98	99.98	**99.97**	99.99
	Mean	64.87	67.72	**78.03**	74.5
[194]	Deg 0	100	99	98	**91**
	Deg 1	7	46	46	**60**
	Deg 2	94	75	75	**69**
	Mean	67	73.33	73	**73.33**
ACVAE	Deg 0	97	**98**	95	91
	Deg 1	85	**73**	60	52
	Deg 2	47	**76**	82	94
	Mean	76.33	**83**	79	79

TABLE V.8: Comparison with previous work, confusion matrix.

		Probability confusion matrix											
		$\alpha_1 = 10, \alpha_2 = 20$			$\alpha_1 = 10, \alpha_2 = 30$			$\alpha_1 = 10, \alpha_2 = 70$			$\alpha_1 = 10, \alpha_2 = 90$		
		Deg 0	Deg 1	Deg 2	Deg 0	Deg 1	Deg 2	Deg 0	Deg 1	Deg 2	Deg 0	Deg 1	Deg 2
[119]	Deg 0	**94.25**	2.7	3.06	**88.74**	9.81	1.44	**66.67**	33.18	0.16	**46.19**	53.78	0.03
	Deg 1	0	**0.39**	99.61	0.04	**14.45**	85.51	3.8	**67.46**	28.74	3.94	**77.33**	18.73
	Deg 2	0	0.02	**99.98**	0	0.02	**99.98**	0	0.03	**99.97**	0	0.01	**99.99**
[194]	Deg 0	**100**	0	0	**99**	1	0	**98**	1	1	**91**	9	0
	Deg 1	45	**7**	48	45	**46**	9	53	**46**	1	38	**60**	2
	Deg 2	0	6	**94**	0	25	**75**	0	25	**75**	0	31	**69**
ACVAE	Deg 0	**97**	3	0	**98**	2	0	**95**	5	0	**91**	9	0
	Deg 1	7	**85**	8	24	**73**	3	37	**60**	3	47	**52**	1
	Deg 2	0	53	**47**	0	24	**76**	0	18	**82**	0	6	**94**

2). This estimation leads to early maintenance and significant lost costs. Contrary to their approach, all the false positive predictions of our approach belong to the less critical degradation class. This latter can have a positive effect in some situations and a negative effect in others. Considering a more compacted time interval leads to more precise maintenance decisions. Thus, assuming that the expert or the maintenance managers are interested in the machine's probability of deteriorating in three different time ranges where α_1 and α_2 are too near (it is also assumed by [119] for the decision making). To be more precise, we assume that the expert is interested in $\alpha_2 \leq 30$. We should point out that the predicted degradation of class 1 is almost nonexistent for $\alpha_2 \leq 30$ in [119], therefore, in order to prevent this lot of false alarms (early maintenance), our proposed approach maximizes the probability of predicted degradation of class 1. It can also be argued that with $\alpha_2 = 30$, both critical degradation classes (Deg 1 and Deg 2) were correctly predicted using our approach compared to both related works. The proportion of the true positive classification for the degradation classes 0, 1, and 2 are 98%, 73%, and 76%, respectively.

Furthermore, the proposed method significantly reduces the maintenance cost rates, where the continuity in the latent space leads to the machine belonging to the state Deg 0 (low degradation) not belonging to Deg 2 (high degradation). On the other hand, in our approach, it can be seen that the sensitivity of Deg 2 is improved, and the sensitivity of Deg 1 decreases with increasing α_2. Contrarily, in [194], the sensitivity of Deg 1 is improved as well as the sensitivity of Deg 2 decreases with increasing α_2.

Future Works: In this work, it was clearly observed that the obtained results show significant improvements in the RUL prediction compared with previous similar works. However, many future works could focus on : $i)$ optimizing conflict zones by identifying outliers and omitting them, $ii)$ giving a formal approach to define the suitable thresholds $(\alpha_1, \alpha_2, \ldots, \alpha_i)$, $iii)$ combining a maintenance strategy with our RUL prediction approach could be useful in the decision-making process, and $IV)$ improving generation performance of synthetic samples using our ACVAE method combined with a generative adversarial network (GAN) [190], $V)$ leveraging the power of the pruning algorithm in the model selection algorithm to stop unpromising trials at the early stages of the training, referred to as automated early-stopping, $VI)$ incorporating an automated method that detects the fault time step of each engine to tackle the problem of RUL that is ill-defined in healthy operation.

V.5 Conclusion

In this chapter, a new strategy for RUL prediction based on ACVAE jointly with a soft voting classifier has been presented as conducive to the predictive maintenance of aero-engines. This approach performs a simultaneous assess the health status of degrading machinery (provides the probabilities of system failure in different time windows) and assigns them to the RUL window estimation. It starts with an automatic extraction of performance degradation features from multiple sensors using the ACVAE method. In this model, the power of the attention layer is to dynamically increase the weights of the useful features in the ACVAE encoding phase to make the network pay attention to these vital features for RUL classes estimation. As articulated in the results section, it was also demonstrated that the ACVAE could cluster the dataset more effectively according to RUL degradation reasoning. In this context, 2D-latent space of ACVAE behaves better than the existing dimension reduction methods (PCA, ISOMAP, and T-SNE), which resulted in a more compacted and better spatial distribution compared with the covered areas by other methods. The conflict zones, which are located near the boundaries between classes, are identified when the classifiers give opposite responses for the same input data. Therefore, to reduce this conflict zone, the soft voting classifier is used. It selects the highest probability class by combining the decisions of different classifiers using the probability classes average.

Chapter VI

Achievements and Conclusions

This chapter summarizes the contributions of this thesis and presents several research directions that require further investigations in the future.

VI.1 Thesis aims

Predictive maintenance is an essential tool for asset management at all levels of operation in many so-called asset-dependent industries, such as oil and gas, aerospace and industrial manufacturing. It reduces system downtime, improves operational safety and saves on operational costs. The maintenance decision within PdM could be based on different characteristics such as the remaining useful life (RUL), the reliability or the cost function. Therefore, the RUL form represents key element of the PdM in decision-making, which is the number of times an asset can accomplish its intended task before requiring maintenance. Indeed, accurate mechanical asset RUL enable manufacturing industries plan future maintenance in advance and ensure a seamless repair and maintenance process with the goal of approaching zero downtime. Whereas an inaccurate estimation of RUL can lead to irreversible environmental damage or safety problems, such as a jet crash caused by engine failure, a rail accident caused by bearing failure, wasted crude oil and ocean pollution caused by mechanical failure on an offshore oil platform, and much more.

