Predictive Maintenance Decisions for a Multi-Component Aircraft based on Prognostics

A Two-Fold Deep Learning Approach

L.C.M. Posthuma

TUDelft

Predictive Maintenance Decisions for a Multi-Component Aircraft based on Prognostics

A Two-Fold Deep Learning Approach

by

L.C.M. Posthuma

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on January 20, 2022.

Student number:	4967585	
Project duration:	March 2021 – January 2022	
Thesis committee:	Member:	Department:
Chair	Prof. Dr. H.A.P. Blom	TU Delft - Air Transport & Operations
Supervisor	Dr. M. Mitici	TU Delft - Air Transport & Operations
Co-Supervisor	Ir. I. I. de Pater	TU Delft - Air Transport & Operations
External Committee Member	Dr. C. Borst	TU Delft - Control & Simulation

TU Delft

Acknowledgements

This thesis is the final work of my MSc Aerospace Engineering at the TU Delft. It has been an enjoyable journey in which I have developed both personally and professionally. The challenges I have faced have taught me to become a stronger person in life.

First of all, I would like to thank my daily supervisor, Mihaela Mitici, for her consistent guidance and support. It was a pleasure to work with you, especially because I was constantly given the space to develop my ideas. The lessons I learned, regarding how to be a better researcher, during my time with you will continue in life. In addition, I would like to thank Ingeborg de Pater for attending the milestone meetings and providing constructive feedback.

Second, I would like to thank my friends who supported me during my studies and in my personal life. The time in Delft was unforgettable, considering the demanding periods interspersed with relaxed ones.

Third, a special thanks goes to my girlfriend, Greta, for always being there for me during every period. You have enriched my life in many different ways.

Finally, I would like to sincerely thank my family for their constant support during my studies. To my parents and siblings, you have always encouraged me to keep going and achieve my dreams. I know I can count on you.

Luuk Posthuma
The Hague, January 2022

Contents

List of Figures

List of Tables

List of Abbreviations

AI	Artificial Intelligence
CBM	Condition-Based Maintenance
CDF	Cumulative Density Function
CM	Corrective Maintenance
CNN	Convolutional Neural Network
CPP	Compound Poisson Process
DL	Deep Learning
DQL	Deep Q-Learning
DQN	Deep Q-Network
FM	Fault Modes
GP	Gamma Process
HMM	Hidden Markov Models
IM	Ideal Maintenance
KPI	Key Performance Indicator
LRU	Line Replaceable Units
LSTM	Long Short-Term Memory
MCTS	Monte-Carlo Tree Search
MDP	Markov Decision Process
ML	Machine Learning
MLP	Multilayer Perceptron
MRO	Maintenance, Repair, and Overhaul
MTBF	Mean Time Between Failure
NN	Neural Network
OC	Operating Conditions
OEM	Original Equipment Manufacturer
OM	Opportunistic Maintenance
PDF	Probability Density Function

PdM	Predictive Maintenance
PHM	Prognostics and Health Management
PL	Predicted Label
PM	Preventive Maintenance
PWL	Piece-Wise Linear
RL	Reinforcement Learning
RM	Reactive Maintenance
RNN	Recurrent Neural Network
RUL	Remaining Useful Life
TBM	Time-Based Maintenance
TBO	Time Between Overhaul
TL	True Label

List of Symbols

Greek Symbols

α	Learning Rate $[-]$
α_s	Smoothing Factor $[-]$
γ	Discount Factor $[-]$
ϵ	Exploration Rate $[-]$
ϵ_{om}	Opportunistic Maintenance Factor $[-]$
θ	Neural Network Weights $[-]$
θ_{cm}	Corrective Maintenance Factor $[-]$
π	Reinforcement Learning Policy $[-]$
σ	Sigmoid Activation Function $[-]$

Latin Symbols

a_t	Action at timestep t $[-]$
AC_t	Aircraft at timestep t $[-]$
C_{cm}	Long-Term Maintenance Cost Corrective Maintenance Policy $[-]$
C_{im}	Long-Term Maintenance Cost Ideal Maintenance Policy $[-]$
C_m	Long-Term Maintenance Cost $[-]$
C_u	Average Component Utilization $[-]$
C_{pdm}	Long-Term Maintenance Cost Predictive Maintenance Policy $[-]$
C_{tbm}	Long-Term Maintenance Cost Time-Based Maintenance Policy $[-]$
dl_1	Degradation Level 1 $[-]$
dl_2	Degradation Level 2 $[-]$
dl_3	Degradation Level 3 $[-]$
E_l	Episode Length $[t]$
$En_{i,t}$	Engine i at timestep t $[-]$
$En^a_{i,t}$	Engine i Actual Lifetime a at timestep t $[t]$
$En^{dl_j}_{i,t}$	Engine i Probability Degradation Level j at timestep t $[-]$
$En^l_{i,t}$	Engine i Lifetime l at timestep t $[t]$
$En^p_{i,t}$	Engine i Identifier p at timestep t $[-]$
$En^s_{i,t}$	Engine i Status at timestep t $[-]$
N_e	Number of Episodes $[-]$

N_f	Number of Features $[-]$
N_o	Number of Outputs $[-]$
N_s	Number of Simulations $[-]$
N_t	Sequence Length $[t]$
N_{cm}	Number of Corrective Maintenance Actions $[-]$
N_{om}	Number of Opportunistic Maintenance Actions $[-]$
N_{pm}	Number of Preventive Maintenance Actions $[-]$
N_{ts}	Number of Timesteps $[t]$
$Q^{\pi}(s_t, a_t)$	Q-(Action-Value) Function $[-]$
r_t	Reward at timestep t $[-]$
R	Total Maintenance Reward $[-]$
R_{cm}	Corrective Maintenance Reward $[-]$
R_{nm}	No Maintenance Reward $[-]$
R_{om}	Opportunistic Maintenance Reward $[-]$
R_{pm}	Predictive Maintenance Reward $[-]$
R_t	Average Component Replacement Time $[t]$
s_t	State Space at timestep t $[-]$
T_a	Average Time Threshold for the TBM policy $[t]$
T_c	Number of Replaced Components $[-]$
T_0	Lower Limit $[t]$
T_1	Upper Limit $[t]$
$V^{\pi}(s_t)$	State-Value Function $[-]$
W_t	Average Wasted Component Lifetime $[t]$
X_t^i	Sensor Value i at timestep t $[-]$

I

Scientific Paper

Predictive Maintenance Decisions for a Multi-Component Aircraft based on Prognostics

L.C.M. Posthuma, *MSc. Student, TU Delft,*
Dr. M. Mitici, *Daily Supervisor, TU Delft,* I.I. de Pater, *Co-Supervisor, TU Delft.*
Section of Air Transport Operations, Faculty of Aerospace Engineering, Delft University of Technology, Kluyverweg 1, 2629HS Delft, The Netherlands

Abstract—**Aircraft maintenance is critical to an airline's operations to ensure the reliability, availability, and safety of their assets. Recently, the approach of using component prognostics in aircraft maintenance has received increasing attention in academic- and industrial research. Predictive maintenance has demonstrated promising results in using sensor-based prognostics for maintenance decisions. In this paper, we propose a novel predictive maintenance framework that is capable of mapping the individual component degradation levels to an optimal maintenance decision. The independent component degradation levels are computed by a supervised learning model, called "Long Short-Term Memory Networks". Subsequently, the computed degradation levels are utilized in a multi-component maintenance decision framework, by using a model-free reinforcement learning technique named "Deep Q-Learning". The predictive maintenance framework aims to minimize a cost objective based on the type and frequency of a maintenance action. In addition, we analyzed several key performance indicators, such as the number of components used, the component utilization level, as well as the wasted component lifetime. The predictive maintenance framework was evaluated using NASA's turbofan degradation dataset. Ultimately, the results of the numerical experiments showed that the proposed predictive maintenance framework resulted in lower costs than when using a time-based and corrective maintenance policy and competitive costs compared to an ideal maintenance policy. The proposed predictive maintenance framework opens new directions for multi-component sensor-based maintenance decisions. The results found form the basis for application suggestions and future research directions in practice.**

Keywords—**Cost Minimization, Deep Q-Learning, Long Short-Term Memory Network, Multi-Component Aircraft, Prognostics and Health Management.**

I. Introduction

Aircraft maintenance is a major expense for airlines and thus plays an important role in their Maintenance, Repair, and Overhaul (MRO) strategies [1]. Therefore, airlines optimize their maintenance activities in order to reduce costs [2]. Traditional maintenance strategies are based on replacing components when they are defective, in other words Corrective Maintenance (CM), or when they have reached a certain time interval, which is known as Time-Based Maintenance (TBM). However, airlines are shifting from a CM and TBM strategy to a Predictive Maintenance (PdM) approach [3]. A PdM policy is based on the health status of the system and indicates if and when a component should be maintained. Being able to detect and respond to the degradation of certain components reduces maintenance cost, aircraft downtime, and unexpected flight delays [4]. The PdM approach utilizes sensor data from components to generate prognostic data, which can then be used to make specific maintenance decisions. Useful sensor data is collected by monitoring the components during operation or inspection. The PdM approach can be split into two parts. The objective of the first part is to predict the degradation levels or remaining-useful-life (RUL) of a component or system. This prognostic data can then be used in the second part to optimize maintenance activities and minimize overall maintenance costs. Extensive research has been conducted separately in the areas of prognostics and maintenance optimization. However, research in an end-to-end PdM framework is notably lacking. Moreover, current research often focuses on single-component systems while ignoring multi-component systems. In this paper, we propose a predictive maintenance framework for a multi-component aircraft, which is able to map the independent component degradation values at each inspection point to make a maintenance decision with a cost minimization objective. To use independent component degradation levels for maintenance decisions, a two-fold Deep Learning (DL) approach is used. The method chosen to obtain the component prognostics is the application of a Long Short-Term Memory (LSTM) network on a turbofan degradation dataset created by NASA. The obtained prognostic data is then used for maintenance decisions in the second part of the framework. The PdM framework is modeled as a Markov Decision Process (MDP) in which a Reinforcement Learning (RL) method called "Deep Q-Learning" (DQL) is used to find an optimal policy. In general, the DQL agent is able to obtain a policy that is capable of minimizing long-term maintenance costs. DQL has been successfully applied on various applications such as robotics and autonomous separation of aircraft [5, 6]. However, studies that use DQL for sensor-based maintenance decision-making in multi-component systems are lacking.

The proposed PdM framework is capable of computing maintenance decisions based on component prognostics. Therefore, the PdM framework is of great interest to airlines since it provides an integrated solution for predictive maintenance. Numerical experiments show promising results and make an important contribution to the body of knowledge. The approach used combines novel techniques in a dynamic PdM framework applicable to various realistic sensor-based problems.

L.C.M. Posthuma is a MSc. Student, Air Transport and Operations, Faculty of Aerospace Engineering, Delft University of Technology.

A. Related Work

Previous work has made important contributions to the knowledge of predictive maintenance. However, little research has been conducted on the complete framework that integrates sensor data and maintenance decisions for multi-component systems. The following section consists of a brief literature review on the computation of component prognostics and methods for maintenance decisions.

Several articles have described the methodology for using supervised learning methods to compute prognostics for systems. Data-driven methodologies are divided into Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), and other approaches. CNNs are predominantly used in studies concerning image and speech recognition. However, [7], showed an application of CNNs for predicting the RUL of turbofan engines. A more recent approach, [8], has performed well in predicting the degradation of turbofan engines. Yet, RNN perform better than CNN because RNN are more suitable for processing sequential data. [9, 10, 11] use a data-driven approach for component prognostics using LSTM networks, which is a specific type of RNN. Beyond that, other approaches are incorporating Hidden Markov Models (HMM) [12] or Multilayer Perceptrons (MLP) [13]. A comprehensive overview can be found in [14].

Alternatively, other studies extend the prognostics to include maintenance decisions in a framework, indicating whether or not a component should be replaced. In [15], a data-driven condition-based maintenance framework is proposed. The authors used an LSTM network to obtain information on component degradation and failure probabilities. This information is then used in a maintenance optimization model based on degradation thresholds. However, their approach only considers one component and short-term maintenance decisions. In, [16], the authors have described a new approach to integrate prognostics into maintenance decision-making, but the decisions are based on a threshold and do not consider multiple components. [17] is a continuation of [16], but lacks the dynamics of a longer-term multi-component framework.

In, [18], a strategy for optimizing the maintenance of a cooling system is proposed. A multi-stage Wiener process is used to characterize degradation trends, and a numerical optimization of the preventive control limit to minimize the cost objective is performed. In [19], a condition-based maintenance model, based on RL is defined. The study was one of the first articles in the field to describe how Q-Learning can be used for maintenance decisions. The authors use a simplified failure distribution for a single component to illustrate its effectiveness for a binary maintenance decision. In, [20], a long-term aircraft optimization strategy is formulated, using a RL approach. It takes into account various input parameters, such as profile data and prognostics, to decide whether or not a mission should be carried out. Moreover, in [21] is described how a RL approach can be used for maintenance decisions based on a normalized health status indicator. They limit their research to a single turbofan component and do not consider opportunistic maintenance. In, [22], a RL approach is used to derive a real-time maintenance policy for a system monitored by sensors. Yet, they used a regression approach for prognostics and only considered a single-component system. In, [23], a RL approach is used to establish a condition-based maintenance model for a multi-component system under dependent competing risks. However, they use a Gamma Process (GP) and a Compound Poisson Process (CPP) as the stochastic deterioration process. [24] showed the application of a RL model to an opportunistic maintenance policy for a machine production system. The paper considers various maintenance strategies, taking into account the reduction of downtime and maintenance costs. Although, they use a Weibull distribution for the component failure probability.

