People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research

Saad Dahlab University - Blida 1
Institute of Aeronautics and Space Studies
Aeronautical Construction Department

Manuscript

Towards the attainment of the degree of Master's in

Aeronautics Engineering

Option: **Avionics**

Thesis Topic

Predictive Maintenance of Aircraft Engines: Remaining Useful Life (RUL)

Submitted by

- Khawla BOUGHERARA
- Nazim BELAID

Supervised by

- Dr.Khireddine CHOUTRI

Jury Members:

Dr.Smain DILMI	President
Dr.Omar ABADA	Examiner

2023/2024

Acknowledgment

First and foremost, I would like to express my gratitude to Allah for all the knowledge I have acquired.

I extend my sincere appreciation to my Supervisor, Dr. Kheireddine CHOUTRI, for his expert guidance and valuable insights. His expertise and mentorship have played a pivotal role in shaping the direction of my research and enhancing the quality of this thesis.

To my beloved family, I am indebted to your encouragement. Your constant belief in my abilities, patience during moments of stress, and words of motivation have been my driving force. Your sacrifices have meant the world to me, and I cannot thank you enough for being my pillars of strength.

I thank the jury members for agreeing to preside over and examine our work. We express our gratitude for their presence and consideration.

Khawla BOUGHERARA

Summary

This thesis presents a comprehensive predictive maintenance system for aircraft engines, with a primary focus on predicting the Remaining Useful Life (RUL) using advanced Machine Learning (ML) techniques. Predictive maintenance is crucial in the aviation industry as it enables the early detection of potential failures, thereby minimizing downtime, reducing maintenance costs, and ensuring the safety and reliability of aircraft operations. The research employs the PHM08 NASA dataset from the first conference on Prognostics and Health Management to train and evaluate various algorithms, including Gradient Boosting (GBM), Support Vector Machines (SVM), Convolutional Neural Networks (CNN), Long Short-Term Memory (LSTM), Gated Recurrent Units (GRU), and Convolutional Neural Networks combined with Long Short-Term Memory (CNN-LSTM) models. The study compares the performance of these methods using Root Mean Square Error (RMSE) as the evaluation metric. The findings indicate that the CNN-LSTM model is particularly effective for predicting the RUL of aircraft engines.
The objectives of this system are to enhance operational efficiency within the aviation industry, reduce downtime, and improve maintenance strategies.

Key Words: Predictive Maintenance, Remaining Useful Life, Aircraft Engine, Machine Learning, Deep Learning, Neural Networks.

ملخص

تقدم هذه الأطروحة نظام صيانة تنبؤية شامل لمحركات الطائرات، مع التركيز الأساسي على التنبؤ بالعمر المتبقي القابل للاستخدام باستخدام تقنيات متقدمة في تعلم الآلة. تعتبر الصيانة التنبؤية أمرًا بالغ الأهمية في صناعة الطيران لأنها تتيح الكشف المبكر عن الأعطال المحتملة، مما يقلل من وقت التوقف عن العمل، ويخفض تكاليف الصيانة، ويضمن سلامة وموثوقية عمليات الطيران. يستخدم البحث مجموعة بيانات من ناسا من المؤتمر الأول لإدارة التنبؤات والصحة لتدريب وتقييم خوارزميات متنوعة، بما في ذلك التعزيز التدريجي ، آلات الدعم المتجهة، الشبكات العصبية التلافيفية ، ذاكرة طويلة وقصيرة المدى ، وحدات التكرار المغلقة ، ونماذج الشبكات العصبية التلافيفية المدمجة مع ذاكرة طويلة وقصيرة المدى. تقارن الدراسة أداء هذه الأساليب باستخدام خطأ الجذر التربيعي المتوسط كمقياس للتقييم. تشير النتائج إلى أن نموذج الشبكات العصبية التلافيفية المدمجة مع ذاكرة طويلة وقصيرة المدى فعال بشكل خاص في التنبؤ بالعمر المتبقي القابل للاستخدام لمحركات الطائرات.تهدف هذه المنظومة إلى تعزيز الكفاءة التشغيلية داخل صناعة الطيران، وتقليل وقت التوقف، وتحسين استراتيجيات الصيانة.

الكلمات الرئيسية: الصيانة التنبؤية، العمر المتبقي القابل للاستخدام، محرك الطائرة، تعلم الآلة، التعلم العميق، الشبكات العصبية.

Contents

List of Figures

List of Tables

Abbreviations list

C-MAPSS	Commercial Modular Aero-Propulsion System Simulation
CSV	Comma-Separated Values
DL	Deep Learning
FD	Flight Data
HPC	High-Pressure Compressor
HPT	High-Pressure Turbine
IoT	Internet of Things
LPC	Low-Pressure Compressor
LPT	Low-Pressure Turbine
ML	Machine Learning
NASA	National Aeronautics and Space Administration
N1	Rotational speed of the low-pressure compressor
N2	Rotational speed of the high-pressure compressor
OP	Operating Setting
PHM08	Prognostics and Health Management in 2008
ReLU	Rectifier Linear Unit
RMSE	Root Mean Square Error
RUL	Remaining Useful Life
S	Sensor Measurement

General Introduction

The aviation industry relies heavily on the reliability and safety of Aircraft engines, they are complex and critical components of an aircraft, responsible for generating thrust and powering the aircraft during flight. They consist of several subsystems, including the compressor, turbine, and exhaust nozzle, each playing a crucial role in the engine's performance[1].

Maintaining aircraft engines poses significant challenges due to their harsh operating conditions, including high temperatures, pressures, and speeds.
To address engine failures, three primary maintenance approaches are utilized. Reactive maintenance involves repairing or replacing components only after they have failed. While straightforward, this approach is unsuitable for the aviation industry as it leads to unexpected downtime and higher costs due to unplanned failures[2].
Preventive maintenance involves regular inspections and scheduled part replacements, no matter their condition. Although it reduces the risk of unexpected failures, this method can waste the usable life of components through early replacements[2].
In contrast, predictive maintenance optimizes maintenance timing by predicting the future condition of components using real-time data and advanced analytics. Making use of techniques such as machine learning and statistical analysis, predictive maintenance forecasts the remaining useful life (RUL) of engine parts. This approach allows maintenance to be performed only when necessary, minimizing downtime and costs while maximizing component life and ensuring efficient and reliable aircraft operation[2].

The thesis is structured as follows:

1. The first chapter provides the state of the art where a quick review is presented on different learning models used in predictive maintenance.

2. The second chapter titled Architecture and Modeling delves into the algorithms used in this thesis.

3. The third chapter details the implementation of the chosen algorithm on the C-MAPSS dataset and the performance evaluation.

At last, we offer a broad conclusion to wrap up our work.

Chapter 01 :

State Of The Art

Chapter 1

State of the Art

1.1 Introduction

This chapter delves into various methods employed for anomaly detection, exploring their Learning, Signal, and model Types to identify the most suitable approach for aircraft engines.
This chapter aims to provide a comprehensive overview of remaining useful life (RUL) prediction, highlighting its profound advantages.

1.2 Anomaly Detection

Anomalies, also known as outliers, are data points that deviate significantly from the expected behavior or norm within a dataset. They are crucial to identify because they can signal potential problems[3].
Anomaly detection in aircraft maintenance offers several key benefits. Firstly, it enables early fault detection, allowing for immediate responses that prevent minor issues from escalating into catastrophic failures[4]. This preventive approach significantly enhances overall flight safety. Secondly, targeting maintenance efforts toward identified anomalies helps optimize resource allocation and prevents unnecessary interventions, leading to more efficient maintenance practices[4]. Also implementing anomaly detection techniques can potentially reduce costs in several ways, including:

- Minimizing the impact of engine failures.
- Optimized maintenance scheduling.

- Extended motor life

However, Implementing anomaly detection in aviation presents significant challenges due to the complexity of aircraft data, the need for high data quality and integrity, the lack of labeled data for training, and the Complicated interconnections among aircraft systems.[4]
The Main Questions Are : Which Model is the most suitable for aircraft engines? Does the chosen anomaly detection method address the challenges and complexities of implementing such techniques?

1.3 Types Of Anomalies

There are three main types of anomalies.

1.3.1 point anomalies

A point anomaly, as described in the study by Chandola, Banerjee, and Kumar (2009)[3], refers to a single data point that significantly deviates from the general pattern in a dataset.

1.3.2 collective anomalies

Collective anomalies are referred to as "a collection or subset of data items that show anomalous behavior when taken as a whole" in the study by Chandola, Tung, and Kumar (2009) [3]. They result from analyzing correlations within data.

1.3.3 Contextual (Conditional) Anomalies

These anomalies depend on the specific context or environment surrounding them. They often occur in time-series data, where patterns can change over time. An example is a sudden spike in temperature during winter within weather data.[3]

1.4 Learning Types

1.4.1 Supervised Learning

Supervised learning is a learning type in which the algorithm is trained using labeled data. The labeled data consists of input features and corresponding output labels or target variables. Supervised learning aims to develop a model that can make accurate predictions or classifications when given new data.[3]
In supervised machine learning, there are two primary types of problems:

Classification

Classification is a supervised learning technique that aims to assign input data to one of several predefined categories or classes. The model is trained on a labeled dataset, where each training example is associated with a class label. Common classification applications include email spam detection, image recognition, and medical diagnosis.[3]

Regression

Regression is a supervised learning technique used to predict a continuous output value based on input features. The model is trained on a labeled dataset with continuous target values. Regression is commonly used in scenarios such as predicting house prices, forecasting stock prices, and estimating the remaining useful life of machinery.[3]

Advantages of supervised learning include the ability to make accurate predictions or classifications when given new data, as well as the ability to handle complex relationships between input features and output labels.
However Supervised learning has limitations as well. These include the need for labeled data, which can be time-consuming and costly. Another limitation is that the model's performance heavily relies on the quality and quantity of the labeled data. [3]

1.4.2 Unsupervised Learning

Unsupervised learning is a learning type in which the algorithm is trained on unlabeled data. Unsupervised learning aims to uncover hidden patterns, structures, and relation-

ships within the data without prior knowledge of the output labels.
Unsupervised learning has two benefits: it may be used to identify patterns in data that are not immediately apparent to people and it doesn't require labeled data. Unsupervised learning has the following limitations: it is less precise than supervised learning. additionally, It is more challenging to put into practice than supervised learning. [3]

1.5 Signal Types

1.5.1 Univariate Method

Univariate methods in prediction refer to analyzing and modeling a single variable or feature to make predictions. This approach is commonly used when there is only one independent variable or when the relationship between multiple variables is not of interest[3].
Univariate methods in signal processing focus on analyzing a single signal, using statistical models and tests like autocorrelation. These techniques are useful for identifying trends, patterns, and anomalies in time-series data.[3]

1.5.2 Multivariate Method

Multivariate methods in prediction involve analyzing and modeling multiple variables simultaneously to make predictions. This approach is used when various independent variables could potentially impact the dependent variable[3].
Multivariate signal analysis methods deal with multiple signals together to detect anomalies in complex systems, which is particularly useful in identifying hidden patterns that cannot be detected using univariate analysis [3].

1.6 Model Types

1.6.1 Statistical Methods

Statistical prediction methods involve using data analysis and probability theory to analyze historical data, and to identify patterns and relationships, which are then used to make informed predictions about future outcomes.[5]

Z-Score

The Z-score is a statistical technique used to determine the number of standard deviations. Z-scores are useful in anomaly detection because they help us find data points that significantly differ from the average. Every data point is then given a z-score, which indicates how far out from the mean it is by a certain number of standard deviations[6].

In a dataset X, the z-score for a given data point is x
If x is outside a certain range determined by the threshold, it is considered an anomaly. A common threshold is ± 3, meaning data points more than 3 standard deviations away from the mean are considered anomalies.

- The principal benefit of the z-score method is its simplicity and interpretability; it is simple to comprehend and apply, also the threshold for identifying an anomaly can be modified according to the specific scenario being used.[6]
- On the other hand, the z-score technique has certain drawbacks, such as its assumption of a normally distributed dataset, which may not always hold, and the possibility of inefficiency in detecting anomalies that are gradual or subtle rather than sudden and extreme.[6]

Interquartile Range(IQR)

The IQR method is a robust measure of statistical distribution. It is calculated as the difference between the upper quartile (Q3) and lower quartile (Q1) of the data [7] The IQR is then Q3 - Q1, where Q1 is the 25th percentile and Q3 is the 75th percentile.

- The Interquartile Range (IQR) technique is beneficial for identifying and removing outliers in continuously distributed data. However, its effectiveness is limited to continuous data, and it may require threshold adjustments for different datasets.[7]

Principal Component Analysis(PCA)

According to the Journal of Multivariate Analysis, Principal Component Analysis (PCA) is a ”statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components.” [8]

It is a widely used tool for analyzing data, particularly in the context of fault prediction. PCA works by transforming a complex and massive dataset into a smaller and simpler set of orthogonal and uncorrelated features known as principal components. These components capture the maximum relevant information from the original dataset, making it easier to handle and visualize the data[8].

By analyzing the correlation and variability of the original variables, PCA creates new variables called principal components. These principal components capture the maximum amount of information in the dataset and can be used to make predictions or classify new data points[8].

- therefore is important to remember that PCA operates under the assumption that relationships between variables are linear, and therefore may not work well in non-linear situations.[8].

Table 1.1: Citations Of Statistical Methods Section

Method	Ref	Year	Topic
Z-Score	[6]	2018	Anomaly detection by robust statistics
IQR	[7]	2018	Detection of outliers using interquartile range technique from intrusion dataset
PCA	[8]	2002	Principal component analysis for special types of data

1.6.2 Machine Learning Methods

Machine learning is a subfield of artificial intelligence that focuses on the development of algorithms and models that allow computers to learn and make predictions or decisions without being explicitly programmed.[5] There are several methods used in machine learning:

Gradient Boosting (GB)

Gradient Boosting is a popular machine learning algorithm that is widely used for regression and classification tasks. It works by combining multiple weak learners (typically decision trees) in a sequential manner, where each learner is trained to correct the mistakes made by the previous learner. By iteratively learning from the residuals of the previous model, Gradient Boosting gradually improves the overall predictive performance.[9]

The algorithm works by optimizing a loss function with gradient descent, hence the name "gradient boosting." Firstly, Gradient Boosting has been shown to have excellent prediction performance. Additionally, Gradient Boosting is known for its ability to handle complex and non-linear relationships between features and the target variable. It is good because it's powerful, handles different types of data well, reduces overfitting, and provides high accuracy.[9]
GBM drawbacks are potential overfitting, longer training time, and sensitivity to noisy data compared to simpler models.[9]

Support Vector Machines(SVMs)

One kind of supervised learning algorithm that may be applied to tasks involving regression or classification is the SVM. Finding the best decision limits and managing high-dimensional data are two of SVM's distinctive qualities. In classification jobs, support vector machines (SVMs) are employed to divide data points into distinct classes according to their characteristics. SVMs function by identifying the ideal hyperplane that maximizes the margin between classes and effectively divides the data points into distinct classes.[10]
When data cannot be separated linearly, SVMs can translate the data into a higher-dimensional space using kernel functions, allowing for linear separation of the data.[10]
Some of the good points of using SVMs for prediction include:

- SVMs can handle both linear and non-linear data by utilizing different types of kernels[10]. Also, SVMs have a robust performance even with limited training data.

SVMs are an effective classification tool. However, they have several drawbacks as well:

- They may be exposed to anomalies and noise in the data. Particularly with complicated data, they might be challenging to comprehend. Also, while training on big datasets, they might be slow.[10]

K-nearest Neighbors(KNN)

An effective strategy for classification and prediction in machine learning is the K-nearest neighbors technique. Since it is non-parametric, no suppositions are made about the distribution of the original data [11].

- Method of Prediction: Based on a distance metric (such as the Euclidean distance), KNN finds the K nearest data points in the training set given a new data point. The majority class or average value of the new data point's K closest neighbors determines the estimated class or value for it. [11]

K-Nearest Neighbors offers the following benefits: non-parametric, easy to implement, no training phase.
KNN Technique drawbacks include its high computational cost, sensitivity to outliers, and requirement for the ideal K value.[11]

Decision Trees (DT)

Decision Trees are a predictive modeling method that uses a tree structure to represent decisions and their possible consequences.[12]

- One of the key advantages of decision trees is their interpretability, as they provide a clear representation of the decision-making process. They also can handle both numerical and categorical data without the need for extensive data preprocessing.

