République algérienne démocratique et populaire
Ministère de l'enseignement supérieur et de la recherche scientifique

Centre Universitaire Abdelhafid Boussouf Mila

Institut des Sciences et Technologie

Département de Mathématiques et Informatique

MÉMOIRE DE MASTER

Pour obtenir le diplôme de Master en Informatique

Option : Sciences et Technologies de l'Information et de la Communication (STIC)

Hybrid Deep Learning Model-based For Remaining Useful Life Estimation

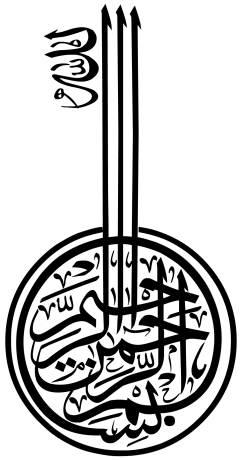
---

Réalisé par : Bekhouche Ikram
Dounyazed Allam

Soutenue devant le jury:

President:	Mr Dib Abderrahim	Centre Universitaire de Mila – Abdelhafid Boussouf
Encadreur:	Dr Khalfi Souheila	Centre Universitaire de Mila – Abdelhafid Boussouf
Co-encadreur:	M^{me} Abderrezek Samira	ENS de Constantine – Assia Dejebar
Examinateur	Dr Boulmerka Aissa	Centre Universitaire de Mila – Abdelhafid Boussouf

Année universitaire 2022–2023



Acknowledgments

We would like to express our sincere gratitude to our supervisors throughout this work, Dr. Khalfi Souheila and Mme. Abderezzek Samira, for the guidance they have provided and the knowledge they have passed to us.

We are also thankful for the committee members, Mr. Dib Abderrahim and Dr. Boulmerka Aissa, for agreeing to evaluate our modest work.

Furthermore, we want to express our profound acknowledgment to the scholars whose academic works have significantly influenced our thesis. Their notable contributions to the field have been an indispensable source of inspiration and insight for our research.

Thank you.

Dedication

الحمد لله و كفى و الصلاة على الحبيب المصطفى و من وفى أما بعد :
الحمد لله الذي وفقنا لتثمين هذه الخطوة في مسيرتنا الدراسية بمذكرتنا هذه ثمرة الجهد و النجاح بفضله تعالى مهداة إلى الوالدين حفظهما الله و أدامهما نورا لدربي لكل العائلة الكريمة التي ساندتني و لا تزال من إخوة و أخوات إلى رفقاء المشوار الذين قاسموني لحظاته من أساتذة و طلاب قسم الرياضيات و الإعلام الآلي إلى زملاء العمل و على رأسهم السيد بودوح فيصل والسيد صخري صلاح الدين

لكم جزيل الشكر.

Dedication

يشرفني ان اهدي هدا العمل المتواضع الذي تم بعون الله و توفيقه و تيسيره إلى والدي الكريمين اللذين لا يمكنني أن أشكرهما بما يكفي على كل ما قدماه لي.

امي الغالية كنت و لازلت سندي في مسيرتي الدراسية، أدامك الله نعمة في حياتي و ادعو ان يرزقك الصحة والسعادة والعمر المديد.

ابي، من يستطيع أن يفتخر ويجد هنا نتيجة سنوات طويلة من التضحية. شكرا على القيم النبيلة والتعليم والدعم الدائم الذي جاء منك، فعلى روحك يا أبي كل الرحمات والمغفرة من الله.

الى زميلتي في هدا العمل »اكرام» اشكرك على مساهمتك التي لا تقدر بثمن و تعاونك الدي كان أساس نجاح اطروحتنا .

الى اخوتي الاعزاء و خاصة اختي الوحيدة »مريم» التي دائما بجانبي اتمنى لك حياة مليئة بالحب والنجاح.

أتقدم بخالص عبارات الشكر و العرفان و المحبة لاصدقائي و الجيران و ادكر بالاخص صديقتي جمانة الغالية و جارتي ربيعة العزيزة في مساهمتها و تقديمها للعون و كدا زملائي في الدراسة و الاساتذة الكرام.

ملخص

يلعب التقدير الدقيق للعمر المفيد المتبقي (RUL) دورا حاسما و هاما في التخفيف من مخاطر فشل الماكينة، و تحديدًا في التشخيص والصيانة القائمة على الحالة. ومن خلال المراقبة المستمرة لظروف و حالة الماكينة والتنبؤ الدقيق بالعمر المتبقي لها، يمكن تنفيذ تدابير استباقية و وقائية لتجنب الأعطال غير المتوقعة.

على مدى السنوات الماضية، أظهرت نماذج التعلم العميق فعالية ملحوظة في التنبؤ بالحياة المفيدة المتبقية للماكنة بدقة من خلال تطبيقات متنوعة. هذه التقنيات قادرة على استخراج أنماط وميزات ذات معنى بشكل فعال من البيانات المعقدة

الهدف الرئيسي من هذا البحث هو اقتراح نموذج هجين قائم على التعلم العميق يستغل تقنيات التعلم العميق المتطورة لتوفير تنبؤ دقيق بالحياة المفيدة المتبقية لمحركات توربوفان الطائرات. بعد ذلك، نحاول تحسين دقة النموذج المقترح من خلال استخدام تقنيات تحسين قائمة على التعلم العميق مثل هندسة تعدد الرؤوس وآلية التركيز.

الكلمات المفتاحية: الحياة المفيدة المتبقية، الذكاء الاصطناعي، التنبؤ وإدارة الصحة، الشبكات العصبية العميقة ، هندسة تعدد الرؤوس آلية التركيز، C-MAPSS .

Abstract

The precise estimation of Remaining Useful Life (RUL) holds crucial importance in mitigating the risks of machine failures, specifically in prognosis and condition-based maintenance (CBM). Through continuous monitoring of machine conditions and accurate RUL prediction, preemptive measures can be implemented to avert unforeseen breakdowns.

Over the past years, deep learning models have demonstrated remarkable effectiveness in accurately predicting RUL across diverse applications. These techniques are able to effectively extract meaningful patterns and features from complex data.

The primary objective of this Master's thesis is to propose a hybrid deep-learning model that leverages advanced deep learning techniques such as multi-head architecture and attention mechanism, to offer an accurate prognosis of the Remaining Useful Life (RUL) for airplane turbofan engines.

Keywords: Remaining Useful Life, Artificial Intelligence, Prognostics and Health Management, Deep Neural Networks, Multi-Head Architecture, Attention Mechanism, C-MAPSS.

Résumé

L'estimation précise de la Durée de Vie Utile Restante (DVUR) revêt une importance primordiale dans les stratégies de pronostic et de maintenance basée sur l'état (MBE). La prédiction de la DVUR joue un rôle essentiel dans la réduction des risques liés aux défaillances des machines. Grâce à la surveillance continue des conditions de la machine et à une prédiction précise de la DVUR, des mesures préventives peuvent être mises en œuvre pour éviter les pannes imprévues.

Les méthodologies d'apprentissage en profondeur ont démontré une efficacité remarquable dans la prévision précise de la DVUR dans diverses applications. En exploitant les capacités des réseaux neuronaux et des algorithmes avancés, ces techniques décryptent avec compétence des données complexes, extrayant des motifs et des caractéristiques saillantes.

L'objectif principal de cee memoire est de proposer un modèle hybride basé sur l'apprentissage en profondeur qui exploite des techniques d'apprentissage en profondeur sophistiquées pour fournir un pronostic précis de la Durée de Vie Utile Restante (DVUR) pour les moteurs turboréacteurs d'avion. Ensuite, nous tentons d'améliorer la précision du modèle proposé en utilisant des techniques d'optimisation basées sur l'apprentissage en profondeur, telles que l'architecture multi-têtes et le mécanisme d'attention.

Mots clés :
Durée de Vie Utile Restante, Intelligence Artificielle, Pronostics et Gestion de la Santé, Réseaux Neuronaux, Architecture Multi-Têtes, Mécanisme d'Attention, C-MAPSS..

Contents

List of Figures

List of Tables

General Introduction

Maintenance has emerged as a critical aspect across diverse industries, including industrial, commercial, and information technology sectors. Its scope has expanded beyond mere repairs to encompass Condition-Based Maintenance (CBM), a proactive approach that monitors equipment or system conditions. By leveraging this approach, organizations can enhance reliability, reduce repair costs, and, crucially, anticipate failures to prevent potential financial losses and human casualties. This preventive aspect of maintenance, coupled with its ability to foresee potential issues, allows organizations to take proactive measures, ensuring operational continuity, minimizing unplanned downtime, and safeguarding against costly repairs or hazardous incidents. By integrating these strategies, maintenance has become a pivotal function that not only ensures the smooth functioning and longevity of equipment and systems but also mitigates risks and safeguards both financial resources and human well-being.

A key objective of Condition-Based Maintenance is to accurately estimate the Remaining Useful Life (RUL) of machines or equipment. RUL represents the amount of time or usage remaining before a piece of equipment reaches a critical state of degradation or failure. This estimation is achieved through the implementation of equipment condition monitoring techniques, including sensor data collection, data analysis, and predictive modeling. RUL estimation serves as a vital tool in Prognostics and Health Management (PHM), offering several benefits such as resource optimization, cost reduction, enhanced safety and reliability, and extended asset lifespan. By accurately predicting the RUL, organizations can strategically plan maintenance activities, allocate resources effectively, minimize downtime, and maximize the operational efficiency of their assets. Ultimately, RUL estimation empowers businesses to make informed decisions, optimize maintenance strategies, and achieve significant improvements in overall equipment performance and longevity.

RUL prognosis approaches can be broadly categorized into three main groups: physics-

based approaches, data-driven approaches, and hybrid approaches. Physics-based approaches emphasize a deep understanding of the equipment or system's physical condition. They leverage comprehensive knowledge of the system's structure, behavior, and underlying physical processes. These approaches require expertise in modeling the system dynamics and degradation mechanisms. On the other hand, data-driven approaches rely on historical data collected from sensors, monitoring systems, or other sources. They use advanced techniques, including deep learning and statistical modeling, to extract valuable insights from the data. By analyzing patterns, trends, and correlations, data-driven approaches can estimate the RUL of equipment or systems. These approaches are particularly effective in capturing complex relationships and adapting to changing operating conditions. Hybrid approaches combine physics-based and data-driven methods to enhance RUL predictions. They harness the best of both worlds by integrating physical understanding with historical data analysis. This allows for more accurate and robust predictions, adapting to varying conditions while incorporating prior knowledge of the system's behavior.

In recent years, Deep Learning (DL) has proven to be an effective tool for modeling of higher-level abstractions through complex and Deep Neural Network (DNN) architectures. It has been widely adopted and has been successful in various domains, including computer vision and natural language processing. This success has also extended its impact on Prognostics and Health Management (PHM) and Remaining Useful Life (RUL) estimation. By leveraging Deep Learning methods, researchers and practitioners have achieved promising results in PHM and RUL estimation tasks. Deep Neural Networks(DNN) with multiple layers are employed to learn complex patterns and extract crucial features from sensor data, historical records, and other relevant information. This enables the identification of hidden correlations and trends that may not be apparent through traditional modeling techniques.

Convolutional Neural Networks (CNN), Gated Recurrent Unit (GRU) and Feed-forward Neural Network (FNN) are among the most commonly used neural network architectures, each designed to address specific types of data and learning tasks. The purpose of a neural network is to process different types of data and perform various tasks such as classification, regression, pattern recognition, and prediction. It is also common to combine two types of neural networks to help surpass their limitations; for instance, CNN has proven to be effective in extracting features from the input data, while GRU has the ability to capture long-term dependencies. A hybrid CNN-GRU model can leverage the strengths of each architecture to enhance overall performance.

Deep learning models can be improved using a range of techniques, including the multi-head model. This approach is particularly effective when dealing with complex

problems. By breaking down the challenge into smaller sub-tasks, each head in the multi-head model can focus on specific aspects of the data. This allows the model to learn diverse aspects and features simultaneously, leading to more accurate predictions. Another powerful technique in Deep Learning is the Attention Mechanism (AM), which enables the model to selectively prioritize and emphasize the most informative parts of the input data. This helps to improve the accuracy of the model's predictions by allowing it to focus on the most relevant temporal or spatial aspects of the problem.

To this end, the main goal of this thesis is to develop a hybrid model that uses deep learning techniques to estimate the Remaining Useful Life (RUL) of airplane turbofan engines. The study will focus on evaluating the effectiveness of the Attention Mechanism and Multi-head architecture on the performance of our proposed model. Additionally, a simpler model will be developed to further improve the study. Finally, the outcomes will be compared with some of the state-of-the-art methods. Leveraging the NASA C-MAPSS dataset as a reference, this research aims to mitigate the severe consequences of unexpected engine failures.

This manuscript is structured as follows:

Chapter 1 discusses several fundamental concepts of Deep Learning, encompassing background notions, the different architectures, and the most commonly used frameworks for development in deep learning.

Chapter 2 explains what maintenance is and its different types, the concept of the Remaining Useful life as well as the various approaches used for its estimation.

In *Chapter 3*, we outline the steps involved in building a Deep learning model and introduced briefly the C-MAPSS data-set. We also described our proposed models and assessed their effectiveness by comparing the results obtained to some state-of-the-art studies.

Chapter 1

Deep Learning

Introduction

Artificial intelligence (AI) is a field of study that focuses on developing intelligent systems that can perform tasks that typically require human intelligence. Machine Learning(ML) is a subset of AI that involves designing algorithms and models that enable computers to learn from and make predictions or decisions based on data without being explicitly programmed. It empowers machines to improve their performance over time through experience. Deep Learning is a class of ML techniques that exploit many layers of non-linear information processing for supervised or unsupervised feature extraction and transformation and for pattern analysis and classification. It has revolutionized pattern recognition, introducing solutions that now power a wide range of technologies in many fields, such as Computer vision (CV), Natural Language Processing (NLP) , and Automatic Speech Recognition. It allows computers to learn from experience and understand the world in terms of a hierarchy of concepts, with each concept defined through its relation to simpler concepts. By gathering knowledge from experience, this approach avoids the need for human operators to formally specify all the knowledge that the computer needs. The hierarchy of concepts enables the computer to learn complicated concepts by building them out of simpler ones.

To successfully apply Deep Learning, a certain level of understanding is required as to how to cast a problem, the basic mathematics of modeling, and the algorithms for fitting models to data. This chapter introduces some preliminaries to facilitate the understanding of deep learning, and covers foundational concepts of DL models. Additionally, it provides an overview of the most widely utilized frameworks for deep learning.

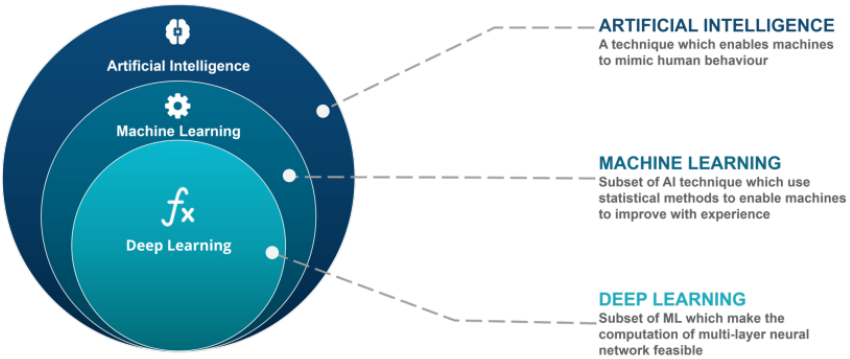


Figure 1.1: Introduction to deep learning [Aggarwal et al., 2022]

1.1 Preliminaries

1.1.1 Artificial Neural Networks

Artificial Neural Networks (ANN) are information processing structures that provide the (often unknown) connection between input and output data by artificially simulating the physiological structure and functioning of human brain structures. The natural Neural Network, instead, consists of a very large number of nerve cells (about ten billion in humans), called neurons, linked together in a complex network. The intelligent behavior is the result of extensive interaction between interconnected units [Gallo, 2015].