B. Research Paper Contribution

The main research gap in the above-mentioned studies is the lack of an end-to-end PdM framework that takes into account sensor data for multi-component systems. To overcome that challenge, we propose a novel PdM framework that uses sensor data to evaluate component degradation and then utilizes these prognostics to make maintenance decisions. To the best of our knowledge, this is the first end-to-end maintenance decision framework which uses RL for data-driven decisions for a multi-component aircraft. Furthermore, the main contributions of this research are the following:

- A novel PdM framework is proposed. The framework is capable of performing maintenance actions for multi-component aircraft systems based on data-driven degradation levels. Moreover, the PdM framework can be applied to various sensor-based systems with N-components.
- The PdM framework is able to learn the degradation behavior of a given component based on historical data and does not require degradation threshold levels. In addition, the framework can handle unknown degradation levels in a large state space to make maintenance decisions.
- The PdM framework is capable of representing a realistic long-term situation. The framework is an end-to-end solution that includes both sensor measurements and maintenance decisions during a specified horizon.
- The PdM framework outperforms traditional maintenance policies in terms of several key performance indicators. Using numerical experiments, it is shown that the PdM framework outperforms corrective and time-based maintenance policies and comes close to ideal maintenance in terms of maintenance cost.

C. Research Paper Structure

The remainder of this paper is organized as follows: Section II introduces the problem statement and describes the system description. Section III describes how the component degradation levels are computed based on sensor data, including the results of the component prognostics. Afterwards, Section IV discusses the formulation of the PdM framework and how prognostics can be used in a decision-making framework. Section V contains the explanation of the numerical experiments that demonstrate the effectiveness of the framework. This section also covers parametric performance analysis and the train- and test strategy. In Section VI we illustrate the performance of the framework compared to traditional maintenance policies and include the results of the research. Subsequently, in Section VII the discussion is given. Finally, Section VIII contains the conclusion of this study and proposes recommendations for future research.

II. Problem Statement

The goal of this study is to create a PdM framework which is capable of making maintenance decisions based on sensor data. In this research, an aircraft with multiple components is considered as the system. The aircraft components are replaceable and independent of each other in terms of degradation and do not interfere with each other. Furthermore, the maintenance framework is a discrete time model, where each timestep t is considered whether or not a maintenance action should be performed. Overall, the objective of the framework is to minimize the total maintenance costs over a specified time period. In addition, the number of components used, the utilization rate of the components, and the average wasted component lifetime are taken into account to analyze the performance of the framework. The problem statement is based on assumptions that are defined as follows:

1) The aircraft and its components are continuously monitored, which means that the maintenance agent knows the degradation values of the components at each timestep t.
2) When a component reaches its maximum lifetime, it must be replaced with a new component and is classified as *corrective maintenance*.
3) If the maintenance agent decides to replace a component based on its dynamic degradation levels and before its maximum lifetime, it is replaced with a new component and classified as *preventive maintenance*.
4) If the maintenance agent decides to replace two components at the same time based on their dynamic degradation levels and before their maximum lifetime, this is considered to be *opportunistic maintenance*.
5) Maintenance decisions are made every timestep t and maintenance actions take place immediately. Each of the maintenance actions described does not cause any system downtime or interruption.
6) There are no limitations in terms of maintenance opportunities or the number of available components.

The maintenance framework description is shown in Figure 1. The aircraft components are continuously monitored and sensor information, such as temperature and pressure, is extracted. The specific sensor measurements used in this research are described in Section III. Based on this sensor data, component degradation levels are computed using an LSTM classifier, resulting in the probability that a component will fail within a given time window. Subsequently, the dynamic state of the components is used in a maintenance decision framework based on an MDP. The agent makes a decision for a component and updates the corresponding system state. Consequently, the environment is updated based on the decision of the agent. Ultimately, the goal of the agent in the PdM framework is to obtain an optimal maintenance policy that minimizes costs. The policy in this research represents

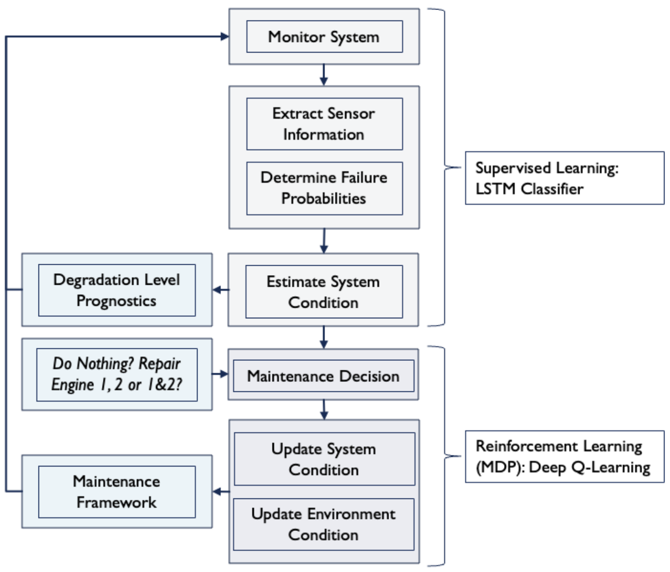

Monitor System
Extract Sensor Information
Determine Failure Probabilities
Supervised Learning: LSTM Classifier
Degradation Level Prognostics
Estimate System Condition
Do Nothing? Repair Engine 1, 2 or 1&2?
Maintenance Decision
Update System Condition
Reinforcement Learning (MDP): Deep Q-Learning
Maintenance Framework
Update Environment Condition

Fig. 1: Overview of the proposed Predictive Maintenance (PdM) framework. The PdM framework maps sensor data from a multi-component system to maintenance decisions.

Variable	*Description*
AC_t	Aircraft at timestep t
$En_{i,t}$	Engine i at timestep t
$En^l_{i,t}$	Engine i lifetime l at timestep t
$En^a_{i,t}$	Engine i actual lifetime a at timestep t
$En^s_{i,t}$	Engine i status s at timestep t
$En^{dl_j}_{i,t}$	Engine i probability degradation level j at timestep t

Table 1: Notation of the variables used in the system description.

the mapping of the observed degradation state of components at each timestep with a specific maintenance action.

A. System Description

We consider a multi-component aircraft (AC_t) for each timestep t, where $t \in \{1, 2, \ldots, T\}$. Each aircraft has two independent engines, denoted by $En_{1,t}$ and $En_{2,t}$ at timestep t. Each engine is independent of any other engine in terms of its sensor data. Therefore, the degradation levels of the engines and the failure probabilities are also independent. To track the component lifetime, the age of engine i at timestep t is defined as $En^l_{i,t} \in \{1, 2, \ldots, L\}$, where $i \in \{1, 2\}$ denotes the number of the aircraft engine, and L the maximum lifetime. Moreover, let $En^a_{i,t} \in \mathbb{R}$ represent the actual lifetime of engine $i \in \{1, 2\}$ at timestep t. Furthermore, the status of an engine at timestep t is described by $En^s_{i,t} \in \{0, 1\}$, where $i \in \{1, 2\}$ indicates whether or not an engine has failed ($En^s_{i,t} = 0$), or has not ($En^s_{i,t} = 1$) at timestep t. If an engine has failed, it will be replaced with another engine and considered new. In addition, we consider the degradation levels (dl_j) for each aircraft engine, which are denoted as

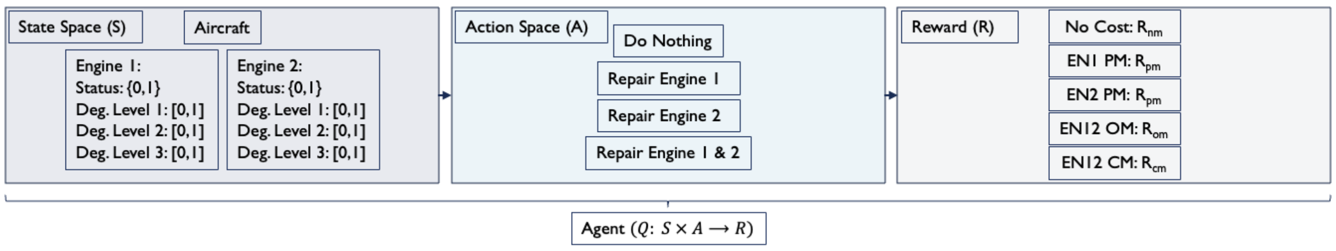

State Space (S)
Aircraft
Engine 1: Status: {0,1} Deg. Level 1: [0,1] Deg. Level 2: [0,1] Deg. Level 3: [0,1]
Engine 2: Status: {0,1} Deg. Level 1: [0,1] Deg. Level 2: [0,1] Deg. Level 3: [0,1]
Action Space (A)
Do Nothing
Repair Engine 1
Repair Engine 2
Repair Engine 1 & 2
Reward (R)
No Cost: R_{nm}
EN1 PM: R_{pm}
EN2 PM: R_{pm}
EN12 OM: R_{om}
EN12 CM: R_{cm}
Agent (Q: $S \times A \longrightarrow R$)

Fig. 2: Reinforcement learning based PdM framework for an aircraft with two engines ($En_{i,t}$, where $i \in \{1, 2\}$). The agent receives a state space $s_t \in S$ representation, takes an action $a_t \in A$ and receives a reward $r_t \in R$.

$En_{i,t}^{dl_j} \in [0, 1]$, where $j \in \{1, 2, 3\}$ and $i \in \{1, 2\}$ for each timestep t. $En_{i,t}^{dl_j} \in [0, 1]$ is the probability that engine *i* has degradation level *j* at timestep *t*. Specifically, dl_1 indicates that the engine is healthy, dl_2 indicates a degraded engine, and dl_3 indicates an engine that is critical to failure. Consequently, each engine has a state that determines whether the system has failed or not, and three different values that reflect the probability that the engine belongs to the specific degradation levels. The notation of the variables used in the remainder of this paper is summarized in Table 1. An illustrative overview of the maintenance decision framework is depicted in Figure 2. The state space (S) contains the engine status $En_{i,t}^{s}$ and the three degradation levels $En_{i,t}^{dl_j}$ for each engine *i*. The action space (A) contains the four different maintenance actions the agent can take, i.e., Do Nothing, Repair Engine 1, Repair Engine 2, or Repair Engines 1 & 2. The rewards (R) are based on the costs associated with a particular maintenance action. Section III describes the calculations of the component degradation levels and Section IV shows how the degradation level probabilities are used in the PdM decision framework.

III. Aircraft Component Degradation Levels

This section first discusses the method of calculating degradation levels for aircraft components based on sensor data. Then the results of the component prognostics are discussed. The main idea is to use supervised learning that has the task of mapping an input X to an output y, such as classification or regression ($X \rightarrow y$). In this research, we consider the sensor data as the input *(X)* and the probability of an engine belonging to a given degradation level as the output *(y)*.

A. LSTM Classifier

The modeling of the component degradation levels is done by means of a Recurrent Neural Network (RNN) architecture called Long Short-Term Memory (LSTM) networks. In general, RNN has been applied to various fields, such as time-series forecasting, language processing, and operations research [25, 26, 27]. RNN is a specific type of neural networks (NN) in which each node in the hidden layer has the ability to use its internal state as input, as opposed to NN that only have feedforward states. RNN use their internal state and form an inner loop to allow information to persist. The advantage of LSTM networks over traditional RNN is that LSTM networks are able to capture long-term dependencies [28, 29]. Long-term dependencies in time series analysis, such as computing prognostics, are often very useful because they can indicate deterioration over time. Moreover, LSTM networks are able to capture nonlinear behavior for time-sequential data.

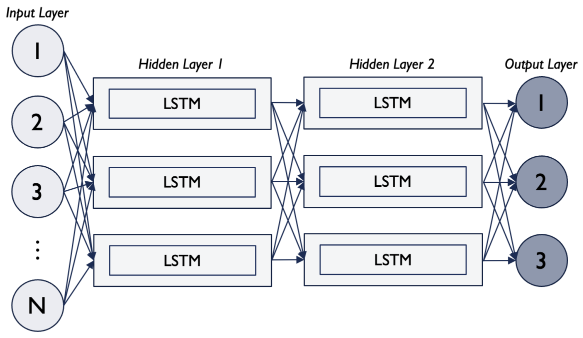


Fig. 3: Long Short-Term Memory network architecture.

A representation of the used LSTM network is illustrated in Figure 3. The input layer for the LSTM network is based on the number of features and the sequence length expected by the network. The shape of the input layer is defined by $N_t \times N_f$, where N_t is the sequence length and N_f is the number of selected features. The input layer is connected to a hidden layer, consisting of multiple LSTM cells, which are capable of analyzing historical and current sensor data. The LSTM layer is then fully connected to another LSTM layer, resulting in two hidden layers. An LSTM cell contains an activation function, which determines the output of the node, to perform the classifier learning task. The output layer is specified by the number of nodes, in this case three different classifications, each for one degradation level. The output layer has a shape of $N_o \times 1$, where N_o is the number of desired outputs, which in this research is 3. An illustrative example of the relationship between the input and the output can be seen in Figure 4, where the input is sensor data of a certain component and the output is a prediction of the probability a component belongs to certain degradation levels. The black bars illustrate the principle of the sliding window determined by the sequence length N_t. Note that not only the class is predicted, but also the probability of belonging to each degradation level, which is useful for indicating the deterioration over time.