- decision trees can be disposed to overfitting, especially when they grow too large and complex. This can lead to poor generalization of unseen data. Additionally, decision trees are sensitive to small variations in the training data, which can result in different tree structures and potentially impact the model's performance. [12]

Random Forest(RF)

Random Forest is a popular machine-learning algorithm known for its high predictive accuracy and robustness. It works by constructing multiple decision trees during training and outputting the mode of the classes (classification) or the average prediction (regression) of the individual trees[13].
Its strengths lie in handling large datasets with high dimensionality and being resistant to overfitting. Random Forest may not perform well with noisy data.[13]

Naive Bayes

Naive Bayes is a probabilistic classification method based on Bayes theorem. Assumes independence of features[14]. It is widely used in various applications due to its simplicity, efficiency, and ability to handle large datasets. [14]

- One of its key strengths is its fast training speed and low computational cost, making it suitable for real-time prediction tasks.[14]

The "naive" assumption of feature independence may not hold in all cases, leading to suboptimal performance when dealing with highly correlated attributes.[14]

Isolation Forest(IF)

Isolation Forest (IF) is a tree-based algorithm for outlier detection. It isolates anomalies by creating decision trees with random divisions. IF is faster and more efficient than other algorithms. However, it may not perform well in high-dimensional data and is sensitive to the number of trees and depth.[15]
The quantity and depth of trees might have an impact on the performance of IF, it needs careful parameter optimization and statistical understanding. Furthermore, IF may not be as accurate when dealing with high-dimensional and connected data.[15]

Local Outlier Factor(LOF)

Local Outlier Factor (LOF) is a density-based method for detecting outliers based on a local density measurement. It assigns a score to each sample, and higher scores indicate a higher probability of being an outlier[16]. It is effective in identifying outliers in complex, high-dimensional data[16].

- LOF is well-known for its ability to locate local outliers in datasets, particularly in high-dimensional spaces. Also, it is an adaptable outlier identification technique that may be used in a variety of domains and applications.[16]
- LOF may need a lot of memory and processing power, particularly when working with big datasets or streaming data. Also, It is sensitive to Parameters, depending on the parameters used, such as the number of neighbors taken into account for density estimation, the performance of LOF may vary.[16]

Table 1.2: Citations Of Machine Learning Methods Section

Method	Ref	Year	Topic
GB	[9]	2024	Machine Learning - Gradient Boosting
SVM	[10]	2024	Support Vector Machine Algorithm
KNN	[11]	2011	A review of various k-nearest neighbor query processing techniques
DT	[12]	2008	Anomaly detection by combining decision trees and parametric densities
RF	[13]	2019	2019 1st International Conference on Smart Systems and Data Science(ICSSD)
NB	[14]	2007	Survey of improving naive bayes for classification
IF	[15]	2019	Functional isolation fores
LOF	[16]	2000	LOF: identifying density-based local outliers

1.6.3 Deep Learning Methods

Deep learning is a branch of machine learning that uses artificial neural networks (ANNs) with multiple layers to learn from data. It involves a computational architecture consisting of input, hidden, and output layers, enabling advanced representation learning.[5]

Autoencoders

Autoencoders are neural networks for dimensionality reduction and representation learning. They use an encoder-decoder structure to reconstruct input[17].
They consist of an encoder that compresses the input data into a latent-space representation and a decoder that reconstructs the input from this representation[17].

- One of the key advantages of autoencoders is their ability to learn efficient representations of data, which can be useful for tasks like anomaly detection.[17]
- autoencoders can suffer from issues like overfitting, especially in complex datasets. Additionally, training autoencoders can be computationally intensive, requiring careful tuning of hyperparameters to achieve optimal performance [17].

Generative Adversarial Networks (GANs)

Generative Adversarial Networks (GANs) are deep learning frameworks. They consist of two components: a generator and a discriminator.
The generator aims to produce artificial data samples that are indistinguishable from real data, while the discriminator's role is to differentiate between real and generated samples[18].
In prediction tasks, GANs can generate realistic and diverse outputs, making them valuable for tasks. However, they can be challenging to train due to instability issues like mode collapse and vanishing gradients [18]

Recurrent Neural Networks (RNNs)

Recurrent Neural Networks are a type of deep learning algorithm that masters in handling sequential and time-series data[17].
Using feedback cycles, RNNs can maintain a memory of previous inputs, making them

well-suited for tasks involving sequential inputs such as natural language processing, speech recognition, machine translation, and time series prediction[19].
RNNs operate by processing input data step by step, with each step incorporating information from the previous steps [19].

- The memory mechanism in RNNs allows them to retain information from previous time steps, enabling the capture of long-term dependencies in data sequences. This flexibility and memory capacity make RNNs well-suited for tasks that involve processing sequential data, where the order and context of the data are crucial for making accurate predictions. [19]

- RNNs can capture complex relationships but suffer from vanishing or exploding gradients, making learning long-term dependencies difficult. Training RNNs is challenging, requiring sophisticated optimization techniques.[19]

Long Short-Term Memory (LSTM)

Long short-term memory (LSTM) is a specialized architecture within Recurrent Neural Networks (RNNs). These networks are designed to capture long-term dependencies in data, enabling the processing of multidimensional information effectively.[17]
LSTM networks represent a sophisticated neural network architecture adapted for time series forecasting. They master identifying long-term dependencies within datasets and are capable of handling multidimensional data inputs effectively[20].
Based on the provided references [20] [17], the strengths and weaknesses of LSTM networks are:
LSTM effectively integrates extensive historical data into predictions, improving time series modeling. The gating mechanisms in LSTM networks filter out unrelated information, leading to higher precision in modeling time-variant behavior. However, their complex architecture can be computationally intensive and often requires more parameters than other RNN types like GRU. Additionally, the encoding and decoding processes in LSTM cells may result in some information loss, although effective feature extraction can reduce this issue.

The Gated Recurrent Unit (GRU)

The Gated Recurrent Unit (GRU) is a neural network architecture employed in deep learning, particularly for tasks like software defect prediction.[21]
It features two primary gates: the update gate (zt) and the reset gate (rt). The update gate controls the extent to which the previous hidden state influences the current state, while the reset gate modulates the transfer of information from past states. Compared to Long Short-Term Memory (LSTM) networks, GRUs are simpler due to their reduced number of gates, resulting in enhanced computational efficiency.[21]

Based on the provided reference [21]The Gated Recurrent Unit (GRU) presents certain limitations:

- Due to its simplified structure, GRU may not capture long-term dependencies as effectively as Long Short-Term Memory (LSTM) networks.
- The gating mechanism in GRU may be less effective in managing complex memory processes compared to LSTM.

Convolutional Neural Networks (CNNs)

Convolutional Neural Networks are a type of artificial neural network that has proven to be highly effective in applications such as pattern recognition and classification. [17]
In prediction, CNNs master at learning graded representations of input data, allowing them to extract suitable features automatically. This makes them highly effective for tasks involving visual or sequential data.[17]

- CNNs may not be the best choice for tasks involving non-visual data structures like graphs or trees.[22]
- Despite its limitations, CNNs remain a powerful tool for a wide range of prediction tasks due to their ability to generalize well to new data and handle inputs of varying sizes and aspect ratios. [22]

Table 1.3: Citations Of Deep Learning Methods Section

Method	Ref	Year	Topic
RNN	[19]	2015	A critical review of recurrent neural networks for sequence learning
	[17]	2017	A Review of Deep Learning Methods Applied on Load Forecasting)
CNN	[22]	2023	Convolutional Neural Networks: A Survey
	[17]	2017	A Review of Deep Learning Methods Applied on Load Forecasting)
LSTM	[20]	2020	Predictive maintenance of aircraft engine using deep learning technique
	[17]	2017	A Review of Deep Learning Methods Applied on Load Forecasting)
GRU	[21]	2021	Attention based GRU-LSTM for software defect prediction
GAN	[18]	2014	Generative adversarial nets
AE	[17]	2017	A Review of Deep Learning Methods Applied on Load Forecasting)

1.7 RUL Concept

The term ”Remaining Useful Life” (RUL) describes the approximate amount of time (which can be in minutes, hours, days, or cycles) that will elapse between now and the anticipated point at which a system, component, or asset is unable to fulfill its intended function satisfactorily.[23]

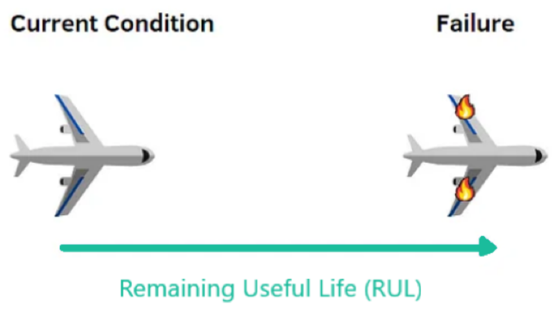


Figure 1.1: Illustration Of RUL

Figure 1.2 is adapted from the article ”Predicting the Remaining Useful Life of Turbofan Engine” by Eugenia Anello [24]
RUL prediction is used in predictive maintenance and prognostics to estimate a system’s remaining operational lifespan based on past performance and other pertinent variables. [23]

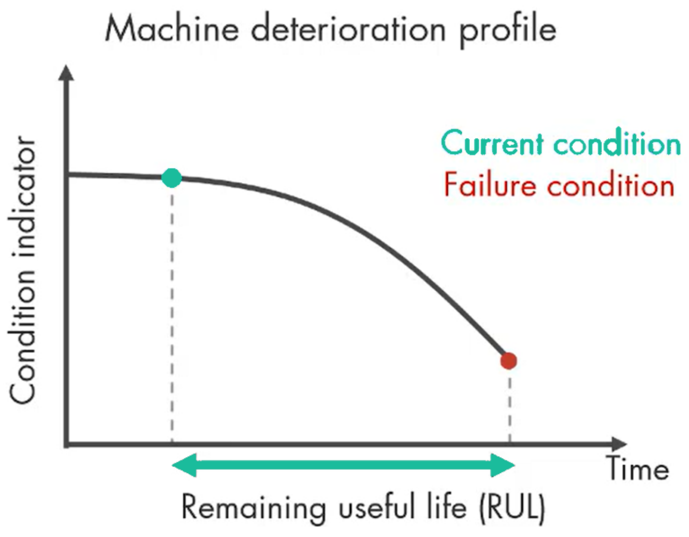


Figure 1.2: Remaining Useful Life (RUL) Timeline

Figure 1.2 is adapted from a video provided by the official website of MathWorks [25].

The smaller the RUL is, the higher the risk of failure. When the RUL is zero, it means that the engine failed.

1.7.1 RUL Methods

The prediction methods for RUL can be categorized as model-based techniques (also known as physics of failure techniques), data-driven techniques, and hybrid-based techniques. These methods are illustrated in Figure 1.3, adapted from the PhD thesis "Remaining Useful Life Prediction of a Turbofan Engine Using Deep Layer Recurrent Neural Networks" [1].

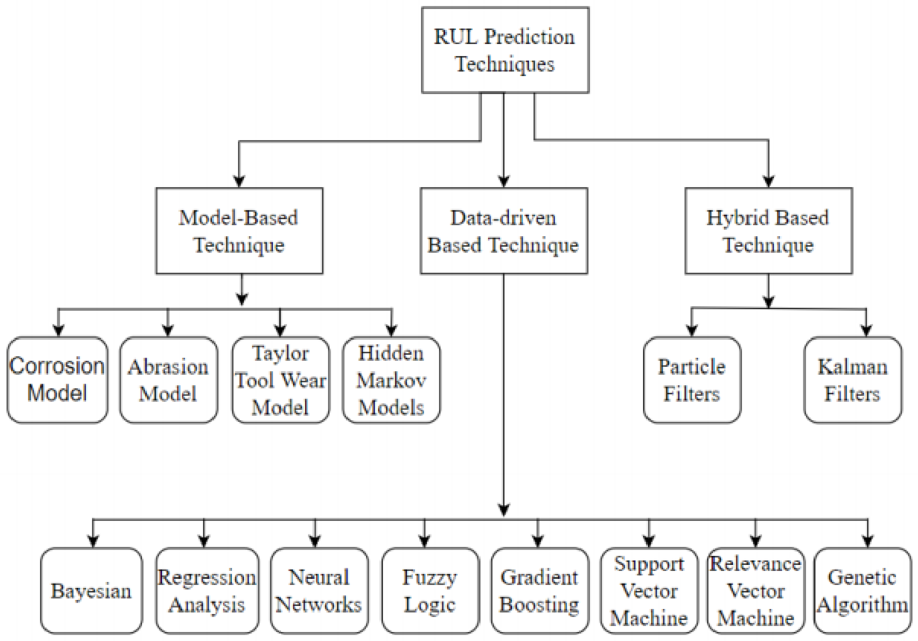


Figure 1.3: RUL prediction techniques

Model-Based Technnique

The model-based approach, also referred to as the physics of failure technique, depends on a precise physical model that reflects the system's dynamics. This method combines the physical model with calculated data to determine model parameters and predict potential behavior. It utilizes "run-to-failure" data and is suitable for maintenance decision-making. The remaining useful life (RUL) is often estimated using this model-driven approach, which helps guide maintenance decisions based on failure thresholds.[1]

Data-Driven Based Technique

The data-driven approach utilizes previously collected data (training data) to examine the characteristics of the current damage state and predict future trends. These methods rely on frequent condition-monitoring data from device metrics gathered daily. Machine learning techniques are employed in data-driven prognostics. Data-driven techniques are quick and easy to implement. However, they require substantial data, and a balanced approach, and carry the risk of overfitting.[1]

Hybrid based Techniques

A hybrid-based technique, also known as a fusion-based technique, integrates model-based and data-driven approaches. It utilizes data to learn model parameters and leverages knowledge of the physical process to determine the appropriate type of regression analysis (such as linear, polynomial, or exponential). This method forecasts Remaining Useful Life (RUL) independently and employs probability theory-based approaches to combine multiple RUL prediction outcomes into a new RUL[1].

Advantages of RUL for Aircraft Engines

In the aviation sector, estimating the Remaining Useful Life (RUL) of aircraft engines is crucial for ensuring operational effectiveness, dependability, and safety.
Predicting RUL offers several key advantages in aviation engine maintenance and operation. First, by predicting RUL, potential issues or failures in aviation engines can be identified proactively, enhancing flight safety and reducing the risk of in-flight incidents due to engine problems. Second, precisely forecasting RUL enables optimized maintenance schedules, ensuring essential maintenance is performed at the right times, reducing downtime, and maintaining engine health. Third, predictive maintenance based on RUL prediction reduces the likelihood of costly problems, extends the life of engine components, and lowers unplanned maintenance expenses, leading to significant cost savings for airlines. Additionally, anticipating RUL helps airlines meet maintenance and safety standards, ensuring airworthiness and compliance with regulations by keeping engines in good working condition.[23]

1.8 Related Works

In the field of aviation, there have been significant advancements in predictive methods for various applications such as engine health monitoring, performance prediction, engine aging prediction, and fault detection.

Table 1.4: Methods For Predicting The Health Of Aircraft Engines

Method	Citation	Title
Deep Neural Network	[26]	Aircraft Engine Health Prediction Using Deep Neural Networks
	[27]	Deep Learning Approach for Aircraft Engine Aging Prediction
	[28]	Deep Learning-Based Fuel Consumption Prediction for Aircraft Engines
	[29]	Deep Learning for Aircraft Engine Fault Detection and Troubleshooting
SVM	[30]	Support Vector Machines for Engine Aging Prediction
RNN	[31]	RNNs for Aircraft Engine Parameter Prediction
GBM	[32]	A light gradient boosting machine for remaining useful life estimation of aircraft engines
	[33]	A proposed algorithm based on gradient boosting for aero-engine thrust estimation on transition state.
	[34]	Aircraft engine reliability analysis using machine learning algorithms.
CNN-LSTM	[35]	Remaining useful life predictions for turbofan engine degradation based on concurrent semi-supervised model
	[36]	A Directed Acyclic Graph Network Combined With CNN and LSTM for Remaining Useful Life Prediction
	[37]	CNNs and LSTM for Engine Health Monitoring

Note that there exist modern models, such as semi-supervised models that combine supervised and unsupervised approaches. These models, when applied to the same data studied in this thesis, have demonstrated superior performance. For instance, the article titled 'Remaining useful life predictions for turbofan engine degradation based on a concurrent semi-supervised model' [35] was published by Springer in 2022.

1.9 Comparison of Anomaly Detection Methods for Engine Applications

Here, we present a comparison table summarizing the characteristics and performance of various anomaly detection methods for engine applications.
This table highlights the sensitivity, complexity, and dataset suitability of these methods, serving as a guide for selecting the most appropriate methods.

Table 1.5: Table of Comparison Between the Different Methods Used in Prediction

Method	Detection Sensitivity	Computational Complexity	Dataset Size Suitability
Z-Score	Medium	Low	Small-Medium
IQR	Low-Moderate	Low	Small-Medium
PCA	Moderate	Low-Medium	Large
GB	High	Medium	Small-Large
IF	High	Low-Medium	Small-Large
Random Forest	High	Medium	Small-Medium
LOF	High	Medium	Small-Large
Decision Trees	Low-Moderate	Low	Small-Medium
SVMs	High	Medium	Small-Medium
KNN	High	Medium	Small-Medium
Naive Bayes	Moderate	Low	Small-Medium
Autoencoders	High	High	Large
CNNs	Moderate-High	High	Large
GANs	High	High	Large
LSTM	High	High	Large
RNNs	Moderate-High	High	Large
GRU	High	Medium	Medium-Large

Following the analysis provided in the table are the top five (5) methods we will consider the most, based on how well they will suit our Remaining Useful Life dataset These methods will; have higher sensitivity in detection, be employed on large datasets. It is important to note that while the methods CNN, LSTM, and GRU are included primarily due to their renowned efficacy in predictive tasks, the remaining methods were chosen for their proven performance on larger datasets.