The increase in complexity of modern physical assets necessitates an increasing number of sensing instruments to accurately measure an asset's behaviour and operating conditions during its life cycle. This increase in sensor measurements translates into large volumes of data representing a great opportunity for the maturity of data-driven PdM methods in industry and academia, mainly relying on machine learning-based computational methods (e.g. linear or non-linear regression) to deliver a more effective and efficient maintenance service. The data acquired from such machinery can be usually high-dimensional or even redundant and unnecessary variables and variables with minimal variance (i.e.low information content) due to multi-sensor measurements, discontinuous due to the wide ranges of parameter variations during continuous sensor measurements and, therefore, estimation of RUL is usually a challenging task. Unfortunately, ML algorithms generally perform poorly on lower data quality, which can yield high rates of inaccurate fault identification and/or time-of-failure (RUL) prediction results. Therefore, the direction of machine learning research has shifted to more complex models i.e. deep learning, given their ability to process highly non-linear and varied data in their raw form without any human intervention. Although there are a large variety of deep learning prognostic methods that have shown promising results in the context of prognostics, building an effective and efficient prognostic approach is still challenging and limited. This thesis focuses primarily on developing, implementing and evaluating automatic and efficient data-driven methods for predicting the RUL and deals with some problems encountered with improving RUL estimation.

VI.2 Thesis Contributions

The problems treated in this thesis gave birth to two main contributions. These contributions can be summarized as follows:

VI.2.1 First Contribution

The first contribution was concerned the leveraging the power of multimodal and hybrid deep neural network techniques for RUL estimation enhancement instead of incorporating a single model. The main objective of this study is to focus primarily on developing, implementing and evaluating automatic and efficient data-driven methods through testing several deep learning techniques. This work proposes two deep end-to-end architectures that jointly optimise the feature reduction and RUL prediction steps in a hierarchical way, intending to achieve data representation in low dimensionality and minimal variable redundancy while preserving critical asset information with minimal preprocessing effort. Firstly, we proposed a CAE with a temporal modeling tool that combines BDGRU and BDLSTM models in a parallel manner for degradation features extraction and RUL prediction. We found that the CAE is more suited for data extraction and reduction rather than conventional approaches. Secondly, for a comprehensive comparison, a second hybrid architecture consisting concurrently of CNN and BDGRU models is developed and applied to capture local and temporal features for the RUL estimation. The GRU has appeared as an improved LSTM model with few parameters to increase the training stage's efficiency and speed. Besides, for the extraction of bidirectional temporal features and to prevent the long-term dependency problem, we used a BDGRU. The proposed hybrid models' evaluated results indicate significant improvements over their counterparts and the most robust literature results in terms of RMSE on the C-MAPSS public NASA dataset. We pointed out that the CAE with DBGRU-BDLSTM outcomes reliably performs higher for FD001, FD002, FD003, and FD004 than our CNN-BDGRU outcomes, in terms of the RMSE value around 14.208%, 6.83%, 3.967%, and 5.537%.

VI.2.2 Second Contribution

Based on the limitation of the first contribution, we treated the prognostics as a regression problem, providing only a single deterministic predicted value of the RUL (estimating the exact time of failure), which has a zero probability of being accurate. The second contribution focuses on developing an innovative RUL estimation strategy that simultaneously assesses degrading machinery's health status (provides the probabilities of system failure in different time windows) and assigns them to the RUL window estimation (predict an interval of time that covers all the distribution that could be more certain). This new RUL prediction approach based on ACVAE jointly with a soft voting classifier has been presented as conducive to the predictive maintenance of aero-engines. The starting point of the proposed method was to automatically extract performance degradation features from multiple sensors using the ACVAE method. The primary objective of applying the ACVAE was to provide a

more structured, disentanglement and lower-dimensional representation of the data that shows the best class distribution over a 2D latent space and demonstrates how well the ACVAE generalises. As articulated in the results section, it was also demonstrated that the ACVAE could cluster the dataset more effectively according to RUL degradation reasoning. In this context, this 2D-latent space of ACVAE also behaved better than the existing dimension reduction methods (PCA, ISOMAP, and T-SNE), which resulted in a more compacted and better spatial distribution compared with the covered areas by other methods. The conflict zones are located near the boundaries between classes, identified when the classifiers give opposite responses for the same input data. Therefore, the soft voting classifier was used to reduce this conflict zone. It selects the highest probability class by combining the decisions of different classifiers using the probability classes average. Furthermore, Automatic Hyperparameters Selection was used to pick out the best configuration of hyperparameters for our hybrid architecture without being time-consuming and avoid error-prone.

VI.3 Perspectives

Although this PhD process culminates with the contributions presented in this dissertation, obviously, no model is perfect. However, the proposed developments on prognostics modeling show good potential and improved performances compared to conventional methods. Future work will include testing the applicability of the proposed methods to other prognostic and real-world applications. Besides, the current developments of this thesis will be further extended as follows:

- **Quantifying uncertainty in DL/ML models:** Several sources of uncertainty often affect the estimation of RUL, either due to gathered data (e.g. due to lack of sufficient quantities of run-to-failure data, sensor noise, and unknown environmental and operating conditions) or degradation mechanisms (e.g., choice of the model's type, estimating its parameters). Therefore, knowing how confident the model is in its own RUL predictions is essential. Besides, decision-making should be based on the bounds of the RUL confidence interval rather than a single value. Furthermore, in future work, we intend to propose a method for assessing, measuring and quantifying the uncertainty in the neural network's prediction to produce reliable classifiers/regressors [49, 103, 141].

- **Anomaly detection triggered RUL estimation:** The remaining useful life of industrial machinery is ill-defined in the absence of degradation (during healthy operation). To address this issue, we aim to use anomaly monitoring during RUL estimator training and deployment, which incorporates an automated method that detects the fault time step of each engine and is considered the starting point of the degradation.

- **Imbalanced data:** Imbalanced data is one of the most critical data problems for data analysis tasks. Many techniques to deal with this issue have been proposed in the literature, including data-level techniques, algorithmic-level

methods, cost-sensitive methods and classifier ensembles. In our work, we focused only on the data-level method and chose the simple method called SMOTE. In future work, we aim to compare and integrate the very promising approach, such as generative adversarial network (GAN), as a data-level or cost-sensitive method in order to be able to solve precision/recall trade-off problems.