An Artificial Neural Network can be created by simulating a network of model neurons in a computer. By applying algorithms that mimic the processes of real neurons, we can make the network 'learn' to solve many types of problems. A model neuron is referred to as a threshold unit. It receives input from a number of other units or external sources, weighs each input and adds them up. If the total input is above a threshold, the output of the unit is one; otherwise it is zero. Therefore, the output changes from 0 to 1 when the total weighted sum of inputs is equal to the threshold [Krogh, 2008].

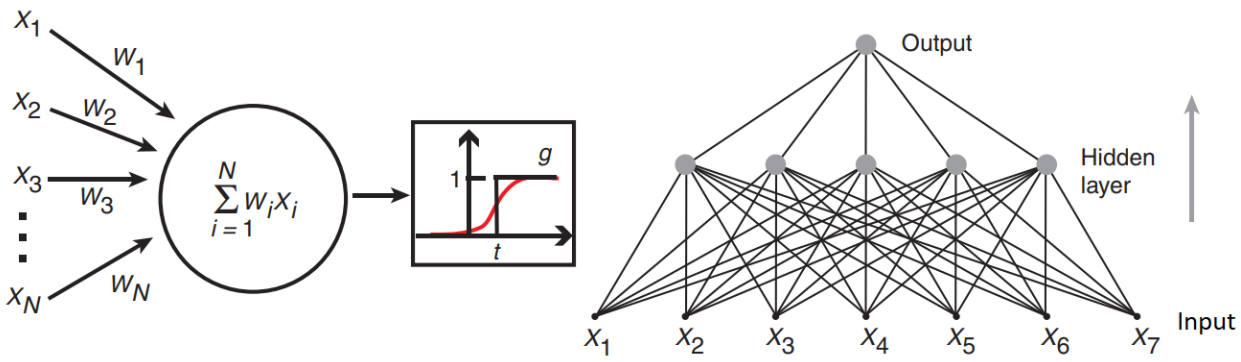


Figure 1.2: A basic artificial neuron network [Krogh, 2008].

1.1.2 Activation Functions

Activation functions are what ANN uses to transform an input signal into an output

signal, which in turn is fed as input to the next layer in the stack. In ANN, we calculate the sum of products of inputs and their corresponding weights apply an activation function to the results to get the output of that particular layer and supply it as the input to the next one. Mainly there are three types of activation functions.

1. Binary step function: this function depends on a threshold, if the input is greater than the threshold, then the neuron is activated, otherwise the neuron remains deactivated, meaning that its output will not be passed on to the next hidden layer. The binary step function, denoted as $H(x)$, can be defined as:

$$H(x) = \begin{cases} 0 & \text{if } x < \text{threshold} \\ 1 & \text{if } x \geq \text{threshold} \end{cases} \tag{1.1}$$

2. Linear activation function: also known as identity function, this function has a linear relationship between the input and output of the neuron network, meaning that the output is a scaled version of the input with no non-linear transformation. Mathematically, a linear activation function can be represented as:

$$f(x) = wx + b \tag{1.2}$$

3. Non-linear activation functions : non-linear activation functions introduce non-linearity to the network, therefor they are crucial to the network for their ability capture non-linear dependencies that may exist between the input and output variables. Some commonly used non-linear activation functions in neural networks include:
   - Sigmoid (Logistic) Function: The sigmoid function maps the input to a range between 0 and 1, providing a smooth non-linear transformation. It is defined as:
   $$f(x) = \frac{1}{1 + e^{-x}} \tag{1.3}$$
   - Hyperbolic Tangent (Tanh) Function: The hyperbolic Tangent Function is similar to the sigmoid function but maps the input to a range between -1 and 1. It is defined as:
   $$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{1.4}$$
   - Rectified Linear Unit (ReLU): The ReLU function is a piecewise linear function that outputs the input directly if it is positive, and 0 if it is negative. It is defined as:
   $$f(x) = \max(0, x) \tag{1.5}$$

- Leaky ReLU : Leaky ReLU is a variation of ReLU activation function that addresses one of its limitations. While ReLU sets all negative values to zero, the Leaky ReLU allows a small, non-zero output for negative inputs. This helps overcome the "dying ReLU" problem, where certain neurons in a network can become inactive and stop learning during training.The Leaky ReLU function is defined as follows:

$$f(x) = \begin{cases} x, & \text{if } x \geq 0 \\ \alpha x, & \text{if } x < 0 \end{cases} \tag{1.6}$$

1.1.3 Loss functions

A loss function is determined as the difference between the target and predicted output values. It measures how well the Neural Network models the training data. The aim is to minimize the average loss between the predicted and target outputs. The hyperparameters are adjusted by finding the weights (w^T) and biases (b), that minimize the value of J (average loss) as demonstrated in the equation:

$$J\left(w^T, b\right) = \frac{1}{m} \sum_{i=1}^{m} L\left(\hat{y}^{(i)}, y^{(i)}\right) \tag{1.7}$$

There are two main types of loss functions:

1. Regression Loss Functions: they are used in regression models where the model predicts the output value through an input value.
   - Mean Squared Error (MSE): MSE takes the difference between the model's predictions and the expected outcome (i.e the ground truth), square it, and average it out across the dataset. The MSE is formally defined by the following equation:

$$MSE = \frac{1}{N} \sum_{i=1}^{N} \left(\hat{y}^{(i)} - y^{(i)}\right)^2 \tag{1.8}$$

     Where N is the number of samples we are testing against.
     One disadvantage of this loss function is that it is very sensitive to outliers, if a predicted value is significantly greater than or less than its target value, this will significantly increase the loss.
   - Mean Absolute Error (MAE): MAE finds the average of the absolute differences between the target and the predicted outputs.

$$MAE = \frac{1}{N} \sum_{i=1}^{N} \left|\hat{y}^{(i)} - y^{(i)}\right| \tag{1.9}$$

This loss function is used as an alternative to MSE in cases when the training data has a large number of outliers to mitigate MSE high sensitivity to outliers, which can dramatically affect the loss because the distance is squared.

- Huber loss: this function has the advantages of both MSE and MAE. If the absolute difference between the actual and predicted value is less than or equal to a threshold value, δ , then MSE is applied. Otherwise, if the error is sufficiently large, MAE is applied.

$$L_\delta = \begin{cases} \frac{1}{N}\sum_{i=1}^{N}\left(\hat{y}^{(i)} - y^{(i)}\right)^2 & if \left|(y^{(i)} - \hat{y}^{(i)})\right| \leq \delta \\ \frac{1}{N}\sum_{i=1}^{N}\delta\left|\hat{y}^{(i)} - y^{(i)} - \frac{1}{2}\delta\right| & if \left|(y^{(i)} - \hat{y}^{(i)})\right| > \delta \end{cases} \tag{1.10}$$

2. classification loss functions:
   - Binary Cross-Entropy/Log Loss: in binary classification, there are only two possible actual outputs,0 or 1. Thus, to accurately determine loss between the actual and predicted values, it needs to compare the actual value (0 or 1) with the probability that the input aligns with that category (p(i) = probability that the category is 1, and 1 — p(i) = probability that the category is 0)

$$BCE - loss = \frac{1}{N}\sum_{i=1}^{N} - \left(y^{(i)}\log(p^{(i)}) + (1 - y^{(i)})\log(1 - p^{(i)})\right) \tag{1.11}$$

   - Categorical Cross-Entropy Loss: in cases where the number of classes is greater than two, categorical cross-entropy loss function is used.

$$CCE - loss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y^{(i,j)}\log(p^{(i,j)}) \tag{1.12}$$

1.1.4 Optimizer

Optimizer is an algorithm that forms the basis on which a machine is able to learn through its experience. It computes gradients and attempts to minimize the loss function [Zaheer and Shaziya, 2019]. There are several ways learning is implemented with different kinds of optimization algorithms, which are described below.

- Stochastic Gradient Descent (SGD): The vanilla gradient descent trains the entire dataset together. Its variant is Stochastic Gradient Descent [Darken et al., 1992] that performs the training on the individual data element.
- Nesterov Momentum: Gradient is computed based on the approximate future positions of the parameters rather than the current parameters. Nesterov momemtum is an improvement over momentum which does not determine the future position of the parameters.

- Adagrad: This method chooses the learning rate based on the situation. Learning Rates are adaptive because the actual rate is determined from parameters. A high gradient for the parameters will have a reduced Learning Rate and parameters with small gradients will have increased learning rate.
- Adadelta: It is an extension of Adagrad. [Zeiler, 2012] presented a modification of Adagrad into Adadelta. Instead of accumulating the gradients, Adadelta makes use of some fixed size window and tracks only the available gradients within the window.
- RMSProp: RMSProp changes the Adagrad in a way how the gradient is accumulated. Gradients are accumulated into an exponentially weighted average. RMSProp discards the history and maintains only recent gradient information. [Mukkamala and Hein, 2017] discusses the RMSProp and its variants. They also examines Adagrad with logarithmic regret bounds.
- Adam: It has derived its name from "adaptive moments". It is a combination of RMSProp and Momentum. The update operation considers only the smooth version of the gradient and also includes a bias correction mechanism. [Kingma and Ba, 2014] discusses the Adam algorithm.

1.1.5 Batch normalization

The change in the distributions of internal nodes of a deep network, in the course of training, is referred to as Internal Covariate Shift. Eliminating it offers a promise of faster training. Batch Normalization takes a step towards reducing internal covariate shift, and in doing so dramatically accelerates the training of Deep Neural Networks. It accomplishes this via a normalization step that fixes the means and variances of layer inputs. Batch Normalization also has a beneficial effect on the gradient flow through the network, by reducing the dependence of gradients on the scale of the parameters or of their initial values. This allows the use of much higher learning rates without the risk of divergence. Furthermore, batch normalization regulates the model and reduces the need for Dropout [Srivastava et al., 2014] and help the Gradient Descent (GD) converge briskly towards the minima. Finally, Batch Normalization makes it possible to use saturating non-linearities by preventing the network from getting stuck in the saturated modes [Ioffe and Szegedy, 2015].

1.1.6 Forward propagation and Backpropagation

1. **Forward Propagation:** The Feed Forward Neural Network (FNN) was the first and arguably simplest type of Artificial Network devised. Forward propagation is where

input data is fed through a network in a forward direction to generate an output. The data is accepted by hidden layers and processed as per the activation function and moves to the successive layer. The forward flow of data is designed to avoid data moving in a circular motion which does not generate an output[Yadav et al., 2015].

2. **Back Propagation:** Among many other learning algorithms, "back-propagation algorithm" is the most popular and the mostly used one for the training of feed-forward neural networks. It is in essence, a means of updating networks synaptic weights [Sazli, 2006]. The back propagation algorithm uses supervised learning, which means that we provide the algorithm with examples of the inputs and outputs we want the network to compute, and then the error (difference between actual and predicted results) is calculated. The idea of the back propagation algorithm is to reduce this error, until the ANN learns the training data. The training begins with random weights, and the goal is to adjust them so that the error will be minimal [Dongare et al., 2012].

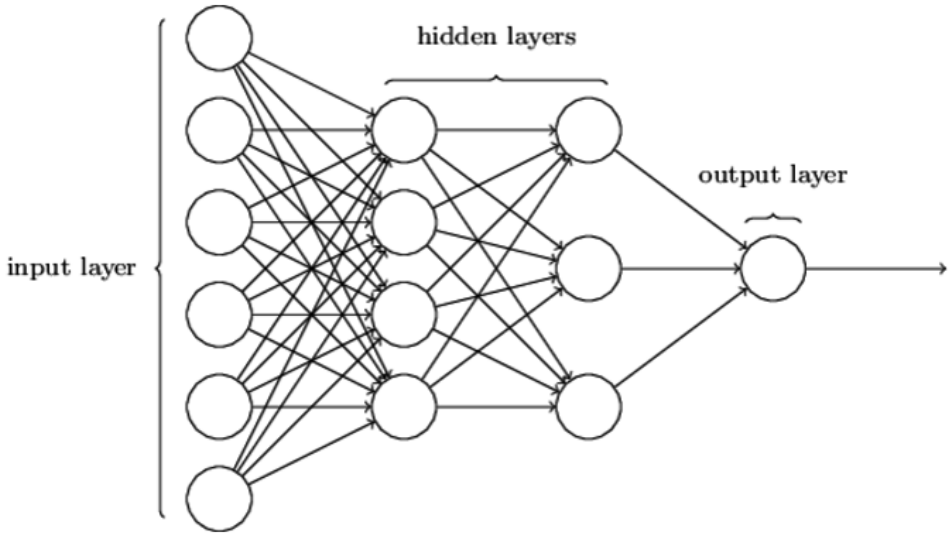


Figure 1.3: Backpropagation expletive illustration.

1.1.7 Evaluation Metrics

Various metrics are commonly employed to assess the effectiveness and performance of models. The following is a selection of widely utilized metrics for evaluating models that tackle the same issue:

- Mean Absolute Error (MAE): MAE measures the average magnitude of the errors in a set of predictions, without considering their direction. It's the average over the test sample of the absolute differences between prediction and actual observation

where all individual differences have equal weight.(see equation (1.9)).

- Root Mean Square Error (RMSE): RMSE is the square root of MSE. It is commonly used as a performance measure since it gives equal weights for both early and late predictions. The equation of the RMSE is given by:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\hat{y}^{(i)} - y^{(i)}\right)^2} \tag{1.13}$$

- Score: The scoring function penalizes late prediction more than early prediction, that is because late prediction usually leads to more severe consequences in many fields such as aerospace industries. The scoring function employed by the International Conference on Prognostic and Health Management Data Challenge is given by [Li et al., 2018]:

$$S = \sum_{i=1}^{N} s_{(i)}, \quad where \quad s_{(i)} = \begin{cases} e^{-\frac{\hat{y}^{(i)} - y^{(i)}}{13}} - 1 & for \quad \hat{y}^{(i)} - y^{(i)} < 0 \\ e^{-\frac{\hat{y}^{(i)} - y^{(i)}}{10}} - 1 & for \quad \hat{y}^{(i)} - y^{(i)} \geq 0 \end{cases} \tag{1.14}$$

- R-squared (R^2): This function measures the coefficient of determination that can take values in the range (-∞, 1] according to the mutual relation between the ground truth and the prediction model. It provides an indication of the goodness of fit and therefore a measure of how likely the invisible samples are to be predicted by the model [Chicco et al., 2021]. It is represented by:

$$R^2(y, \bar{y}) = 1 - \frac{\sum_{i=1}^{N}\left(y^{(i)} - \hat{y}^{(i)}\right)^2}{\sum_{i=1}^{N}\left(y^{(i)2} - \bar{y}\right)^2}, \quad where \quad \bar{y} = \frac{1}{N}\sum_{i=1}^{N} y^{(i)} \tag{1.15}$$

These metrics provide different perspectives on the performance of a regression model, and the choice of the most appropriate metric depends on the specific problem and requirements. Additionally, it is worth mentioning that these metrics are not exhaustive.