1.10 Conclusion

In conclusion, this thesis will focus on Gradient Boosting, Support Vector Machines, Convolutional Neural Networks, Long Short-Term Memory networks, and Gated Recurrent Units. These methods are chosen due to their high sensitivity in detection processes and their suitability for working with large datasets, making them optimal for predictive maintenance.

Chapter 02 :

Architecture And Modeling

Chapter 2

Architecture And Modeling

2.1 Introduction

This chapter presents a review of all the Machine and Deep learning methods we chose in the first chapter to be employed in predicting the Remaining Useful Life for aircraft engines. The methods to be examined include Gradient Boosting, Support Vector Machines, Long Short-Term Memory, Gated Recurrent Units, and Convolutional Neural Networks. The chapter delves into the methodologies architecture, including their structures and algorithms for predicting the Remaining Useful Life (RUL).

2.2 Architecture Of the Used Prediction Models

2.2.1 Gradient Boosting

Gradient Boosting, known as the thrust method, is a powerful boosting algorithm that combines several weak learners into strong learners, in which each new model is trained to minimize the loss function such as mean squared error of the previous model using gradient descent. In each iteration, the algorithm computes the gradient of the loss function concerning the predictions of the current ensemble and then trains a new weak model to minimize this gradient. The predictions of the new model are then added to the ensemble, and the process is repeated until a stopping criterion is met.[9]
The following explanation is inspired by the official website GeeksforGeeks [9].
There is a technique called the Gradient Boosted Trees whose base learner is CART (Classification and Regression Trees) as shown in figure 2.1 [9].

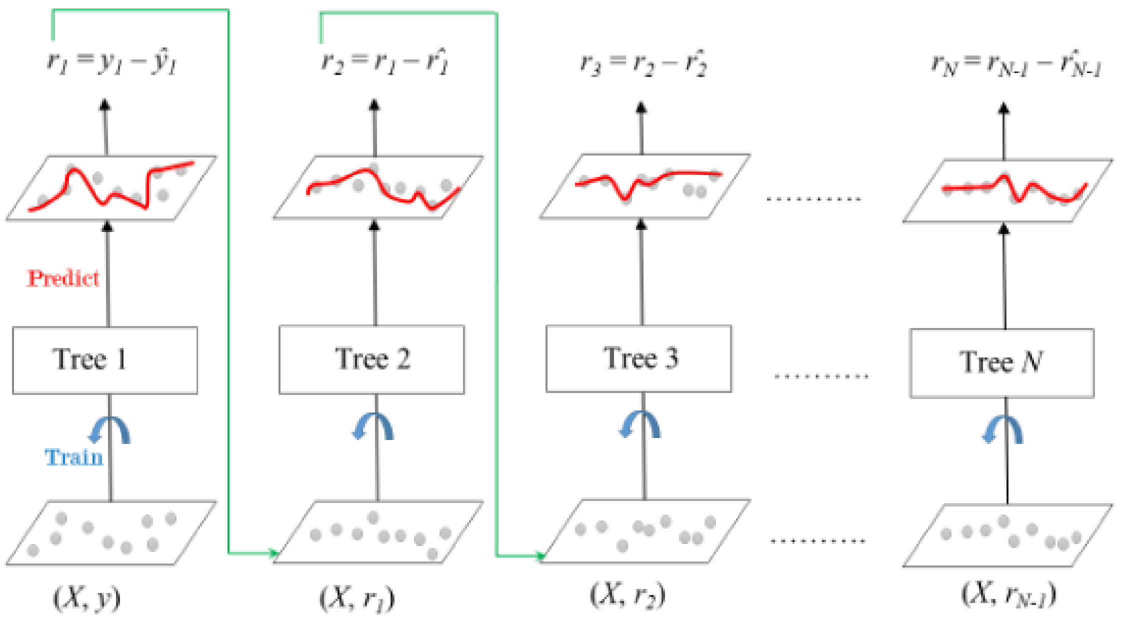


Figure 2.1: Gradient Boosted Trees for Regression

The ensemble consists of N trees. Tree 1 is trained using the feature matrix X and the labels y. The predictions labeled y1(hat) determine the training set residual errors r1. Tree 2 is then trained using the feature matrix X and the residual errors r1 of Tree 1 as labels. The predicted results r1(hat) are then used to determine the residual r2. The process is repeated until all the ensemble's N trees are trained. There is an important parameter used in this technique known as Shrinkage. Shrinkage refers to the fact that the prediction of each tree in the ensemble is shrunk after it is multiplied by the learning rate (eta) which ranges between 0 and 1. There is a trade-off between eta and the number of estimators, decreasing learning rate needs to be compensated with increasing estimators to reach certain model performance. Since all trees are trained now, predictions can be made. Each tree predicts a label and the final prediction is given by the formula,

$$y_{\text{pred}} = y_1 + \eta \cdot r_1 + \eta \cdot r_2 + \cdots + \eta \cdot r_N \tag{2.1}$$

Gradient Boosting Implemented in Python

Gradient boosting is well-established through Python libraries. Here are the main four:

- XGBoost: eXtreme Gradient Boosting
- LightGBM: Light Gradient Boosting Machine

- CatBoost: Categorical Boosting
- Scikit-learn: Has two estimators for regression and classification

The first three libraries are similar to each other: Dominant performance, GPU support, Rich set of hyperparameters, and Very high community support.
A popular alternative to all these three libraries is Scikit-learn, on its own, is more popular than the three libraries put together[9].
Besides, gradient boosting models built with Scikit-learn could be integrated into its rich ecosystem like cross-validation estimators, data processors, etc.
However, we have to remember that Scikit-learn, has the disadvantage of being CPU-only. Since gradient boosting is a computation-heavy algorithm, running it on a CPU could be infeasible for large datasets.
Note that in this thesis, we will be using the Scikit-learn library.

Gradient Boosting Algorithm

This algorithm is inspired by the article titled ' A comparative analysis of gradient boosting algorithms ' published by Springer in 2021[38].

Given a training dataset $D = \{(x_i, y_i)\}_{i=1}^N$, the goal of gradient boosting is to find an approximation, $\hat{F}(x)$, of the function $F^*(x)$, which maps instances x to their output values y, by minimizing the expected value of a given loss function, $L(y, F(x))$. Gradient boosting builds an additive approximation of $F^*(x)$ as a weighted sum of functions:

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x), \tag{2.2}$$

where γ_m is the weight of the m-th function, $h_m(x)$. These functions are the models of the ensemble (e.g. decision trees). The approximation is constructed iteratively. First, a constant approximation of $F^*(x)$ is obtained as

$$F_0(x) = \arg\min_c \sum_{i=1}^N L(y_i, c). \tag{2.3}$$

Subsequent models are expected to minimize

$$(\gamma_m, h_m) = \arg\min_{\gamma,h} \sum_{i=1}^N L(y_i, F_{m-1}(x_i) + \gamma h(x_i)). \tag{2.4}$$

However, instead of solving the optimization problem directly, each h_m can be seen as a greedy step in a gradient descent optimization for F^*. For that, each model, h_m, is trained on a new dataset $D = \{(x_i, r_{mi})\}_{i=1}^{N}$, where the pseudo-residuals, r_{mi}, are calculated by

$$r_{mi} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}. \tag{2.5}$$

The value of γ_m is subsequently computed by solving a line search optimization problem.

Note: Gradient boosting employs gradient descent to solve optimization problems due to its straightforward application. However, gradient descent is also known for its potential instability.

2.2.2 Support Vector Machines(SVM)

The most basic idea of the SVM algorithm is to find a hyperplane in the sample space based on the training set, separating the samples of different categories.[39]
This algorithm is inspired by the article titled 'Research and Application of AdaBoost Algorithm Based on SVM' published by IEEE in 2019 [39].

Mathematical Formulation

The following linear equation can describe the division of the hyperplane:

$$\mathbf{w}^T\mathbf{x} + b = 0 \tag{2.6}$$

As can be seen from the above expression, the hyperplane is determined by the normal vector $\mathbf{w}$ and the displacement b. Now assume that a point p is known on the plane, the distance from p to the hyperplane is represented by d, and the formula for calculating d is as follows:

$$d = \frac{\mathbf{w}^T\mathbf{x} + b}{\|\mathbf{w}\|} \tag{2.7}$$

To allow the hyperplane to classify the training samples correctly, we hope that the two heterogeneous support vectors on both sides of the hyperplane can have a "maximum interval:

$$\max = \frac{2}{\|\mathbf{w}\|} \tag{2.8}$$

For the convenience of calculation, it is usually rewritten as:

$$\min = \frac{1}{2}\|\mathbf{w}\|^2 \tag{2.9}$$

Where:

- $\mathbf{w}$ is the weight vector.
- b is the bias term.

As illustrated in Figure 2.2, the token is derived from the article "Research and Application of AdaBoost Algorithm Based on SVM" (2019) [39].

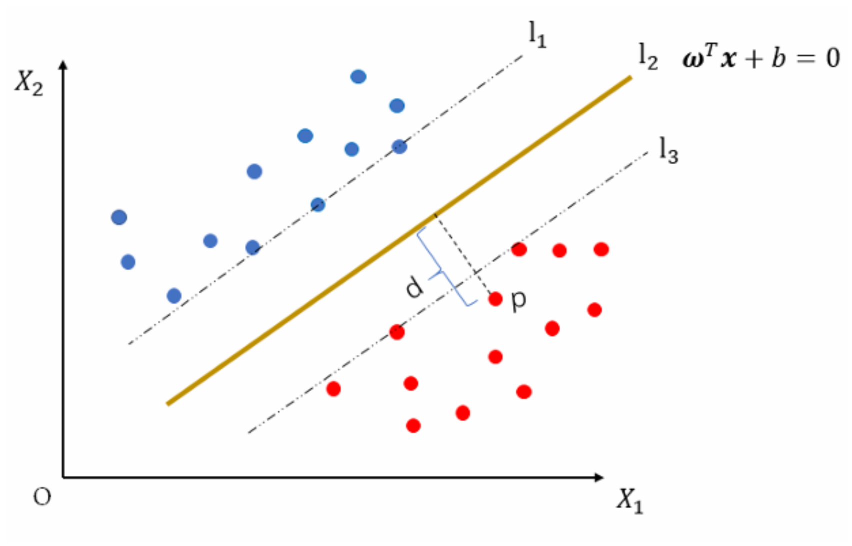


Figure 2.2: Ilustration Of Support Vector Machines(SVM)

2.2.3 Long Short Term Memory (LSTM)

A traditional RNN has a single hidden state that is passed through time, making it difficult for the network to learn long-term dependencies. LSTMs address this problem by introducing a memory cell, which is a container that can hold information for an extended period. LSTM networks are capable of learning long-term dependencies in sequential data[20].
The memory cell is controlled by three gates: the input gate, the forget gate, and the output gate. These gates decide what information to add to, remove from, and output from the memory cell. The input gate controls what information is added to the memory cell. The forget gate controls what information is removed from the memory cell. And the output gate controls what information is output from the memory cell.[17]

This algorithm is inspired by the articles titled "A Review of Deep Learning Methods Applied on Load Forecasting" (IEEE, 2017) [17] and "Predictive Maintenance of Aircraft Engine Using Deep Learning Technique" (IEEE, 2020) [20].

LSTM Cell Structure

An LSTM network consists of a series of LSTM cells, each with the following components:

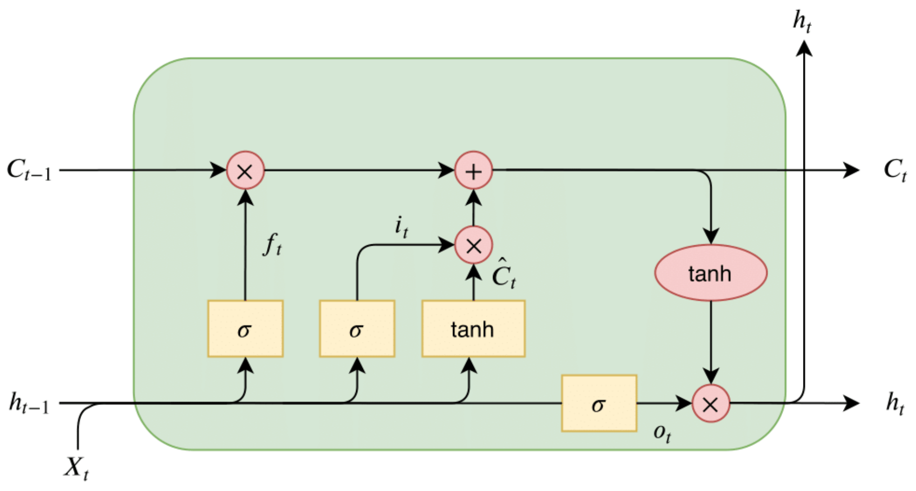


Figure 2.3: Structure of an LSTM cell

With:

- Input gate (i_t)

- Forget gate (f_t)

- Output gate (o_t)

- Cell state (C_t)

- Hidden state (h_t)

Mathematical Formulation

At each time step t, the LSTM cell updates its states and outputs based on the following equations:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2.10}$$

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2.11}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{2.12}$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{2.13}$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \tag{2.14}$$

$$h_t = o_t \odot \tanh(C_t) \tag{2.15}$$

where:

- x_t is the input at time step t.

- h_{t-1} is the hidden state from the previous time step.

- i_t, f_t, and o_t are the input, forget, and output gates, respectively.

- $\tilde{C}_t$ is the candidate cell state.

- C_t is the cell state.

- σ is the sigmoid activation function.

- tanh is the hyperbolic tangent function.

- W_i, W_f, W_o, W_C are weight matrices.

- b_i, b_f, b_o, b_C are bias vectors.

- $\odot$ denotes element-wise multiplication.

Explanation of Gates

- **Input Gate (i_t):** Determines how much of the new information from the input $\tilde{C}_t$ should be added to the cell state C_t.

- **Forget Gate** (f_t): Decides how much of the previous cell state C_{t-1} should be carried forward to the current cell state C_t.
- **Output Gate** (o_t): Controls the amount of the cell state C_t to be exposed to the output as hidden state h_t.
- **Candidate Cell State** ($\tilde{C_t}$): Represents the new information generated from the current input and previous hidden state.
- **Cell State** (C_t): Acts as the memory of the network, accumulating important information across time steps.
- **Hidden State** (h_t): Represents the output of the LSTM cell at each time step, incorporating both current input and past information.

2.2.4 Gated Recurrent Unit (GRU)

Gated Recurrent Unit (GRU) is a type of recurrent neural network (RNN) that was introduced by Cho et al. in 2014 it is designed to address the vanishing gradient problem and improve the learning of long-term dependencies.[21]
The basic idea behind GRU is to use gating mechanisms to selectively update the hidden state of the network at each time step. The gating mechanisms are used to control the flow of information in and out of the network. The GRU has two gating mechanisms, called the reset gate and the update gate [21].
The reset gate determines how much of the previous hidden state should be forgotten, while the update gate determines how much of the new input should be used to update the hidden state[21].

This algorithm is inspired by the article titled 'Attention-Based GRU-LSTM for Software Defect Prediction'[21].

GRU Cell Structure

A GRU network consists of a series of GRU cells, each with the following components:

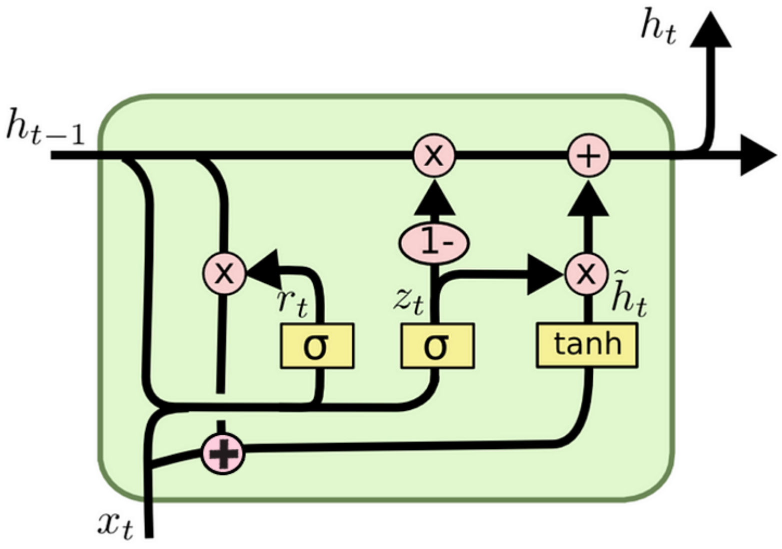


Figure 2.4: Structure of a GRU cell

With:

- Update gate (z_t)

- Reset gate (r_t)

- Hidden gate (h_t)

- Candidate hidden state $\tilde{h}_t$

Mathematical Formulas

Given input sequence $\{x_1, x_2, \ldots, x_T\}$:

- **Update gate z_t:**

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z) \tag{2.16}$$

- **Reset gate r_t:**

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t] + b_r) \tag{2.17}$$

- **Candidate hidden state $\tilde{h}_t$:**

$$\tilde{h}_t = \tanh(W \cdot [r_t \odot h_{t-1}, x_t] + b) \tag{2.18}$$

- **New hidden state h_t:**

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \tag{2.19}$$

With: σ: Sigmoid activation function.

tanh: Hyperbolic tangent activation function.