- **Optimisation of the conflict zones:** The conflict zones are located near the boundaries between classes, which are identified when the classifiers give opposite responses for the same input data. Therefore, future developments may focus on the optimisation of the conflict zones, where it is recommended to identify the outliers and omit them from the training dataset, which may help improve the classifier's predictions. Besides that, one could think about is to propose an approach that can automatically determine the number of states (clusters) from multidimensional data or propose a formal approach to setting the appropriate thresholds ($\alpha_1, \alpha_2, \ldots, \alpha_i$) without the human assumption that minimise the conflict zones.

- **Post-prognostics decision-making:** The decision process results from the prognosis module, as stated in the PHM architecture, which is supposed to use available prognostic information (RUL estimation). This raises the direct question of how this information is integrated into the decision-making process. We intend to extend our work on the whole PHM process by integrating our RUL prediction approach with the decision-making process (some condition-based maintenance strategies).

Bibliography

[1] A Agogino and K Goebel. "Mill data set". In: *BEST lab, UC Berkeley. NASA Ames PrognosticsDataRepository,[http://ti. arc. nasa. gov/projec t/prognostic-data-repository], NASA Ames, Moffett Field, CA* (2007).

[2] Ali Al-Dulaimi et al. "A multimodal and hybrid deep neural network model for remaining useful life estimation". In: *Computers in Industry* 108 (2019), pp. 186–196.

[3] Farzana Anowar, Samira Sadaoui, and Bassant Selim. "Conceptual and empirical comparison of dimensionality reduction algorithms (PCA, KPCA, LDA, MDS, SVD, LLE, ISOMAP, LE, ICA, t-SNE)". In: *Computer Science Review* 40 (2021), p. 100378.

[4] Oluseun Aremu. "Achieving a representation of asset data conducive to machine learning driven predictive maintenance". In: (2020).

[5] Sylvain Arlot and Alain Celisse. "A survey of cross-validation procedures for model selection". In: *Statistics surveys* 4 (2010), pp. 40–79.

[6] Olgun Aydin and Seren Guldamlasioglu. "Using LSTM networks to predict engine condition on large scale data processing framework". In: *2017 4th International Conference on Electrical and Electronic Engineering (ICEEE)*. IEEE. 2017, pp. 281–285.

[7] Giduthuri Sateesh Babu, Peilin Zhao, and Xiao-Li Li. "Deep convolutional neural network based regression approach for estimation of remaining useful life". In: *International conference on database systems for advanced applications*. Springer. 2016, pp. 214–228.

[8] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate". In: *arXiv preprint arXiv:1409.0473* (2014).

[9] Pierre Baldi. "Autoencoders, unsupervised learning, and deep architectures". In: *Proceedings of ICML workshop on unsupervised and transfer learning*. JMLR Workshop and Conference Proceedings. 2012, pp. 37–49.

[10] Horace B Barlow. "Unsupervised learning". In: *Neural computation* 1.3 (1989), pp. 295–311.

[11] Ron Bekkerman, Mikhail Bilenko, and John Langford. *Scaling up machine learning: Parallel and distributed approaches*. Cambridge University Press, 2011.

[12] Oguz Bektas, Amjad Alfudail, and Jeffrey A Jones. "Reducing dimensionality of multi-regime data for failure prognostics". In: *Journal of Failure Analysis and Prevention* 17.6 (2017), pp. 1268–1275.

[13] Abir Belaala et al. "Computer-Aided Diagnosis for Spitzoid Lesions Classification Using Artificial Intelligence Techniques". In: *International Journal of Healthcare Information Systems and Informatics (IJHISI)* 16.1 (2021), pp. 16–37.

[14] Abir Belaala et al. *Skin cancer and deep learning for dermoscopic images classification: A pilot study.* 2020.

[15] Yoshua Bengio. "How auto-encoders could provide credit assignment in deep networks via target propagation". In: *arXiv preprint arXiv:1407.7906* (2014).

[16] James Bergstra and Yoshua Bengio. "Random search for hyper-parameter optimization." In: *Journal of machine learning research* 13.2 (2012).

[17] James Bergstra et al. "Algorithms for hyper-parameter optimization". In: *25th annual conference on neural information processing systems (NIPS 2011)*. Vol. 24. Neural Information Processing Systems Foundation. 2011.

[18] Hadrien Bertrand. "Hyper-parameter optimization in deep learning and transfer learning: applications to medical imaging". PhD thesis. Université Paris-Saclay, 2019.

[19] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. "Variational inference: A review for statisticians". In: *Journal of the American statistical Association* 112.518 (2017), pp. 859–877.

[20] Giuseppe Bonaccorso. *Machine learning algorithms*. Packt Publishing Ltd, 2017.

[21] Omar Bougacha. "Contribution to post-pronostic decision: a new framework based on prongnostic/decision interaction". PhD thesis. Université Bourgogne Franche-Comté, 2020.

[22] V Bourne. *a. GD c. ServiceMax, After the fall: The costs, causes & consequences of unplanned downtime*. Tech. rep. Technical Report, GE Digital, 2017.

[23] Leo Breiman. "Bagging predictors". In: *Machine learning* 24.2 (1996), pp. 123–140.

[24] J Brownlee. "Ensemble Learning Algorithms With Python: Make Better Prediction with Bagging, Boosting, and Stacking". In: *Machine Learning Mastery* (2021).

[25] Diego Cabrera et al. "Fault diagnosis of spur gearbox based on random forest and wavelet packet decomposition". In: *Frontiers of Mechanical Engineering* 10.3 (2015), pp. 277–286.

[26] Smith WA Canchumuni, Alexandre A Emerick, and Marco Aurélio C Pacheco. "Towards a robust parameterization for conditioning facies models using deep variational autoencoders and ensemble smoother". In: *Computers & Geosciences* 128 (2019), pp. 87–102.

[27] Rich Caruana. "Multitask learning". In: *Machine learning* 28.1 (1997), pp. 41–75.

[28] Thyago P Carvalho et al. "A systematic literature review of machine learning methods applied to predictive maintenance". In: *Computers & Industrial Engineering* 137 (2019), p. 106024.

[29] Anthony L Caterini and Dong Eui Chang. *Deep neural networks in a mathematical framework*. Springer, 2018.