A general LSTM cell structure is shown in Figure 5. x_t is the cell input and h_t is the cell output value, which in this research represents the sensor data and the degradation

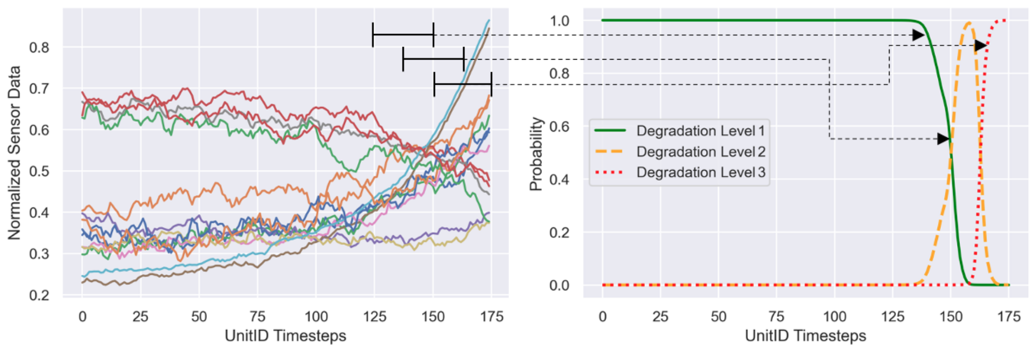


Fig. 4: Illustrative example of the relationship between sensor data *(left)* and corresponding degradation level probabilities *(right)* for one specific engine with an actual lifetime of 175 timesteps.

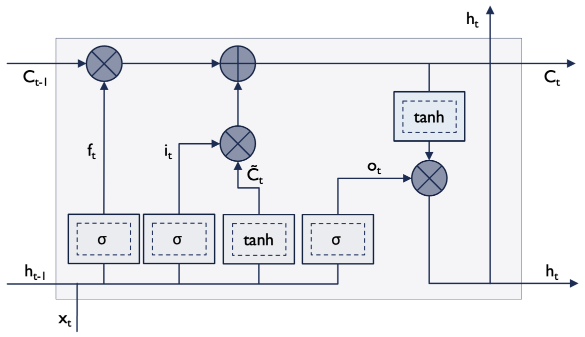


Fig. 5: Long-Short Term Memory cell structure.

level classification, respectively. Each LSTM cell contains a cell state (C_t) that depends on the previous cell state (C_{t-1}), characterized by Equation 1.

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

Where C_t is the new cell state at timestep t, f_t is the forget layer at timestep t, C_{t-1} is the old cell state at timestep $t-1$, i_t is the input layer at timestep t, and $\tilde{C}_t$ are the candidate values at timestep t. The information can be altered by the gates, which are deciding which information is added or removed. The three gates, described below, consist of a sigmoid function that outputs a value between 0 and 1. If the value is zero, no information is added, while if the value is one, all information is passed through. For all below stated equations, we consider W_* as the trainable weight matrix and b_* as the bias matrix for each corresponding gate (f_t, i_t, o_t). The following three gate layers can be identified:

1) **Forget Gate Layer** (f_t): considers h_{t-1} and x_t and outputs a value between 0 and 1 to decide what information is carried along and what information is omitted. Characterized by the equation:

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \tag{2}$$

Where f_t is the forget gate at timestep t, σ is the sigmoid activation function, h_{t-1} is the output at timestep $t-1$ and x_t is the input at timestep t.

2) **Input Gate Layer** (i_t): consists of two layers, where the sigmoid layer considers h_{t-1} and x_t and decides which values are updated. The next layer, based on a tanh activation function, computes a vector of values ($\tilde{C}_t$) which could be added to the state. These two layers are then multiplied to generate an update to the current cell state and are defined by the following equations:

$$\begin{aligned} i_t &= \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right) \\ \tilde{C}_t &= \tanh\left(W_C \cdot [h_{t-1}, x_t] + b_C\right) \end{aligned} \tag{3}$$

Where i_t is the input gate at timestep t and $\tilde{C}_t$ is the vector of new possible information at timestep t.

3) **Output Layer** (o_t): the last layer determines what the output of the cell will be. The output is a filtered version of the cell state. The signal is passed through a sigmoid layer to determine which part will be the output and then multiplied by a tanh function to obtain values in the range $[-1, 1]$ and subsequently provides the desired output values. The output layer is defined as follows:

$$\begin{aligned} o_t &= \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right) \\ h_t &= o_t * \tanh\left(C_t\right) \end{aligned} \tag{4}$$

Where o_t is the output gate at timestep t and h_t is the output value of the current cell and input for the next cell at timestep t.

In the LSTM network (Figure 5), various sigmoid and tanh functions are used to specify which information is transmitted and which is not, and to regulate the network. The sigmoid function is characterized by Equation 5 and has a range of $[0, 1]$. The tanh function, defined in Equation 6, has a range of $[-1, 1]$.

$$\sigma(x) = \frac{1}{1+e^{-x}} \quad (5) \qquad tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (6)$$

In this research, a softmax function is used as the activation function in the output layer of the LSTM network, since the PdM framework requires failure probabilities between $[0, 1]$. The softmax function converts the real values of the LSTM layer into a vector of probabilities, summing to 1, characterized by Equation 7. The used LSTM architecture has 3 output nodes, where each node gives the probability that the component belongs to that particular degradation class. In addition, dropout layers are used in each LSTM layer to avoid over-fitting of the training data and to improve the

Dataset	*FD001*	*FD002*	*FD003*	*FD004*
Train Units	100	260	100	249
Test Units	100	259	100	248
Operating Conditions (OC)	1	6	1	6
Failure Modes (FM)	1	1	2	2

Table 2: Overview of the different subsets (FD001-FD004) within the C-MAPSS dataset.

performance of the model [8]. The dropout layers act as a regularization method by probabilistically excluding input and recurrent connections.

$$S(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{J} e^{x_j}} \quad \text{for} \quad i = \{1, \ldots, J\} \tag{7}$$

B. Dataset Overview & Exploration

The dataset used in the numerical experiments is the "Turbofan Engine Degradation Simulation Dataset", provided by the Prognostics Center of Excellence at NASA Ames [30]. The dataset contains four different subsets (FD001-FD004) of a simulated turbofan engine under various operating conditions, which mimic the degradation behavior. The system model is a representation of an aircraft engine in C-MAPPS and is widely used in prognostics studies [31]. The specifications of the subsets are shown in Table 2. The subsets FD001 and FD003 have only a single operating condition, while the subsets FD002 and FD004 are more complex as they have six operating conditions. Furthermore, in FD001 and FD002 there is only one failure condition, while in FD003 and FD004 there are two failure modes. The two different failure modes (FM) are: high-pressure degradation and fan degradation. The operating conditions (OC) are a combination of different altitude, Mach number, and throttle resolver angle settings.

Each subset contains a total of 26 columns of data, including the engine Unit ID, time cycles, operational settings, and sensor measurements. The first two columns display the Unit ID and time cycles, while the next three columns characterize the operating condition settings. The consecutive 21 columns correspond to the sensor measurements. The sensor readings, which are considered observation variables, are given in Table 3.

Various descriptive statistics are used to better understand the different datasets. For the sake of simplicity, FD001 is used in the remainder of the paper, including during the numerical experiments of the PdM framework. The values calculated for the training data of the selected subset are provided in Table 4. The table shows that there are 100 engines with different maximum life cycles, ranging from 128 to 362 cycles. Table 4 shows that operational settings (Op Set) 1 and 2 have a very small standard deviation. Operational setting 3 has a standard deviation of 0. This indicates that only one combination of operational settings can be identified, since the settings within the dataset do not change.

C. Data Preprocessing

Data preprocessing is an essential step within the steps of the PdM framework. All sensor data must be labeled,

Sensor	*Description*	*Unit*
1	Total temperature at fan inlet	°R
2	Total temperature at low pressure compressor outlet	°R
3	Total temperature at high pressure compressor outlet	°R
4	Total temperature at low pressure turbine outlet	°R
5	Pressure at fan inlet	psia
6	Total pressure in bypass-duct	psia
7	Total pressure at high pressure compressor outlet	psia
8	Physical fan speed	rpm
9	Physical core speed	rpm
10	Engine pressure ratio	–
11	Static pressure at high pressure compressor outlet	psia
12	Ratio of fuel flow	pps/psi
13	Corrected fan speed	rpm
14	Corrected core speed	rpm
15	Bypass ratio	–
16	Burner fuel-air ratio	–
17	Bleed enthalpy	–
18	Demanded fan speed	rpm
19	Demanded corrected fan speed	rpm
20	High pressure turbine coolant bleed	lbm/s
21	Low pressure turbine coolant bleed	lbm/s

Table 3: Overview of the 21 different sensors within the C-MAPSS dataset.

Variable	*UnitID*	*Time [Cycles]*	*Op Set 1*	*Op Set 2*	*Op Set 3*
count	20631	100	20631	20631	20631
mean	51.51	206.31	-0.000009	0.000002	100.0
std	29.23	46.34	0.002187	0.000293	0.0
min	1.00	128.00	-0.008700	-0.000600	100.0
25%	26.00	177.00	-0.001500	-0.000200	100.0
50%	52.00	199.00	0.000000	0.000000	100.0
75%	77.00	229.25	0.001500	0.000300	100.0
max	100.00	362.00	0.008700	0.000600	100.0

Table 4: Descriptive statistics of the engines and the operational settings of subset FD001 (C-MAPSS dataset).

normalized, smoothed and sequenced to obtain accurate prognostics [31]. The sensor data is logged for each timestep t and forms the vector: $X_t^i = [x_t^1, x_t^2, x_t^3, ..., x_t^N]$ for each selected feature $i \in \{1, 2, 3, \ldots, N\}$ and timestep $t \in \{1, 2, 3, \ldots, T\}$.

1) Data Labelling: The LSTM network calculates the probability that the system belongs to each degradation level. The degradation levels are associated with a certain time window. The boundaries of these windows (T_0 and T_1) can be varied and depend on operational requirements. The choice of a classification method over a regression method has the advantage that these windows can be chosen according to operational planning requirements and may differ for a given application. A second advantage of predicting probabilities rather than just classes is that it reveals the degradation behavior of a component. It shows the duration and intensity that a component belongs to a certain level, rather than a single class value. The sensor data is labeled according to three defined windows: *Degradation Level 1, Degradation Level 2 and Degradation Level 3*. The probability that a given component, based on its sensor data, belongs to each of these windows is calculated by the LSTM network. The different windows are defined as follows:

- dl_1 - Degradation Level 1: RUL $> T_1$

Window configuration	*1*	*2*	*3*	*4*	*5*	*6*	*7*
Lower Limit T_0	10	10	10	10	10	10	10
Upper Limit T_1	20	30	40	50	60	70	80

Table 5: Overview of the window configurations used to compute different degradation level probabilities.

- dl_2 - Degradation Level 2: $T_1 \geq \text{RUL} > T_0$
- dl_3 - Degradation Level 3: $T_0 \geq \text{RUL}$

The system belongs to the first class, dl_1, when the remaining-useful-life (RUL) is greater than the specified upper bound of the window (T_1). The second class, dl_2, describes the window in which the RUL of the component falls between the two limits (T_1 and T_0). When the system belongs to the third class, dl_3, the component's RUL is below the lower window bound (T_0).

2) Determination of Window Sizes: The windows in this research are based on the frequency of A- and C-checks for the Airbus A319, A320, and A321. The A-check interval is approximately 750 Flight Hours, 750 Flight Cycles or 120 Calendar Days [32]. The C-check interval is approximately 7500 Flight Hours, 5000 Flight Cycles, or 730 Calendar Days. In addition, the lifetime distribution of the selected system is important to determine the appropriate window size. The distribution of the true engine lifetimes within the training dataset is given in Table 4 and shows a mean value of approximately 200 rows. If we assume that the Time Between Overhaul (TBO) for an engine is 15,000 flight hours [33, 34] and the average lifetime in the dataset is 200 rows, every row represents about 75 flight hours. An A-check interval is approximately 750 hours and corresponds to about 10 rows. Therefore, we set the lower limit (T_0) to 10 units. The upper limit (T_1) should be at least two A-checks and at most the C-check interval, so it ranges from 20 to 80 units. By combining the TBO for an engine with the intervals between A/C-checks and the average engine lifespan in the dataset, the seven window configurations shown in Table 5 are determined.

3) Feature Selection: Feature selection is the reduction of the number of input variables of a dataset for use in a predictive model [35]. Removing redundant or ambiguous features from a given dataset reduces computation time and can improve model performance. In this research, we use a feature selection method based on the statistical characteristics of the dataset. Features with values below a certain variance threshold are removed. Features with a zero variance or below a predetermined threshold have similar values in all samples and are therefore not considered useful in the prediction of component prognostics. All features that do not meet the requirement in Equation 8 are omitted.

$$\text{Var}(X_t^i) \geq 0.001 \quad \text{for} \quad i = \{1, \ldots, N\} \tag{8}$$

Features with a variance value close to zero, such as Sensor 6, shown in Figure 6, are discarded as they do not improve model performance. An overview of the statistical values of

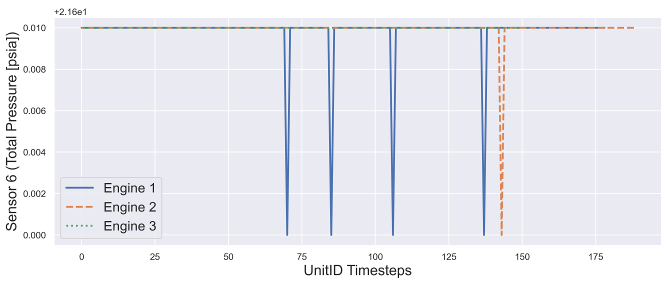


Fig. 6: Overview of Sensor 6 for three arbitrary engines of subset FD001 (C-MAPSS Dataset).