$\odot$: Element-wise multiplication.

W_z, W_r, W: Weight matrices for the update gate, reset gate, and candidate hidden state.

b_z, b_r, b: Bias vectors for the update gate, reset gate, and candidate hidden state.

h_t: Hidden state at time step t.

h_{t-1}: Hidden state at the previous time step $t - 1$.

x_t: Input at time step t.

2.2.5 Convolutionnal Neural Network (CNN)

Convolutional Neural Networks (CNNs) are a class of deep learning models specifically designed for processing structured grid data, such as images. The key components of a CNN are convolutional layers, pooling layers, and fully connected layers.[17]

This algorithm is inspired by the article titled 'A Review of Deep Learning Methods Applied on Load Forecasting' published by IEEE in 2017 [17].

Architecture

A typical CNN consists of the following layers:

- Input Layer
- Convolutional Layer
- Activation Function (ReLU)
- Pooling Layer
- Fully Connected Layer
- Output Layer

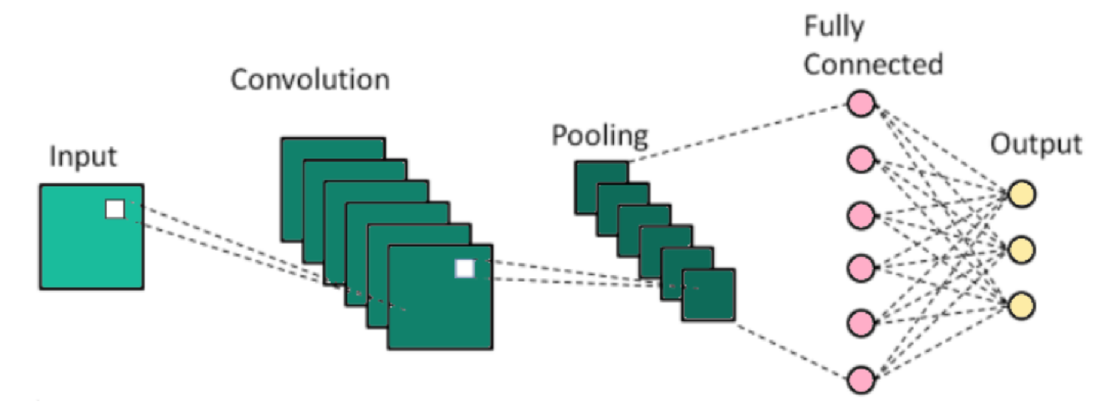


Figure 2.5: Architecture of a CNN Model

Explanation of Terms

- **Convolutional Layer:** This layer applies a convolution operation(involving a filter (or kernel) across the input data, computing the dot product between the filter and the input at every position) to the input, passing the result to the next layer.
- **Filter (Kernel):** A small matrix is used to detect features in the input.
- **Stride:** The number of pixels by which the filter matrix is moved across the input matrix.
- **Padding:** Adding extra pixels around the input matrix to control the spatial size of the output.
- **Pooling Layer:** Reduces the dimensionality of each feature map while retaining the most important information.
- **Max Pooling:** A pooling operation that selects the maximum element from the region of the feature map covered by the filter.
- **Flattening:** Converts the pooled feature map into a single-column vector.
- **Fully Connected Layer:** A layer where each neuron is connected to every neuron in the previous layer.
- **Activation Function:** A function applied to each neuron output to introduce non-linearity (e.g., ReLU, sigmoid).

Algorithm Steps

1. Convolutional Layer The convolutional layer applies a set of learnable filters (kernels) to the input image to extract features such as edges, textures, and patterns.

- **Convolution Operation:**

$$y_{i,j} = (x * w)_{i,j} = \sum_{m=0}^{k_H-1} \sum_{n=0}^{k_W-1} \sum_{c=0}^{C-1} x_{i+m,j+n,c} w_{m,n,c} \tag{2.20}$$

  where x is the input image, w is the filter, and y is the feature map.
  Each filter w slides over the input x and computes the dot product between the filter and a local region of the input image to produce a feature map y.

2. Activation Function The activation function introduces non-linearity into the model. The most common activation function used in CNNs is the Rectified Linear Unit (ReLU).[22]

- **ReLU Function:**

$$\text{ReLU}(z) = \max(0, z) \tag{2.21}$$

where z is the input to the activation function.

ReLU sets all negative values in the feature map to zero, allowing the network to learn non-linear relationships.

3. Pooling Layer The pooling layer reduces the dimensionality of the feature maps, making the computation more efficient and reducing the risk of overfitting. The most common type of pooling is max pooling.[22]

- **Max Pooling:**

$$y_{i,j} = \max_{0 \le m \le P_H - 1} \max_{0 \le n \le P_W - 1} x_{i+m\,,\,j+n} \tag{2.22}$$

where x is the input feature map, and y is the pooled feature map.

Max pooling takes the maximum value from a set of neighboring values (usually in a 2×2 window), reducing the spatial dimensions of the feature map.

4. Fully Connected Layer The fully connected layer takes the high-level features produced by the convolutional and pooling layers and uses them to classify the image into a label.

$$\hat{y} = \sigma(W \cdot x + b) \tag{2.23}$$

where W is the weight matrix, x is the input vector (flattened feature map), b is the bias vector, and σ is an activation function.

Note: The convolutional neural network uses stochastic gradient descent (SGD) to solve its optimization problem.

2.2.6 CNN-LSTM Approach

The combination of Convolutional Neural Networks (CNN) and Long Short-Term Memory networks (LSTM) forms a hybrid architecture that combines the strengths of both networks by first using CNN layers to extract spatial features from the input data, and then feeding these features into LSTM layers to learn the temporal dependencies. This approach is particularly useful for time-series prediction tasks.[35][40]

This approach is inspired by the article titled 'Intrusion Detection System in the Advanced Metering Infrastructure: A Cross-Layer Feature-Fusion CNN-LSTM-Based Approach' (2021) [40].

Architecture Overview

- **Input Layer**: The raw input data is typically a time-series signal or a sequence of observations over time.
- **CNN Layers**: Convolutional layers are applied to the input data to extract high-level features. This may include multiple convolutional and pooling layers to capture different levels of abstraction.
- **Flatten Layer**: The output of the CNN layers is flattened to a 1D vector, which serves as the input to the LSTM layers.
- **LSTM Layers**: LSTM layers process the sequential data, capturing the temporal dependencies and providing a rich representation of the temporal dynamics.
- **Dense Layers**: Fully connected layers may be used after the LSTM layers to perform the final regression or classification task, such as predicting the RUL.
- **Output Layer**: The final output layer provides the prediction, which in this case would be the RUL.

Mathematical Formulation of CNN-LSTM

Let X_t represent the input sequence at time t. The CNN layer applies a series of convolutions and pooling operations to extract features F_t:

$$F_t = \text{CNN}(X_t) \tag{2.24}$$

The features F_t are then fed into the LSTM layer:

$$h_t = \text{LSTM}(F_t, h_{t-1}) \tag{2.25}$$

The final prediction is obtained through a dense(fully connected) layer:

$$\hat{y}_t = \text{Dense}(h_t) \tag{2.26}$$

The output is $\mathbf{y} \in \mathbb{R}$, the predicted RUL. The dense layer operation is given by:

$$\mathbf{y} = \mathbf{W}_d \cdot \mathbf{h}_t + \mathbf{b}_d \tag{2.27}$$

The dense layer applies a linear transformation to h_t using weights $\mathbf{W}_d$ and biases $\mathbf{b}_d$ to produce the final prediction y.

Advantages of CNN-LSTM

- **Spatial-Temporal Feature Extraction**: CNNs effectively capture local spatial patterns, while LSTMs capture long-term dependencies.
- **Improved Performance**: By leveraging both spatial and temporal features, CNN-LSTM models often perform better in sequential data tasks than using CNN or LSTM alone.
- **Flexibility**: This architecture can be applied to various time-series prediction problems.

2.3 Data Processing

Data Processing refers to the series of operations performed on the data to transform it into a structured, clean, and meaningful format. This step is crucial in ensuring that the data is accurate, consistent, and free from errors or noise that could affect the performance of machine learning models.

2.3.1 StandardScaler

The `StandardScaler` is a preprocessing tool from the `sklearn.preprocessing` module that standardizes features by removing the mean and scaling to unit variance. This process is also known as z-score normalization.
The Z-Score normalization is inspired by the article titled ' Reliability Engineering and System Safety' (2019) [41]

Mathematically, the `StandardScaler` transforms each feature x_i in the dataset as follows:

$$z_i = \frac{x_i - \mu}{\sigma} \tag{2.28}$$

With: For each feature x in the dataset, calculate the mean (μ) and the standard deviation (σ).

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{2.29}$$

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2} \tag{2.30}$$

where N is the number of samples.
And:

- x_i is the original feature value,
- μ is the mean of the feature values,
- σ is the standard deviation of the feature values,
- z_i is the standardized feature value.

This transformation results in a new feature z_i with a mean of 0 and a standard deviation of 1. The process is applied to each feature independently.

2.3.2 Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a dimensionality reduction technique that transforms data into a set of orthogonal (uncorrelated) components. This transformation reduces the dimensionality of the dataset while preserving as much variance as possible.

PCA computes the eigenvectors of the covariance matrix and maps the original data onto a lower dimensional feature space defined by eigenvectors with large eigenvalues.[42]

Architecture Overview

This explanation is inspired by the article titled 'Comparison of Feature Extraction with PCA and LTP Methods and Investigating the Effect of Dimensionality Reduction in the Bat Algorithm for Face Recognition '[42].

The training dataset $\{X_1, X_2, X_3, \ldots, X_N\}$ is considered. The average face of the set is defined by this relation:

$$\bar{X} = \frac{1}{N}\sum_{i=1}^{N} X_i \tag{1}$$

The estimation covariance matrix should be calculated to show the degree of dispersion in all the feature vectors that are related to the mean vector. The covariance matrix Q is defined by the following:

$$Q = \frac{1}{N}\sum_{i=1}^{N}(X - X_i)(X - X_i)^T \tag{2.31}$$

The eigenvectors and their corresponding eigenvalues are calculated using

$$QV = \lambda V \quad (V \neq 0) \tag{2.32}$$

where V is the set of the matrix of eigenvectors Q, which is related to its own eigenvalue, i.e., λ. All educational data of the i-th entity is embedded in a specialized subspace:

$$y_i^k = W^T(x_i) \quad (i = 1, 2, \ldots, N) \tag{2.33}$$

where y_i^k are called principal components.

2.3.3 Windowing Process

Windowing is a crucial data preprocessing technique used in time-series analysis to convert a sequence of observations into smaller, overlapping segments called windows. This approach helps in capturing temporal patterns and dependencies within the data, making it suitable for models like CNN-LSTM that require fixed-size inputs.

Architecture Overview

Given an input sequence $\{x_1, x_2, \ldots, x_T\}$, the windowing process creates a set of overlapping windows, each of length W. For a window size W and stride S, the i-th window W_i is defined as:

$$W_i = \{x_{iS+1}, x_{iS+2}, \ldots, x_{iS+W}\}$$

where $i = 0, 1, 2, \ldots, \left\lfloor \frac{T-W}{S} \right\rfloor$.

This means that for each window W_i, the elements are taken from the input sequence starting from index $iS + 1$ to $iS + W$.

2.4 Performance Assessment

In most recent research, the root-mean-square error (RMSE) is used to assess models[41][17]. The RMSE is a parameter that is normally used in data processing problems[41].
The Root Mean Squared Error (RMSE) is a standard way to measure the error of a model in predicting quantitative data. It is particularly useful in the context of regression models.
The mathematical formulas are inspired by the article titled 'A Review of Deep Learning Methods Applied on Load Forecasting' published by IEEE in 2017 [17].

Compute RMSE for each fold

$$\text{RMSE}_j = \sqrt{\frac{1}{N_{\text{val},j}} \sum_{i=1}^{N_{\text{val},j}} (y_{\text{val},j,i} - \hat{y}_{\text{val},j,i})^2} \tag{2.34}$$

Average RMSE

$$\text{Average RMSE} = \frac{1}{k} \sum_{j=1}^{k} \text{RMSE}_j \tag{2.35}$$

With:

- $N_{\text{val},j}$ is the number of samples in the validation set of the j-th fold.
- $y_{\text{val},j,i}$ is the true value of the i-th sample in the validation set of the j-th fold.
- $\hat{y}_{\text{val},j,i}$ is the predicted value of the i-th sample in the validation set of the j-th fold.
- k is the number of folds in the cross-validation.

2.5 Conclusion

This chapter has provided a review of the Machine and Deep Learning methods employed in predicting the Remaining Useful Life (RUL) of aircraft engines. The methods examined include Gradient Boosting, Isolation Forest, Local Outlier Factor, Support Vector Regression, Long Short-Term Memory (LSTM), Gated Recurrent Units (GRU), and Convolutional Neural Network (CNN), as well as a combined CNN-LSTM approach.
The chapter detailed the architecture of these methods, encompassing their structures and algorithms. Additionally, it discussed the data processing algorithms used, such as StandardScaler, PCA normalization, and windowing, along with the performance assessment metric, Root Mean Square Error (RMSE).

Chapter 03 :

Implementation

Chapter 3

Implementation

3.1 Introduction

This chapter provides an overview of the development tools and software used to obtain the results discussed in this thesis. It also includes an explanation of turbofan engines as per the NASA PHM08 dataset. Additionally, it applies the methodology outlined in the second chapter to achieve the primary objective of this thesis: developing a model for predicting the Remaining Useful Life (RUL) of aircraft engines, along with the evaluation of results.

3.2 Gas Turbine Engine

A gas turbine engine is a type of turbine engine that spins pressurized gas to produce electricity or supply kinetic energy for an aircraft [1].
There are primarily five categories of gas turbine engines: turbojet, turbofan, turboprop, turboshaft, and ramjet engines.[1]
Turbofan engines are widely used in airliners due to their high thrust and efficiency. These engines work by converting fuel energy into shaft power, which drives the aircraft forward. A turbofan engine consists of several components, including an air intake, compressors, a combustion chamber, turbines, and an exhaust nozzle[1].
Here's how it works: Air is first drawn in and compressed to high pressure. It then enters the combustion chamber, mixing with fuel and igniting, producing high-velocity, hot gases. These gases flow through turbines, extracting energy to drive the compressor.

Finally, the high-speed exhaust gases exit through the nozzle, creating thrust that propels the aircraft[1].
One of the main advantages of turbofan engines is their fuel efficiency and relatively low noise levels compared to turbojet engines. However, they are larger and less efficient at very high altitudes, which are considered their drawbacks.[1]

3.2.1 Parts of a Gas Turbofan Engine

The operation of a turbofan engine is explained in detail below. The engine comprises five primary components, as depicted in Figure 3.1, adapted from Shutterstock's turbofan website [43].

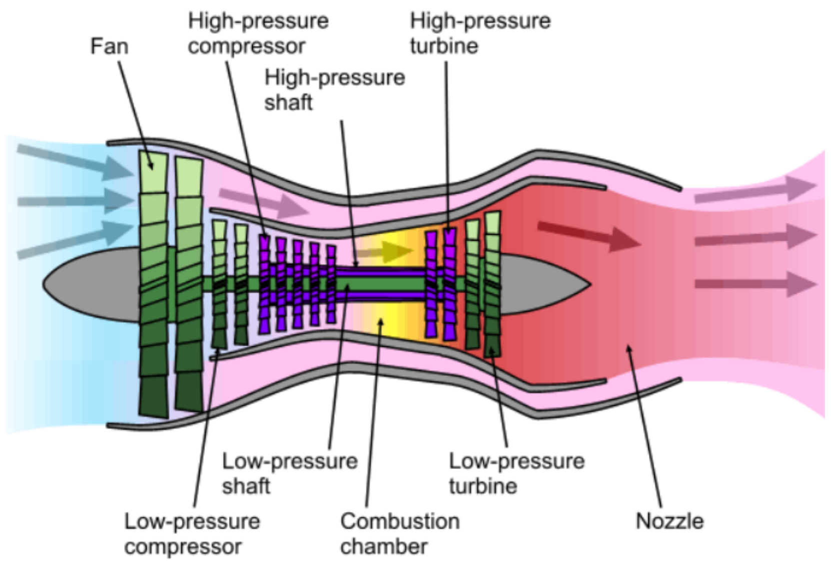


Figure 3.1: Illustration Of The Main Components of Turbofan Aircraft Engines

Figure 3.1 is adapted from the "NASA Predictive Maintenance (RUL)" notebook available on Kaggle [44].

Air inlet

Air flows through the engine from left to right, passing through the fan. The fan plays a critical role in generating the majority of the thrust produced by the turbofan engine. It is connected to the low-pressure compressor and the low-pressure turbine via a shaft. Much of the air that passes through the fan bypasses the engine's core, a pathway known as bypass air. This air never comes into contact with other engine components and is

directly expelled from the rear of the engine.[1]
The remaining air flows into the core of the engine and then enters the low-pressure compressor.