[30] Gavneet Singh Chadha, Arfyan Rabbani, and Andreas Schwung. "Comparison of semi-supervised deep neural networks for anomaly detection in industrial processes". In: *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*. Vol. 1. IEEE. 2019, pp. 214–219.

[31] Varun Chandola, Arindam Banerjee, and Vipin Kumar. "Anomaly detection: A survey". In: *ACM computing surveys (CSUR)* 41.3 (2009), pp. 1–58.

[32] Jinglong Chen et al. "Gated recurrent unit based recurrent neural network for remaining useful life prediction of nonlinear deterioration process". In: *Reliability Engineering & System Safety* 185 (2019), pp. 372–382.

[33] Xuefeng Chen et al. "Basic research on machinery fault diagnostics: Past, present, and future trends". In: *Frontiers of Mechanical Engineering* 13.2 (2018), pp. 264–291.

[34] Zhenghua Chen et al. "Machine remaining useful life prediction via an attention-based deep learning approach". In: *IEEE Transactions on Industrial Electronics* 68.3 (2020), pp. 2521–2531.

[35] Feifan Cheng, Q Peter He, and Jinsong Zhao. "A novel process monitoring approach based on variational recurrent autoencoder". In: *Computers & Chemical Engineering* 129 (2019), p. 106515.

[36] Junyoung Chung et al. "Empirical evaluation of gated recurrent neural networks on sequence modeling". In: *arXiv preprint arXiv:1412.3555* (2014).

[37] Camila Ferreira Costa and Mario A Nascimento. "Ida 2016 industrial challenge: Using machine learning for predicting failures". In: *International Symposium on Intelligent Data Analysis*. Springer. 2016, pp. 381–386.

[38] Paulo Roberto de Oliveira da Costa et al. "Attention and long short-term memory network for remaining useful lifetime predictions of turbofan engine degradation". In: *International Journal of Prognostics and Health Management* 10 (2019), p. 034.

[39] Michael Crawshaw. "Multi-task learning with deep neural networks: A survey". In: *arXiv preprint arXiv:2009.09796* (2020).

[40] Li Deng and Dong Yu. "Deep learning: methods and applications". In: *Foundations and trends in signal processing* 7.3–4 (2014), pp. 197–387.

[41] Balbir S Dhillon. *Engineering maintenance: a modern approach*. cRc press, 2002.

[42] Remco M Dijkman et al. "Business models for the Internet of Things". In: *International Journal of Information Management* 35.6 (2015), pp. 672–678.

[43] Dina Elreedy and Amir F Atiya. "A comprehensive analysis of synthetic minority oversampling technique (SMOTE) for handling class imbalance". In: *Information Sciences* 505 (2019), pp. 32–64.

[44] Dina Elreedy and Amir F Atiya. "A novel distribution analysis for smote oversampling method in handling class imbalance". In: *International Conference on Computational Science*. Springer. 2019, pp. 236–248.

[45] Manuel Esperon-Miguez, Philip John, and Ian K Jennions. "A review of Integrated Vehicle Health Management tools for legacy platforms: Challenges and opportunities". In: *Progress in Aerospace Sciences* 56 (2013), pp. 19–34.

[46] Ya Ju Fan. "Autoencoder node saliency: Selecting relevant latent representations". In: *Pattern Recognition* 88 (2019), pp. 643–653.

[47] Dean K Frederick, Jonathan A DeCastro, and Jonathan S Litt. *User's guide for the commercial modular aero-propulsion system simulation (C-MAPSS)*. Tech. rep. 2007.

[48] MA Ganaie, Minghui Hu, et al. "Ensemble deep learning: A review". In: *arXiv preprint arXiv:2104.02395* (2021).

[49] Jakob Gawlikowski et al. "A survey of uncertainty in deep neural networks". In: *arXiv preprint arXiv:2107.03342* (2021).

[50] Ali Ghodsi. "Dimensionality reduction a short tutorial". In: *Department of Statistics and Actuarial Science, Univ. of Waterloo, Ontario, Canada* 37.38 (2006), p. 2006.

[51] Kai Goebel et al. *Prognostics: The science of making predictions*. 2017.

[52] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[53] Anjana Gosain and Saanchi Sardana. "Handling class imbalance problem using oversampling techniques: A review". In: *2017 international conference on advances in computing, communications and informatics (ICACCI)*. IEEE. 2017, pp. 79–85.

[54] R Gouriveau et al. "PHM-Prognostics and health management-De la surveillance au pronostic de défaillances de systèmes complexes". In: *Techniques de l'Ingénieur* 9 (2013).

[55] Rafael Gouriveau, Kamal Medjaher, and Noureddine Zerhouni. *Du concept de PHM à la maintenance prédictive 1: Surveillance et pronostic*. Vol. 3. ISTE Group, 2017.

[56] Rafael Gouriveau, Kamal Medjaher, and Noureddine Zerhouni. *From prognostics and health systems management to predictive maintenance 1: Monitoring and prognostics*. John Wiley & Sons, 2016.

[57] Margherita Grandini, Enrico Bagli, and Giorgio Visani. "Metrics for Multi-Class Classification: an Overview". In: *arXiv preprint arXiv:2008.05756* (2020).

[58] John M Gross. *Fundamentals of preventive maintenance*. AMACOM/American Management Association, 2002.

[59] Chen Guo. "Study on the recognition of aero-engine blade-casing rubbing fault based on the casing vibration acceleration". In: *Measurement* 65 (2015), pp. 71–80.

[60] Liang Guo et al. "A recurrent neural network based health indicator for remaining useful life prediction of bearings". In: *Neurocomputing* 240 (2017), pp. 98–109.

[61] Jayant Sen Gupta et al. "Characterization of prognosis methods: an industrial approach". In: *PHM Society European Conference*. Vol. 1. 1. 2012.

[62] Hashem M Hashemian. "State-of-the-art predictive maintenance techniques". In: *IEEE Transactions on Instrumentation and measurement* 60.1 (2010), pp. 226–236.

[63] Kaiming He and Jian Sun. "Convolutional neural networks at constrained time cost". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 5353–5360.

[64] Felix O Heimes. "Recurrent neural networks for remaining useful life estimation". In: *2008 international conference on prognostics and health management*. IEEE. 2008, pp. 1–6.

[65] Aiwina Heng et al. "Rotating machinery prognostics: State of the art, challenges and opportunities". In: *Mechanical systems and signal processing* 23.3 (2009), pp. 724–739.