1.1.8 Hyper-parameters

They are parameters whose values are set before starting the model training process such as Gradient Descent, Learning Rate (LR), epochs, Batch-size. Choosing appropriate values for hyper-parameters to tune DNN is difficult because they affect different aspects of the network: structure, optimization process, data processing, learning process control, how well our model performs and how accurate it is, by directly influencing other parameters of the model such as weights [Lakhmiri et al., 2021].

1. **Gradient Descent:** GD is one of the most algorithms of optimizing Neural Networks. It is used to find the values of the parameters (coefficients) of a function that decrease

the loss function as much as possible. For this it needs to compute the objective function gradient by applying a first-order derivative with respect to the network parameters. Next, the parameter is updated in the reverse direction of the gradient to reduce the error [Haji and Abdulazeez, 2021]. There are three types of Gradient Descent learning algorithms:

- Batch Gradient Descent:

$$\theta = \theta - \eta.\nabla_\theta J(\theta) \tag{1.16}$$

Here, θ represents the model parameters, η is the learning rate, and $\nabla_\theta J(\theta)$ is the gradient of the loss function J.

- Stochastic Gradient Descent:

$$\theta = \theta - \eta.\nabla_\theta J\left(\theta; x^{(i)}; y^{(i)}\right) \tag{1.17}$$

Where $x^{(i)}$ and $y^{(i)}$ represent the input features and corresponding target labels of the i-th training example. The gradient $\nabla_\theta J\left(\theta; x^{(i)}; y^{(i)}\right)$ is computed by evaluating the loss function J on the single training sample ($x^{(i)}$,$y^{(i)}$).

- Mini-batch Gradient Descent

$$\theta = \theta - \eta.\nabla_\theta J\left(\theta; x^{(i:i+n)}; y^{(i:i+n)}\right) \tag{1.18}$$

Where $x^{(i:i+n)}$ and $y^{(i:i+n)}$ represent a mini-batch of training set, consisting of n consecutive samples starting from the i-th index. The gradient $\nabla_\theta J\left(\theta; x^{(i:i+n)}; y^{(i:i+n)}\right)$ is calculated by evaluating the loss function J on this mini-batch.

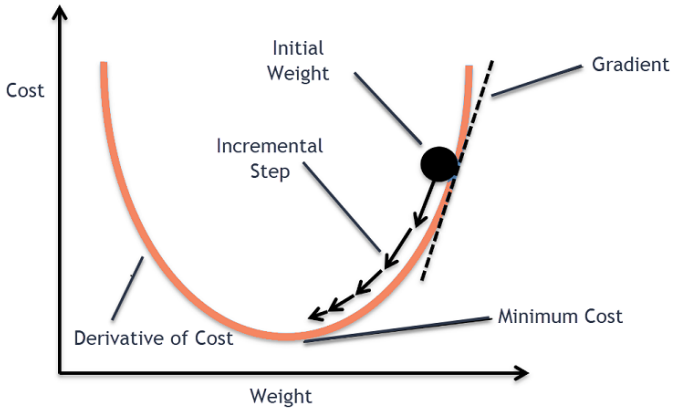


Figure 1.4: Gradient Descent Algorithm.

2. **Learning Rate:** The Learning Rate may be the most important hyper-parameter when configuring a Neural Network.The learning rate is multiplied by the gradients to determine the step size for weight updates Which means it controls the rate or speed at which the model learns, Choosing the Learning Rate is challenging as it

may have a significant impact on the convergence of the GD algorithms. There is a trade-off between rate of convergence and overshoot when determining the rate of learning, If the Learning Rate is too high, we can go beyond the threshold and start bouncing without meeting the minimum, On the other hand, if the LR is too poor, training may take a long time [Haji and Abdulazeez, 2021].

3. **Number of iterations or Epochs:** We can define an Epoch as the number of passages of a training data-set by an algorithm, one pass is equivalent to one round trip. An epoch is made up of an aggregation of "Batches" or "lots" of data and iterations where data-sets are generally broken down into batches, especially when the volume of data is massive because processing it in one epoch can be computationally challenging. The number of epochs can reach several thousand, because the procedure repeats itself until the error rate of the model is sufficiently reduced.

1.2 Basic Deep Learning Architectures

Deep learning architecture refers to the design and arrangement of deep neural networks, which are composed of multiple layers of interconnected nodes called neurons. These networks are designed to process data in a hierarchical manner, where each layer extracts increasingly abstract features from the input data. In the following, we highlight some of the most common deep learning architectures.

1.2.1 Convolutional Neural Network

CNN is one of the most popular models of deep neural networks inspired by the structure of the brain, just like neurons in the brain, and has been widely applied in a range of different fields, including computer vision, speech processing, face recognition, . . . etc. It is a special class of forward neural networks. The artificial neurons or nodes in CNNs have the ability to automatically learn features from the input data and get rid of manual extraction, so they take the input, process it and send the result as output [Patel and Patel, 2018]. This process uses a very small number of parameters and only small areas of the scene are processed by these cells rather than the entire scene, which simplifies the training process and speeds up the network [Alzubaidi et al., 2021].

The CNN architecture (figure 1.5) consists of a number of layers (or so-called multi-building blocks). Each layer in the CNN architecture, including its function, is described in detail below.

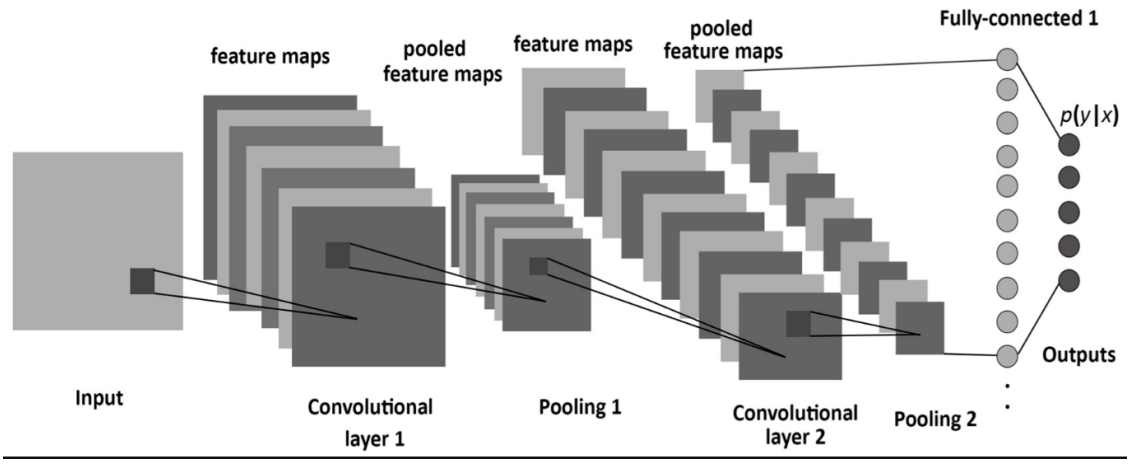


Figure 1.5: Architecture of the CNN.

- The Convolution layer is the initial layer in a neural network. Its purpose is to analyze input images and identify specific features. It performs a mathematical operation that involves taking an input image matrix and a filter or kernel. As a result, this layer produces a set of feature maps, which represent the detected features within the input images [Narejo et al., 2020].
- The Pooling layer, which is usually applied between two convolution layers. It takes the feature maps generated by the previous convolutional layer as input and reduces their size and dimensionality while preserving their most essential characteristics. Two commonly used pooling operations are max-pooling and average pooling. Max-pooling selects the largest element from the rectified feature map, while average pooling calculates the average value within the filter window. These pooling operations help reduce the size of the feature maps while retaining important information. The output of the pooling layer consists of the same number of feature maps as the input, but in a considerably compressed form.
- The activation layer, also known as the non-linearity layer, sets negative input values to zero. This layer introduces non-linearity to the model, enabling it to capture complex relationships and dependencies in the data [Narejo et al., 2020].
- The fully-connected layer, it is often located at the end of each CNN structure. Within this layer, each neuron is connected to all neurons in the previous layer, which is called the full connection (FC) approach. The inputs of the fully-connected layer come from the last pooling or CNN layer. This input is in the form of a vector, which is created from the feature maps after flattening. The output of the fully-connected layer represents the final CNN output [Alzubaidi et al., 2021].

1.2.2 LSTM

LSTM networks are a type of Recurrent Neural Networks (RNNs) that address the challenge of long-term dependencies. Introduced by Hochreiter and Schmidhuber in 1997 [Hochreiter and Schmidhuber, 1997], LSTMs effectively handle vanishing and exploding gradients in RNNs. They have the inherent ability to retain information over long sequences, making them well-suited for processing, predicting, and classifying time-series data. LSTM networks consist of an input layer, memory cells, and an output layer. The key feature of LSTM networks lies in the hidden layer, which contains memory cells [Hu et al., 2018]. Each memory cell is equipped with three gates that regulate and modify its cell state:

- The forget gate defines what information is removed from the cell state.
- The input gate specifies what information is added to the cell state
- The output gate specifies what information from the cell state is used.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{1.19}$$
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{1.20}$$
$$\tilde{c}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \tag{1.21}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \tag{1.22}$$
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{1.23}$$
$$h_t = o_t \odot \tanh(c_t) \tag{1.24}$$

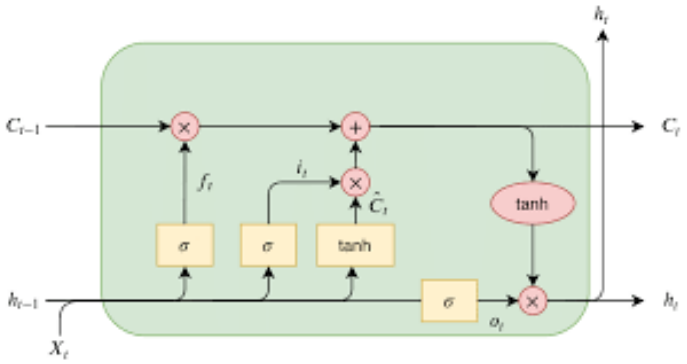


Figure 1.6: The architecture of LSTM memory block.

1.2.3 GRU

A Gated Recurrent Unit is a type of gating mechanism used in RNN. It shares similarities with LSTM but offers faster computation due to its simpler internal structure.

GRU is easier to train compared to LSTM as it involves fewer computations [Haji and Abdulazeez, 2021]. The LSTM's three gates are replaced by two: the reset gate and the update gate.The update gate regulates the flow of information into the memory, while the reset gate controls the flow of information out of the memory. These gates determine which information is retained and passed on to the output. Through training, the update gate can preserve relevant past information, while the reset gate can discard irrelevant information, enhancing the model's predictive capabilities.

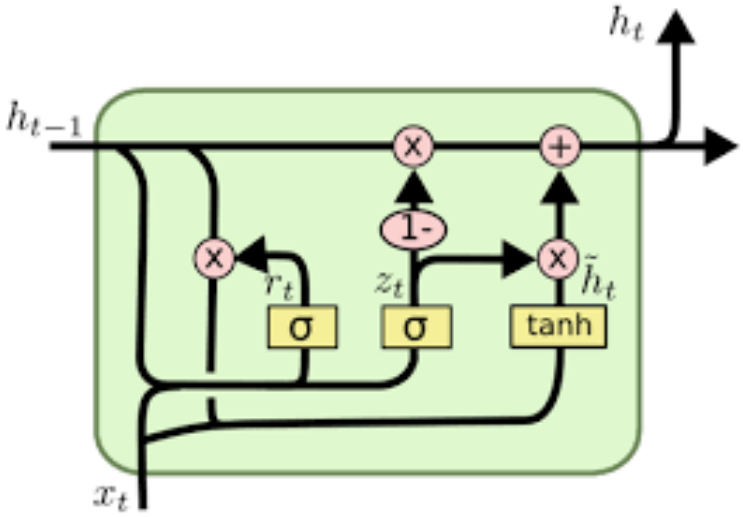


Figure 1.7: The architecture of Gated Recurrent Unit.

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t]) \tag{1.25}$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t]) \tag{1.26}$$

$$\tilde{h}_t = \tanh(W_h \cdot [r_t \odot h_{t-1}, x_t]) \tag{1.27}$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \tag{1.28}$$

1.3 Deep Learning Frameworks

Deep learning frameworks are software libraries that provide a high-level interface and tools for building and training deep neural networks. These frameworks offer pre-defined functions and optimized implementations of complex operations, making it easier to design, train, and deploy deep learning models. Below are some popular deep learning frameworks:

1.3.1 TensorFlow

TensorFlow is an open source software library developed by researchers and engineers

from the Google Brain team within the Google artificial intelligence organization. It comes with strong support for machine learning and deep learning, and its flexible numerical computation core is used across many other scientific fields. It is a numerical library for dataflow programming that provides the basis for deep learning research and development. TensorFlow programming interfaces include APIs for Python and C++, and developments for Java. [Nguyen et al., 2019].

1.3.2 Keras

Keras is an open-source deep-learning framework for Python that provides a convenient way to define and train almost any kind of deep-learning model. This tool can work on TensorFlow, Theano, Microsoft Cognitive Toolkit, and PlaidML. The USP (Unique Selling Proposition) of Keras is its speed; it comes with built-in support for data parallelism, and thus, it can handle massive volumes of data while speeding up training time for models. Since it's written in Python, it's incredibly easy to use and extensible. Additionally, it is excellent for beginners who are just starting their journey in this field, and it allows the same code to run seamlessly on CPU or GPU [Chollet, 2021].

Apart from the mentioned frameworks, there are several other popular options available for deep learning, such as PyTorch, Theano, and Caffe.

Conclusion

In this chapter, we explored the general concept of Deep Learning and its fundamental principles. Additionally, we covered some widely employed approaches in deep learning, namely Convolutional Neural Networks (CNN), Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU). These approaches are extensively utilized in various domains and have proven to be successful in addressing complex problems.

In the next chapter, we will go through the concept of maintenance, focusing on the notion of Remaining Useful Life (RUL). We will explore several approaches employed for estimating RUL, including deep-learning-based approaches.

Chapter 2

An Overview On Remaining Useful Life

Introduction

Maintenance and prognostics are essential in various industries, including aerospace, manufacturing, automotive, and heavy industries with high-risk engineering systems. An unexpected malfunction can lead to significant financial loss or even human casualties. Therefore, it is crucial to accurately evaluate the equipment's operational condition and estimate its Remaining Useful Life. This information can assist operators in making critical decisions, enabling timely maintenance before any irreversible damage occurs[Chen et al., 2018; Yu et al., 2019].

Prognostic and health management technologies, also known as Condition-Based Maintenance, are showing promise for high-risk industries. Remaining useful life estimation of engineering systems is the most important task within the field of PHM [Wang et al., 2008; Khelif et al., 2014], using physics-based, data-driven, or hybrid methods. Data-driven methods are proving to be the most reliable for RUL estimation as they are easier to implement and rely on routinely collected sensor data.

This chapter first discusses the concept of maintenance and its various types, followed by the concept of "remaining useful life" and its modeling methods.

2.1 Maintenance Definition

Maintenance corresponds to all the technical, administrative and management actions to be carried out to maintain an efficient information technology infrastructure. Whether you are an individual or a professional, it is important to regularly check the status of your devices. Maintenance meets several objectives: preserve the performance of hardware and software, update the computer system, maintain and correct any problems with the computer equipment and restore it to a state in which it can perform the required function

[Yende, 2018]. Maintenance tasks vary by work environment which includes, for example, visual inspection, test, measurement, change of consumables (greasing, lubrication, oil filters), adjustment, repair, tightening of bolts, cleaning, fault detection, etc [Mosallam, 2014].