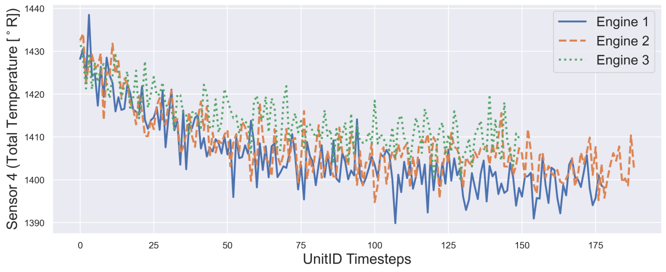


Fig. 7: Overview of Sensor 4 for three arbitrary engines of subset FD001 (C-MAPSS Dataset).

the sensor measurements is given in Table 6. Table 6 shows that the standard deviation for Sensor 1, 5, 6, 10, 16, 18, and 19 is zero and will therefore be omitted. The sensors that are selected are Sensor 2, 3, 4, 7, 8, 9, 11, 12, 13, 14, 15, 17, 20, and 21. Furthermore, we examined the trends of the sensor data, which are also summarized in Table 6 as ascending (*A*), constant (*C*), descending (*D*), and irregular (*I*). For example, Sensor 4, depicted in Figure 7, shows a descending trend.

4) Data Normalization: Normalizing sensor data is necessary because LSTM networks are unable to handle large absolute differences, as these networks are using relative distances between sensor data. Scaling sensor data improves the efficiency and robustness of the LSTM network [36]. In this research, we use minimum-maximum normalization, shown in Equation 9 [37], since the sensor data may have different ranges. Table 6 shows the large variation in absolute numbers, underscoring the importance of normalizing the features. Normalizing the sensor data results in all feature data being in the range $[0, 1]$.

$$x_{norm} = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{9}$$

Where x_{norm} is the normalized sensor value, x is the current sensor value, $\min(x)$ is the minimum sensor value and $\max(x)$ is the maximum sensor value in the input vector X_t^i.

5) Sensor Smoothing: Sensor smoothing is applied to the remaining sensor data to reduce signal noise and improve model performance. It reduces the signal noise while preserving its characteristics trends. Exponential smoothing is introduced as powerful yet fairly simple to implement [38]. The

Sensor	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*	*16*	*17*	*18*	*19*	*20*	*21*
Mean	518.7	642.7	1590.5	1408.9	14.6	21.6	553.4	2388.1	9065.2	1.3	47.5	521.4	2388.0	8143.7	8.44	0.03	393.2	2388.0	100.0	38.8	23.3
Std	0.00	0.50	6.1	9.0	0.00	0.00	0.89	0.071	22.1	0.00	0.27	0.74	0.072	19.1	0.038	0.00	1.55	0.00	0.00	0.18	0.11
Min	518.7	641.2	1571.0	1382.2	14.6	21.6	549.8	2387.9	9021.7	1.3	46.8	518.6	2387.8	8099.9	8.32	0.03	388.0	2388.0	100.0	38.1	22.8
Max	518.7	644.5	1616.9	1441.4	14.6	21.6	556.0	2388.5	9244.5	1.3	48.5	523.3	2388.5	8293.7	8.58	0.03	400.0	2388.0	100.0	39.4	23.6
Trend	*C*	*D*	*D*	*D*	*C*	*C*	*A*	*D*	*I*	*C*	*D*	*A*	*D*	*I*	*D*	*C*	*D*	*C*	*C*	*A*	*A*

Table 6: Descriptive statistics of the sensor values of subset FD001 (C-MAPSS Dataset).

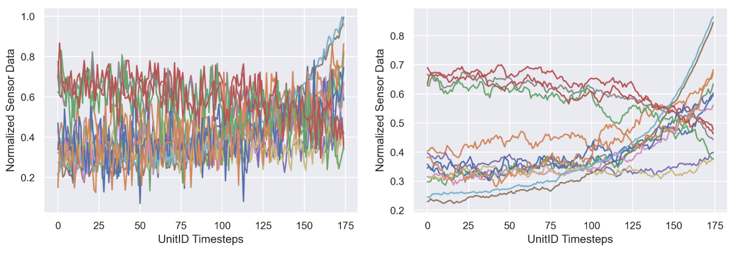


Fig. 8: The influence of exponential smoothing on sensor data from one specific engine. In the left graph $\alpha_s = 0.9$ was used, in the right graph $\alpha_s = 0.1$.

formula for exponential smoothing is given in Equation 10.

$$x_t = \alpha_s x_t + (1 - \alpha_s) x_{t-1} \tag{10}$$

Where x_t is the sensor value at timestep t, α_s is the smoothing factor and x_{t-1} is the previous sensor value. For example, if $\alpha_s = 0.9$, the sensor value at timestep t consists of 90% of the current value and 10% of the previous smoothed value. Figure 8 shows the influence of the smoothing value. The left graph uses $\alpha_s = 0.9$ and the right graph uses the finally chosen parameter, $\alpha_s = 0.1$.

6) Data Sequencing: The sensor data must be prepared for the LSTM network by creating sequences. The sequence input ensures that the network is able to calculate classification predictions based on the previous individual timesteps of the data. An illustration of the sequence length N_t is shown in Figure 4. The use of sequences allows the LSTM network to retain information over a certain specified time period. The larger the value N_t of the data sequences, the more the network is able to look back [39]. However, the value should not be too large, as this increases the computation time and may reduce the performance of the model. If the input vector for one specific feature ($i = 1$) is defined as $X_t^1 = [x_1^1, x_2^1, x_3^1, ..., x_{15}^1]$ and we assume a sequence length $N_t = 4$, then the samples would be ordered as $[x_1^1, x_2^1, x_3^1, x_4^1]$, $[x_2^1, x_3^1, x_4^1, x_5^1]$ up to $[x_{12}^1, x_{13}^1, x_{14}^1, x_{15}^1]$.

D. *Performance Evaluation Methods*

To analyze the performance of the model, evaluation metrics are introduced. As discussed in the previous sections, we compute the probabilities that the system belongs to a certain degradation level rather than predicting a single value. Since we propose a classification formulation rather than a regression formulation, we cannot use traditional evaluation metrics such as the PHM08 score, MSE, MAPE(%) or MAE [40]. Consequently, we adopt the approach proposed in [16]. A confusion matrix is widely used in evaluating the performance of deep learning models [41, 42]. The confusing matrix is a specific table layout that allows visualization of the performance of an algorithm. The columns in a confusion matrix are representing the *predicted* classes $\hat{y}$, while the rows are representing the *true* classes y. The diagonal entries are representing the correct predictions for each label. The confusion matrix is characterized by Equation 11. In this case, the entry within the confusion matrix (M_{ij}) is based on the predicted and true class, described by the degradation levels $En_t^{dl_j}$, where $j \in \{1, 2, 3\}$ for each timestep t.

$$M_{ij} = \sum_{x=1}^{N} \left((\hat{y}_x = dl_j) \cap (y_x = dl_i) \right) \tag{11}$$

In this research, we extend the prediction by including the probability that the input belongs to a particular class, rather than only predicting the class. Predicting the probabilities has the advantage that we can evaluate the degradation progression over time for each component, visualized in Figure 4. The confusion matrix, described in Equation 11, is extended by taking the average probability of each prediction for each class. The confusion matrix based on probabilities is given in Equation 12.

$$\hat{M}_{ij} = \frac{\sum_{x=1}^{N} \mathbb{P}\left((\hat{y}_x = dl_j) \cap (y_x = dl_i) \right)}{\sum_{x=1}^{N} (y_x = dl_i)} \tag{12}$$

Where $\mathbb{P}((\hat{y}_x = dl_j) \cap (y_x = dl_i))$ is the predicted probability that the engine sensor reading x belongs to dl_j while its true class is dl_i.

E. *LSTM Network Architecture*

The network parameters used in the LSTM network have a strong influence on the performance of the model [43]. The following parameters are important in the calculation of the degradation levels: smoothing factor α_s, sequence length N_t, number of layers, nodes per layer, number of epochs, and the optimizer. Table 7 contains the network and simulation parameters for the predictions of the prognostics. The parameters are determined based on a combination of experimental analysis and literature review [44, 45, 46]. We implemented the model using a Tensorflow backend for the LSTM network, written in Python 3.8. All experiments are performed on an M1 processor with 8GB of RAM. The computational performance of the experiments is discussed in Subsection VI-F. Furthermore, the following functions are used within the LSTM network:

- **Adam Optimizer** is a stochastic gradient descent method that is based on adaptive estimation of first-order and second-order moments [47]. The adaptive optimization

Description	*Value*
Number of Epochs	21
Smoothing Factor (α_s)	0.1
Sequence Length (N_t)	30
Number of Layers	2
Nodes per Layer	256
Dropout Value	0.1
LSTM Layer Activation Function	Tanh
Output Layer Activation Function	Softmax
Batch Size	64
Optimizer	Adam

Table 7: Network and simulation parameters for LSTM Network.

algorithm updates the weights of the neural network. The Adam optimizer is widely used in deep learning models because it provides strong performance in broad applications [27, 48, 49].

- **Categorical Cross Entropy** is used to analyze the class classification loss function. The loss function measures the prediction error. The function calculates the cross-entropy loss between the labels and predictions for multi-label classes. The loss function is defined as:

$$L_C = -\sum_{i=1}^{N} y_{ij} * \log \hat{y}_{ij} \tag{13}$$

Where $\hat{y_{ij}}$ is the predicted probability in the model output that predicts that element i belongs to class j. y_{ij} is the target value and 1 if class j is consistent with element i, and 0 otherwise. N is the number of classes in the model.

F. Degradation Level Probabilities Results

We use the confusion probability matrix described in Subsection III-D to analyze the prognostics results. The confusion probability matrix also indicates whether the algorithm predicts the degradation levels (dl_i) too early or too late. As mentioned in Subsubsection III-C1, the degradation levels were defined as: Degradation Level 1 (RUL $> T_1$), Degradation Level 2 ($T_1 \geq$ RUL $> T_0$), and Degradation Level 3 ($T_0 \geq$ RUL). Figure 9 shows the confusion matrix for the second configuration, where $T_0 = 10$ and $T_1 = 30$. The x-axis and y-axis represent the predicted labels $\hat{y}$ and the true labels y, respectively. The first row describes the predictions for Degradation Level 1 and exhibits high accuracy. As one can see, when the engine belongs to dl_1, the predicted state, $\hat{y}$, is also mainly dl_1. A value of 0.91 means that the class prediction was correct, i.e., $\hat{y} = y$ and the probability values for this class averaged 0.91. However, when the engine belongs to dl_1 the model predicted in some cases dl_2, with an average probability of 0.075. On the other hand, the model predicted that the system belongs to dl_1 while the true class was dl_2, with an average probability of 0.0038. However, these relatively small values are caused by the transition in classes, mainly driven by the abrupt change in the true labels of the degradation levels. Moving to $y = dl_2$, we see that the predictions are dominated by the predicted labels dl_2 and dl_3. When the engine belongs to dl_2, the predicted state probabilities are equal to 0.3 for the correct prediction ($\hat{y} = dl_2$) and 0.7 for the incorrect prediction ($\hat{y} = dl_3$). Moving to the last row, indicating that the engine belongs to dl_3, shows a probability of 1 for the correct prediction ($\hat{y} = dl_3$) and 0.0036 for the incorrect prediction ($\hat{y} = dl_2$). However, the probability of 0.7 for the true class $y = dl_2$ and predicted class $\hat{y} = dl_3$ indicates that the model predicts dl_3 slightly too early.

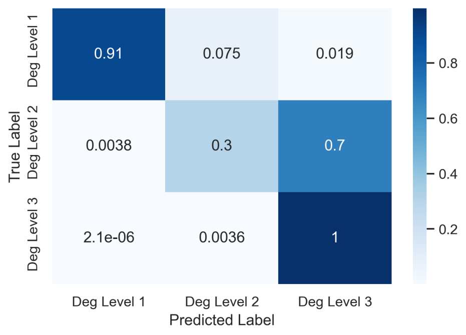


Fig. 9: Confusion probability matrix for window configuration 2, where $T_0 = 10$ and $T_1 = 30$.