[66] Geoffrey Hinton et al. "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups". In: *IEEE Signal processing magazine* 29.6 (2012), pp. 82–97.

[67] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. "A fast learning algorithm for deep belief nets". In: *Neural computation* 18.7 (2006), pp. 1527–1554.

[68] Geoffrey E Hinton and Ruslan R Salakhutdinov. "Reducing the dimensionality of data with neural networks". In: *science* 313.5786 (2006), pp. 504–507.

[69] Geoffrey E Hinton and Richard Zemel. "Autoencoders, minimum description length and Helmholtz free energy". In: *Advances in neural information processing systems* 6 (1993).

[70] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[71] Kenneth Holmberg et al. *E-maintenance*. Springer Science & Business Media, 2010.

[72] Xianxu Hou et al. "Improving variational autoencoder with deep feature consistent and generative adversarial training". In: *Neurocomputing* 341 (2019), pp. 183–194.

[73] Che-Sheng Hsu and Jehn-Ruey Jiang. "Remaining useful life estimation using long short-term memory deep learning". In: *2018 IEEE International Conference on Applied System Invention (ICASI)*. IEEE. 2018, pp. 58–61.

[74] Runqing Huang et al. "Residual life predictions for ball bearings based on self-organizing map and back propagation neural network methods". In: *Mechanical systems and signal processing* 21.1 (2007), pp. 193–207.

[75] Amit Kumar Jain and Bhupesh Kumar Lad. "Predicting Remaining Useful Life of high speed milling cutters based on Artificial Neural Network". In: *2015 International Conference on Robotics, Automation, Control and Embedded Systems (RACE)*. IEEE. 2015, pp. 1–5.

[76] Andrew KS Jardine, Daming Lin, and Dragan Banjevic. "A review on machinery diagnostics and prognostics implementing condition-based maintenance". In: *Mechanical systems and signal processing* 20.7 (2006), pp. 1483–1510.

[77] Kamran Javed, Rafael Gouriveau, and Noureddine Zerhouni. "State of the art and taxonomy of prognostics approaches, trends of prognostics applications and open issues towards maturity at different technology readiness levels". In: *Mechanical Systems and Signal Processing* 94 (2017), pp. 214–236.

[78] Feng Jia et al. "Deep neural networks: A promising tool for fault characteristic mining and intelligent diagnosis of rotating machinery with massive data". In: *Mechanical systems and signal processing* 72 (2016), pp. 303–315.

[79] Stephen B Johnson et al. *System health management: with aerospace applications*. John Wiley & Sons, 2011.

[80] Tinu Theckel Joy et al. "Hyperparameter tuning for big data using Bayesian optimisation". In: *2016 23rd International Conference on Pattern Recognition (ICPR)*. IEEE. 2016, pp. 2574–2579.

[81] H. Kagermann et al. *Recommendations for Implementing the Strategic Initiative INDUSTRIE 4.0: Securing the Future of German Manufacturing Industry ; Final Report of the Industrie 4.0 Working Group*. Forschungsunion, 2013. URL: `https://books.google.it/books?id=AsfOoAEACAAJ`.

[82] Utkarsh Mahadeo Khaire and R Dhanalakshmi. "Stability of feature selection algorithm: A review". In: *Journal of King Saud University-Computer and Information Sciences* (2019).

[83] Asifullah Khan et al. "A survey of the recent architectures of deep convolutional neural networks". In: *Artificial intelligence review* 53.8 (2020), pp. 5455–5516.

[84] Diederik P Kingma and Max Welling. "Auto-encoding variational bayes". In: *arXiv preprint arXiv:1312.6114* (2013).

[85] Diederik Pieter Kingma. "Variational inference & deep learning: A new synthesis". In: (2017).

[86] Mojtaba Kordestani et al. "Failure prognosis and applications—A survey of recent literature". In: *IEEE transactions on reliability* (2019).

[87] Ranganath Kothamasu, Samuel H Huang, and William H VerDuin. "System health monitoring and prognostics—a review of current paradigms and practices". In: *The International Journal of Advanced Manufacturing Technology* 28.9 (2006), pp. 1012–1024.

[88] Pradeep Kundu and Bhupesh Kumar Lad. "PCA-ANN Based Approach for Remaining Useful Life Prediction for Roller Ball Bearings". In: *International Conference on Business Analytics and Intelligence*. Vol. 17. 2015. 2015.

[89] Ahmed Lahmadi, Labib Terrissa, and Noureddine Zerhouni. "A data-driven method for estimating the remaining useful life of a composite drill pipe". In: *2018 International Conference on Advanced Systems and Electric Technologies (IC_ASET)*. IEEE. 2018, pp. 192–195.

[90] Heiner Lasi et al. "Industry 4.0". In: *Business & information systems engineering* 6.4 (2014), pp. 239–242.

[91] Yann Le Cun and Françoise Fogelman-Soulié. "Modèles connexionnistes de l'apprentissage". In: *Intellectica* 2.1 (1987), pp. 114–143.

[92] Mitchell Lebold and Michael Thurston. "Open standards for condition-based maintenance and prognostic systems". In: *Maintenance and reliability conference (MARCON)*. Vol. 200. May. 2001.

[93] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *nature* 521.7553 (2015), pp. 436–444.

[94] Min Su Lee et al. "AESNB: active example selection with naïve Bayes classifier for learning from imbalanced biomedical data". In: *2009 Ninth IEEE International Conference on Bioinformatics and Bioengineering*. IEEE. 2009, pp. 15–21.

[95] Seulki Lee et al. "Process monitoring using variational autoencoder for high-dimensional nonlinear processes". In: *Engineering Applications of Artificial Intelligence* 83 (2019), pp. 13–27.

[96] Yaguo Lei et al. "Machinery health prognostics: A systematic review from data acquisition to RUL prediction". In: *Mechanical systems and signal processing* 104 (2018), pp. 799–834.

[97] Jialin Li, Xueyi Li, and David He. "A directed acyclic graph network combined with CNN and LSTM for remaining useful life prediction". In: *IEEE Access* 7 (2019), pp. 75464–75475.

[98] Jing Li et al. "Attention mechanism-based CNN for facial expression recognition". In: *Neurocomputing* 411 (2020), pp. 340–350.

[99] Xiang Li, Qian Ding, and Jian-Qiao Sun. "Remaining useful life estimation in prognostics using deep convolution neural networks". In: *Reliability Engineering & System Safety* 172 (2018), pp. 1–11.