2.2 Maintenance Types

In the context of maintenance, two main types can be distinguished: corrective maintenance and preventive maintenance, depending on the moment of intervention in relation to the failure. The first type of maintenance is applied after the occurrence of the failure, while the second type applies before the latter [Nihal, 2022].

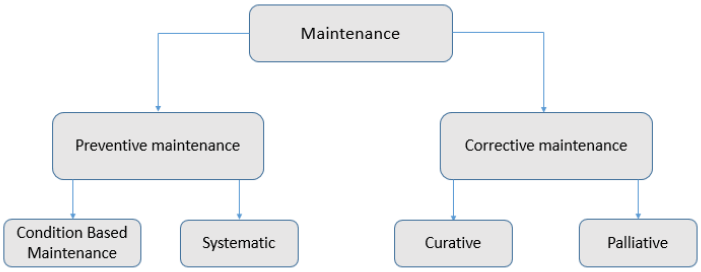


Figure 2.1: Predictive maintenance.

2.2.1 Corrective maintenance

Corrective maintenance corresponds to an operation undertaken following a breakdown, a disaster, or a hazard. Its objective is to restore the good working order of the machines by troubleshooting, repairing, or replacing defective parts[Amira et al., 2019]. This maintenance includes two types:

- Palliative maintenance: avoids a complete stoppage of production by temporarily returning the machine or tool to its original condition pending repair or replacement.
- Curative maintenance: This type of maintenance fixes the causes and consequences of a breakdown. This is an in-depth procedure that works in the long run, often by replacing the defective part with a new one. Then, the equipment will resume normal production.

2.2.2 Preventive maintenance

This type of maintenance is carried out at predetermined intervals or according to prescribed criteria and intended to reduce the probability of failure or the degradation

of the functioning of an asset. Preventive maintenance is therefore, as its name suggests, prevents total failure through the regular maintenance of a machine or asset. It consists of carrying out check-ups according to established standards or in accordance with the manufacturer's instructions, but also according to human experience and the history of interventions carried out on the asset [Hao et al., 2010].

1. **Systematic maintenance:** When preventive maintenance is performed at predetermined intervals, we are talking about systematic maintenance, the maintenance operation is carried out in accordance with a timetable and no intervention takes place before the predetermined deadline.Optimization of systematic preventive maintenance involves determining the most appropriate frequency for performing maintenance operations based on various factors such as time, number of operating cycles, number of parts produced, and other relevant parameters.[Deloux, 2008].
2. **Condition Based Maintenance:** CBM is a maintenance strategy that monitors the actual condition of an asset to decide what maintenance needs to be done. CBM dictates that maintenance should only be performed when certain indicators show signs of decreasing performance or upcoming failure. The heart of CBM is the Condition Monitoring (CM) and evolution of significant parameters of the degradation or drop in performance of an entity. The significant degradation parameters can either be measurements of the physical characteristics of the system (thickness of a material, degree of erosion, temperature, pressure, . . .), or information about the Remaining useful life (this is called predictive maintenance) [Deloux, 2008].
   Predictive maintenance is a subset of Condition-Based Maintenance that uses advanced technologies, data analysis, and predictive algorithms to forecast equipment failures and decide which maintenance activity to perform. Predictive analysis relies on data collected from measuring devices/sensors connected to machines and tools, such as vibration data, thermal images, ultrasound data, etc. The predictive model processes the information through predictive algorithms, detects trends, and determines when equipment will need to be repaired or retired Instead of operating a part of the equipment or a component in a state of failure or replacing them when they still have a useful life, This would help companies to improve their strategies by performing maintenance activities only when absolutely necessary [Ran et al., 2019].

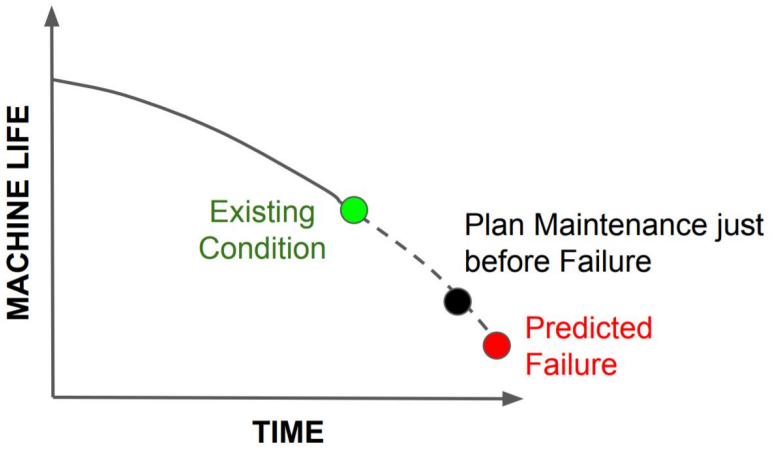


Figure 2.2: Predictive maintenance.

2.3 Remaining Useful Life of an equipment

The Remaining Useful Life (RUL) is a subjective estimate of the remaining time that an item, component, or system can function according to its intended purpose before requiring maintenance or replacement. RUL is calculated by measuring the time between the current moment, after detecting degradation, and when the degradation reaches the failure threshold.

The remaining useful life is estimated by observing similar items, components, or systems, or by using average estimates or a combination thereof. For example, if a PVC membrane roof was installed about seven years ago and poorly maintained, its remaining useful life might be around ten years.

The Remaining Useful Life of building components and systems is also known as Property Condition Assessment. Accurately predicting RUL allows maintenance to be planned in advance, reducing costs and time by avoiding unnecessary maintenance. Figure 2.3 illustrates the general concept of RUL by showing the evolution of the deterioration of a given system. In this case, RUL is the time between the current time, corresponding to condition A, and the time when a maximum acceptable condition of deterioration B is reached.

The maximum acceptable condition of deterioration B can correspond to the moment of failure or to a predefined threshold in a lower value. Therefore, useful life can be defined from the beginning of life to failure or until the time of condition B. Proper RUL prediction is useful in carrying out maintenance from condition A to B, saving resources and improving processes [Obando, 2018].

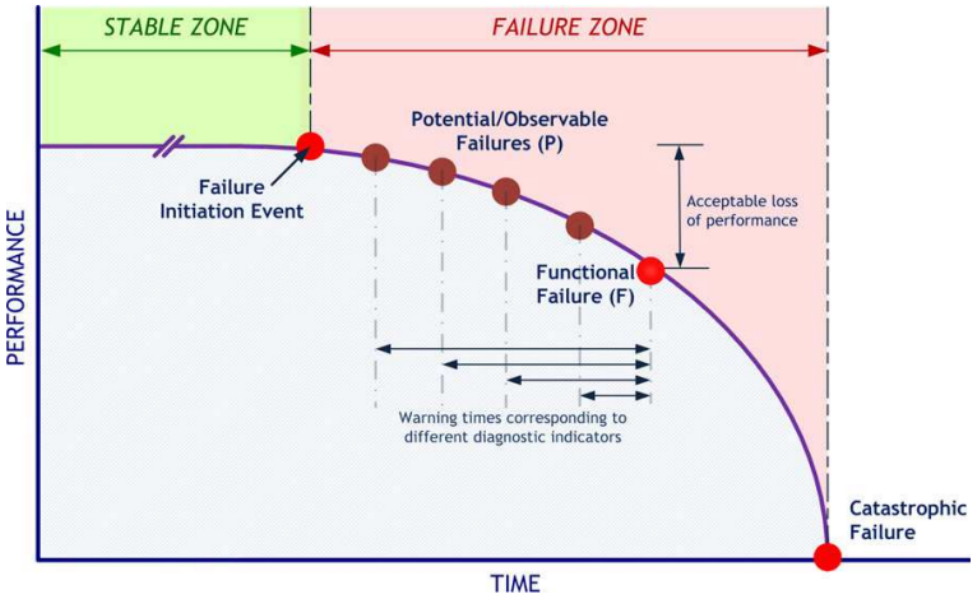


Figure 2.3: The concept of remaining useful life (RUL) [Sikorska et al., 2011].

2.4 Modeling Methods of RUL Estimation

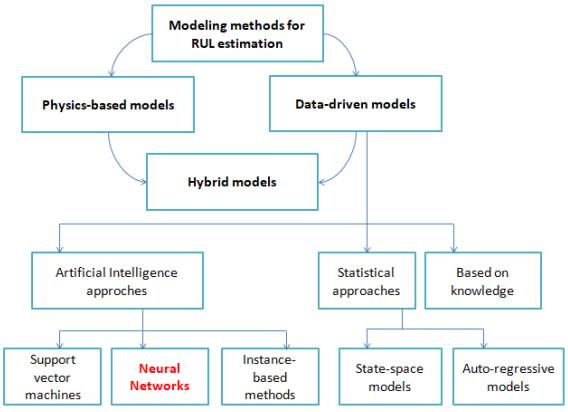


Figure 2.4: Modeling methods of RUL estimation.

2.4.1 Physics-based models

Failure physics-based approaches or physics-based models [Cubillo et al., 2016; Liu, 2020] create analytical models that are directly linked to the physical processes that affect the health of a component. These approaches require knowledge of physical laws such as mechanics, chemistry, electricity, and hydraulics. The physical model is usually described using dynamical systems such as differential equations, non-linear equations, state representations, etc.

The principle of prognosis based on a physical model can be seen in figure (2.5). In this approach, the behavior of the system is represented by an analytical model. A coherence

test is carried out by comparing the sensor data from the real system with the outputs of the analytical model. The residues generated during this test are then evaluated. If there is a malfunction, the residuals will exceed a fault detection threshold.

Projections of the future behavior and degradation of the system are used to estimate the remaining time before failure. Physical models are used to address problems such as material wear, crack growth, fatigue-related breakages, or corrosion.

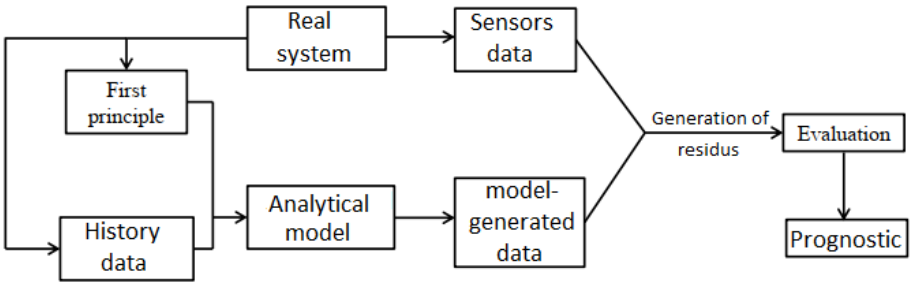


Figure 2.5: Flow chart of physics-based models approach

2.4.2 Data-driven models

Data-driven approaches , data-driven [Khelif et al., 2016; Wang et al., 2019] seek to extract from a history of monitoring data models of the evolution of the functioning of the monitored system, going as far as its degradation. These approaches have two phases. An offline phase dedicated to understanding and learning the behavior of degradation, and an online phase that estimates the current state of the system and predicts its functioning time before failure. Data-driven models can be classified into two categories: statistical approaches and artificial intelligence based approaches.

1. **Statistical approaches:**

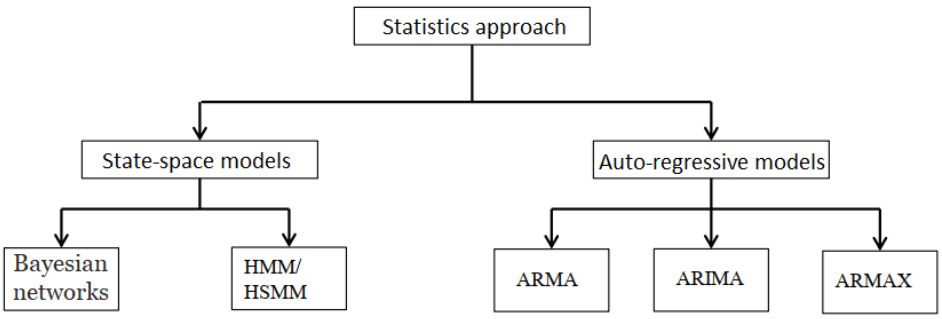


Figure 2.6: Statistical approaches classification.

- **State-space models:** State-space models are a powerful framework for modeling and predicting time series data, including the prediction of RUL, they combine two key components: a dynamic system model and an observation model.The dynamic system model describes the underlying behavior of the system over time.

It typically consists of a state transition equation that captures the evolution of the system's latent or hidden states, and a process noise term that accounts for the uncertainties in the state evolution. Two of the commonly used techniques for stat-space models are:

Hidden Markov Model (HMM): HMM is a statistical model for sequential data, with hidden states representing unobservable sequences and observations related to states through probability distributions. HMMs are used in speech and handwriting recognition [Khorasgani et al., 2016] [Zhao et al., 2017] and degradation modeling, predicting the Remaining Useful Life with meaningful states [Dong and He, 2007; Son et al., 2016; Alaswad and Xiang, 2017].

Bayesian networks: Bayesian networks (BN) are probabilistic graphical models that allow for integrating historical data and expert knowledge to overcome the uncertain, incomplete, and even conflicting properties of the data [Nguyen et al., 2023].Bayesian networks are used in aircraft brake wear prediction, RUL estimation, modeling degradation and failures in complex systems, and capturing uncertainty in instrument systems like helicopter stability augmentation.

- **Auto-regressive models:**

auto-regressive moving average models (ARMA), integrated auto-regressive moving average models (ARIMA) and Auto-Regressive Moving Average with Exogenous (ARMAX) are the reference methods in modelling temporal series . The ARMA and ARMAX models are only used for stationary data. They are improved by applying an integration operation in the ARIMA model. Regressive models do not require a degradation history and must be evaluated recursively until a certain threshold is reached, which generates an accumulation of systematic error and deteriorates the performance of the predictor.

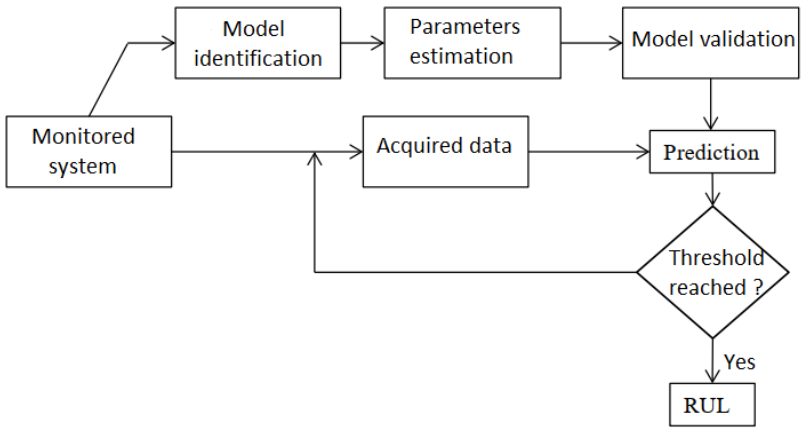


Figure 2.7: Flowchart of auto-regressive approaches for RUL prediction.