Compressor

The compressor's primary role is to raise the air's pressure and temperature, preparing it for combustion[1]. There are two main types of compressors: the low-pressure compressor (LPC) and the high-pressure compressor (HPC). The LPC is linked to the fan and the low-pressure turbine by the low-pressure shaft. The HPC is positioned just after the LPC and just before the combustor. It is connected to the high-pressure turbine by the high-pressure shaft.[1]

Combustor

The hot, high-pressure air that moves through the compressor mixes with fuel and is ignited in the combustion chamber. This process, known as combustion, leads to the production of extremely hot and fast-moving gases. These gases are then used to drive the turbine.[1]
The combustion chamber, located immediately after the compressor, is constructed from materials capable of withstanding the high temperatures produced during combustion. It is situated at the core of the engine.

Turbine

The turbine is made up of a row of spinning blades. It receives high-temperature, high-velocity gas from the combustor, which passes through the turbine, extracting energy. This energy is then transferred to the compressor and fan via a connecting shaft, helping them spin. As a result of the energy extraction by the turbine, the temperature and pressure of the air leaving the turbine are relatively low.[1]
There are two main types of turbines: the high-pressure turbine (HPT) and the low-pressure turbine (LPT). The HPT is positioned downstream of the combustor or burner and upstream of the LPT. In contrast, the LPT is located downstream of the HPT and upstream of the exhaust nozzle. [1]

Exhaust Nozzle

The exhaust nozzle is positioned at the far right of the engine, immediately after the turbine. It serves as the final passage for air exiting the engine. This tube-shaped component plays a crucial role in generating extra thrust, aiding the engine in its forward movement [1]. Moreover, it also helps regulate pressure within the engine which helps other components functioning properly.

3.2.2 Turbofan Engine Malfunction

All systems degrade over time due to factors like usage and environmental conditions. Gas turbine engines, for example, can experience harsh environmental and operational conditions such as corrosion, wear, and erosion. If a "run-to-failure" approach is taken, these issues can eventually lead to expensive and catastrophic failures.[1]
To effectively monitor gas turbine engine performance, over two hundred sensors are typically installed in each engine. These sensors help track the engine's health, which deteriorates over time. Various engine variables, known as health parameters, greatly influence engine condition. Since different engines have unique health parameters, monitoring and assessing these variables is crucial for enhancing engine lifespan, performance, and reliability.[1]
However, due to aircraft complexity and the occasional unavailability of devices to measure certain parameters, directly measuring all health parameters can be challenging.[1] Therefore, it is crucial to have fault diagnosis prognostics, and health management (PHM) for engines. Predicting an engine's remaining useful life (RUL) is particularly difficult and challenging within PHM.[1]

3.3 Data Sourcing

This thesis uses a data-driven method of anomaly identification by predicting RUL after examining the sensor values.
An available dataset from the Kaggle repository was utilized to predict RUL after examining the sensor's values. The NASA Predictive Maintenance dataset is an industrial dataset from the Kaggle repository https://www.kaggle.com/datasets/behrad3d/nasa-cmaps [45], Its sensor data is collected from industrial turbofan engines during operation.

3.3.1 C-MAPSS for Turbofan Engines RUL Prediction

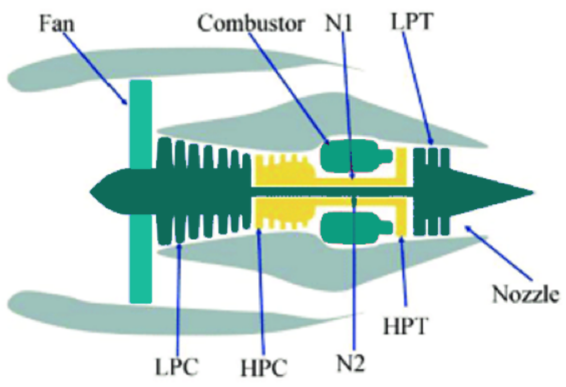


Figure 3.2: Simplified Diagram Of Engine Simulated C-MAPSS

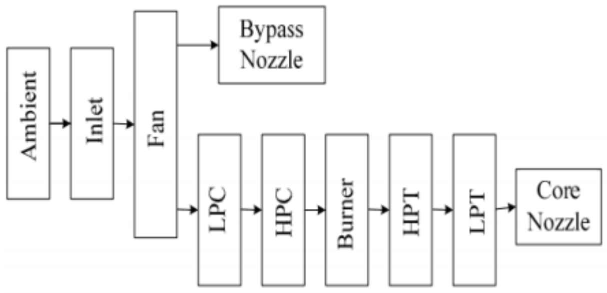


Figure 3.3: C-MAPSS modular components

Figures 3.2 and 3.3 are both referenced from the PhD thesis entitled "Remaining useful life prediction of a turbofan engine using deep layer recurrent neural networks," published in 2021 by Thakkar [1].

In this study, we distinguish between training and testing trajectories. Training trajectories are sets of full run-to-failure data that serve to train the algorithms. Conversely, test trajectories are shorter instances of data leading up to a failure point, used for testing purposes. Four different operating conditions are simulated across these trajectories, as outlined in Table 3.1 [1].
Data for each trajectory is stored in a CSV file, with each row representing a time step (measured in cycles) and containing 21 sensor measurements, three operating conditions,

and additional relevant information (see Table 3.2)[1].
In the C-MAPSS dataset, the data represents a group of similar engines, with each engine being identified by a Unit Number [1].

Table 3.1: C-MAPSS Dataset[1]

Data-Set	**Train Trajectory**	**Test Trajectory**	**Condition**	**Fault Mode**
FD-001	100	100	One (Sea Level)	One (HPC)
FD-002	260	259	six	One (HPC)
FD-003	100	100	One (Sea Level)	Two (HPC,Fan)
FD-004	248	249	Six	Two(HPC,Fan)

Table 3.2: Variables C-MAPSS Dataset[1]

Data Description
Unit number
Time (in cycles)
Operational setting 1
Operational setting 2
Operational setting 3
Sensor measurement 1
Sensor measurement 2
...
Sensor measurement 21

The engine's performance is significantly impacted by three operational settings recorded in the dataset. Each trajectory in the training sets starts with the engine operating normally, experiencing an unknown initial level of deterioration before failure is identified. Additionally, sensor noise has affected the data. In the test set, the time series stops just before the engine fails, and the Remaining Useful Life (RUL) for each test engine is provided in a separate CSV file.[1]

3.3.2 Understanding the NASA Dataset

This section delves into the characteristics of the NASA Predictive Maintenance dataset obtained from the Kaggle repository [45], which serves as the foundation for the research on anomaly detection in aircraft engines.

Data Structure and Content

These datasets comprise three operational settings and twenty-one (21) sensor measurements susceptible to sensor noise. The data is divided into three sets: training, testing, and RUL.[1]

The first two columns in the dataset indicate the unit number and operational time in cycles, while columns three, four, and five represent operational settings[1]. The remaining columns contain sensor measurements as detailed in Table 3.3

Table 3.3: Sensor Measurements[1]

	Symbol	**Description**	**Unit**
Sensor 1	T2	Total temperature at fan inlet	°R
Sensor 2	T24	Total temperature at LPC outlet	°R
Sensor 3	T30	Total temperature at HPC outlet	°R
Sensor 4	T50	Total temperature at LPT outlet	°R
Sensor 5	P2	Pressure at fan inlet	psia
Sensor 6	P15	Total pressure in bypass-duct	psia
Sensor 7	P30	Total pressure at HPC outlet	psia
Sensor 8	Nf	Physical fan speed	rpm
Sensor 9	Nc	Physical core speed	rpm
Sensor 10	epr	Engine pressure ratio	-
Sensor 11	Ps30	Static pressure at HPC outlet	psia
Sensor 12	phi	Ratio of fuel flow to Ps30	pps/psi
Sensor 13	NRf	Corrected fan speed	rpm
Sensor 14	NRc	Corrected core speed	rpm
Sensor 15	BPR	Bypass ratio	-
Sensor 16	farB	Burner fuel-air ratio	-
Sensor 17	htBleed	Bleed Enthalpy	-
Sensor 18	Nf-dmd	Demanded fan speed	rpm
Sensor 19	PCNfR-dmd	Demanded corrected fan speed	rpm
Sensor 20	W31	HPT coolant bleed	lbm/s
Sensor 21	W32	LPT coolant bleed	lbm/s

Data Goal

The primary objective associated with this dataset is to predict the Remaining Useful Life (RUL) of each engine within the test set and compare it with the true RUL provided.

3.4 Presentation of Development tools

Here is a comprehensive list of the tools utilized in this project. These tools have played a pivotal role in successfully executing the project tasks and objectives.

Table 3.4: Table of the Tools Used In The Thesis

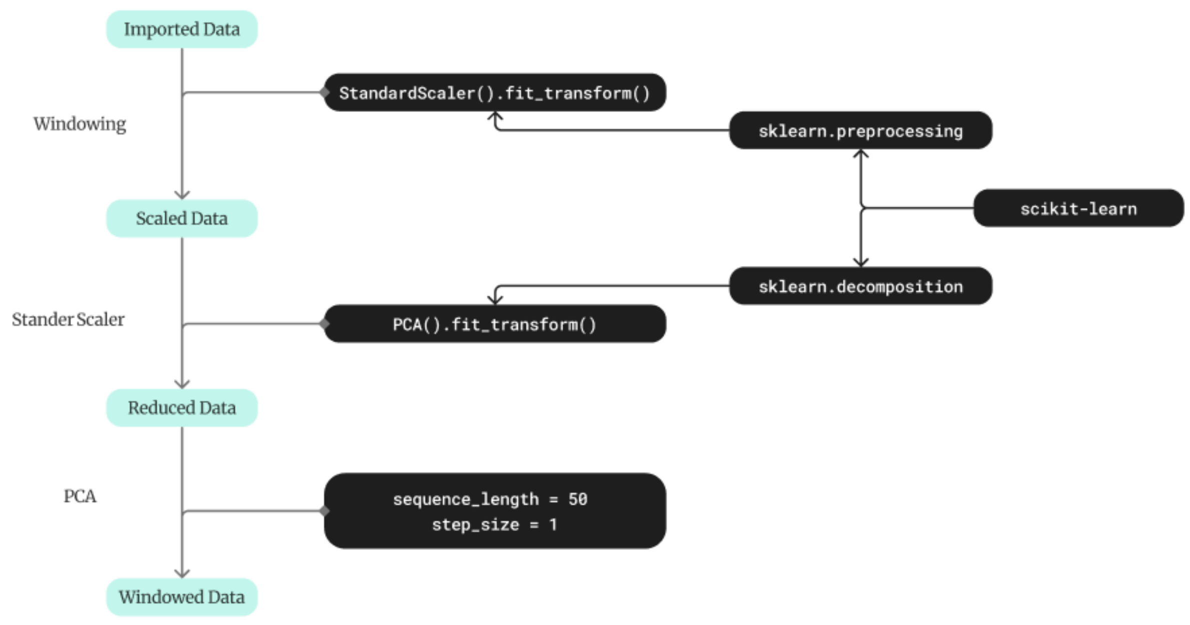
Category	**Tool**	**Description**
PROGRAMMING LANGUAGE	Python	Python is an interpreted programming language that is used for a variety of applications such as web development, data analysis, artificial intelligence, and scientific computing.[46]
PROGRAMMING TOOLS	Jupyter Notebook	Jupyter Notebook is the latest web-based interactive development environment for notebooks, code, and data. [47]
COLLABORATION TOOLS	Overleaf	Overleaf is an online LaTeX editor that allows you to create and collaborate on documents in real time. [48]
	Kaggle	Kaggle is an online community platform for data scientists and machine learning enthusiasts, it provides datasets and notebooks. [49]
	GitHub	GitHub is a web-based platform that hosts and manages Git repositories made for collaborative coding projects.[50]

3.5 Evaluation and Analysis of Results

3.5.1 Processing Datasets

In our implementation, we employ the StandardScaler class from the 'sklearn.preprocessing' module, which is part of the 'scikit-learn' library in Python. The StandardScaler standardizes the features by removing the mean and scaling them to unit variance. This is achieved through the 'fit-transform' method, which first calculates the mean and standard deviation of the training data (fit) and then applies the transformation (transform) to standardize the data. The specific code utilized is 'StandardScaler().fit-transform()', ensuring that each feature contributes equally to the model training process by having a mean of 0 and a variance of 1.

Figure 3.4: Diagram Of The main steps of Data Processing Followed IN the Code



Next, we apply Principal Component Analysis (PCA) using the PCA class from the 'sklearn.decomposition' module of the 'scikit-learn' library. PCA is utilized to reduce the dimensionality of the data by transforming it into a set of orthogonal components. These components capture the majority of the variance present in the original data. The method 'PCA().fit-transform()' is employed to fit the PCA model to the data and transform it into its principal components, retaining only the most significant components.

Finally, we use a custom windowing function to segment the data into overlapping windows, essential for capturing temporal patterns in time series data. In our implementation, we set the 'sequence-length' to 50 and the 'step-size' to 1, ensuring that each window consists of 50 data points with a stride of 1 between consecutive windows. This approach effectively preserves the temporal dependencies within the data, which is crucial for accurate time series analysis.

Each step ensures that the data is optimally prepared for the subsequent stages of model training and evaluation, as shown in Figure 3.4.

3.5.2 Compute Remaining Useful Life

The Remaining Useful Life can be computed in several ways. One method is to get each unit-id maximum time cycle first, and then make the following difference:

Maximun Time Cycle For Each Unit_id - Time Cycle For Each Row With The Specific Unit_id = Remaining Useful Life

This explanation is based on the article "Predicting the Remaining Useful Life of Turbofan Engine" by Eugenia Anello [24]

Assessment Of Models Performances

The assessment of model performance based on Expected vs. Predicted Remaining Useful Life, as illustrated in the figures below. Among the models evaluated, SVM displays considerable discrepancies between expected and predicted RUL, indicating limitations in its predictive capabilities.
In contrast, LSTM, CNN-LSTM, and CNN models exhibit notably superior performance, with expected and predicted RUL closely aligned.
Figures 3.5 to 3.28 illustrate the predicted and expected Remaining Useful Life (RUL) for the four datasets across all applied models. In these figures, red denotes the expected RUL, while green indicates the predicted RUL, as shown below:

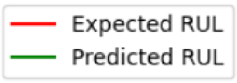


Figure 3.5: Comparison of Expected and Predicted RUL of CNN Model FD-001

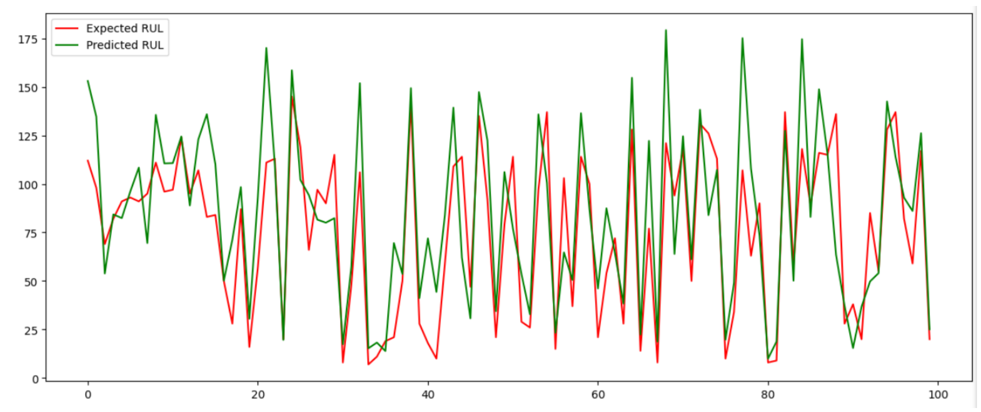


Figure 3.6: Comparison of Expected and Predicted RUL of CNN Model FD-002

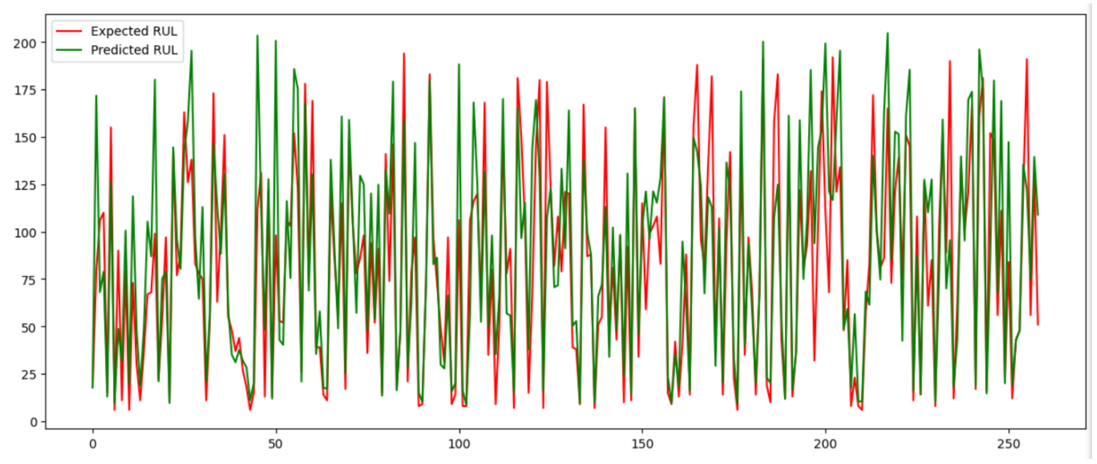


Figure 3.7: Comparison of Expected and Predicted RUL of CNN Model FD-003

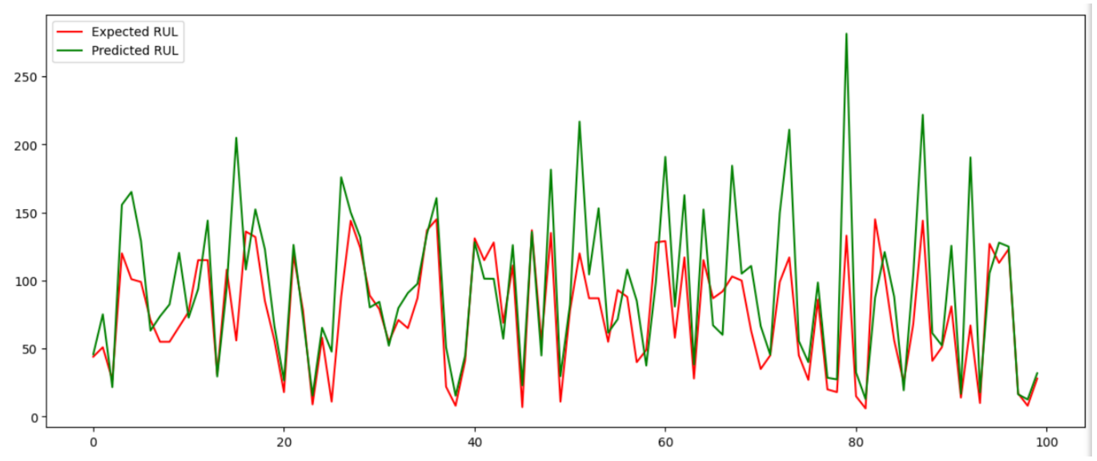


Figure 3.8: Comparison of Expected and Predicted RUL of CNN Model FD-004

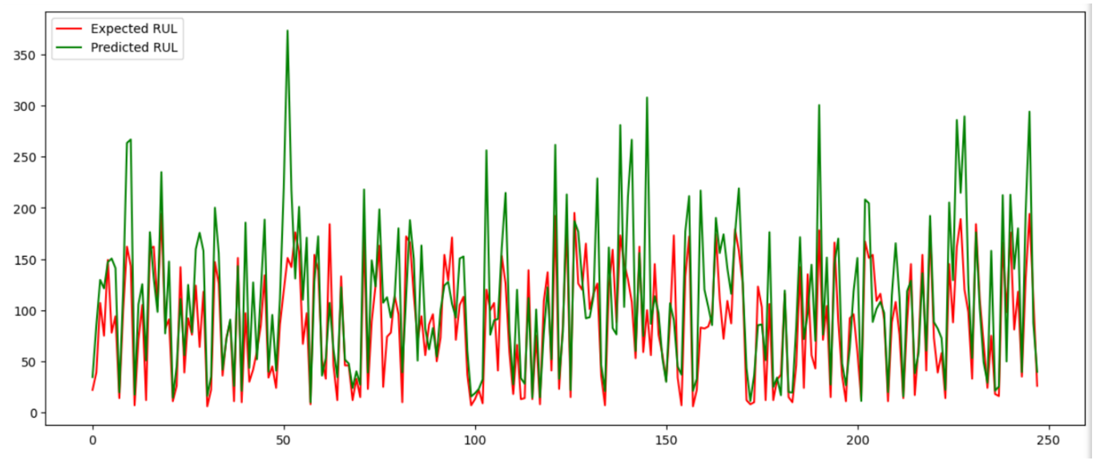


Figure 3.9: Comparison of Expected and Predicted RUL of CNN-LSTM Model FD-001

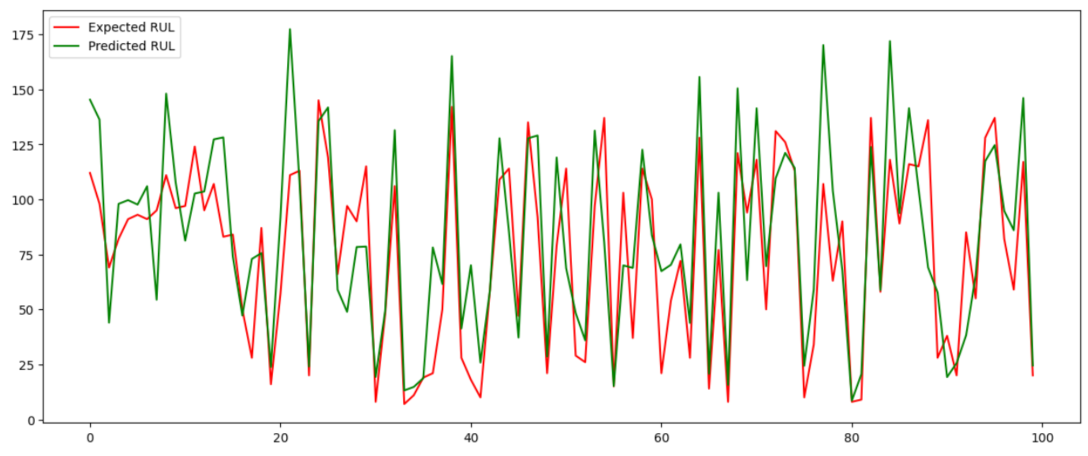


Figure 3.10: Comparison of Expected and Predicted RUL of CNN-LSTM Model FD-002

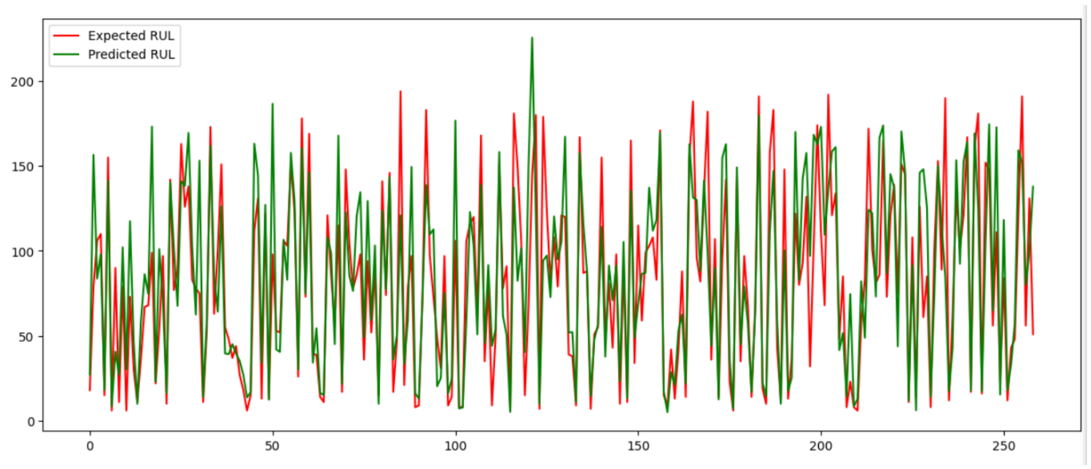


Figure 3.11: Comparison of Expected and Predicted RUL of CNN-LSTM Model FD-003

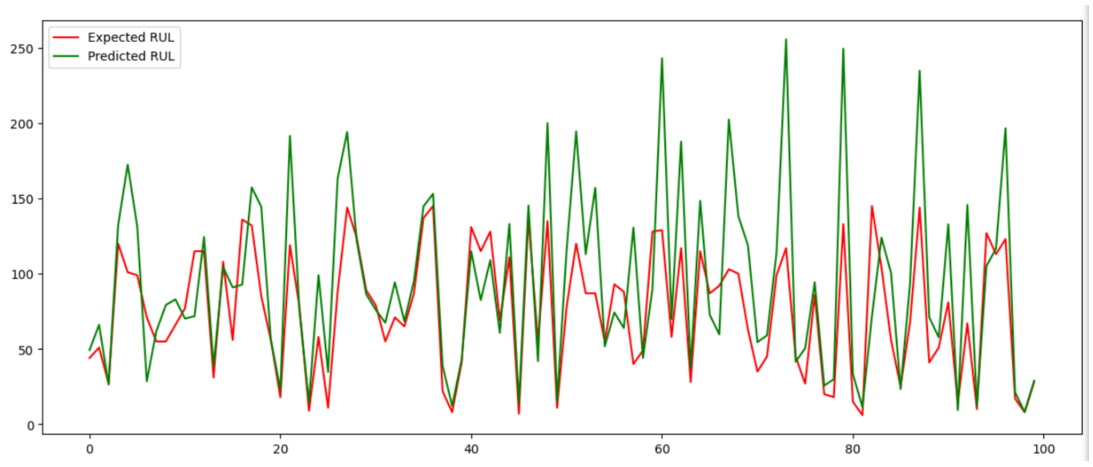


Figure 3.12: Comparison of Expected and Predicted RUL of CNN-LSTM Model FD-004

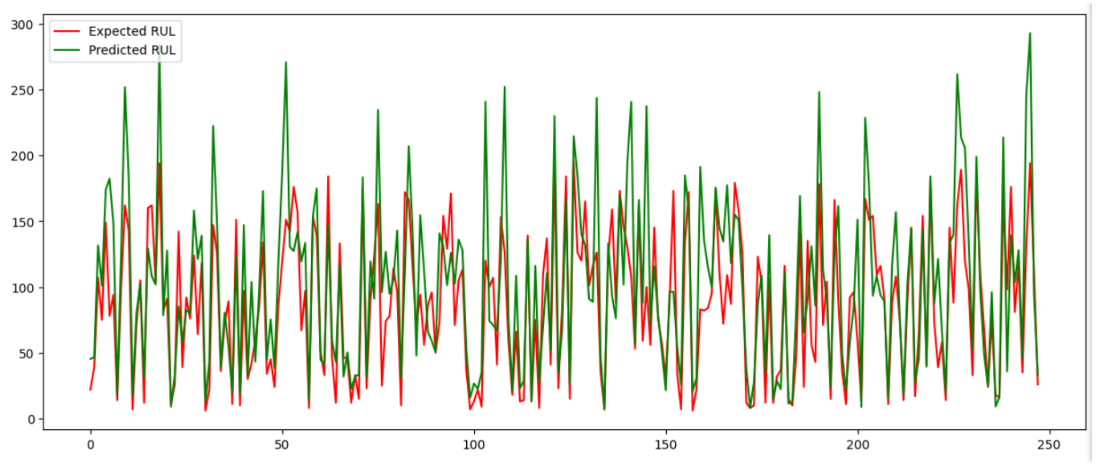


Figure 3.13: Comparison of Expected and Predicted RUL of LSTM Model FD-001

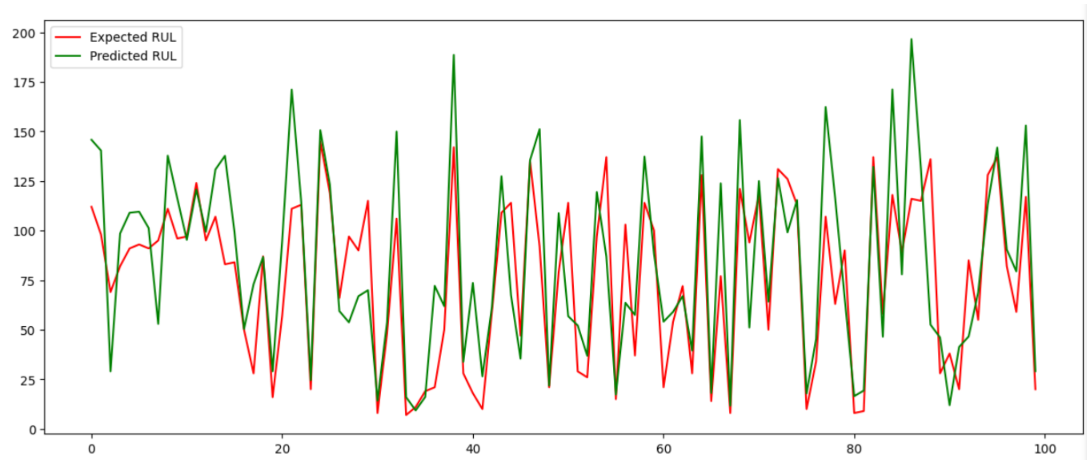


Figure 3.14: Comparison of Expected and Predicted RUL of LSTM Model FD-002

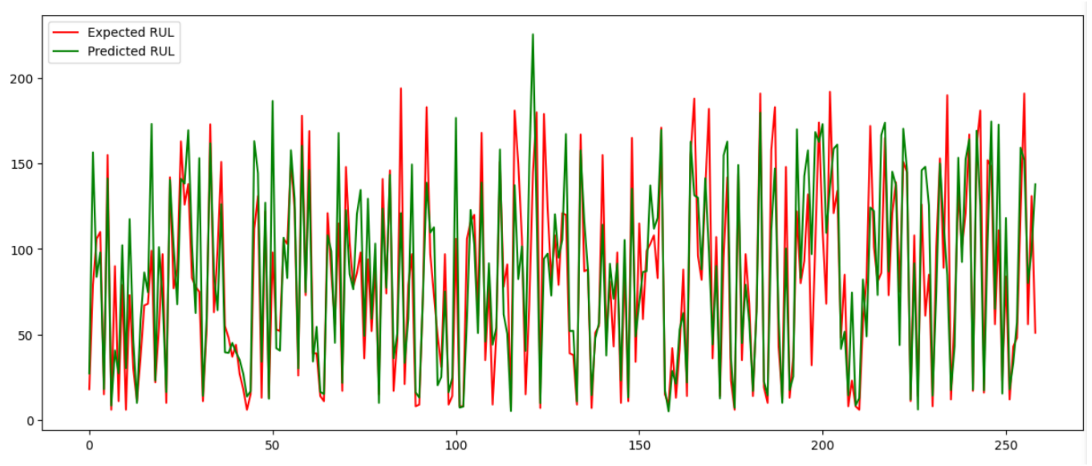


Figure 3.15: Comparison of Expected and Predicted RUL of LSTM Model FD-003

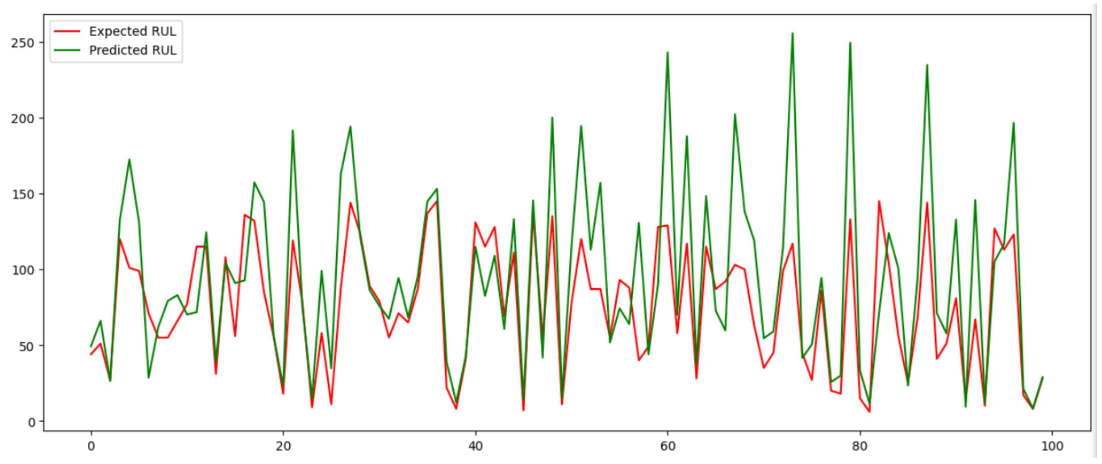


Figure 3.16: Comparison of Expected and Predicted RUL of LSTM Model FD-004

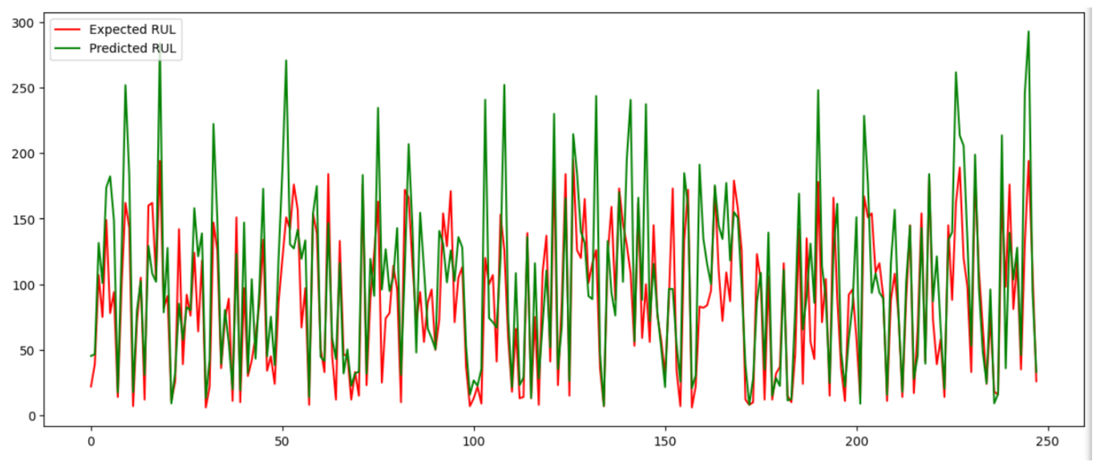


Figure 3.17: Comparison of Expected and Predicted RUL of GBM Model FD-001

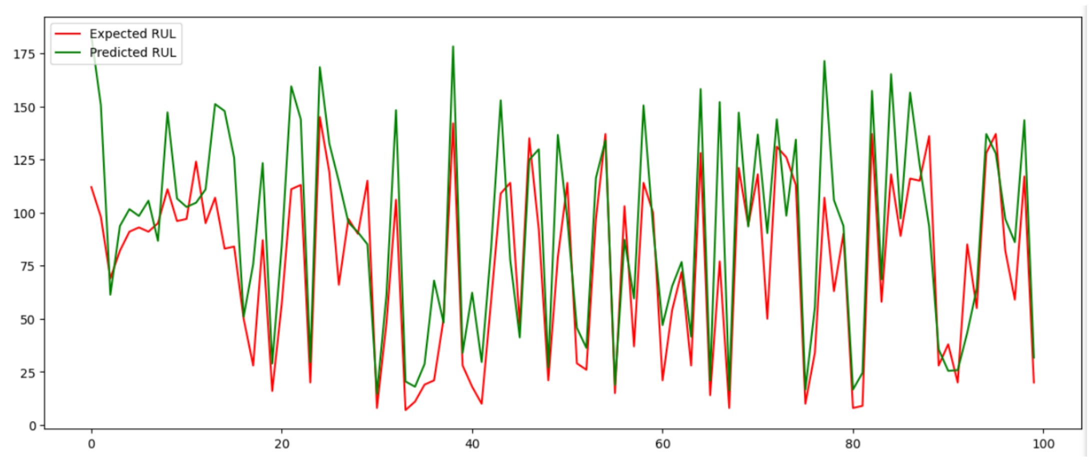


Figure 3.18: Comparison of Expected and Predicted RUL of GBM Model FD-002

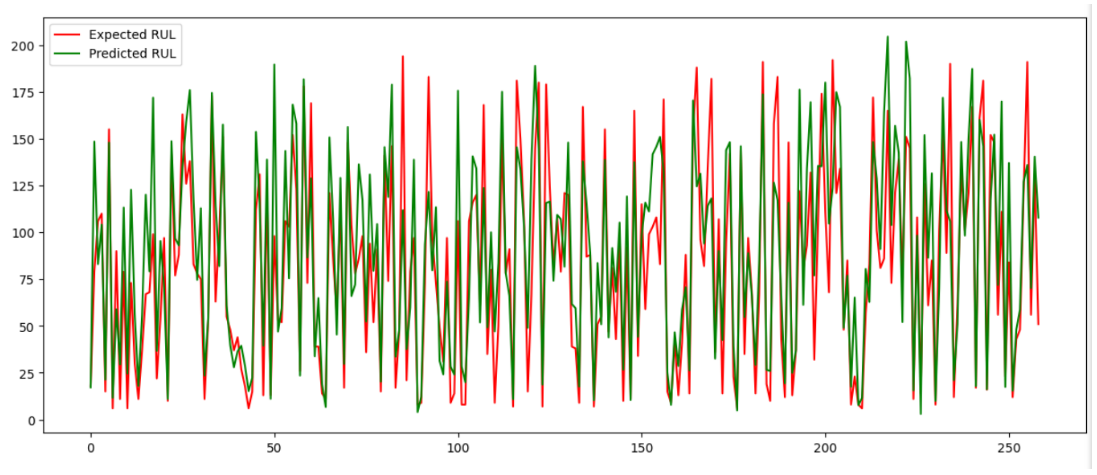


Figure 3.19: Comparison of Expected and Predicted RUL of GBM Model FD-003

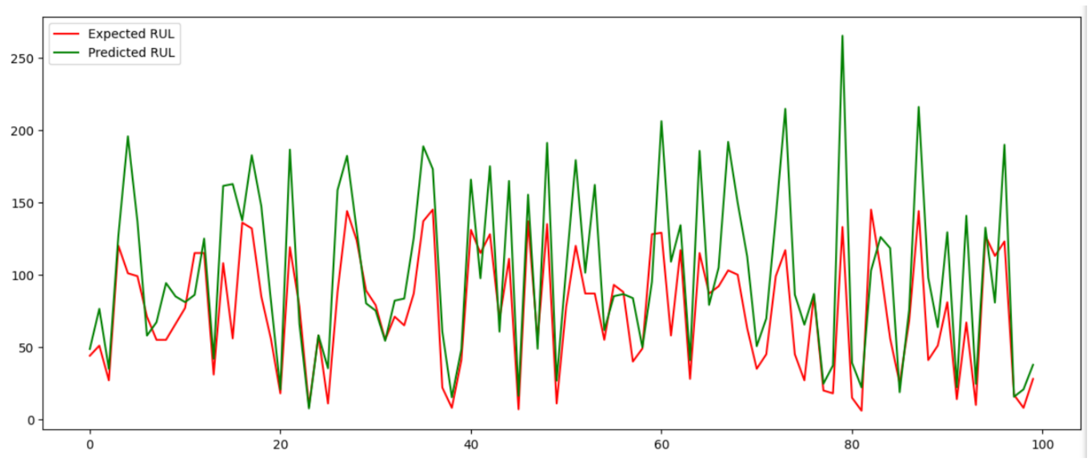


Figure 3.20: Comparison of Expected and Predicted RUL of GBM Model FD-004

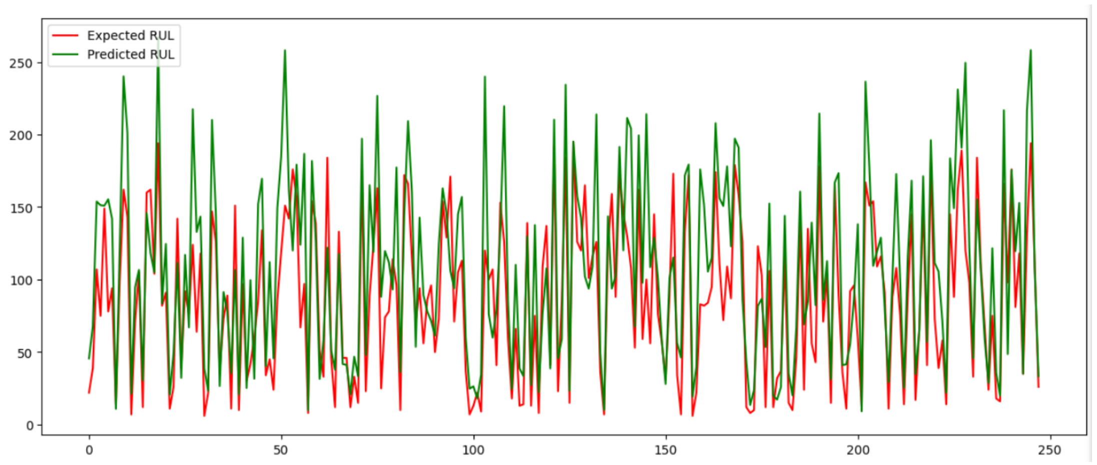


Figure 3.21: Comparison of Expected and Predicted RUL of GRU Model FD-001

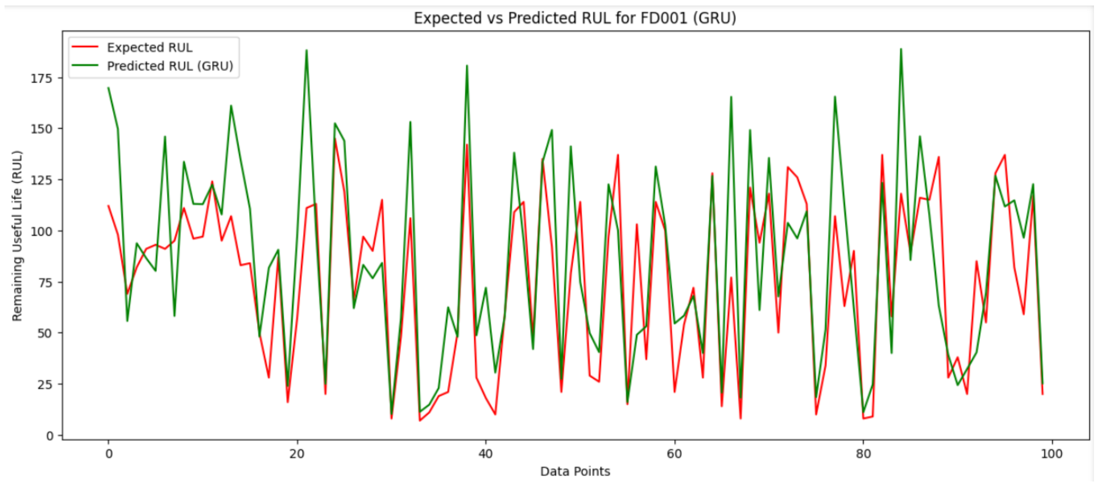


Figure 3.22: Comparison of Expected and Predicted RUL of GRU Model FD-002

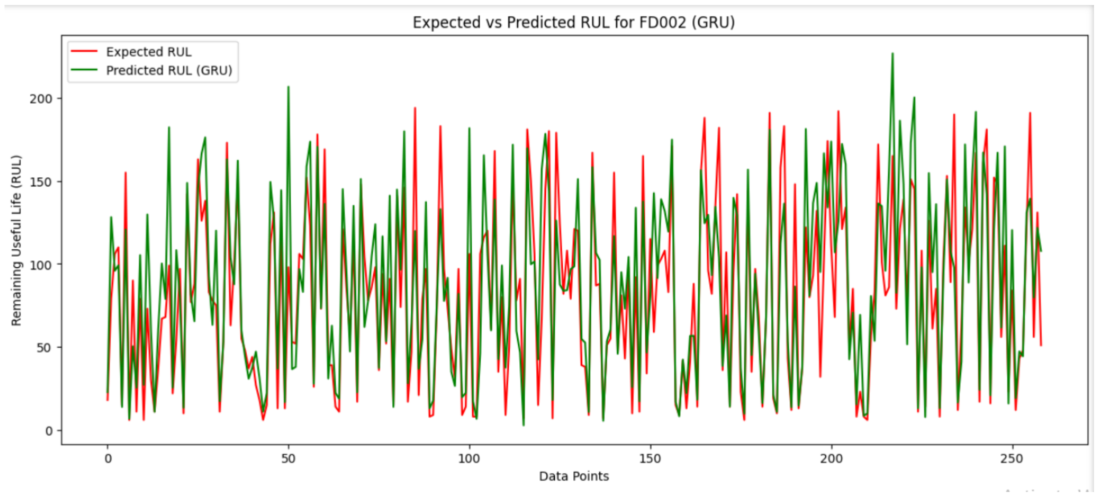


Figure 3.23: Comparison of Expected and Predicted RUL of GRU Model FD-003

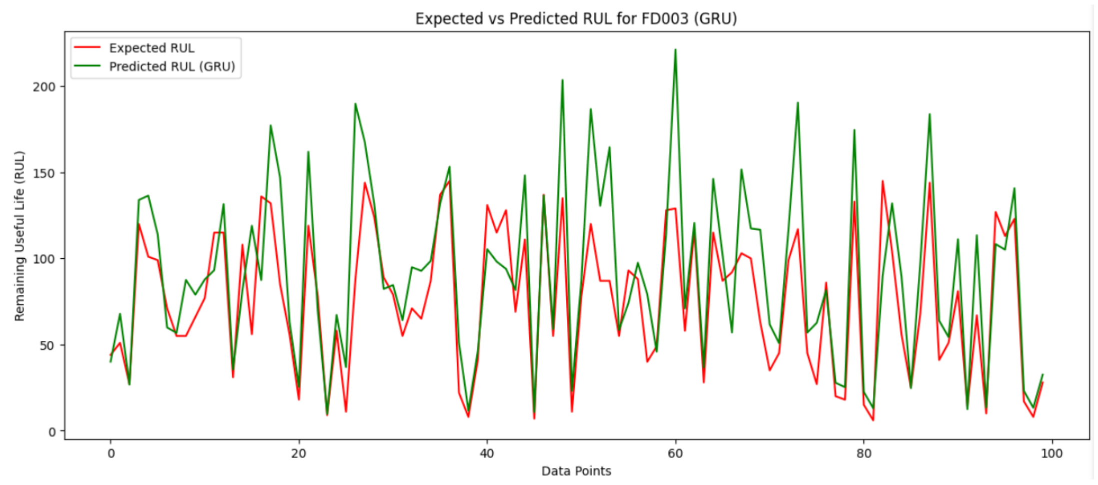


Figure 3.24: Comparison of Expected and Predicted RUL of GRU Model FD-004

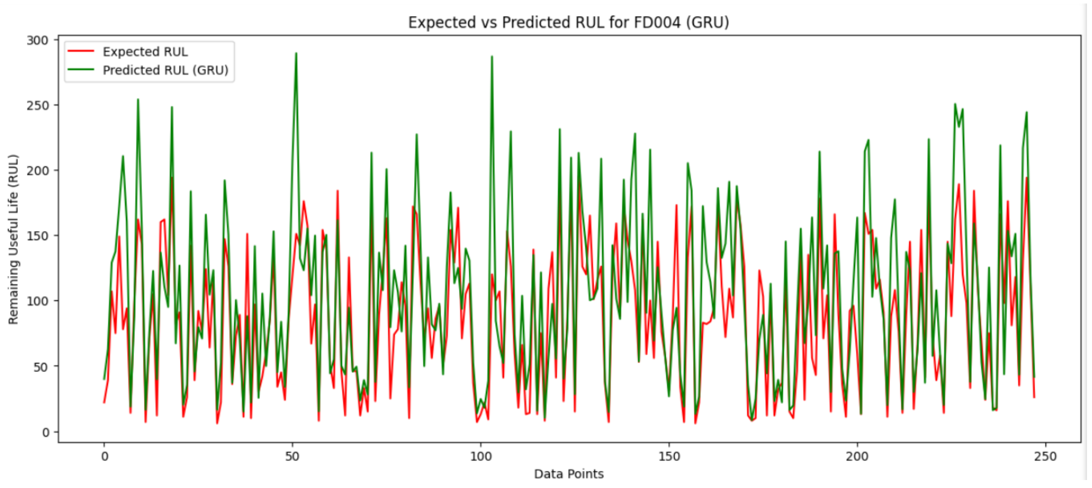


Figure 3.25: Comparison of Expected and Predicted RUL of SVM Model FD-001

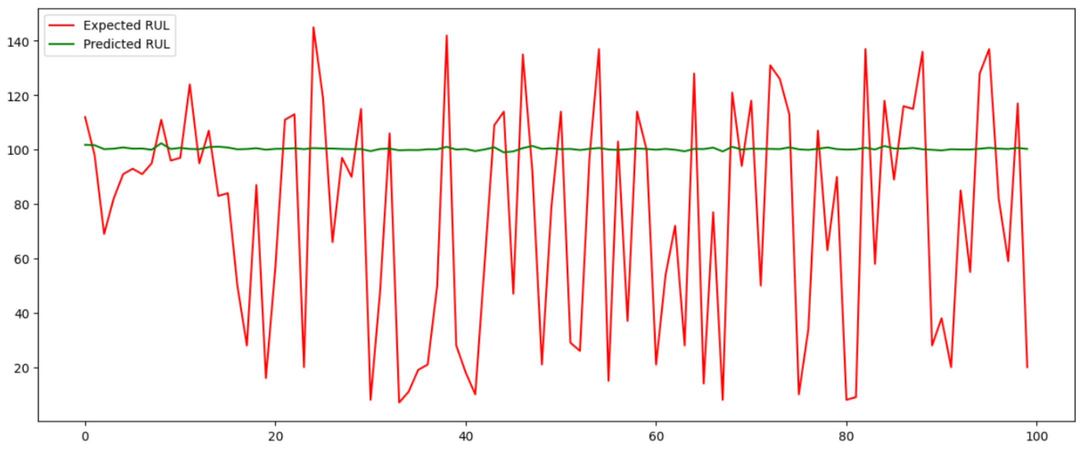


Figure 3.26: Comparison of Expected and Predicted RUL of SVM Model FD-002

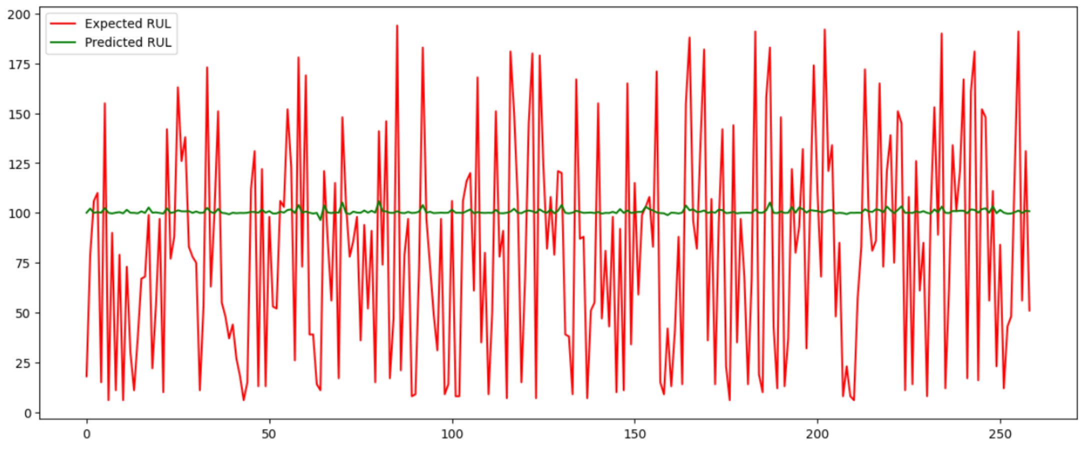


Figure 3.27: Comparison of Expected and Predicted RUL of SVM Model FD-003

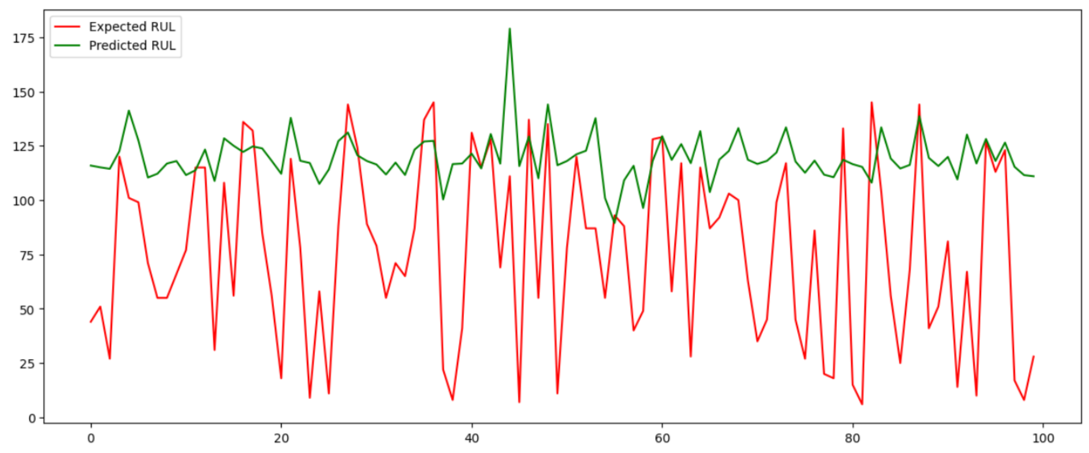
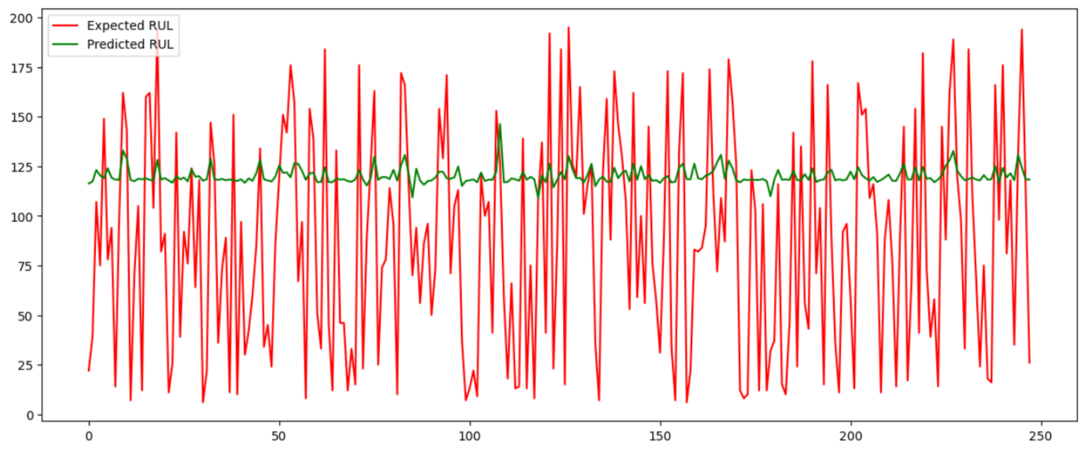


Figure 3.28: Comparison of Expected and Predicted RUL of SVM Model FD-004



The evaluation of model performance shown in the figures above centers on the comparison of Expected vs. Predicted Remaining Useful Life. Among the models scrutinized, SVM reveals significant disparities between expected and predicted RUL, underscoring constraints in its predictive accuracy. Conversely, LSTM, CNN-LSTM, and CNN models showcase notably robust performance, demonstrating closely aligned expected and predicted RUL values. This disparity highlights their efficacy in accurately forecasting the remaining useful life of the engines under examination.

3.5.3 Performance Comparison Of The Used Methods

In selecting the optimal model for predicting Remaining Useful Life (RUL) across different datasets, the key criterion is to choose a model that consistently achieves the lowest Root Mean Square Error (RMSE) across all datasets.

Table 3.5: RMSE Results for the Applied Models

Method	**FD-001**	**FD-002**	**FD-003**	**FD-004**
GBM	28.18	29.80	40.99	41.08
LSTM	27.14	31.04	40.98	40.64
SVM	48.18	56.54	57.79	62.18
GRU	30.8	30.37	30.83	40.79
CNN	27.15	30.80	30.53	42.00
CNN-LSTM	24.14	29.50	35.52	38.74

As Table 3.7 shows among the models evaluated, the CNN-LSTM model stands out as the top performer among the applied models for the FD-001, FD-002, and FD-004 datasets, with an RMSE of 24.14, 29.50, and 38.84 respectively. The Convolutional Neural Network (CNN) model exhibits superior performance for the FD-003 dataset, recording an RMSE of 30.53.

Despite these individual achievements, if one seeks a single model that consistently performs well across all datasets, the CNN-LSTM model emerges as the most effective choice, as shown in Table 3.9.

Table 3.6: The Average RMSE Of The Applied Models

Model	**GBM**	**GRU**	**LSTM**	**CNN**	**CNN-LSTM**	**SVM**
RMSE	35.0125	33.1975	34.95	32.62	31.97	56.1725

The CNN-LSTM model showcases competitive performances across all datasets.

3.6 Conclusion

This chapter has presented an in-depth overview of the dataset employed in this thesis, as well as a comprehensive explanation of the focal point of the study, which is the Turbofan engine.
Additionally, this chapter has shown results indicating that the CNN-LSTM model demonstrates a slightly improved overall performance.
Consequently, the CNN-LSTM model could be considered the optimal choice for implementation in predicting the Remaining Useful Life (RUL) across diverse datasets within the realm of aircraft engine maintenance.