[100] Yuan Liao, Linxuan Zhang, and Chongdang Liu. "Uncertainty prediction of remaining useful life using long short-term memory network based on bootstrap method". In: *2018 IEEE International Conference on Prognostics and Health Management (ICPHM)*. IEEE. 2018, pp. 1–8.

[101] Yun Lin et al. "Adversarial attacks in modulation recognition with convolutional neural networks". In: *IEEE Transactions on Reliability* 70.1 (2020), pp. 389–401.

[102] Xing Liu, Jianbo Yu, and Lyujiangnan Ye. "Residual attention convolutional autoencoder for feature learning and fault detection in nonlinear industrial processes". In: *Neural Computing and Applications* (2021), pp. 1–17.

[103] Titouan Lorieul. "Uncertainty in predictions of Deep Learning models for fine-grained classification". PhD thesis. Université Montpellier, 2020.

[104] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. "Effective approaches to attention-based neural machine translation". In: *arXiv preprint arXiv:1508.04025* (2015).

[105] Jian Ma et al. "Predicting the remaining useful life of an aircraft engine using a stacked sparse autoencoder with multilayer self-learning". In: *Complexity* 2018 (2018).

[106] Laurens Van der Maaten and Geoffrey Hinton. "Visualizing data using t-SNE." In: *Journal of machine learning research* 9.11 (2008).

[107] Alireza Makhzani and Brendan Frey. "K-sparse autoencoders". In: *arXiv preprint arXiv:1312.5663* (2013).

[108] Arnaz Malhi and RX Gao. "Recurrent neural networks for long-term prediction in machine condition monitoring". In: *Proceedings of the 21st IEEE Instrumentation and Measurement Technology Conference (IEEE Cat. No. 04CH37510)*. Vol. 3. IEEE. 2004, pp. 2048–2053.

[109] Arnaz Malhi, Ruqiang Yan, and Robert X Gao. "Prognosis of defect propagation based on recurrent neural networks". In: *IEEE Transactions on Instrumentation and Measurement* 60.3 (2011), pp. 703–711.

[110] Wentao Mao et al. "Imbalanced fault diagnosis of rolling bearing based on generative adversarial network: A comparative study". In: *IEEE Access* 7 (2019), pp. 9515–9530.

[111] Marcello Mastroleo et al. "Automatic analysis of faulty low voltage network asset using deep neural networks". In: *The Journal of Engineering* 2018.15 (2018), pp. 851–855.

[112] Tom M Mitchell et al. "Machine learning. 1997". In: *Burr Ridge, IL: McGraw Hill* 45.37 (1997), pp. 870–877.

[113] R Keith Mobley. *An introduction to predictive maintenance*. Elsevier, 2002.

[114] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2018.

[115] Condition Monitoring. "Diagnostics of machines-prognostics part 1: General guidelines". In: *ISO13381-1:(e). vol. ISO/IEC Directives Part 2, IO f. S* 14 (2004).

[116] Alexandre Muller, Adolfo Crespo Marquez, and Benoit Iung. "On the concept of e-maintenance: Review and current research". In: *Reliability Engineering & System Safety* 93.8 (2008), pp. 1165–1187.

[117] Patrick Nectoux et al. "PRONOSTIA: An experimental platform for bearings accelerated degradation tests." In: *IEEE International Conference on Prognostics and Health Management, PHM'12.* IEEE Catalog Number: CPF12PHM-CDR. 2012, pp. 1–8.

[118] Giang Nguyen et al. "Machine learning and deep learning frameworks and libraries for large-scale data mining: a survey". In: *Artificial Intelligence Review* 52.1 (2019), pp. 77–124.

[119] Khanh TP Nguyen and Kamal Medjaher. "A new dynamic predictive maintenance framework using deep learning for failure prognostics". In: *Reliability Engineering & System Safety* 188 (2019), pp. 251–262.

[120] Sangyoon Oh, Min Su Lee, and Byoung-Tak Zhang. "Ensemble learning with active example selection for imbalanced biomedical data classification". In: *IEEE/ACM transactions on computational biology and bioinformatics* 8.2 (2010), pp. 316–325.

[121] Nabil Omri. "Knowledge extraction from SME data for the implementation of PHM process". PhD thesis. Université Bourgogne Franche-Comté, 2021.

[122] Tom O'Malley et al. "Keras Tuner". In: *Retrieved May* 21 (2019), p. 2020.

[123] Sinno Jialin Pan and Qiang Yang. "A survey on transfer learning". In: *IEEE Transactions on knowledge and data engineering* 22.10 (2009), pp. 1345–1359.

[124] Jyoti Pareek and Joel Jacob. "Data Compression and Visualization Using PCA and T-SNE". In: *Advances in Information Communication Technology and Computing*. Springer, 2021, pp. 327–337.

[125] S Patro and Kishore Kumar Sahu. "Normalization: A preprocessing stage". In: *arXiv preprint arXiv:1503.06462* (2015).

[126] Antoine Proteau et al. "Dimension reduction and 2D-visualization for early change of state detection in a machining process with a variational autoencoder approach". In: *The International Journal of Advanced Manufacturing Technology* 111.11 (2020), pp. 3597–3611.

[127] Qinglin Qi and Fei Tao. "Digital twin and big data towards smart manufacturing and industry 4.0: 360 degree comparison". In: *IEEE Access* 6 (2018), pp. 3585–3593.

[128] Emmanuel Ramasso and Rafael Gouriveau. "Remaining useful life estimation by classification of predictions based on a neuro-fuzzy system and theory of belief functions". In: *IEEE Transactions on Reliability* 63.2 (2014), pp. 555–566.

[129] Ikram Remadna et al. "An overview on the deep learning based prognostic". In: *2018 International Conference on Advanced Systems and Electric Technologies (IC_ASET)*. IEEE. 2018, pp. 196–200.

[130] Ikram Remadna et al. "Leveraging the Power of the Combination of CNN and Bi-directional Lstm Networks for Aircraft Engine RUL Estimation". In: *2020 Prognostics and Health Management Conference (PHM-Besançon)*. IEEE. 2020, pp. 116–121.

[131] Ikram Remadna et al. "RUL Estimation Enhancement using Hybrid Deep Learning Methods". In: *International Journal of Prognostics and Health Management* 12.1 (2021).