2. **Artificial intelligence or machine learning:**

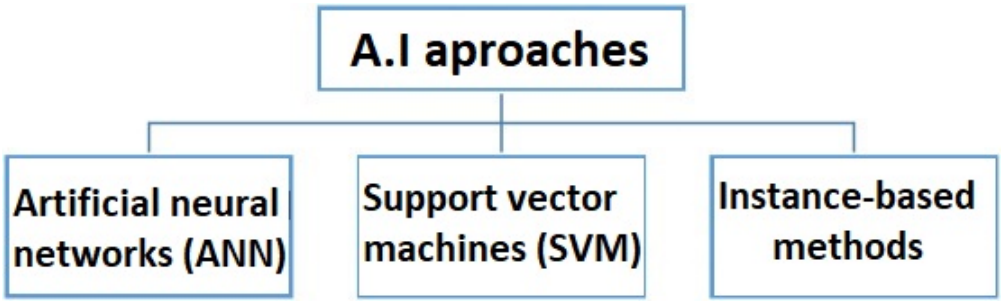


Figure 2.8: Machine learning approaches.

- **Neural Networks (NN):** Machine learning methods have been studied for RUL prediction, NN received a lot of attention given their ability to approximate functions directly from raw data [Li et al., 2018]. In fact, ANNs are known for their ability to model complex, multidimensional, and nonlinear systems with no need for a physical understanding of system behavior. An ANN is a nonlinear approximation function with multiple inputs and outputs and can be used for different purposes including prognosis. According to the inputs of the ANN model, the latter has the flexibility to perform different tasks, such as the prediction of the time remaining before failure for example [Khelif, 2015]. Recently, Deep NN methods have been proposed to predict problems containing large amounts of temporal input data [da Costa et al., 2020].
- **Support Vector Machines:** Support vector machines (SVM) were developed by Vapnick and his colleagues. SVMs are the basis of a learning system that represents a space of input features in a higher-dimensional space. This is called the kernel trick and it gives SVMs the ability to process nonlinear data. These methods have a good generalization and have two types of application: support vector classification, used to solve diagnostic problems, and support vector regression, used to solve the prognostic problems.
- **Instance-based methods:** Instance-based approaches are based on a data-set, on which learning techniques are applied. They are based on the principle: "the experiences acquired during the resolution of a problem can be used to solve similar cases". An algorithm often used for this type of approach is the k-nearest neighbors. An instance is generally represented by a vector in an n-dimensional Euclidean space and the objective is to find similar instances. The advantage of these methods is incremental learning. When a problem is successfully solved, the experience is kept in order to solve new similar problems. Instance-based methods are part of case-based reasoning (RaPC) which is a paradigm of reasoning by analogy. New problems are solved based on specific knowledge gained from

solving old problems. RaPC is based on the philosophy: "Similar cases produce similar results".

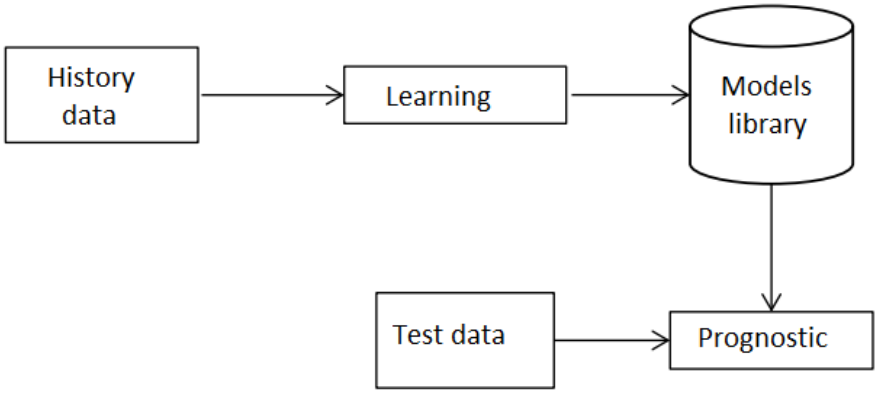


Figure 2.9: Machine learning approaches workflow.

2.4.3 Hybrid models

By combining model-based and data-driven approaches, hybrid approaches aim to leverage the advantages of both and mitigate their drawbacks [Baraldi et al., 2013]. For example, the recursive Bayesian estimator is an important approach for integrating system models with sensor data, enabling so-called hybrid models. Recursive Bayesian estimators typically involve two common procedures at each iteration: prediction and update. Two significant variants of the Bayesian estimator are the Kalman filter and the particle filter. The Kalman filter is suitable for linear state-space models with Gaussian noise, while the particle filter can be used for nonlinear models with non-Gaussian noise [Khelif, 2015].

2.5 Deep Learning Techniques For RUL Estimation-Related Works-

Several Deep Learning techniques have been proposed and have shown promising results in RUL estimation, and further research in this area is going to improve the performance and efficiency of these models. The following are some related works in which the CMAPSS data-set was used:

- [Sateesh Babu et al., 2016] proposed a CNN based regression model for estimating the Remaining Useful Life. The architecture of the proposed CNN-based regression model consists of several convolutional layers followed by fully connected layers. The authors used a 1D CNN architecture that takes the time-series input data and

extracts relevant features from it. The extracted features are then fed into fully connected layers that predict the RUL.

- [Li et al., 2018] suggested a Deep Convolutional Neural Network (DCNN) for predicting the Remaining Useful Life (RUL). The authors argued that their model has the potential to improve RUL estimation in prognostics, especially when dealing with complex and high-dimensional data. The proposed model consists of an input layer, four stacked convolutional layers with the same configuration to extract features, and another convolutional layer with one filter to merge the previous feature maps into a unique one. The feature map is then flattened and connected to a dense layer, followed by one neuron for RUL estimation.
- [Ayodeji et al., 2021] investigated the potential benefits of incorporating soft Self-Attention Mechanisms into a multi-head model for analyzing the CMAPSS dataset subsets FD001 and FD003. The proposed model was a fully connected neural network (FNN) which was evaluated in comparison to other models such as simple recurrent neural network (SRNN), GRU, LSTM, CNN and other variants of those models, before incorporating Self-Attention and observing its effect on each model.
- [Azyus et al., 2022] experimented with the GRU architecture by building three GRU architectures and testing them on the CMAPSS dataset. The GRU architectures have different numbers of GRU layers expending from two to four layers after the input layer, followed by a Fully Connected layer and an output layer. ([Azyus et al., 2022]) also stated some of the architecture configurations such as a batch size of 512, a learning rate with Adam optimizer of 0.001, and a total of 200 epochs.

Conclusion

This chapter reviewed two main parts: the first part explains what maintenance is, focusing on predictive and Condition-Based Maintenance, while detailing the types of each and citing its convenience. The second part discussed the concept of Remaining Useful Life (RUL) passing through several approaches used for its estimation along with some similar ideas and related works.

In the next chapter, we will explain our proposed models, the steps we followed to build them, as well as display and discuss the obtained results.

Chapter 3

Contribution: Development of a Hybrid Deep Learning Model for RUL Estimation

Introduction

In this chapter, we will address the task of estimating the RUL for airplane turbofan engines using a data-driven approach, specifically Deep Learning. First, we will outline the common steps used for building a model, and then we will briefly introduce the C-MAPSS dataset used for this purpose. To enhance the predictive performance of the proposed models, two techniques will be evaluated, specifically the Attention Mechanism and Multi-head architecture. Finally, we will discuss the results obtained and compare them to some of the state-of-the-art methods mentioned in the previous chapter.

3.1 Methodology: Common Steps for Building a Model for a Regression Problem

Developing a Deep Neural Network for a regression problem [1] is an iterative process. The following stages must be revisited to fine-tune both the model parameters and its architecture to enhance its performance.

- **Loading the Dataset:** First, we will load the C-MAPSS dataset. It is divided into four subsets (FD001, FD002, FD003, and FD004), with each subset containing data for training, testing, and RUL. All subsets have the same number of features (25 columns) but differ in the number of samples collected under various operating conditions and fault modes, as illustrated in Table 3.2. For more details about the dataset, please refer to Section 3.3.

[1] Remaining Useful Life (RUL) can be defined as a regression problem since it provides a numerical value (the output) representing the number of cycles the engine will last in service before it fails.

- **Data Normalization:** Data normalization is an important step that directly impacts the performance and convergence speed of Deep Learning models. Its purpose is to standardize all input features to a common scale without distorting the differences in the value ranges.
- **Neural Network Architecture Design:** Neural Network architecture refers to the structure and organization of Artificial Neural Networks. The process of selecting and organizing the layers connections and parameters of the network to effectively address a given problem requires careful consideration of factors such as the nature of the input data, task complexity, and available computational resources. When constructing a Neural NetworK, it is essential to identify the input and output layers, any hidden layers, the number of neurons, and the activation functions. To ensure an effective architecture, hyperparameters such as the Learning Rate, number of Epochs, validation split, and Batch size must be adjusted to improve results.
- **Training the Model:** The objective of training the model is to learn the relationship between the training data and labels and then store this knowledge. The compilation step involves configuring the model with specific settings before training can commence. This includes defining the optimizer, loss function, and evaluation metrics used during training to assess the model's performance (as discussed in Chapter 1). During the training phase, the provided input sequences are divided into a training set and a validation set. The training set undergoes a series of computations to generate predictions over a predefined number of Epochs. After each epoch, the model's performance is evaluated on the validation set to identify potential issues like overfitting or convergence problems. Once the model is compiled and trained, it becomes important to save it for future use or deployment. Saving the trained model entails storing its architecture, trained weights, optimizer state, and any additional configurations. This allows the model to be loaded and used on the testing set without the need for re-training.
- **Model Evaluation and Tuning:** By adjusting hyperparameters such as the Learning Rate and Batch size, the model's performance can be enhanced to achieve improved accuracy or other desired metrics. Tuning helps identify the optimal configuration that maximizes the model's predictive capability while avoiding issues of underfitting and overfitting. Once the model has been tuned, it should be evaluated to verify that the improvements made during tuning have indeed resulted in enhanced performance. Model evaluation allows us to compare different regression models (or algorithms targeting the same problem), enabling us to determine which one offers the best solution for that problem.
- **Predicting New Data:** Once the model is trained and validated, it becomes capable

of predicting labels for new, unseen data (test data), which differs from the data used during the training phase. The accuracy of these predictions on the test data relies heavily on the generalization ability of the trained model.

3.2 Motivation

Hybrid models, which combine multiple types of Deep Learning architectures, have gained significant attention and popularity due to their ability to leverage the strengths of different types of Neural Networks and overcome their individual limitations, leading to more reliable and robust predictions. This can be particularly useful in complex and uncertain domains where a single model may struggle to capture the full complexity of the data. In recent years, there has been growing interest in utilizing two promising architectural advancements, namely Multi-head models and Attention Mechanisms, to enhance the predictive capabilities of Deep Learning models. Unlike traditional approaches, the Multi-head approach employs independent "heads" to process sensor data individually. This approach offers notable benefits such as improved feature extraction. Additionally, Attention Mechanisms enable Deep Learning models to automatically identify and prioritize important features while suppressing less influential ones. By incorporating Attention Mechanisms, information processing in DL models is likely to become more efficient and accurate, leading to a potential improvement in model performance. Despite the significance of these advancements, comprehensive investigations into the individual effects of these mechanisms on Deep Learning models, particularly in the context of Remaining Useful Life predictions, remain limited. On the other hand, to the best of our current knowledge, only a few studies have employed the CNN-GRU architecture for RUL prediction. The dominant choice in the literature has been the CNN-LSTM hybrid model. Moreover, not all research papers provide comprehensive architectural details for their proposed models. Motivated by the aforementioned considerations, we propose two models: a CNN-GRU model and a straightforward FNN model. Subsequently, we analyze the impact of the previously mentioned techniques on their performance.

3.3 Data Description - C-MAPSS Dataset -

The dataset used in this study was generated with the Commercial Modular Aero-Propulsion System Simulation (C-MAPSS). It provides simulated run-to-failure trajectories of a small fleet of large turbofan engines, with the components depicted in Figure 3.1.

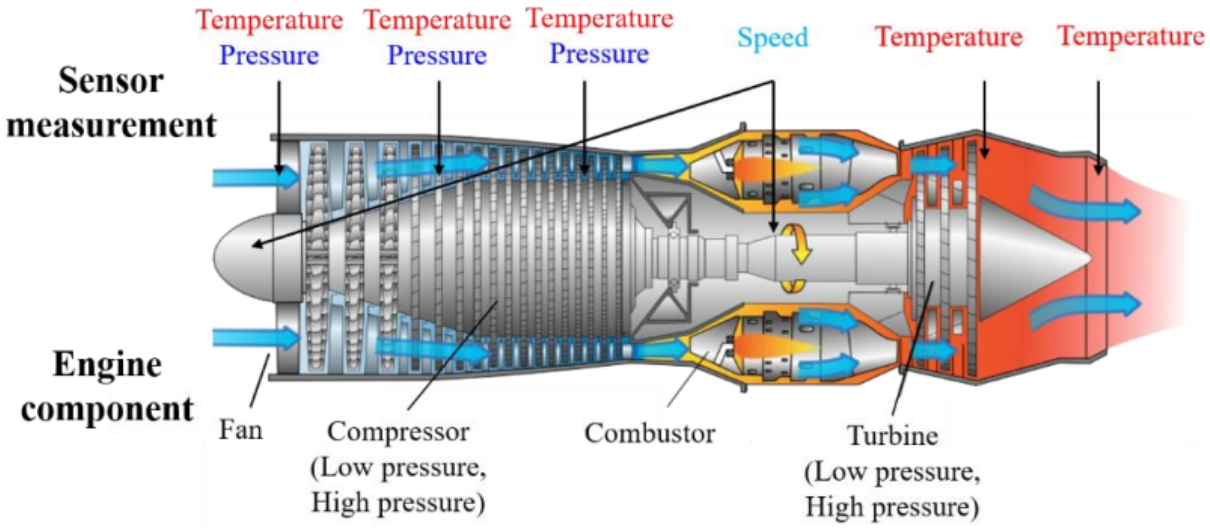


Figure 3.1: Diagram of turbofan engine modules [Hong et al., 2020].

The C-MAPSS dataset comprises data collected from twenty-one sensors that measure engine status and information from three operating settings that significantly impact engine performance. It is important to note that the data is affected by sensor noise. Table 3.1 displays the list and physical meanings of C-MAPSS dataset parameters and margins that were used for health index calculation only and were not available explicitly. It should be noted that we treated the previous sensors as a series of numbers from 1 to 21 without going into their specific details.

As mentioned earlier, the C-MAPSS dataset consists of four subsets (see Table 3.2) configured with variations in the number of engines, operating conditions, and failure types. Each subset comprises training data, test data, and its respective Remaining Useful Life. In practical scenarios, domain experts and decision-makers can utilize sensor data to monitor the condition of turbofan engines, detect and address malfunctions, and, most importantly, predict performance degradation and RUL.

3.4 Data Pre-processing

Data pre-processing is a crucial step in building a model and has a significant impact on its performance. We start by reading the files containing our data. As previously mentioned, each subset of the C-MAPSS dataset comprises three files: "train" for training, "test" for testing, and "RUL," which contains the corresponding labels for the test data. Subsequently, we calculate the RUL values for each train and test subset. Two methods

Sensor Number	Symbol	Description	Unit
Parameters available to participants as sensor data			
1	T2	Total temperature at fan inlet	°R
2	T24	Total temperature at LPC outlet	°R
3	T30	Total temperature at HPC outlet	°R
4	T50	Total temperature at LPT outlet	°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 (P50/P2)	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_dmd	Demanded fan speed	rpm
19	PCNfR_dmd	Demanded corrected fan speed	rpm
20	W31	HPT coolant bleed	lbm/s
21	W32	LPT coolant bleed	lbm/s
Parameters for calculating the Health Index			
1	T48 (EGT)	Total temperature at HPT outlet	°R
2	SmFan	Fan stall margin	–
3	SmLPC	LPC stall margin	–
4	SmHPC	HPC stall margin	–

Table 3.1: C-MAPSS outputs to measure system response. [Saxena et al., 2008].