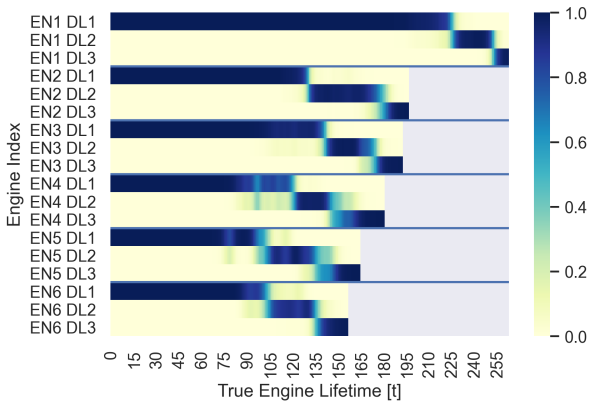
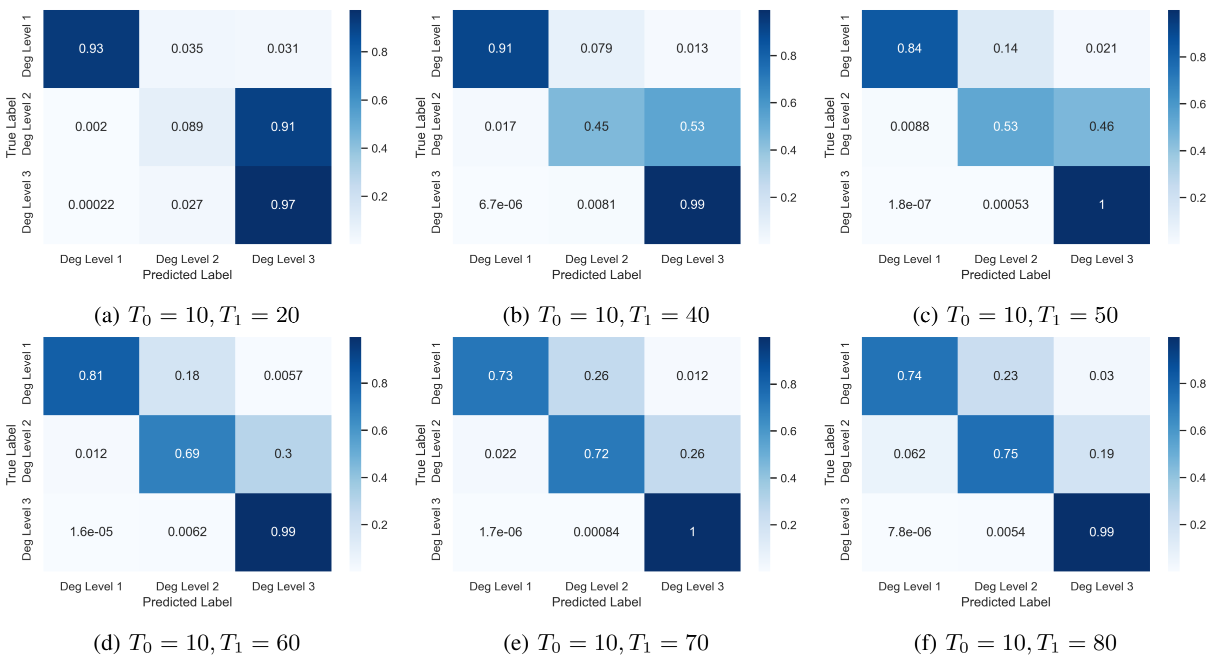


Fig. 10: Overview of degradation level probabilities (dl_j) for six different engines (En_i).

An overview of six different component prognostics is shown in Figure 10, where the horizontal axis represents the true engine lifetime. Each engine is represented by three rows, which indicate the different degradation levels. The engines are separated by a blue horizontal line. A dark color indicates a value close to one, while a light color indicates a value close to zero. At the beginning of an engine lifetime, the probability that the component belongs to dl_1 is almost one. This means that the failure probability for this aircraft engine is nearly zero. As time passes, the probability that the component belongs to dl_2 increases. When reaching the maximum lifetime of the engine, the probability of failure increases and so does the probability that the component belongs to dl_3. It is clear that the engines exhibit different degradation behavior and failure probabilities. Furthermore, the distribution of the selected engine lifetime lengths is evident, ranging from about 155 to 260 life cycles.

A summary of the results for the other configurations is depicted in Figure 11. The visible common trend is that the accuracy for $y = dl_1$, $\hat{y} = dl_1$ decreases, as the class



(a) $T_0 = 10, T_1 = 20$ (b) $T_0 = 10, T_1 = 40$ (c) $T_0 = 10, T_1 = 50$

(d) $T_0 = 10, T_1 = 60$ (e) $T_0 = 10, T_1 = 70$ (f) $T_0 = 10, T_1 = 80$

Fig. 11: Overview of the confusion probability matrices for several configurations, where T_0 and T_1 are varied.

length increases from $T_1 = 20$ to $T_1 = 80$. The value for the first setup is 0.93, while the value for the last setup is only 0.74, about 20% lower. In contrast, the accuracy for $y = dl_2, \hat{y} = dl_2$ increases as the middle window size becomes larger. The value ranges from 0.09 in the first setup to 0.75 in the last setup. The consequence of increasing the width of the middle window is clearly reflected in the prognostic accuracy. Nevertheless, the large middle window also leads to an instability problem, which is shown in the appendix [50].

IV. Predictive Maintenance Decision Framework

This section contains the proposed predictive maintenance decision framework. The component prognostics information computed in Section III should now be utilized within a maintenance framework to represent a realistic problem setting.

Reinforcement Learning (RL) has shown promising results in the field of predictive and condition-based maintenance [20]. In RL, an agent interacts with an environment and receives a reward after performing an action. The agent learns how to act by selecting actions that gives larger rewards. The goal is to maximize the total reward based on the policy between a state and an action. An overview of the interaction within an RL model is shown in Figure 12. An agent receives a representation of an environment in a state space s_t on which it takes an action a_t. Subsequently, it moves to s_{t+1} and receives a reward r_t indicating how good the action was. The maintenance environment is modelled as a discrete time Markov Decision Process (MDP). An MDP provides a mathematical formulation for decision-making under uncertainties. Moreover, the state transitions satisfy the Markov property, which implies that the future states are independent of the past states, given the current state [22]. The MDP is represented as a five item tuple (S, A, P, R, γ), where S is the set of states,

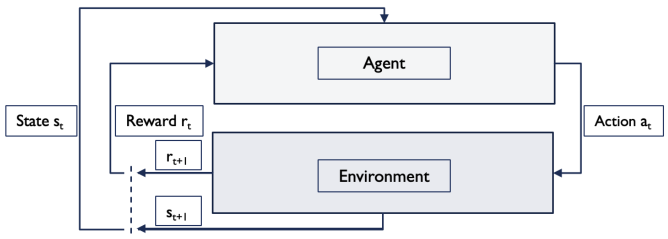


Fig. 12: Interaction between the Agent and the Environment.

A is the set of actions, P is the transition function, R is the reward formulation, and γ is the discount factor. The goal is to find an optimal policy that maximizes the total expected return R. The total return is defined as the sum of discounted rewards over a given time horizon T and is defined in Equation 14.

$$R = \sum_{t=0}^{T} \gamma^t r_t \tag{14}$$

Where γ is the discount factor ($0 \leq \gamma \leq 1$), and r_t is the instantaneous reward. When γ is large, future rewards are considered more than when γ is small.

A. State Space: $s_t \in S$

The multi-component aircraft is denoted as AC_t for each timestep t. The aircraft has two independent engines, denoted by $En_{1,t}$ and $En_{2,t}$ at timestep t. The state space vector s_t is defined in Equation 16, and contains the engine status $En^s_{i,t}$ and the degradation level probabilities $En^{dl_j}_{i,t}$ for each timestep t. Each engine has a status $En^s_{i,t}$, which indicates whether it has failed or not, and is defined as:

$$En^s_{i,t} = \begin{cases} 0, & \text{Engine } i \text{ has failed at timestep } t \\ 1, & \text{Engine } i \text{ has not failed at timestep } t \end{cases} \quad (15)$$

Moreover, $En^{dl_j}_{i,t} \in [0,1]$ defines the probability that engine $i \in \{1,2\}$ belongs to degradation level $j \in \{1,2,3\}$ for each timestep *t*. The degradation level probabilities are the outputs of the LSTM network as described in Section III. The state space s_t of the environment is defined as:

$$s_t(AC_t) = \{En^s_{1,t}, En^s_{2,t}, En^{dl_1}_{1,t}, En^{dl_2}_{1,t}, En^{dl_3}_{1,t} \\ En^{dl_1}_{2,t}, En^{dl_2}_{2,t}, En^{dl_3}_{2,t}\} \quad (16)$$

Intuitively, the initial state space ($s_{t=1}$) of a simulation is defined in Equation 17. Both engines are considered new and have a value of 1 for their engine state, $En^s_{i,1}$, as they have not failed. In addition, both engines have a value of 1.0 for $En^{dl_1}_{i,1}$, and 0.0 for both $En^{dl_2}_{i,1}$ and $En^{dl_3}_{i,1}$. As time progresses, the engines will degrade and the values in the state space will change.

$$s_t(AC_1) = \{1, 1, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0\} \quad (17)$$

An illustrative example for $t = 175$ is shown in Figure 13. The left graph indicates $En_{1,175}$ and the right graph $En_{2,175}$. The green lines represent $En^{dl_1}_{i,t}$, the orange lines $En^{dl_2}_{i,t}$, the red lines $En^{dl_3}_{i,t}$, the blue lines the scaled true remaining-useful-life, and the black vertical lines $t = 175$. The corresponding state space representation is given in Equation 18. The first two values indicate that the engines are still in operation, the subsequent three values are the probabilities that the first engine belongs to each specific degradation level, and the last three values represent the same for the second engine.

$$s_t(AC_{175}) = \{1, 1, 0.0, 0.369, 0.631, 0.145, 0.855, 0.0\} \quad (18)$$

B. Action Space: $a_t \in A$

The PdM framework action space is modelled as a vector of discrete actions. The agent must take an action a_t at each timestep *t* for the aircraft. Let A be the set of actions a_t, i.e., $a_t \in A$, the agent can take for the aircraft AC_t. The action space is defined as follows:

$$A = \{a_t\},\ a_t \in \{0, 1, 2, 3\} \quad (19)$$

Where the actions related to the aircraft AC_t are defined by the following rules:

$$a_t = \begin{cases} 0, & \text{Do Nothing } AC_t \\ 1, & \text{Repair } En_{1,t} \text{ (PM)} \\ 2, & \text{Repair } En_{2,t} \text{ (PM)} \\ 3, & \text{Repair } En_{1,t} \text{ and } En_{2,t} \text{ (OM)} \end{cases} \quad (20)$$

The agent can take only one of the four actions at each timestep *t*. The agent can decide to do nothing, repair one of the two engines as preventive maintenance (PM), or repair both engines at the same time as opportunistic maintenance (OM). As a result, we obtain the action space:

$$A = \{\text{DN } AC_t, \text{Repair } En_{1,t}, \text{Repair } En_{2,t}, \\ \text{Repair } En_{1,t} \text{ and } En_{2,t}\} \quad (21)$$

Parameter	*Description*	*Value [-]*	*Maintenance*
$-R_{nm}$	Do Nothing AC_t	0	None
$-R_{pm}$	Repair $En_{1,t}$	e^1	Preventive
$-R_{pm}$	Repair $En_{2,t}$	e^1	Preventive
$-R_{om}$	Repair $En_{1,t}$ and $En_{2,t}$	$2 \cdot R_{pm} - \epsilon_{om}$	Opportunistic
$-R_{cm}$	Repair $En_{1,t}$ and $En_{2,t}$	$2 \cdot e^{\theta_{cm}}$	Corrective

Table 8: Predictive maintenance framework cost-related reward parameters.

C. Reward Function

The reward function is described as a function of the state, action, and next state: R(s_t, a_t, s_{t+1}). The agent receives an immediate reward r_t after each action based on the outcome of the next state. The values of the reward act as feedback to improve the agent's policy, and thus have a large influence on the learning process. The objective in this research is to minimize the maintenance cost while considering the failure probabilities for each engine.

To guide the agent's actions, a reward formulation is proposed in Equation 22. If the agent takes an action a_t and in the next state s_{t+1} the engine does not fail ($En^s_{i,t+1} = 1$), the agent receives the rewards associated with the specific action. However, if the agent takes an action and an engine fails in the next timestep ($En^s_{i,t+1} = 0$), the agent receives the reward for corrective maintenance. Integrating the maintenance cost related parameters from Table 8 into a reward function motivates the agent to perform a preventive maintenance action before experiencing a corrective maintenance action based on the state space s_t. However, since the agent's goal is to maximize the cumulative reward, it is also encouraged to perform the do nothing action, since this results in an immediate reward of zero.

$$r_t = \begin{cases} -R_{nm}, & \text{if } a_t = 0 \cap En^s_{i,t+1} = 1 \\ -R_{pm}, & \text{if } a_t = 1 \cap En^s_{i,t+1} = 1 \\ -R_{pm}, & \text{if } a_t = 2 \cap En^s_{i,t+1} = 1 \\ -R_{om}, & \text{if } a_t = 3 \cap En^s_{i,t+1} = 1 \\ -R_{cm}, & \text{if } a_t = 0,1,2,3 \cap En^s_{i,t+1} = 0 \end{cases} \quad (22)$$

The parameters for the maintenance cost are shown in Table 8. The reward for the do nothing action (R_{nm}) is zero, while the agent receives a small negative reward (R_{pm}), for replacing one of the engines. If the agent decides to repair both engines at the same time, classified as opportunistic maintenance, it receives a reward (R_{om}) slightly smaller than double the preventive reward. However, if an engine has failed and needs to be correctively replaced, the agent receives a larger penalty (R_{cm}). Note that the corrective maintenance activity is not an action, but a consequence of performing an action too late because the engine has failed.