[132] Shaoqing Ren et al. "Faster r-cnn: Towards real-time object detection with region proposal networks". In: *Advances in neural information processing systems* 28 (2015).

[133] Behnoush Rezaeianjouybari and Yi Shang. "Deep learning for prognostics and health management: State of the art, challenges, and opportunities". In: *Measurement* 163 (2020), p. 107929.

[134] Marc Rovira, Klas Engvall, and Christophe Duwig. "Identifying key features in reactive flows: A tutorial on combining dimensionality reduction, unsupervised clustering, and feature correlation". In: *Chemical Engineering Journal* (2022), p. 135250.

[135] Sebastian Ruder. "An overview of gradient descent optimization algorithms". In: *arXiv preprint arXiv:1609.04747* (2016).

[136] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. *Learning internal representations by error propagation*. Tech. rep. California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[137] Hojjat Salehinejad et al. "Recent advances in recurrent neural networks". In: *arXiv preprint arXiv:1801.01078* (2017).

[138] Arthur L Samuel. "Some studies in machine learning using the game of checkers". In: *IBM Journal of research and development* 3.3 (1959), pp. 210–229.

[139] Arthur L Samuel. "Some studies in machine learning using the game of checkers. II—Recent progress". In: *IBM Journal of research and development* 11.6 (1967), pp. 601–617.

[140] Gabriel San Martin et al. "Deep variational auto-encoders: A promising tool for dimensionality reduction and ball bearing elements fault diagnosis". In: *Structural Health Monitoring* 18.4 (2019), pp. 1092–1128.

[141] Téo Sanchez et al. "Deep Learning Uncertainty in Machine Teaching". In: *27th International Conference on Intelligent User Interfaces*. 2022, pp. 173–190.

[142] Fernando Fernandes dos Santos et al. "Analyzing and increasing the reliability of convolutional neural networks on GPUs". In: *IEEE Transactions on Reliability* 68.2 (2018), pp. 663–677.

[143] Sharifah Saon, Takashi Hiyama, et al. "Predicting remaining useful life of rotating machinery based artificial neural network". In: *Computers & Mathematics with Applications* 60.4 (2010), pp. 1078–1087.

[144] Abhinav Saxena and Kai Goebel. "Turbofan engine degradation simulation data set". In: *NASA Ames Prognostics Data Repository* (2008), pp. 1551–3203.

[145] Abhinav Saxena et al. "Damage propagation modeling for aircraft engine run-to-failure simulation". In: *2008 international conference on prognostics and health management*. IEEE. 2008, pp. 1–9.

[146] Abhinav Saxena et al. "Metrics for evaluating performance of prognostic techniques". In: *2008 international conference on prognostics and health management*. IEEE. 2008, pp. 1–17.

[147] Dominik Scherer, Andreas Müller, and Sven Behnke. "Evaluation of pooling operations in convolutional architectures for object recognition". In: *International conference on artificial neural networks*. Springer. 2010, pp. 92–101.

[148] Jürgen Schmidhuber. "Deep learning in neural networks: An overview". In: *Neural networks* 61 (2015), pp. 85–117.

[149] Mike Schuster and Kuldip K Paliwal. "Bidirectional recurrent neural networks". In: *IEEE transactions on Signal Processing* 45.11 (1997), pp. 2673–2681.

[150] Oscar Serradilla et al. "Deep learning models for predictive maintenance: a survey, comparison, challenges and prospects". In: *Applied Intelligence* (2022), pp. 1–31.

[151] Burr Settles. "Active learning literature survey". In: (2009).

[152] Bobak Shahriari et al. "Taking the human out of the loop: A review of Bayesian optimization". In: *Proceedings of the IEEE* 104.1 (2015), pp. 148–175.

[153] Connor Shorten and Taghi M Khoshgoftaar. "A survey on image data augmentation for deep learning". In: *Journal of big data* 6.1 (2019), pp. 1–48.

[154] Joanna Z Sikorska, Melinda Hodkiewicz, and Lin Ma. "Prognostic modelling options for remaining useful life estimation by industry". In: *Mechanical systems and signal processing* 25.5 (2011), pp. 1803–1836.

[155] Padhraic Smyth and David Wolpert. "Stacked density estimation". In: *Advances in neural information processing systems* 10 (1997).

[156] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. "Practical bayesian optimization of machine learning algorithms". In: *arXiv preprint arXiv:1206.2944* (2012).

[157] Jorge Sola and Joaquin Sevilla. "Importance of input data normalization for the application of neural networks to complex industrial problems". In: *IEEE Transactions on nuclear science* 44.3 (1997), pp. 1464–1468.

[158] Ya Song et al. "Remaining useful life prediction of turbofan engine using hybrid model based on autoencoder and bidirectional long short-term memory". In: *Journal of Shanghai Jiaotong University (Science)* 23.1 (2018), pp. 85–94.

[159] V Sugumaran, GR Sabareesh, and KI Ramachandran. "Fault diagnostics of roller bearing using kernel based neighborhood score multi-class support vector machine". In: *Expert Systems with Applications* 34.4 (2008), pp. 3090–3098.

[160] Jiayu Sun et al. "Learning sparse representation with variational auto-encoder for anomaly detection". In: *IEEE Access* 6 (2018), pp. 33353–33361.

[161] Jiedi Sun, Changhong Yan, and Jiangtao Wen. "Intelligent bearing fault diagnosis method combining compressed data acquisition and deep learning". In: *IEEE Transactions on Instrumentation and Measurement* 67.1 (2017), pp. 185–195.

[162] Yu Sun et al. "Condition-based maintenance for the offshore wind turbine based on long short-term memory network". In: *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability* (2020), p. 1748006X20965434.

[163] Christian Szegedy et al. "Going deeper with convolutions". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1–9.

[164] Wei Ming Tan and T Hui Teo. "Remaining Useful Life Prediction Using Temporal Convolution with Attention". In: *AI* 2.1 (2021), pp. 48–70.

[165] Maintenance Terminology. "The European Standard EN 13306: 2001". In: *European committee for standardization* (2001).

[166] Zhigang Tian. "An artificial neural network method for remaining useful life prediction of equipment subject to condition monitoring". In: *Journal of Intelligent Manufacturing* 23.2 (2012), pp. 227–237.

[167] Jesper E Van Engelen and Holger H Hoos. "A survey on semi-supervised learning". In: *Machine Learning* 109.2 (2020), pp. 373–440.