Subset ID	FD001	FD002	FD003	FD004
Number of engines	100	260	100	249
Number of training samples	20,631	53,579	24,270	61,249
Number of test samples	100	259	100	248
Number of the data column	26	26	26	26
Average life span (cycles)	206	206	247	245
Operating conditions	1	6	1	6
Fault conditions	1	1	2	2

Table 3.2: Summary of C-MAPSS Dataset Subset [Saxena et al., 2008].

can be employed for this task (see Figure 3.2)[2]:

- Linear Degradation: in this method, we assume that the degradation state begins with the first cycle. For example, in the case of the first engine of the FD001 train, which has 192 cycles, this implies that the degradation state starts with the first cycle, leaving a total of 191 cycles remaining until failure.
- Piecewise Linear Degradation: this method proposes assigning a fixed number, referred to as 'early-RUL', to the RUL so that the degradation state does not commence from the first cycle but rather starts when RUL values reach 'early-RUL'.

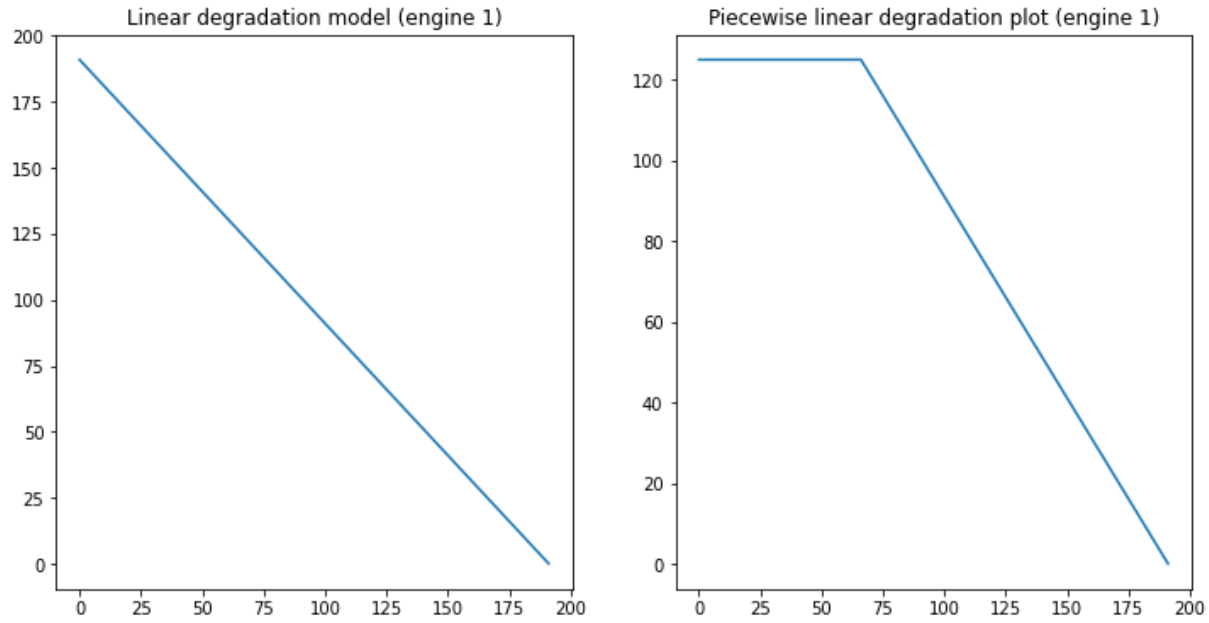


Figure 3.2: Visual Comparison of Linear Degradation vs. Piecewise Linear Degradation.

The final step is data normalization, a fundamental pre-processing technique employed to standardize and scale data before inputting it into the model. As previously mentioned (see Section 3.1), the objective of data normalization is to ensure that all features share a consistent scale and distribution, ultimately enhancing model performance and convergence. Various methods are commonly used for data normalization, with Min-Max Scaling being one of them, which we have adopted. This technique scales the data to a fixed range, typically between 0 and 1 [Eesa and Arabo, 2017]. It computes the minimum and maximum values within the dataset and scales each data point accordingly, as indicated in Equation 3.1.

$$X_{\text{scaled}} = \frac{X - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}} \tag{3.1}$$

[2] https://github.com/biswajitsahoo1111/rul_codes_open/blob/master/notebooks/cmapss_notebooks/CMAPSS_data_description_and_preprocessing.ipynb

Where X represents the original data, and X_{scaled} represents the normalized value. X_{max} and X_{min} respectively denote the maximum and minimum values of the original sensor measurements [Li et al., 2018].

3.5 Contributions

3.5.1 The first proposed model: CNN-GRU hybrid

The first proposed model for RUL prediction is a CNN-GRU hybrid model. The idea behind this combination is to employ CNN ability to automatically extract features from the input data, and GRU ability to capture long-term dependencies in sequential data. While GRU and LSTM are both Recurrent Neural Network architectures known for handling long-term dependencies in sequential data, GRU has a simpler structure and faster computations, making it easier to train and more computationally efficient than LSTM. GRU's simplicity leads to faster inference, lower memory requirements, and reduced risk of over-fitting. For these reasons, we chose to use GRU instead of LSTM.
The architecture of the model is sequential (i.e the layers are arranged sequentially, one after another). It consists of three CNN layers with the identical configurations for feature extraction, followed by a Dropout layer to prevent over-fitting. Then two GRU layers are added to handle long-term dependencies, followed by another Dropout layer, and final, a Fully-Connected layer. A detailed description of this architecture is presented in Table 3.3. The results are displayed in Table 3.6.

3.5.2 The second proposed model: FNN model

The second model we proposed in this work is the FNN model. The fundamental architecture of FNNs consists of multiple layers of interconnected neurons in which information flows in a forward direction from the input layer through the hidden layers to the output layer. In the FNN sequential model, we used two dense layers that are responsible for transforming the input data by applying a linear transformation, followed by a nonlinear activation function. Each neuron in the dense layer takes the output from all neurons in the previous layer as input and produces an output value. To mitigate the risk of overfitting, we added a dropout layer after the first dense layer. This description is better presented in Table 3.4. The results are displayed in Table (3.6).

It is important to note that during the pre-processing phase of both models construction, the linear degradation method was employed.

Hyperparameter	Configurations			
	FD001	FD002	FD003	FD004
CNN1	Filters = 64, kernels = 4			
Leaky-ReLU	$\alpha = 0.001$			
MaxPooling	Pool size = 2			
CNN2	Filters = 64, kernels = 4, activation = ReLU			
Leaky-ReLU	$\alpha = 0.001$			
CNN3	Filters = 64, kernels = 4, activation = ReLU			
Leaky-ReLU	$\alpha = 0.001$			
Dropout	Rate = 0.2			
GRU1	Units = 128, Activation = LeakyReLU($\alpha = 0.001$)			
GRU2	Units = 80, Activation = LeakyReLU($\alpha = 0.001$)			
Dropout	Rate = 0.2			
Flatten	/			
Dense	Units = 1, Activation = Linear			
Optimizer	Adam(learning rate = 3×10^{-4})			
Seq-length	90			
Epochs	100			
Batch size	512			
Split-val	0.05			

Table 3.3: Hyper-parameters and configurations of the CNN-GRU hybrid model.

Hyperparameter	Configurations			
	FD001	FD002	FD003	FD004
Dense1	units = 128, activation=relu			
Drop-out	0.3			
Flatten	/			
Dense2	units=120, activation=relu			
Dense	Units = 1, Activation = Linear			
Batch size	200			
Optimizer	Adam(lr=3e-3)			
Seq-length	100			
Split validation	0.2			
Epochs	50			

Table 3.4: Hyper-parameters and configuration of FNN model.

Regarding the evaluation metrics used to assess the models' performance, we selected MAE (Equation 1.9), RMSE (Equation 1.13) and r-squared (Equation 1.15).

3.5.3 The Effect of the Attention Mechanism

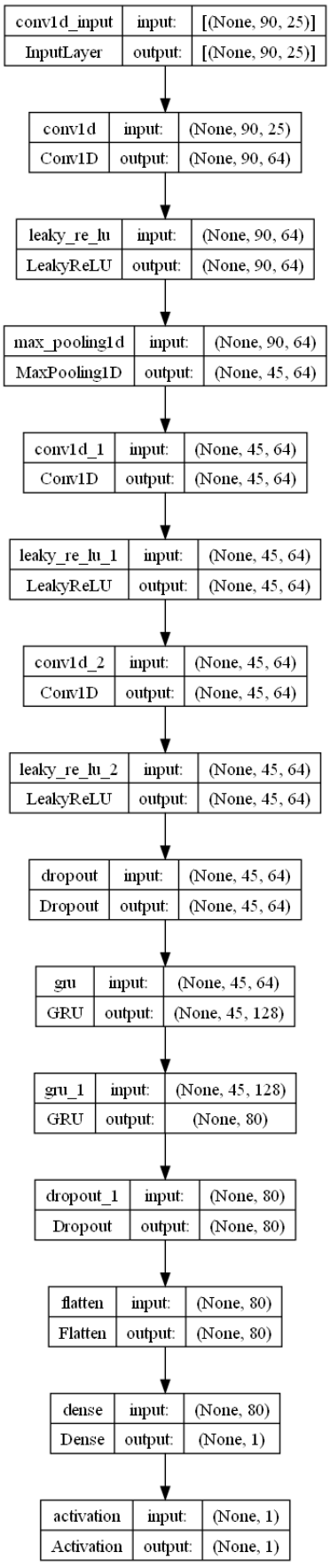
Sequential self-attention, also known as Self-Attention or intra-attention, is a powerful mechanism that has gained significant attention in Natural Language Processing and sequence modeling tasks such as time series data. This Attention Mechanism allows the model to focus on different parts of the input sequence when generating representations, providing several benefits. It assigns weights to each feature in the input sequence based on its relevance to other features in the same sequence. These weights indicate the importance or attention given to each feature during the prediction process. In an attempt to observe the effect of Attention Mechanism the performance of a model, we incorporated a Sequential Self-Attention layer into both proposed models. This layer was added before the first GRU layer in the CNN-GRU hybrid model and before the Flatten layer in the FNN model. The attention-activation was set to $relu$, and the attention-width was set to 15 in both models. The attention width determines how many samples the model considers simultaneously when calculating attention weights. The results obtained are displayed in Table 3.7.

3.5.4 The Effect of Multi-Head Architecture

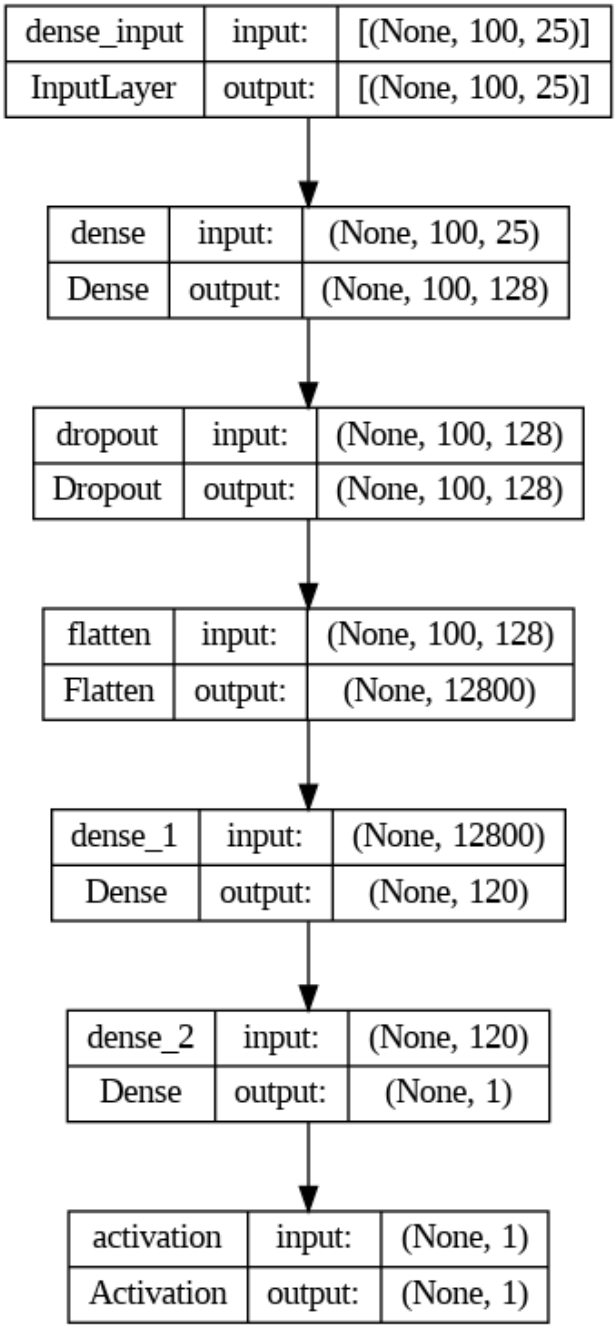
In the context of RUL prognosis for the C-MAPSS dataset using Multi-head architecture, each sensor's output is processed independently by a dedicated head. This setup enables the network to effectively capture spatial representations. Moreover, the outputs from each subspace are combined through concatenation with the addition of a Fully-Connected layer, which improves the contextual information within each time series [Canizo et al., 2019]. The final results are displayed in Table (3.8).

Additionally, the Attention Mechanism was applied to the Multi-head models to observe their combined effect on the accuracy of the models. The results are displayed in Table 3.9.

Furthermore, we would like to mention that during our experiments, we observed that incorporating an additional Dense layer With 20 units or (neurons) after concatenating the outputs of all branches resulted in improvements in the FNN Multi-head model. This additional layer can enhance the network's capacity to learn complex patterns and high-level representations from the concatenated features. The results of these modifications are presented in Table 3.10.

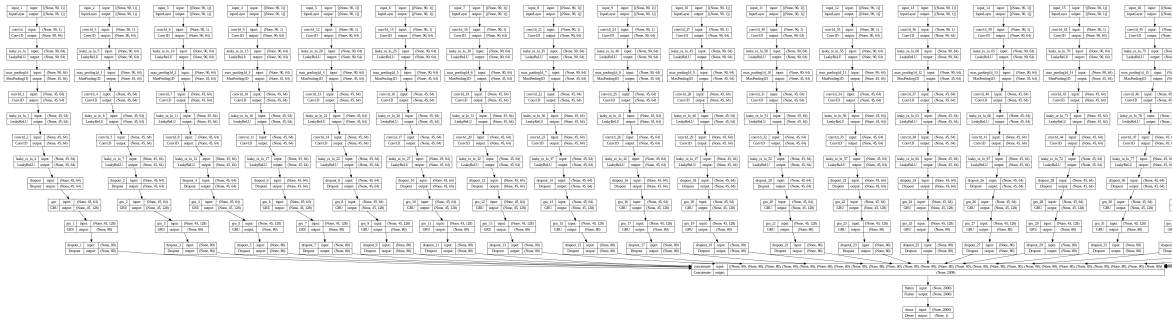


CNN-GRU hybrid model.