The total reward R for each episode is shown in Equation 23. The agent's goal is to minimize the sum of the immediate rewards r_t related to the maintenance actions a_t

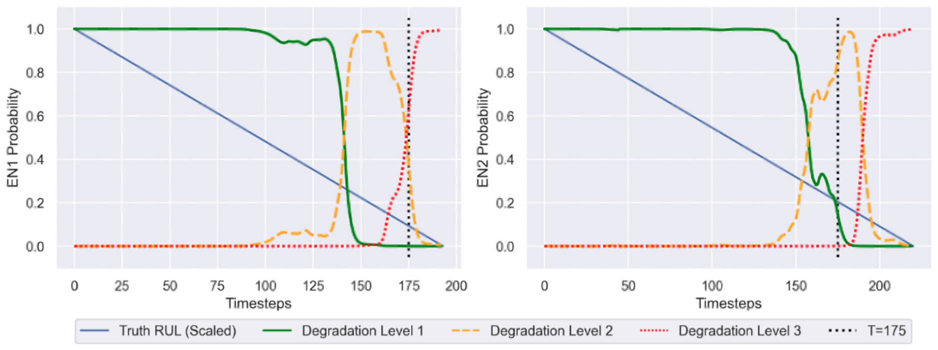


Fig. 13: State space s_t for timestep $t = 175$. Showing the degradation level probabilities and scaled true RUL for En_1 and En_2. Corresponding state space: $s_t(AC_{175})$ = {1, 1, 0.0, 0.369, 0.631, 0.145, 0.855, 0.0}.

during the time horizon T.

$$\text{Minimize } R = \sum_{t=0}^{T} \gamma^t r_t \tag{23}$$

To account for the economic dependence within this research, we introduce a parameter ϵ_{om} for the opportunistic maintenance reward R_{om}. If the agent decides to replace both components at the same time, it can gain an economic advantage over replacing both components individually. The reward for R_{om} then amounts to $2 \cdot R_{pm}$, discounted by a value ϵ_{om}. Where, the opportunistic maintenance factor ϵ_{om} holds the following values:

$$\epsilon_{om} = \{0.6, 0.7, 0.8\} \tag{24}$$

In addition, the reward for corrective maintenance, R_{cm}, is a function of θ_{cm}. The larger the value of θ_{cm}, the more negative the corrective maintenance reward becomes. This will affect the agent's behavior in terms of preventive maintenance actions. The larger the value of θ_{cm}, the more careful the agent will be about running the engine to its maximum lifetime as the reward will be more negative. The corrective maintenance factor θ_{cm} holds the following values:

$$\theta_{cm} = \{3, 4, 5, 6, 7, 8\} \tag{25}$$

Furthermore, to analyze the agent's behavior and the model's performance, we let the discount factor γ include the values stated in Equation 26. The influence of the discount factor determines the agent's cautiousness. With a lower discount factor, the agent will only consider rewards in the near future and therefore run the engines longer. The utilization of the components will be higher, but so will the chance of failure and consequently the corrective maintenance reward. The larger the discount factor is, the more the agent will take future rewards into account and thus be more prudent.

$$\gamma = \{0.5, 0.6, 0.7, 0.8, 0.9, 0.925, 0.95, 0.99, 0.999\} \tag{26}$$

An additional varied parameter is the learning rate α and includes the values stated in Equation 27. The learning rate determines how much the network values are updated and thus affects the convergence time and performance.

$$\alpha = \{0.01, 0.001, 0.0001\} \tag{27}$$

D. State Transition Function

The state transition function determines how the system environment evolves from s_t to s_{t+1}. At each timestep t, the agent processes the engine states s_t and decides what maintenance action $a_t \in A$ should be taken. The following steps are taken in one episode:

1) Compute degradation level probabilities $En_{i,t}^{dl_j}$ for each engine i at timestep t.
2) The agent selects a maintenance decision $a_t \in A$ for aircraft AC_t at timestep t.
3) Update the engine system condition $En_{i,t+1}^{dl_j}$ based on taken maintenance decision a_t at timestep t.
4) Update the environment and state space s_{t+1}.
5) Repeat until the maximum timestep is reached and terminate the episode.

The transition of each aircraft engine state depends on this maintenance decision and is shown in Equation 28. Let p be the engine identifier in the pool of P engines, i.e., $p \in \{1, \ldots, P\}$. If the agent takes the decision to do nothing, the state evolves to the next engine state: $En_{i,t+1}^{p}$, based on the prognostics calculated by the LSTM network. If the agent makes the decision to repair one of the two engines, the respective engine state resets to a new engine from the pool P and corresponding initial engine state: $En_{i,t=1}^{p+1}$.

$$s_{t+1}(En_i) = \begin{cases} En_{i,t+1}^{p}, & \text{if } a_t = \text{Do Nothing } AC_t \\ En_{i,t=1}^{p+1}, & \text{if } a_t = \text{Repair } En_{i,t} \end{cases} \tag{28}$$

E. Q-Learning

The objective in the MDP is to find an optimal policy π^* that maximizes the total reward. The policy π defines the agent's actions in different states, and is represented as an action a_t taken in state s_t. Formally, the policy π is defined as a probability $\pi(a|s)$ of taking an action $a \in A$ in state $s \in S$ [51].

$$\pi : A \times S \rightarrow [0, 1] \tag{29}$$

The state-value function $V^{\pi}(s_t)$ describes how good it is to be in state s_t. It represents the value of a state given a policy π. The state-value function is defined as the expected cumulative reward of following the policy π from state s_t at timestep t.

$$V^{\pi}(s_t) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_t, \pi\right] \tag{30}$$

Q-Learning is a model-free reinforcement learning algorithm that aims to learn the quality value of an action in a given state [20]. Q-Learning uses an state-action value function $Q^{\pi}(s_t, a_t)$ that specifies the quality of taking a specific action in a state s_t. The Q-values represent the expected reward of a state-action pair.

$$Q : S \times A \rightarrow \mathbb{R} \tag{31}$$

The Q-value function describes how good the state-action pair is and focuses on a specific action in a specific state. The Q-value function is defined as the expected cumulative reward of taking action a_t in state s_t and following the policy π. The Q-value function Q^{π} is described as:

$$Q^{\pi}(s_t, a_t) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_t, a_t, \pi\right] \tag{32}$$

The main difference is that $V^{\pi}(s_t)$ considers only the expected discounted rewards in a given state s_t over all actions according to the policy π. While $Q^{\pi}(s_t, a_t)$ considers the expected discounted rewards based on state s_t, following policy π and taking action a_t.

The Q-Learning agent initializes a table consisting of the Q-values $Q(s_t, a_t)$, which are state-action quality values and are used for selecting the optimal actions based on a state. The agent aims to learn the optimal policy π^*, which yields sequential decisions with the highest expected discounted reward. Consequently, the optimal Q-value function $Q^{\pi^*}(s_t, a_t$ is the maximum expected cumulative reward achievable from a given (s_t, a_t) pair. The optimal policy π^*, is described as:

$$\pi^*(s_t) = \arg\max_{a_t} Q^{\pi^*}(s_t, a_t) \tag{33}$$

During training, at each timestep *t*, the agent takes an action a_t in state s_t and observes the reward r_t. The quality value of this state-action pair is computed and the Q-table is updated as follows:

$$\begin{aligned} Q(s_t, a_t) \leftarrow (1-\alpha) Q(s_t, a_t) + \\ \alpha\left(r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1})\right) \end{aligned} \tag{34}$$

Where $(1-\alpha)Q(s_t, a_t)$ is the old value, weighted by the learning rate α. αr_t is the reward obtained by taking an action weighted by the learning rate α. And $\alpha\gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1})$ is the maximum reward obtained from the state s_{t+1}, weighted by the learning rate α and discount factor γ.

1) Exploration vs Exploitation: A well-known dilemma in training an agent is the exploration versus exploitation trade-off [52]. The agent must choose between exploration (making new decisions) or exploitation (repeating experienced decisions). At the beginning of the training phase, the agent must perform random actions because it does not have complete knowledge of the environment. As the episodes progress, the agent must choose the most favorable action over exploring new actions. An exponentially decaying epsilon parameter ($0 \leq \epsilon < 1$), based on the ratio of the initial and final epsilon, is chosen. The agent chooses the action with the highest Q-value with $P(1-\epsilon)$ and a random action otherwise [53]. The exploration probability decays over the total amount of episodes from an initial value of 1 to a final value of 0.01.

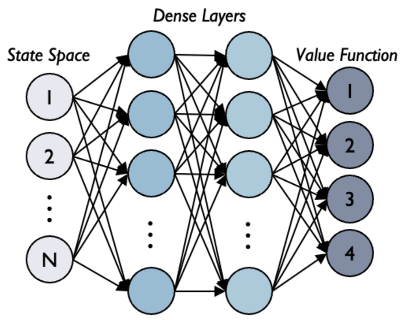


Fig. 14: Neural network used to estimate the value function in the Deep Q-Learning Network. The state space is the input and the Q-values are the output.

F. Deep Q-Learning

The disadvantage of using traditional Q-Learning is the increase in size of the Q-table when the action- or state- space increases in size. The original Q-Learning algorithm is not scalable because it must compute $Q(s_t, a_t)$ for each pair. In this research, the state space is large because of the many values it can hold. Each entry in the state space can have any value between $[0, 1]$. And since we consider a long-term horizon that includes two components, the unpredictability caused by replacing a component at any given point also contributes to the size of the Q-table. An improvement to traditional Q-Learning is Deep Q-Learning (DQL), which uses a neural network to estimate the Q-value, developed by Mnih et al. in 2015 [54]. DQL uses a function approximator to estimate the action-value function. This is done by utilizing a neural network (Figure 14) with network parameters θ, also known as weights. DQL approximates the Q-values by a function: $Q(s, a; \theta) \approx Q^*(s, a)$. The essence is that two similar states ($s_t^1 \approx s_t^2$), are about equally good or bad to be in and therefore also have a similar Q-value: $Q^{\pi}(s_t^1, a_t) \approx Q^{\pi}(s_t^2, a_t)$ for a specific a_t.

The used Deep Q-Network (DQN), shown in Figure 14, consists of an input layer, dense layers and an output layer. The input layer of the neural network receives the environment state space (s_t). The input is passed through the fully connected dense layers in the network, where the computational processes are performed. The nodes within the dense layer consist of non-linear transformation functions with network parameters θ. The output layer produces the Q-values for each possible action (a_t) for the input state (s_t). The output layer has a dimension of 4, represented by $Q(s_t, a_0)$, $Q(s_t, a_1)$, $Q(s_t, a_2)$, $Q(s_t, a_3)$.

1) Training Strategy: The DQN can be trained by iteratively minimizing the loss function $L_i(\theta_i)$ formulated in

Equation 35. The network parameters θ_i are updated with a stochastic gradient descent algorithm by minimizing the mean squared error between the target Q-values and the predicted Q-values.

$$L_i(\theta_i) = (y - Q(s_t, a_t; \theta_i))^2 \quad \text{where } y = r_t + \gamma \max_{a_{t+1}} Q\left(s_{t+1}, a_{t+1}; \theta_i^-\right) \tag{35}$$

Where the first term, y, are the Q-values of the target network, and the second term are the Q-values of the online network. The architecture of the target network is identical to that of the online network, except for the network parameters θ_i. The target network parameters (θ_i^-) are kept constant when calculating the loss function of the online network parameters (θ_i). The target network parameters are updated only every N_θ steps, which prevents the network from slipping into a worse policy. In addition, an experience sampling method is used to select a random batch of transitions to avoid correlated experiences. Each timestep, the transition s_t, a_t, r_t, s_{t+1} is stored in the replay memory buffer D. Instead of updating the network parameters θ_i based on only the last transition, a batch of experiences from D is used. Experience replay is used to ensure that the Deep Q-Network is trained not only on consecutive samples but on random mini-batches and prevents the DQN from learning the correlation between transitions.

G. Data Augmentation

Data augmentation is performed to improve the performance of the RL model [55]. In general, data augmentation is a method to increase the amount of data by adding slightly modified copies of the existing data. Data augmentation helps to reduce overfitting and acts as a regulizer. Several methods have been identified by [49], e.g. jittering, averaging, flipping, scaling or warping. We have used a method called pattern mixing, where one or more patterns are used to create new ones. Magnitude domain mixing, the most direct application of pattern mixing, uses a linear combination of the patterns at each timestep. For example, by averaging the sensor data of a specific engine between two timesteps, and using this value as a new point as augmented data. The general trend of the data remains the same, but the exact data values at a given time may vary slightly. Figure 15 shows the degradation level probabilities for a given unit, where it can be seen that the data for all timesteps t is relatively coarse. The original data points (En_t^o), around $t = 211$, are given in Equation 36, and the augmented data points (En_t^a) in Equation 37. It can be seen that the trends of the data remain the same, but the exact values differ.

$$\begin{bmatrix} s_t(En_{210}^o) \\ s_t(En_{211}^o) \\ s_t(En_{212}^o) \end{bmatrix} = \begin{bmatrix} 0.8002 & 0.1992 & 0.0004 \\ 0.7207 & 0.2784 & 0.0007 \\ 0.6282 & 0.3705 & 0.0012 \end{bmatrix} \begin{bmatrix} dl_1 \\ dl_2 \\ dl_3 \end{bmatrix} \tag{36}$$

$$\begin{bmatrix} s_t(En_{210}^a) \\ s_t(En_{211}^a) \\ s_t(En_{212}^a) \end{bmatrix} = \begin{bmatrix} 0.8343 & 0.1651 & 0.0003 \\ 0.7606 & 0.2389 & 0.0006 \\ 0.6745 & 0.3245 & 0.0010 \end{bmatrix} \begin{bmatrix} dl_1 \\ dl_2 \\ dl_3 \end{bmatrix} \tag{37}$$

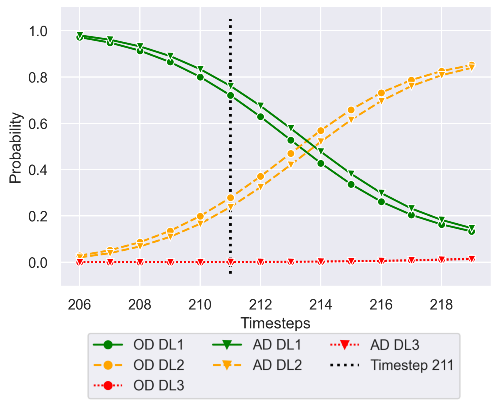


Fig. 15: Original *(OD)* and augmented data *(AD)*, zoomed in on $t = [206 - 220]$ for one specific engine.

V. Experimental Setup and Training Environment

This section first describes the terminology commonly used in the experimental design and remaining sections. Subsequently, the key performance indicators are provided. In addition, an introduction to the working principle of the PdM framework is provided. The section concludes with a parametric performance analysis and the train- and test strategy.