General Conclusion and Perspectives

This thesis has presented a comprehensive study on the implementation of a predictive system for aircraft engine maintenance, focusing on the application of Machine Learning (ML) and Deep Learning (DL) methods to predict the Remaining Useful Life (RUL) of the Aircraft Engine.

The comparison of the models, namely Gradient Boosting, Support Vector Machines, Gated Recurrent Unit, Convolutional Neural Networks, Long Short Term Memory, and CNN-LSTM using the PHM08 NASA dataset has provided valuable insights.

The results have shown that the CNN-LSTM Model outperforms the other methods regarding Root Mean Square Error (RMSE), making it the most suitable method for RUL prediction in this context. This finding underscores the importance of selecting the appropriate Machine Learning algorithm for predictive maintenance tasks, as it can significantly impact the accuracy and reliability of the predictions.

By accurately predicting the RUL of aircraft engines, maintenance schedules can be optimized, leading to reduced downtime and operational costs. Moreover, the system can enable proactive maintenance strategies, preventing unexpected failures and ensuring the continued airworthiness of the engines.

Looking ahead, there are several avenues for further research and development in the field of predictive maintenance for aircraft engines. For example, the integration of advanced sensor technologies, such as IoT devices, enables real-time monitoring and early detection of potential issues.

In conclusion, this thesis highlighted the importance of machine learning methods in improving the reliability and safety of aircraft engines. The findings and methodologies presented in this research can serve as a valuable reference for researchers and practitioners in the field, guiding future efforts toward more efficient and effective predictive maintenance practices in the aviation industry.

Bibliography

[1] Unnati Rajeshkumar Thakkar. *Remaining useful life prediction of a turbofan engine using deep layer recurrent neural networks.* Phd thesis, Carleton University, 2021.

[2] Khalid Khan, Muhammad Sohaib, Azaz Rashid, Saddam Ali, Hammad Akbar, Abdul Basit, and Tanvir Ahmad. Recent trends and challenges in predictive maintenance of aircraft's engine and hydraulic system. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, 43:1–17, 2021.

[3] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Computing Surveys (CSUR)*, 41(3):15, 2009.

[4] Luis Basora, Paloma Bry, Xavier Olive, and Floris Freeman. Aircraft fleet health monitoring with anomaly detection techniques. *Aerospace*, 8(4):103, 2021.

[5] Iqbal H Sarker. Machine learning: Algorithms, real-world applications and research directions. *SN computer science*, 2(3):160, 2021.

[6] Peter J Rousseeuw and Mia Hubert. Anomaly detection by robust statistics. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(2):e1236, 2018.

[7] HP Vinutha, B Poornima, and BM Sagar. Detection of outliers using interquartile range technique from intrusion dataset. In *Information and decision sciences: Proceedings of the 6th international conference on ficta*, pages 511–518. Springer, 2018.

[8] Ian T Jolliffe. *Principal component analysis for special types of data.* Springer, 2002.

[9] GeeksforGeeks. Machine learning - gradient boosting, 2024. Accessed: 2024-05-23.

[10] GeeksforGeeks. Support vector machine algorithm, 2024. Accessed: 2024-06-21.

[11] Subramaniam Dhanabal and SJIJCA Chandramathi. A review of various k-nearest neighbor query processing techniques. *International Journal of Computer Applications*, 31(7):14–22, 2011.

[12] Matthias Reif, Markus Goldstein, Armin Stahl, and Thomas M Breuel. Anomaly detection by combining decision trees and parametric densities. In *2008 19th international conference on pattern recognition*, pages 1–4. IEEE, 2008.

[13] MAISSAE HADDOUCHI and ABDELAZIZ BERRADO. A survey of methods and tools used for interpreting random forest. In *2019 1st International Conference on Smart Systems and Data Science (ICSSD)*, pages 1–6, 2019.

[14] Liangxiao Jiang, Dianhong Wang, Zhihua Cai, and Xuesong Yan. Survey of improving naive bayes for classification. In *Advanced Data Mining and Applications: Third International Conference, ADMA 2007 Harbin, China, August 6-8, 2007. Proceedings 3*, pages 134–145. Springer, 2007.

[15] Guillaume Staerman, Pavlo Mozharovskyi, Stephan Clémençon, and Florence d'Alché Buc. Functional isolation forest. In *Asian Conference on Machine Learning*, pages 332–347. PMLR, 2019.

[16] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. Lof: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 93–104, 2000.

[17] Abdulaziz Almalaq and George Edwards. A review of deep learning methods applied on load forecasting. In *2017 16th IEEE international conference on machine learning and applications (ICMLA)*, pages 511–516. IEEE, 2017.

[18] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[19] Zachary C Lipton, John Berkowitz, and Charles Elkan. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*, 2015.