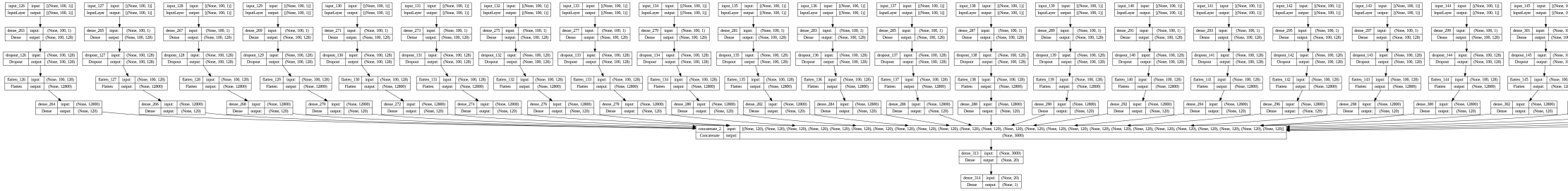


FNN model.

Figure 3.3: Flowchart for the sequential models.



CNN-GRU hybrid model.



FNN model.

Figure 3.4: Flowcharts for multi-head models.

3.6 Experimental Setup

The models were compiled using two different setups, as described in Table 3.5. The first computer was used to run the CNN-GRU hybrid sequential model and the CNN-GRU hybrid sequential model with the Attention Mechanism, while the second computer was used to run the FNN sequential model, the FNN sequential model with the Attention Mechanism and the FNN Multi-head model as well as the FNN multi-head model with Attention Mechanism. For the CNN-GRU hybrid Multi-head model, both with and without the Attention Mechanism and the FNN Multi-head model with the Attention Mechanism, Google Colab was utilized. The table also contains information about the versions of the used frameworks.

Configuration	1st computer	2nd computer
OS	win 10 x64 bits	win 11 x64 bits
RAM	6 GB	8 GB
ROM	500 GB	256 GB
CPU name	Intel core i3	Intel core i5
CPU cores	2	2
CPU speed	2 Ghz	2.60 Ghz
GPU	HD Graphics 5500	HD Graphics 620
Anaconda-v	2022.10	
Python-v	3.9.13	
keras-v	2.11.0	
Tensorflow-v	2.11.0	
Google colab		
GPU	T4	
Python-v	3.10.12	
Keras-v	2.12.0	
Tensorflow-v	2.12.0	

Table 3.5: Characteristics of the computers used in this work.

3.7 Results and Discussion

3.7.1 Performance Analysis of the Proposals

Both models were trained using the train data, and to evaluate their performance after the training phase, we fed the test data to our models and obtained the next results.

The results presented in Table 3.6 indicate that the FNN model generally outperformed the CNN-GRU hybrid model, with a decrease in RMSE by 32.49% in FD002, 5.77% in

FD003, and 13.20% in FD004. However, within the FD001 subset, the CNN-GRU hybrid model outperformed the FNN model by 15.77%.

Subsets	CNN-GRU hybrid			FNN		
	MAE	RMSE	R-squared	MAE	RMSE	R-squared
FD001	7.59	11.00	0.91	9.07	13.06	0.83
FD002	15.79	21.11	0.74	10.61	14.25	0.86
FD003	8.82	12.66	0.89	7.67	11.93	0.88
FD004	20.82	29.68	0.61	18.26	25.76	0.71

Table 3.6: The results for the CNN-GRU hybrid and FNN sequential models.

When applying the Attention Mechanism, as shown in Table 3.7, the results for the CNN-GRU hybrid model highlight the effectiveness of the sequential Self-attention layer on FD002 and FD004 subsets, with improvements in RMSE by 15.30% and 11.89%, respectively. As for the FD001 and FD003 subsets, there was no improvement recorded. Upon transitioning to the FNN model, the Attention Mechanism helped improve RMSE results for FD001 and FD004 by 7.73% and 10.44%, respectively, while no improvement was recorded for the FD002 and FD003 subsets.

Subsets	CNN-GRU hybrid			FNN		
	MAE	RMSE	R-squared	MAE	RMSE	R-squared
FD001	7.66	11.46	0.91	7.79	12.05	0.86
FD002	12.62	17.88	0.80	10.05	14.49	0.85
FD003	15.36	20.72	0.62	8.16	13.98	0.86
FD004	17.93	26.15	0.68	17.09	23.07	0.77

Table 3.7: The results for the CNN-GRU hybrid and FNN sequential models with attention mechanism.

While examining the impact of multi-head architecture on both proposed models, the results in Table 3.8 revealed no improvement for the CNN-GRU compared to the sequential architecture. Regarding the FNN model, we noticed an improvement of 2.22% in FD001 compared to the sequential architecture. However, the other subsets did not show any improvement.

The inclusion of the attention mechanism in multi-head models showed no significant improvement in the performance of the CNN-GRU hybrid model, as indicated in Table 3.9, except for the FD002 subset where RMSE improved by 8.73% compared to the multi-head model without an attention mechanism. Overall, it is evident that the sequential architecture outperformed the multi-head architecture when applied to the CNN-GRU hybrid model. As for FNN, there was a slight improvement of 1.09% in FD001. Notably,

Subsets	CNN-GRU hybrid			FNN		
	MAE	RMSE	R-squared	MAE	RMSE	R-squared
FD001	9.75	13.55	0.88	9.12	12.77	0.84
FD002	27.20	34.12	0.33	26.68	31.97	0.26
FD003	15.07	20.02	0.70	14.40	18.36	0.72
FD004	32.28	39.63	0.31	36.47	45.19	0.15

Table 3.8: The results for the CNN-GRU hybrid and FNN multi-head models.

we recorded a modest improvement of 10.47% in FD002 and a substantial improvement of 23.98% in FD004. However, in the case of FD003, there was no improvement compared to the results obtained from the multi-head model without the Attention Mechanism.

Subsets	CNN-GRU hybrid			FNN		
	MAE	RMSE	R-squared	MAE	RMSE	R-squared
FD001	9.97	14.62	0.85	9.76	12.63	0.86
FD002	25.69	31.14	0.46	23.75	28.62	0.37
FD003	16.35	21.88	0.64	13.80	18.37	0.75
FD004	37.42	43.81	0.15	28.68	34.35	0.51

Table 3.9: The results for the CNN-GRU hybrid and FNN multi-head models with attention mechanism.

To enhance the model's performance following the addition of the last dense layer, it exhibited improvements in both subsets FD001 and FD003 by 10.72% and 47.60% respectively when compared to the previous multi-head model. Conversely, no improvements were observed in the FD002 and FD004 subsets. In view of the results obtained, we can say that the multi-head architecture in some sub-sets provided better performance compared to the sequential architecture when applied on the FNN model.

Subsets	FNN		
	MAE	RMSE	R-squared
FD001	7.21	11.40	0.87
FD002	23.88	34.22	0.13
FD003	6.74	9.62	0.93
FD004	34.91	47.79	0.06

Table 3.10: The results for the improved FNN multi-head model (without attention mechanism).

As a last note, the RUL predictions made by both models, the CNN-GRU hybrid model with Attention Mechanism and FNN model with Multi-Head Architecture, for the four

subsets are visualized in Figures 3.5 and 3.6, respectively. The horizontal axis indicates the number of test engines, while the vertical axis corresponds to the RUL values. Within each subfigure, the disparity between the two graphs represents the error in the RUL prediction.

It is crucial to consider the computational burden and extended run time associated with the multi-head model when compared to the sequential approach. These factors should be taken into account when deciding on the appropriate model for RUL prediction, considering the trade-off between improved accuracy and computational efficiency.

3.7.2 Comparing the Best Models to State-Of-The-Art-Results

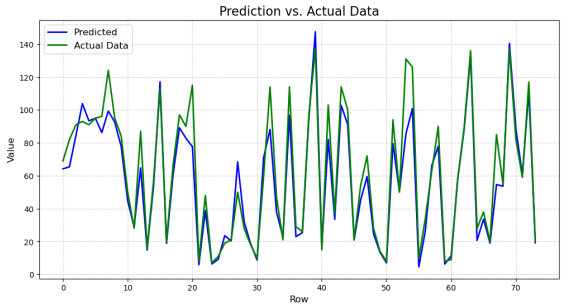
Table 3.11 compares the best results obtained in this work with other recent state-of-the-art approaches for RUL prognosis using the C-MAPSS dataset. The selected approaches are CNN [Sateesh Babu et al., 2016], DCNN ([Li et al., 2018]), FNN ([Ayodeji et al., 2021]), and lastly GRU ([Azyus et al., 2022]).

When evaluating using the RMSE metric, for the FD001 subset, the results of our proposed FNN multi-head model rank second, following only the work of [Ayodeji et al., 2021]. Conversely, for the FD004 subset, the results obtained with our proposed CNN-GRU hybrid model with an attention mechanism rank second, with [Li et al., 2018] being the top performing one. However, in the FD002 and FD003 subsets, our proposed models outperformed all state-of-the-art works used for comparison.

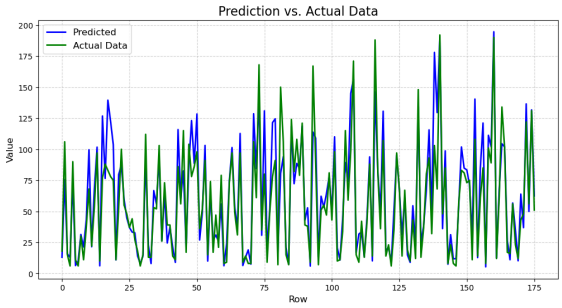
It is evident that deep-learning approaches show a promising future in the field of RUL prognosis, especially with emerging architectures such as multi-head models and attention mechanisms. This work concluded that perhaps there are other factors influencing the overall effect of the multi-head architecture and attention mechanism on the model's performance that should also be explored.

Algorithms	RMSE			
	FD001	FD002	FD003	FD004
CNN [Sateesh Babu et al., 2016]	18.44	30.29	19.81	29.15
DCNN [Li et al., 2018]	12.61	22.36	12.64	**23.31**
FNN [Ayodeji et al., 2021]	**8.68**	/	9.69	/
GRU [Azyus et al., 2022]	20.55	34.28	31.74	44.75
CNN-GRU sequential W/AM (proposed)	11.46	**17.88**	20.72	26.15
FNN multi-head (proposed)	11.40	34.22	**9.62**	47.79

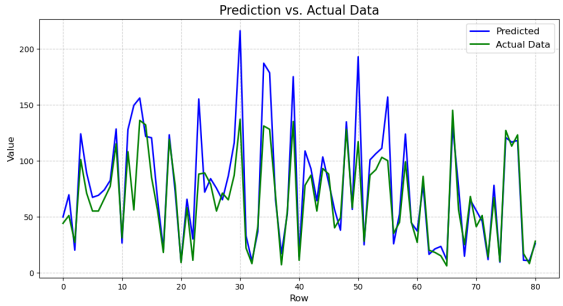
Table 3.11: Comparing our models with some of the state-of-art results.



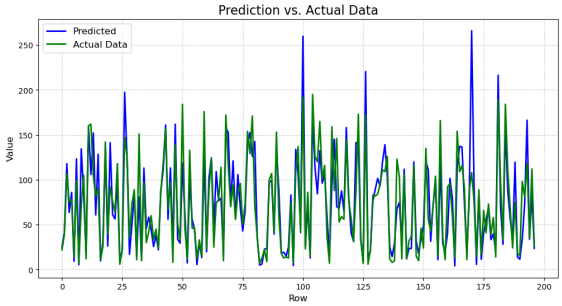
FD001



FD002

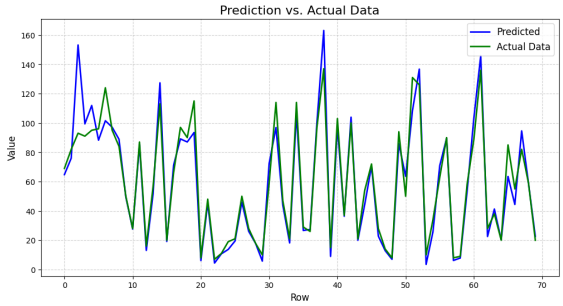


FD003

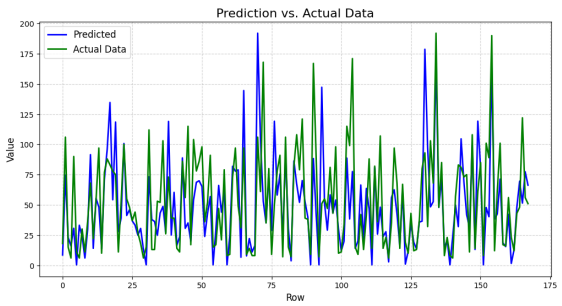


FD004

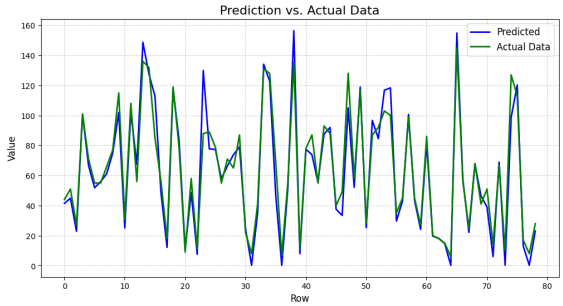
Figure 3.5: Predicted RUL by CNN-GRU hybrid model W/AM.



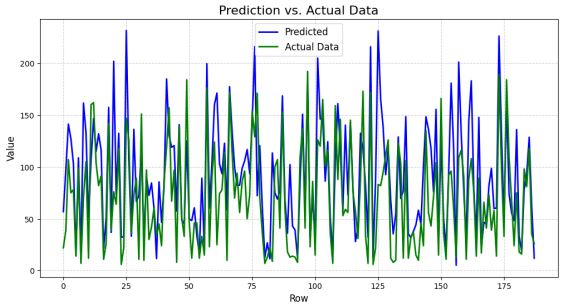
FD001



FD002



FD003



FD004

Figure 3.6: Predicted RUL by FNN multi-head model.

Conclusion

In this chapter, we proposed two models: a CNN-GRU hybrid model and an FNN model. Both models employ the linear degradation method in the data pre-processing phase. To improve the accuracy of both models, we integrated the attention mechanism and multi-head architecture, observing their individual and combined effects on the performance of our proposed models. Despite encountering some limitations, both models show promise in improving accuracy and overall performance.

General Conclusion

In this work, we directed our attention toward the prognosis estimation of Remaining Useful Life (RUL) for airplane turbofan engines, utilizing the C-MAPSS data-set. The main objective was to propose a hybrid deep learning model incorporating CNN (Convolutional Neural Network) and GRU (Gated Recurrent Unit) deep neural networks for this purpose. CNN is designed to automatically learn hierarchical patterns and features from input data through the use of convolutional layers, pooling layers, and fully connected layers. On the other hand, GRU is a type of recurrent neural network (RNN) commonly used for sequential data processing tasks, including natural language processing and time series analysis. GRU networks have gating mechanisms that allow them to selectively retain or update information at each time step, thereby rendering them effective in capturing long-term dependencies in sequential data. In addition, we introduced a second model, FNN (Feedforward Neural Network), to address the same problem as the hybrid model. FNN is a fundamental and general-purpose type of Artificial Neural Network (ANN) in which the information flows in a forward direction, from the input layer through one or more hidden layers to the output layer. To improve the performance of the models presented, we opted to incorporate the Attention Mechanism. The AM enables models to effectively focus on specific parts of the input sequence, giving more importance to relevant information while suppressing irrelevant or redundant information. Furthermore, we employed a Multi-head architecture in an effort to enhance the accuracy of both models. Multi-head architecture allows the parallel processing of multiple heads (or sub-models), with each one specializing in a distinct feature. This approach enables the model to leverage the strengths of each specialized sub-model and capture a wider range of information. Finally, we compared the results obtained with those of other relevant studies that tackled the same task, specifically predicting Remaining Useful Life (RUL), using the identical C-MAPSS dataset.