A. Commonly Used Terminology

The following terminology is commonly used in the descriptions of the experiments and in the results section.

- **Timestep:** Each timestep, the agent takes an action and the environment responds. The agent receives a reward based on the state-action pair.
- **Episode:** A sequential decision period of 1200 timesteps in which the agent considers several components. Each episode returns a value of the total cumulative reward.
- **Simulation:** A full run of 200 episodes for a specific combination of parameters. One simulation returns a list of results based on the total reward in each episode.
- **Experiment:** A total of 10 runs of the same simulation, with random initial parameters.

One episode has a fixed length of E_l timesteps. The episode will terminate when the predetermined number of timesteps is reached:

$$E_l = 1200 \text{ timesteps} \tag{38}$$

The length of the episode is based on the average number of components replaced within one episode. As the agent decides to do nothing or to replace a component preemptively, the average number of components considered in the first set of experiments is about 20, as shown in Table 15.

B. Key Performance Indicators PdM Framework

The long-term maintenance cost for a specific episode (with episode length E_l) is defined as follows:

$$C_{\text{pdm}} = \frac{R_{cm} \cdot N_{cm} + R_{pm} \cdot N_{pm} + R_{om} \cdot N_{om}}{E_l} \tag{39}$$

Where R_{cm} is the corrective maintenance reward, N_{cm} is the number of corrective maintenance actions, R_{pm} is the preventive maintenance reward, N_{pm} is the number of preventive maintenance actions, R_{om} is the opportunistic maintenance reward and N_{om} is the number of opportunistic maintenance actions. The long-term maintenance costs are used to compare with other maintenance policies, described in Subsection VI-A. In general, the goal is to establish a policy that determines the optimal maintenance action based on the given state space and minimizes the long-term maintenance cost C_{pdm}. Since we divide a multiplication of reward and frequency by a scalar representing a time horizon, C_{pdm} is considered a dimensionless cost rate.

Using only long-term maintenance costs to evaluate the performance of the PdM framework is not sufficient. Therefore, we introduce the following Key Performance Indicators (KPIs): the number of replaced components T_c, the average component utilization C_u, the average wasted component lifetime W_t, and the average component replacement time R_t. The KPIs are defined as follows:

1) **Number of Replaced Components:** The number of replaced components is interesting to observe the efficiency of the framework in terms of component usage. It is complementary to the long-term cost, as it only takes into account the use of the components and is not affected by the maintenance cost ratio. The number of replaced components per episode is calculated as follows:

$$T_c = N_{cm} + N_{pm} + N_{om} \tag{40}$$

Where N_{cm} is the number of components replaced by corrective maintenance, N_{pm} is the number of components replaced by preventive maintenance and N_{om} is the number of components replaced by opportunistic maintenance.

2) **Average Component Utilization:** The average component utilization indicates the ratio of replaced lifetime to true lifetime. It denotes the proportion of the component's lifetime that is used. The average component utilization for one episode is calculated as follows:

$$C_u = \frac{1}{N}\sum_{n=1}^{N}\frac{En_n^l}{En_n^a} \tag{41}$$

Where N is the number of components replaced, En_n^l is the replaced lifetime of engine n, and En_n^a is the actual true lifetime of engine n.

3) **Average Wasted Component Lifetime:** The wasted component lifetime is the difference between the true lifetime and the replaced lifetime. It is a measure of how well the components are utilized and how much lifetime is wasted. The average wasted component lifetime per episode is calculated as:

$$W_t = \frac{1}{N}\sum_{n=1}^{N}(En_n^a - En_n^l) \tag{42}$$

Where N is the number of components replaced, En_n^a is the actual true lifetime of engine n, and En_n^l is the replaced lifetime of engine n.

Description	*Value*
Training Episodes	200
Test Episodes	100
Discount Factor (γ)	0.999
Learning Rate (α)	0.001
Hidden Layers	2
Nodes in Dense Layer	64
Replay Memory Capacity	10^5
Batch Size	32
Exploration Rate (ϵ)	$[1 \rightarrow 0.01]$

Table 9: Network and simulation parameters for the Deep Q-Network.

4) **Average Component Replacement Time:** The average component replacement time indicates the replacement age at which an engine is replaced, and is defined as:

$$R_t = \frac{1}{N}\sum_{n=1}^{N}En_n^l \tag{43}$$

Where N is the number of components replaced and En_n^l is the replaced lifetime of engine n.

C. PdM Working Principle and Network Parameters

We use a Tensorforce backend, which is based on Tensorflow, for the Deep-Q Network, written in Python 3.8. All experiments were run on an M1 processor with 8GB of RAM. Table 9 shows the network and simulation parameters for the maintenance decision framework. The values chosen are based on experimental analysis and commonly used parameters in other studies [56, 57, 58] and work well for this application.

We consider 1200 timesteps as one episode, in which the agent decides each timestep whether an aircraft engine should be replaced, either by preventive or opportunistic maintenance. The decisions of the PdM framework are illustrated in Figure 16, where it can be seen that the agent takes different actions based on the degradation levels of the components represented by the state space. If a component is replaced by a maintenance decision, a random engine is selected from the dataset as the new engine. The vertical colored lines represent the maintenance actions taken by the agent. For example, the green line represents the action Repair $En_{1,t}$ at timestep t. The lines labeled "En" represent the three degradation levels for each engine ($En_{i,t}^{dl_j}$) at timestep t. An example of the state space, when the agent takes the decision to replace both engines at $t = 268$, indicated by the orange line, is given in Table 10.

Figure 17 depicts the reward values, described in Subsection IV-C, for one episode. For each maintenance decision, it receives a negative reward. The goal for the agent in one episode is to minimize the total reward received, resulting in the minimized maintenance cost. Each timestep that the agent decides to do nothing does not change the cumulative reward. Once the agent decides to perform a maintenance action, the negative reward value increases.

Figure 18 shows the reward development for 10 simulations ($N_s = 10$) and 200 episodes ($N_e = 200$). The reward convergence shows that the agent is able to learn the optimal

Timestep (t)	*State Space* (s_t)	*Action* (a_t)	*Reward* (r_t)
267	1, 1, 0.506, 0.494, 0.0, 0.523, 0.475, 0.003	Do Nothing	$-R_{nm}$
268	1, 1, 0.389, 0.611, 0.0, 0.422, 0.574, 0.004	Repair Engine 1 and 2	$-R_{om}$
269	1, 1, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0	Do Nothing	$-R_{nm}$

Table 10: Example for the state space s_t representation and corresponding maintenance actions a_t. Representing the working principle of the PdM framework for $t = [267 - 269]$.

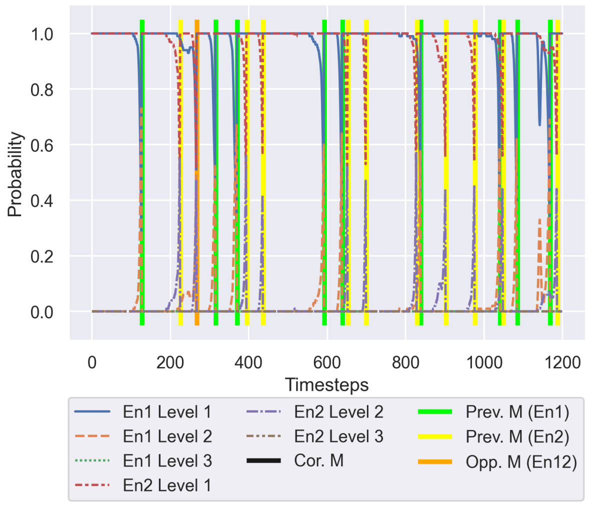


Fig. 16: Working principle of the proposed maintenance decision framework for $N_e = 1$ and $E_l = 1200$.

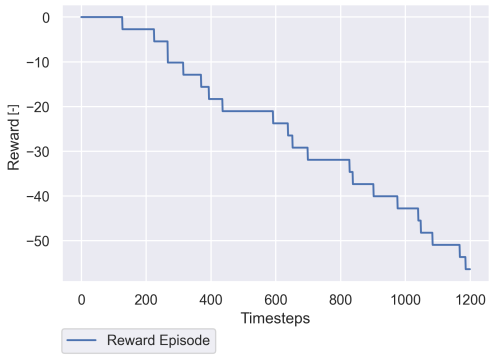


Fig. 17: Corresponding reward for $N_e = 1$ and $E_l = 1200$.

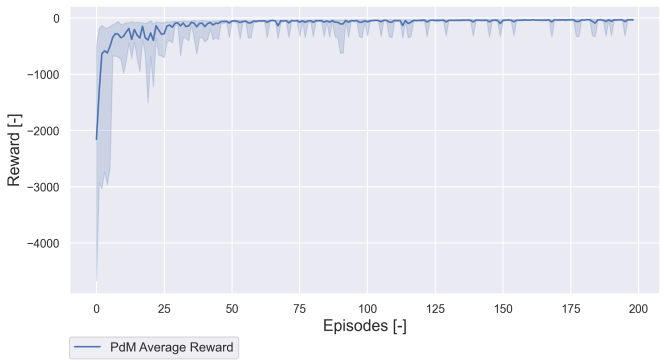


Fig. 18: Reward development for $N_e = 200$ and $N_s = 10$ during training.

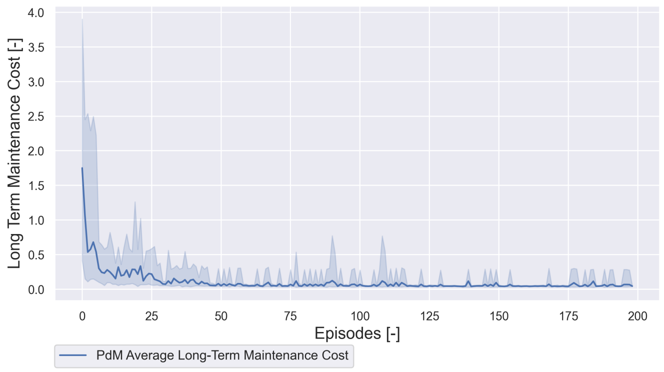


Fig. 19: Long-term maintenance cost for $N_e = 200$ and $N_s = 10$ during training.

policy based on the state space and corresponding actions. The blue line represents the sum of rewards received in one episode. The reward converges as the agent learns to perform the do nothing action until the aircraft engine needs to be replaced. It can be seen that in the first episodes the agent obtains a very large negative reward for replacing too many engines or experiencing corrective maintenance.

The long-term maintenance costs associated with the predictive maintenance strategy are shown in Figure 19. The maintenance-related cost, consisting of the preventive, opportunistic, and corrective maintenance cost, decreases over time and exhibits a convergent behavior. The agent decides in the first episodes to replace the engines frequently or to run them to their maximum lifetime, and these costs increase rapidly. As the episodes progress and the agent updates its policy, long-term maintenance costs decrease and converge to a value of 0.0524 for the last 50 episodes. Furthermore, the number of replaced components per episode is depicted in Figure 20. At the beginning of the training process, the number of components used in one episode is about 1000, but it converges to 19.2 components as an average of the last 50 episodes.

D. *Agent Action Analysis*

The action space for 10 simulations, 200 episodes of 1200 timesteps, is shown in Figure 21. The agent must take a total of $1200 \cdot 200 = 240000$ actions per simulation. It can be seen that the agent is taking the *Do Nothing* action most of the time. Whereas, the *Repair* $En_{1,t}$ or *Repair* $En_{2,t}$ actions are nearly equally distributed. As expected, the action *Repair* $En_{1,t}$ *and* $En_{2,t}$ is performed the least. The corrective

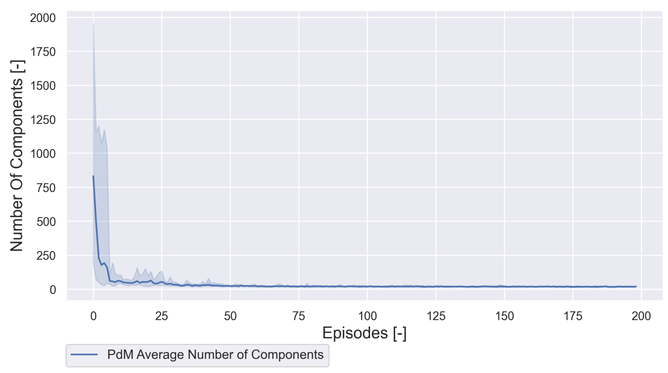


Fig. 20: Number of components for $N_e = 200$ and $N_s = 10$ during training.

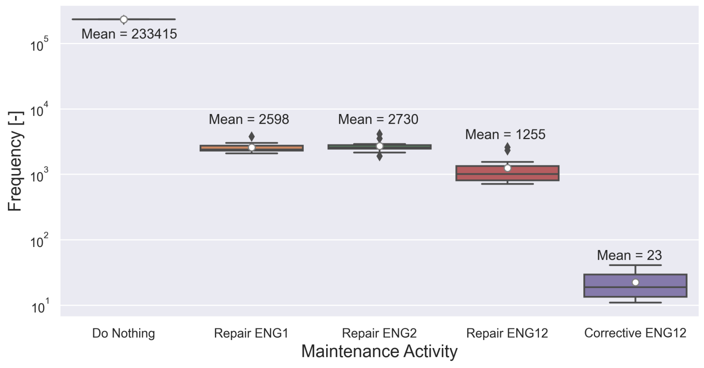


Fig. 21: Maintenance activity analysis for $N_e = 200$ and $N_s = 10$ during training. Note: y-axis uses log scale.