[20] Ade Pitra Hermawan, Dong-Seong Kim, and Jae-Min Lee. Predictive maintenance of aircraft engine using deep learning technique. In *2020 International Conference on*

Information and Communication Technology Convergence (ICTC), pages 1296–1298. IEEE, 2020.

[21] Hafiz Shahbaz Munir, Shengbing Ren, Mubashar Mustafa, Chaudry Naeem Siddique, and Shazib Qayyum. Attention based gru-lstm for software defect prediction. *Plos one*, 16(3):e0247444, 2021.

[22] Moez Krichen. Convolutional neural networks: A survey. *Computers*, 12(8), 2023.

[23] Tarek Berghout and Mohamed Benbouzid. A systematic guide for predicting remaining useful life with machine learning. *Electronics*, 11(7):1125, 2022.

[24] Towards AI. Predicting the remaining useful life of turbofan engine, 2024. Accessed: 2024-06-25.

[25] MathWorks. Predictive maintenance, part 3: Remaining useful life estimation, 2018. Accessed: 2024-06-24.

[26] Amare Desalegn Fentaye, Valentina Zaccaria, and Konstantinos Kyprianidis. Aircraft engine performance monitoring and diagnostics based on deep convolutional neural networks. *Machines*, 9:337, 2021.

[27] Hao Li, Zhuojian Wang, and Zhe Li. An enhanced cnn-lstm remaining useful life prediction model for aircraft engine with attention mechanism. *PeerJ. Computer Science*, 8:e1084, 2022.

[28] Xianwei Xie, Baozhi Sun, Xiaohe Li, Tobias Olsson, Neda Maleki, and Fredrik Ahlgren. Fuel consumption prediction models based on machine learning and mathematical methods. *Journal of Marine Science and Engineering*, 11(4):738, 2023.

[29] Anurag Upadhyay, Jun Li, Steve King, and Sri Addepalli. A deep-learning-based approach for aircraft engine defect detection. *Machines*, 11(2):192, 2023.

[30] Bob Johnson. Research on yield prediction technology for aerospace engine production lines based on cnn-isvr. *Aerospace Science and Technology*, XX:XX–XX, 2022.

[31] Steven Lee. Prediction of performance degradation in aircraft engines with fuel flow parameter. *Neural Computing and Applications*, 36(1):12–25, 2024.

[32] Fei Li, Li Zhang, Bin Chen, Dianzhu Gao, Yijun Cheng, Xiaoyong Zhang, Yingze Yang, Kai Gao, Zhiwu Huang, and Jun Peng. A light gradient boosting machine for remainning useful life estimation of aircraft engines. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 3562–3567. IEEE, 2018.

[33] Yong-Ping Zhao, Yao-Bin Chen, and Zhi-Qiang Li. A proposed algorithm based on long short-term memory network and gradient boosting for aeroengine thrust estimation on transition state. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 235(15):2182–2192, 2021.

[34] Deepankar Singh, Mithilesh Kumar, KV Arya, and Sunil Kumar. Aircraft engine reliability analysis using machine learning algorithms. In *2020 IEEE 15th International Conference on Industrial and Information Systems (ICIIS)*, pages 443–448. IEEE, 2020.

[35] Tiancheng Wang, Di Guo, and Xi-Ming Sun. Remaining useful life predictions for turbofan engine degradation based on concurrent semi-supervised model. *Neural Computing and Applications*, 34(7):5151–5160, 2022.

[36] Jialin Li, Xueyi Li, and David He. A directed acyclic graph network combined with cnn and lstm for remaining useful life prediction. *IEEE Access*, 7:75464–75475, 2019.

[37] Khouloud Abdelli, Helmut Griesser, and Stephan Pachnicke. A hybrid cnn-lstm approach for laser remaining useful life prediction. *IEEE Access*, 11:5366–5384, 2023.

[38] Candice Bentéjac, Anna Csörgő, and Gonzalo Martínez-Muñoz. A comparative analysis of gradient boosting algorithms. *Artificial Intelligence Review*, 54:1937–1967, 2021.

[39] Yanqiu Zhang, Ming Ni, Chengwu Zhang, Shuang Liang, Sheng Fang, Ruijie Li, and Zhouyu Tan. Research and application of adaboost algorithm based on svm. In *2019 IEEE 8th joint international information technology and artificial intelligence conference (ITAIC)*, pages 662–666. IEEE, 2019.

[40] Ruizhe Yao, Ning Wang, Zhihui Liu, Peng Chen, and Xianjun Sheng. Intrusion detection system in the advanced metering infrastructure: a cross-layer feature-fusion cnn-lstm-based approach. *Sensors*, 21(2):626, 2021.

[41] André Listou Ellefsen, Emil Bjørlykhaug, Vilmar Æsøy, Sergey Ushakov, and Houxiang Zhang. Remaining useful life predictions for turbofan engine degradation using semi-supervised deep architecture. *Reliability Engineering & System Safety*, 183:240–251, 2019.

[42] Azita Mousavi, Hadis Arefanjazi, Mona Sadeghi, Ali Mojarrad Ghahfarokhi, Fatemehalsadat Beheshtinejad, and Mahsa Madadi Masouleh. Comparison of feature extraction with pca and ltp methods and investigating the effect of dimensionality reduction in the bat algorithm for face recognition. *International Journal of Robotics and Control Systems*, 3(3):501–509, 2023.

[43] Shutterstock. Search results for turbofan, 2024. Accessed: 2024-06-21.

[44] Wassim Derbel. Nasa predictive maintenance rul, 2021. Accessed: 2024-06-25.

[45] Behrad. NASA C-MAPSS Data Set. Accessed: June 23, 2024.

[46] JavaTpoint. Javatpoint - tutorials list, 2024. Accessed: April 25, 2024.

[47] Jupyter. Accessed: April 25, 2024.

[48] Overleaf. Theorems and proofs - overleaf, online latex editor, 2024. Accessed: April 25, 2024.

[49] DataCamp. What is kaggle. Accessed: April 25, 2024.

[50] Kinsta. Git vs github: Quelle est la différence ?, 2024. Accessed: 2024-06-23.