It is inevitable to encounter certain challenges while conducting research, and our

work is no exception. Among these challenges, we list the following:

- Limited access to articles and resources, particularly recent ones.
- Another challenging requirement is writing the master's thesis in English, considering that the majority of resources and tutorials are in English as well, adding additional difficulty due to our educational background in French.
- The lack of powerful equipment that can provide a convenient environment for running our codes is another challenge. Although we used Google Colab to mitigate this issue, our sessions are automatically terminated upon the completion of the allocated computing time.

The results we have obtained are promising. Nevertheless, it is important to note that these findings are based on certain restrictive assumptions and have the potential for further improvement by exploring alternative research avenues. One such avenue is to investigate alternative architectures, such as transformers and ensemble learning. The latter was initially part of our research agenda but unfortunately had to be excluded due to the pressing deadline. Additionally, it would be advantageous to assess our models on the new CMAPSS dataset. Another intriguing direction that we intended to explore involved the use of optimization metaheuristics for neural architecture search. However, in our pursuit of achieving state-of-the-art results, we ultimately decided to focus our time and effort into deep learning-based methods, with the aim of enhancing the performance of our proposed models, rather than exploring an entirely separate research discipline.

This project has proven beneficial to us on several levels. It allowed us to acquire new knowledge and strengthen existing skills. We delved into the field of maintenance, with a special focus on the Remaining Useful Life and, most importantly, Deep Learning techniques. All of these research areas required thorough literature analysis, as they were unfamiliar topics not covered during our Master's program. Additionally, it provided us with the opportunity to learn a new programming language, Python, as well as Deep Learning frameworks.

Bibliography

Karan Aggarwal, Maad M Mijwil, Abdel-Hameed Al-Mistarehi, Safwan Alomari, Murat Gök, Anas M Zein Alaabdin, Safaa H Abdulrhman, et al. Has the future started? the current growth of artificial intelligence, machine learning, and deep learning. *Iraqi Journal for Computer Science and Mathematics*, 3(1):115–123, 2022.

Anders Krogh. What are artificial neural networks? *Nature biotechnology*, 26(2):195–197, 2008.

Joanna Z Sikorska, Melinda Hodkiewicz, and Lin Ma. Prognostic modelling options for remaining useful life estimation by industry. *Mechanical systems and signal processing*, 25 (5):1803–1836, 2011.

Chang Woo Hong, Changmin Lee, Kwangsuk Lee, Min-Seung Ko, Dae Eun Kim, and Kyeon Hur. Remaining useful life prognosis for turbofan engine using explainable deep neural networks with dimensionality reduction. *Sensors*, 20(22):6626, 2020.

Abhinav Saxena, Kai Goebel, Don Simon, and Neil Eklund. Damage propagation modeling for aircraft engine run-to-failure simulation. In *2008 international conference on prognostics and health management*, pages 1–9. IEEE, 2008.

Crescenzio Gallo. Artificial neural networks tutorial. In *Encyclopedia of Information Science and Technology, Third Edition*, pages 6369–6378. IGI Global, 2015.

Raniah Zaheer and Humera Shaziya. A study of the optimization algorithms in deep learning. In *2019 third international conference on inventive systems and control (ICISC)*, pages 536–539. IEEE, 2019.

Christian Darken, Joseph Chang, John Moody, et al. Learning rate schedules for faster stochastic gradient search. In *Neural networks for signal processing*, volume 2, pages 3–12. Citeseer, 1992.

Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

Mahesh Chandra Mukkamala and Matthias Hein. Variants of rmsprop and adagrad with logarithmic regret bounds. In *International conference on machine learning*, pages 2545–2553. PMLR, 2017.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015.

Neha Yadav, Anupam Yadav, Manoj Kumar, et al. *An introduction to neural network methods for differential equations*, volume 1. Springer, 2015.

Murat H Sazli. A brief review of feed-forward neural networks. *Communications Faculty of Sciences University of Ankara Series A2-A3 Physical Sciences and Engineering*, 50(01), 2006.

AD Dongare, RR Kharde, Amit D Kachare, et al. Introduction to artificial neural network. *International Journal of Engineering and Innovative Technology (IJEIT)*, 2(1):189–194, 2012.

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

Davide Chicco, Matthijs J Warrens, and Giuseppe Jurman. The coefficient of determination r-squared is more informative than smape, mae, mape, mse and rmse in regression analysis evaluation. *PeerJ Computer Science*, 7:e623, 2021.

Dounia Lakhmiri, Sébastien Le Digabel, and Christophe Tribes. Hypernomad: Hyperparameter optimization of deep neural networks using mesh adaptive direct search. *ACM Transactions on Mathematical Software (TOMS)*, 47(3):1–27, 2021.

Saad Hikmat Haji and Adnan Mohsin Abdulazeez. Comparison of optimization techniques based on gradient descent algorithm: A review. *PalArch's Journal of Archaeology of Egypt/Egyptology*, 18(4):2715–2743, 2021.

Sanskruti Patel and Atul Patel. Deep leaning architectures and its applications: a survey. *International journal of computer sciences and engineering*, 6(6):1177–1183, 2018.

Laith Alzubaidi, Jinglan Zhang, Amjad J Humaidi, Ayad Al-Dujaili, Ye Duan, Omran Al-Shamma, José Santamaría, Mohammed A Fadhel, Muthana Al-Amidie, and Laith Farhan. Review of deep learning: Concepts, cnn architectures, challenges, applications, future directions. *Journal of big Data*, 8:1–74, 2021.

Sanam Narejo, Shahnawaz Talpur, Madeha Memon, and Amna Rahoo. An automated system for traffic sign recognition using convolutional neural network. *3c Tecnología: glosas de innovación aplicadas a la pyme*, 9(1):119–135, 2020.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Caihong Hu, Qiang Wu, Hui Li, Shengqi Jian, Nan Li, and Zhengzheng Lou. Deep learning with a long short-term memory networks approach for rainfall-runoff simulation. *Water*, 10(11):1543, 2018.

Giang Nguyen, Stefan Dlugolinsky, Martin Bobák, Viet Tran, Álvaro López García, Ignacio Heredia, Peter Malík, and Ladislav Hluchỳ. Machine learning and deep learning frameworks and libraries for large-scale data mining: a survey. *Artificial Intelligence Review*, 52:77–124, 2019.

Francois Chollet. *Deep learning with Python*. Simon and Schuster, 2021.

Gaige Chen, Jinglong Chen, Yanyang Zi, Jun Pan, and Wei Han. An unsupervised feature extraction method for nonlinear deterioration process of complex equipment under multi dimensional no-label signals. *Sensors and Actuators A: Physical*, 269:464–473, 2018.

Wennian Yu, II Yong Kim, and Chris Mechefske. Remaining useful life estimation using a bidirectional recurrent neural network based autoencoder scheme. *Mechanical Systems and Signal Processing*, 129:764–780, 2019.

Tianyi Wang, Jianbo Yu, David Siegel, and Jay Lee. A similarity-based prognostics approach for remaining useful life estimation of engineered systems. In *2008 international conference on prognostics and health management*, pages 1–6. IEEE, 2008.

Racha Khelif, Simon Malinowski, Brigitte Chebel-Morello, and Noureddine Zerhouni. Rul prediction based on a new similarity-instance based approach. In *2014 IEEE 23rd International Symposium on Industrial Electronics (ISIE)*, pages 2463–2468. IEEE, 2014.

Raphael Yende. Support de cours de maintenance informatique. 2018.

Ahmed Fikri Ali Mosallam Mosallam. *Remaining useful life estimation of critical components based on Bayesian Approaches.* PhD thesis, Université de Franche-Comté, 2014.

Hassani Nihal. *Une approche d'optimisation pour une meilleure efficacité d'un modèle d'estimation de temps restant utile du moteur à double flux à base de deep learning*. PhD thesis, university center of abdalhafid boussouf-MILA, 2022.

Messaoud Amira, Aissam Boulechefar, and Rachid Encadreur Belhadef. *Optimisation et amélioration de la maintenance par la fiabilité: Cas d'une turbine d'une centrale électrique*. PhD thesis, Université de Jijel, 2019.

Qi Hao, Yunjiao Xue, Weiming Shen, Brian Jones, and Jie Zhu. A decision support system for integrating corrective maintenance, preventive maintenance, and condition-based maintenance. In *Construction Research Congress 2010: Innovation for Reshaping Construction Practice*, pages 470–479, 2010.

Estelle Deloux. *Politiques de maintenance conditionnelle pour un système à dégradation continue soumis à un environnement stressant*. PhD thesis, Université de Nantes, 2008.

Yongyi Ran, Xin Zhou, Pengfeng Lin, Yonggang Wen, and Ruilong Deng. A survey of predictive maintenance: Systems, purposes and approaches. *arXiv preprint arXiv:1912.07383*, 2019.

Diego Jair Rodriguez Obando. *From Deterioration Modeling to Remaining Useful Life Control: a comprehensive framework for post-prognosis decision-making applied to friction drive systems*. PhD thesis, Université Grenoble Alpes, 2018.

Adrian Cubillo, Suresh Perinpanayagam, and Manuel Esperon-Miguez. A review of physics-based models in prognostics: Application to gears and bearings of rotating machinery. *Advances in Mechanical Engineering*, 8(8):1687814016664660, 2016.

Jing Liu. A dynamic modelling method of a rotor-roller bearing-housing system with a localized fault including the additional excitation zone. *Journal of Sound and Vibration*, 469:115144, 2020.

Racha Khelif, Brigitte Chebel-Morello, Simon Malinowski, Emna Laajili, Farhat Fnaiech, and Noureddine Zerhouni. Direct remaining useful life estimation based on support vector regression. *IEEE Transactions on industrial electronics*, 64(3):2276–2285, 2016.

Biao Wang, Yaguo Lei, Naipeng Li, and Tao Yan. Deep separable convolutional network for remaining useful life prediction of machinery. *Mechanical systems and signal processing*, 134:106330, 2019.

Hamed Khorasgani, Gautam Biswas, and Shankar Sankararaman. Methodologies for system-level remaining useful life prediction. *Reliability Engineering & System Safety*, 154:8–18, 2016.

Zeqi Zhao, Bin Liang, Xueqian Wang, and Weining Lu. Remaining useful life prediction of aircraft engine based on degradation pattern learning. *Reliability Engineering & System Safety*, 164:74–83, 2017.

Ming Dong and David He. A segmental hidden semi-markov model (hsmm)-based diagnostics and prognostics framework and methodology. *Mechanical systems and signal processing*, 21(5):2248–2266, 2007.

Junbo Son, Shiyu Zhou, Chaitanya Sankavaram, Xinyu Du, and Yilu Zhang. Remaining useful life prediction based on noisy condition monitoring signals using constrained kalman filter. *Reliability Engineering & System Safety*, 152:38–50, 2016.

Suzan Alaswad and Yisha Xiang. A review on condition-based maintenance optimization models for stochastically deteriorating system. *Reliability engineering & system safety*, 157:54–63, 2017.

Khanh TP Nguyen, Kamal Medjaher, and Do T Tran. A review of artificial intelligence methods for engineering prognostics and health management with implementation guidelines. *Artificial Intelligence Review*, 56(4):3659–3709, 2023.

Racha Khelif. *Estimation du RUL par des approches basées sur l'expérience: de la donnée vers la connaissance.* PhD thesis, Université de Franche-Comté, 2015.

Paulo Roberto de Oliveira da Costa, Alp Akçay, Yingqian Zhang, and Uzay Kaymak. Remaining useful lifetime prediction via deep domain adaptation. *Reliability Engineering & System Safety*, 195:106682, 2020.

Piero Baraldi, Michele Compare, Sergio Sauco, and Enrico Zio. Ensemble neural network-based particle filtering for prognostics. *Mechanical Systems and Signal Processing*, 41(1-2): 288–300, 2013.

Giduthuri Sateesh Babu, Peilin Zhao, and Xiao-Li Li. Deep convolutional neural network based regression approach for estimation of remaining useful life. In *Database Systems for Advanced Applications: 21st International Conference, DASFAA 2016, Dallas, TX, USA, April 16-19, 2016, Proceedings, Part I 21*, pages 214–228. Springer, 2016.

Abiodun Ayodeji, Wenhai Wang, Jianzhong Su, Jianquan Yuan, and Xinggao Liu. An empirical evaluation of attention-based multi-head models for improved turbofan engine remaining useful life prediction. *arXiv preprint arXiv:2109.01761*, 2021.

Adryan Fitra Azyus, Sastra Kusuma Wijaya, Mohd Naved, et al. Determining rul predictive maintenance on aircraft engines using gru. *Journal of Mechanical, Civil and Industrial Engineering*, 3(3):79–84, 2022.

Adel S Eesa and Wahab Kh Arabo. A normalization methods for backpropagation: a comparative study. *Science Journal of University of Zakho*, 5(4):319–323, 2017.

Mikel Canizo, Isaac Triguero, Angel Conde, and Enrique Onieva. Multi-head cnn–rnn for multi-time series anomaly detection: An industrial case study. *Neurocomputing*, 363: 246–260, 2019.

Acronyms

AI Artificial Intelligence

ML Machine Learning

DL Deep Learning

CV Computer vision

NLP Natural Language Processing

DNN Deep Neural Network

ANN Artificial Neural Network

Tanh Hyperbolic Tangent Function

ReLU Rectified Linear Unit

Leaky ReLU Leaky Rectified Linear Unit

MSE Mean Squared Error

MAE Mean Absolute Error

BCE Binary Cross-Entrop

SGD Stochastic Gradient Descen

CCE Categorical Cross-Entropy

Adagrad Adaptive Gradient Algorithm

Adadelta Adaptive Delta Algorithm

RMSProp Root Mean Square Propagation

Adam Adaptive Moments

RMSE Root Mean Square Error

PHM Prognostics and Health Management

LR Learning Rate

GD Gradient Descent

CNN Convolutional Neural Network

FC Full Connection

RNN Recurrent Neural Network

PL Pooling Layer

LSTM Long Short-Term Memory

GRU Gated Recurrent Unit

CPU Central Processing Unit

API Application Programming Interface

USP Unique Selling Proposition

RUL Remaining Useful Life

CBM Condition-Based Maintenance

CM Condition Monitoring

HMM Hidden Markov Model

ARMA Auto-Regressive Moving Average model

BN Bayesian networks

ARIMA Integrated Auto-Regressive Moving Average model

SVR Support Vector Regression

IBM Instance-Based Methods

NN Neural Network

SVM Support Vector Machines

DCNN Deep Convolutional Neural Network

SRNN Simple Recurrent Neural Network

RaPC Retrieve and Propose with Case-based reasoning

C-MAPSS Commercial Modular Aero-Propulsion System Simulation

AM Attention Mechanism

FNN Feedforward Neural Network

MLPs Multi-Layer Perceptron

NASA National Aeronautics and Space Administration