[168] Tran Van Tung and Bo-Suk Yang. "Machine fault diagnosis and prognosis: The state of the art". In: *International Journal of Fluid Machinery and Systems* 2.1 (2009), pp. 61–71.

[169] A Helen Victoria and G Maragatham. "Automatic tuning of hyperparameters using Bayesian optimization". In: *Evolving Systems* 12 (2021), pp. 217–223.

[170] A Vidhyalakshmi and C Priya. "Medical big data mining and processing in e-health care". In: *An Industrial IoT Approach for Pharmaceutical Industry Growth*. Elsevier, 2020, pp. 1–30.

[171] Pascal Vincent et al. "Extracting and composing robust features with denoising autoencoders". In: *Proceedings of the 25th international conference on Machine learning*. 2008, pp. 1096–1103.

[172] Jiujian Wang et al. "Remaining useful life estimation in prognostics using deep bidirectional lstm neural network". In: *2018 Prognostics and System Health Management Conference (PHM-Chongqing)*. IEEE. 2018, pp. 1037–1042.

[173] Kai Wang et al. "Systematic development of a new variational autoencoder model based on uncertain data for monitoring nonlinear processes". In: *IEEE Access* 7 (2019), pp. 22554–22565.

[174] Long Wang et al. "Wind turbine gearbox failure identification with deep neural networks". In: *IEEE Transactions on Industrial Informatics* 13.3 (2016), pp. 1360–1368.

[175] Yasi Wang, Hongxun Yao, and Sicheng Zhao. "Auto-encoder based dimensionality reduction". In: *Neurocomputing* 184 (2016), pp. 232–242.

[176] You-ren Wang et al. "Planetary gearbox fault feature learning using conditional variational neural networks under noise environment". In: *Knowledge-Based Systems* 163 (2019), pp. 438–449.

[177] Zhaobin Wang et al. "Various frameworks and libraries of machine learning and deep learning: a survey". In: *Archives of computational methods in engineering* (2019), pp. 1–24.

[178] Paul J Werbos. "Generalization of backpropagation with application to a recurrent gas market model". In: *Neural networks* 1.4 (1988), pp. 339–356.

[179] David H Wolpert. "Stacked generalization". In: *Neural networks* 5.2 (1992), pp. 241–259.

[180] Jia Wu et al. "Hyperparameter optimization for machine learning models based on Bayesian optimization". In: *Journal of Electronic Science and Technology* 17.1 (2019), pp. 26–40.

[181] Jun Wu et al. "Data-driven remaining useful life prediction via multiple sensor signals and deep long short-term memory neural network". In: *ISA transactions* 97 (2020), pp. 241–250.

[182] Xun Xu. "From cloud computing to cloud manufacturing". In: *Robotics and computer-integrated manufacturing* 28.1 (2012), pp. 75–86.

[183] Gang Xue, Shifeng Liu, and Yicao Ma. "A hybrid deep learning-based fruit classification using attention model and convolution autoencoder". In: *Complex & Intelligent Systems* (2020), pp. 1–11.

[184] Bo-Suk Yang et al. "An intelligent condition-based maintenance platform for rotating machinery". In: *Expert Systems with Applications* 39.3 (2012), pp. 2977–2988.

[185] Zichao Yang et al. "Hierarchical attention networks for document classification". In: *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*. 2016, pp. 1480–1489.

[186] Shujian Yu and Jose C Principe. "Understanding autoencoders with information theoretic concepts". In: *Neural Networks* 117 (2019), pp. 104–123.

[187] Yong Yu et al. "A review of recurrent neural networks: LSTM cells and network architectures". In: *Neural computation* 31.7 (2019), pp. 1235–1270.

[188] Mei Yuan, Yuting Wu, and Li Lin. "Fault diagnosis and remaining useful life estimation of aero engine using LSTM neural network". In: *2016 IEEE International Conference on Aircraft Utility Systems (AUS)*. IEEE. 2016, pp. 135–140.

[189] Matthew D Zeiler. "Hierarchical convolutional deep learning in computer vision". PhD thesis. New York University, 2013.

[190] Ryad Zemouri. "Semi-supervised adversarial variational autoencoder". In: *Machine Learning and Knowledge Extraction* 2.3 (2020), pp. 361–378.

[191] Ryad Zemouri and Daniel Racoceanu. "Innovative deep learning approach for biomedical data instantiation and visualization". In: *Deep Learning for Biomedical Data Analysis*. Springer, 2021, pp. 171–196.

[192] Ryad Zemouri et al. "Deep convolutional variational autoencoder as a 2D-visualization tool for partial discharge source classification in hydrogenerators". In: *IEEE Access* 8 (2019), pp. 5438–5454.

[193] Ryad Zemouri et al. "Deep Variational Autoencoder: An efficient tool for PHM frameworks". In: *2020 Prognostics and Health Management Conference (PHM-Besançon)*. IEEE. 2020, pp. 235–240.

[194] Ryad Zemouri et al. "Hybrid architecture of deep convolutional variational auto-encoder for remaining useful life prediction". In: (2020).

[195] Jianjing Zhang et al. "Long short-term memory for machine remaining life prediction". In: *Journal of manufacturing systems* 48 (2018), pp. 78–86.

[196] Miao Zhang et al. "Convolutional neural networks-based lung nodule classification: A surrogate-assisted evolutionary algorithm for hyperparameter optimization". In: *IEEE Transactions on Evolutionary Computation* 25.5 (2021), pp. 869–882.

[197] Zehan Zhang et al. "Automated feature learning for nonlinear process monitoring–An approach using stacked denoising autoencoder and k-nearest neighbor rule". In: *Journal of Process Control* 64 (2018), pp. 49–61.

[198] Zehan Zhang et al. "Gaussian feature learning based on variational autoencoder for improving nonlinear process monitoring". In: *Journal of Process Control* 75 (2019), pp. 136–155.

[199] Rui Zhao et al. "Deep learning and its applications to machine health monitoring". In: *Mechanical Systems and Signal Processing* 115 (2019), pp. 213–237.

[200] Shuai Zheng et al. "Long short-term memory network for remaining useful life estimation". In: *2017 IEEE international conference on prognostics and health management (ICPHM)*. IEEE. 2017, pp. 88–95.

[201] Zhi-Hua Zhou. *Ensemble methods: foundations and algorithms*. Chapman and Hall/CRC, 2019.