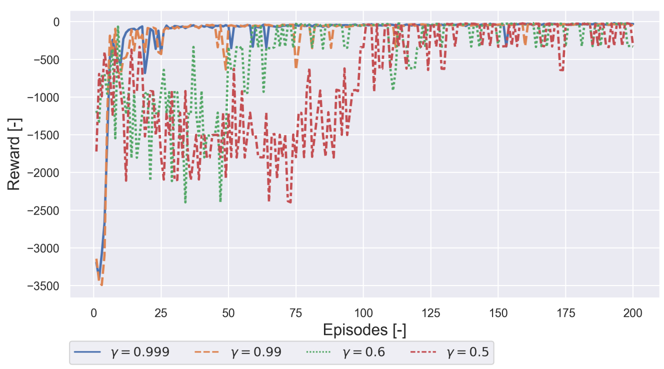


Fig. 22: Reward development for $N_e = 200$ and $N_s = 1$ during the parametric performance analysis for the discount factor γ. Note: for clarity, only the top two and bottom two γ values are plotted.

DF γ [-]	*Reward [-]*	*CM12 [-]*	*C_u [-]*	*C_{pdm} [-]*	*MF [%]*
0.999	-154.0	13	0.688	0.128	0.251
0.99	-172.7	25	0.667	0.143	0.476
0.95	-241.5	77	0.684	0.201	1.55
0.925	-200.0	87	0.751	0.167	3.06
0.9	-233.7	113	0.735	0.195	4.28
0.8	-267.4	134	0.773	0.223	4.80
0.7	-371.6	214	0.831	0.309	9.18
0.6	-473.4	284	0.825	0.395	12.50
0.5	-870.3	558	0.801	0.725	25.84

Table 11: Overview of relevant results for the parametric performance analysis for the discount factor γ. $N_e = 200$ and $N_s = 1$ during testing for every single discount factor.

maintenance activity is appended. This is not an action of the agent, but a consequence of performing a maintenance action too late because the engine failed. The distribution of boxplots indicates the agent's ability to deal with the environment in different simulations with random initial parameters.

E. *Parametric Performance Analysis*

We performed a parametric performance analysis in which we varied the model and network parameters. We analyzed the influence of the network parameters, such as the discount factor (γ) and the learning rate (α) on the performance of the model. In addition, we analyzed the influence of the model parameters, such as corrective (θ_{cm}) and opportunistic (ϵ_{om}) maintenance costs. The parameters that were ultimately chosen for the final scenarios are shown in bold in the respective table of the parameter results.

The reward development for the discount factor (γ) is shown in Figure 22. The discount factor indicates the importance of the reward in the future. If $\gamma = 0$, the agent cares only about the immediate reward. If $\gamma = 1$, the agent cares about all future rewards. The discount factor is important since the future rewards are heavily influenced by the actions of the agent. Figure 22 indicates that small discount factors ($\gamma = \{0.5, 0.6\}$), shown by the green and red lines, trigger unstable agent behavior.

Table 11 clearly shows that smaller discount factors perform worse than larger discount factors in terms of long-term maintenance cost (C_{pdm}). Moreover, the average reward for smaller γ is remarkably more negative than for larger γ. In addition, the number of corrective maintenance (CM12) actions is noticeably higher for smaller γ than for larger γ. This can be explained by the fact that the agent only cares about the next step and reward, and therefore often runs the engines up to their maximum lifetime. The component utilization (C_u) is higher with a smaller γ, but so are the long-term maintenance (C_{pdm}) costs. The last column (MF %) indicates the percentage of failed components. The percentage of failed components for small γ is higher than for large γ, explained by the myopic policy of the agent.

The reward development for the learning rate (α) is shown in Figure 23. The learning rate is a network parameter that determines the step size at each iteration, which involves a trade-off between convergence speed and overshoot. If the value of α is too large, the agent learns inefficiently and continues to overshoot, shown by the blue line in Figure 23. If the value of α is too small, the convergence rate is slow. This can be seen by the green dotted line, which shows slow convergence in the first 25 episodes. The orange line, where $\alpha = 0.001$, gives the best performance in terms of convergence speed and maintenance decision-making.

Table 12 shows that a learning rate $\alpha = 0.001$ performs the best in terms of the smallest average reward, highest compo-

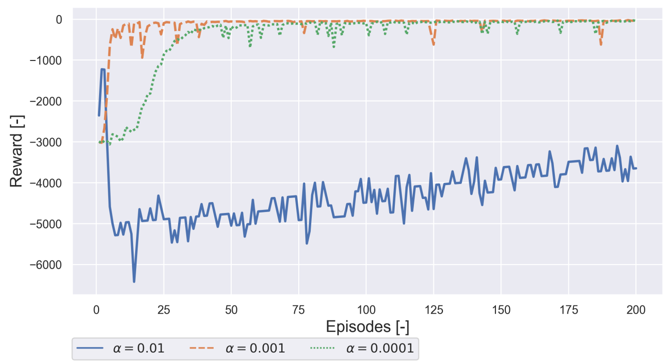


Fig. 23: Reward development for $N_e = 200$ and $N_s = 1$ during the parametric performance analysis for the learning rate α.

LR α [-]	*Reward [-]*	*CM12 [-]*	*C_u [-]*	*C_{pdm} [-]*	*MF [%]*
0.01	-5189.7	1347	0.087	4.324	1.13
0.001	-151.9	22	0.671	0.127	0.475
0.0001	-454.0	22	0.445	0.378	0.134

Table 12: Overview of relevant results for the parametric performance analysis for the learning rate α. $N_e = 200$ and $N_s = 1$ during testing for every single learning rate.

nent utilization, and lowest long-term maintenance costs.

The reward development for the corrective maintenance factor (θ_{cm}) is shown in Figure 24. The θ_{cm} value determines how heavily the maintenance activity of corrective maintenance is penalized. The reward for corrective maintenance is specified as $e^{\theta_{om}}$, which becomes larger as θ_{cm} increases. Figure 24 shows the reward development for four different values of θ_{cm}. The peaks in the reward development are caused by the increase in the corrective maintenance reward. For example, if $\theta_{cm} = 8$, indicated by the red line, it can be seen that the negative value of the reward becomes very large. As the reward of corrective maintenance increases, the agent will be more hesitant to run the engine to its maximum lifetime.

The results listed in Table 13 show that the frequency of the corrective maintenance action (CM12) decreases when the value of θ_{cm} increases. Logically, the component utilization (C_u) also decreases when θ_{cm} becomes larger.

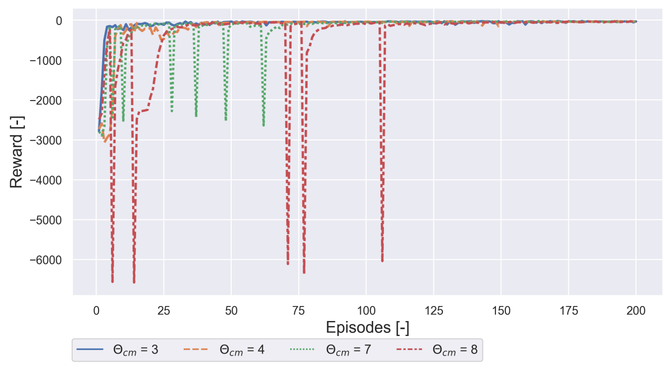


Fig. 24: Reward development for $N_e = 200$ and $N_s = 1$ during the parametric performance analysis for the corrective maintenance factor θ_{cm}. Note: for clarity, only the top two and bottom two θ_{cm} values are plotted.

CM θ_{cm} [-]	*Reward [-]*	*CM12 [-]*	*C_u [-]*	*C_{pdm} [-]*	*MF [%]*
3	-93.6	86	0.751	0.078	2.88
4	-171.2	51	0.659	0.143	0.91
5	-152.0	15	0.652	0.127	0.31
6	-175.8	14	0.653	0.147	0.30
7	-240.6	9	0.652	0.200	0.16
8	-413.9	5	0.518	0.345	0.05

Table 13: Overview of relevant results for the parametric performance analysis for the corrective maintenance factor θ_{cm}. $N_e = 200$ and $N_s = 1$ during testing for every single corrective maintenance factor.

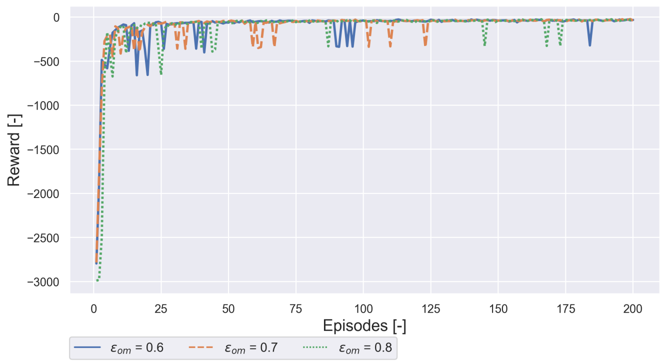


Fig. 25: Reward development for $N_e = 200$ and $N_s = 1$ during the parametric performance analysis for the opportunistic maintenance factor ϵ_{om}.

The reward development for the opportunistic maintenance factor (ϵ_{om}) is shown in Figure 25. The reward for the opportunistic maintenance action is specified as $R_{om} = 2 \cdot R_{pm} - \epsilon_{om}$. As the value of ϵ_{om} becomes larger, the opportunistic maintenance reward becomes smaller. Figure 25 indicates that there is no major difference in agent performance when ϵ_{om} is varied.

Table 14 lists the relevant KPIs and indicates that the frequency of opportunistic maintenance actions (OM12) increases when the maintenance reward R_{om} decreases. The average reward and component utilization remain about the same, indicating that the value of ϵ_{om} does not have a major impact on the agent's performance.

OM ϵ_{om} [-]	*Reward [-]*	*OM12 [-]*	*C_u [-]*	*C_{pdm} [-]*	*MF [%]*
0.6	-116.3	873	0.667	0.096	0.57
0.7	-112.7	804	0.667	0.093	0.51
0.8	-129.9	1490	0.674	0.108	0.33

Table 14: Overview of relevant results for the parametric performance analysis for the opportunistic maintenance factor ϵ_{om}. $N_e = 200$ and $N_s = 1$ during testing for every single opportunistic maintenance factor.

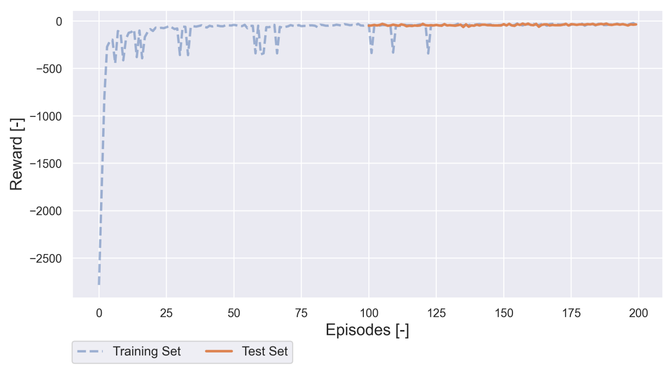


Fig. 26: Reward development for $N_e = 200$ on the training dataset *(blue)* and $N_e = 100$ on the test dataset *(orange)*.

F. Training and Test Overview

We successively conduct a training and a testing phase to evaluate the agent in the PdM framework. A clear distinction is made between training data and testing data. For the experiments, we use 80% of the dataset to train the agent and 20% of the dataset for the test part. At the beginning of each simulation, the train-test distribution is randomized, which ensures that the agent does not overfit certain engines. We let the agent train for 200 episodes on the selected 80% of the dataset. In this process, the agent observes the state, takes an action, observes the next state, receives a reward, and updates the network weights. For the testing part, we let the agent only act and not update the network weights on the remaining 20%, which consists of "held-out" data that the agent has never seen before.

In Figure 26 we show the reward development for the training set (blue) and the test set (orange). It can be seen that the agent is able to make the same decisions on the test set due to the similarity in the reward development. Figure 27 shows the rewards for the last 50 episodes to get a better perspective of the results. The moving average (MA) lines, with $n = 5$, for both training and testing show the same trend, indicating that the agent is able to transfer its knowledge and decisions onto a "held-out" dataset.

VI. Results - Predictive Maintenance Framework

This section elaborates on the results with respect to the KPIs described in Subsection V-B, where we consider the maintenance-related cost, the number of components, the average component utilization, the average wasted lifetime, and the average replacement time. For the remaining figures in this section, the shaded area is defined by the *minimum-maximum* values. First, the maintenance policy definitions used for comparison are described. Then the results for the first scenario are given. Followed by the results for the second scenario.

The PdM framework will be evaluated in two scenarios. In the first scenario, a discount factor of $\gamma = 0.999$ will be used, while in the second scenario, a discount factor of $\gamma = 0.7$ will be used. As described in Subsection V-E, the discount factor has a major impact on the agent's decisions. If

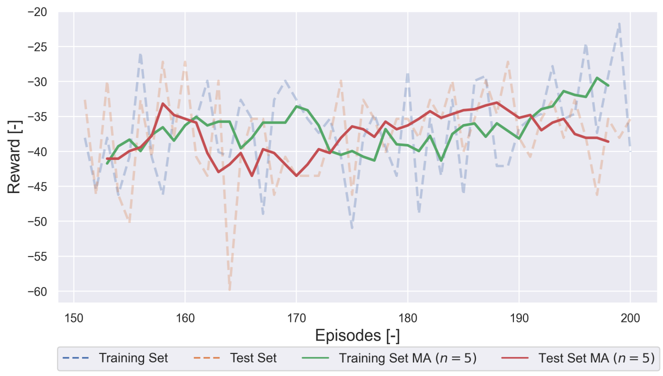


Fig. 27: Zoomed reward development, showing similar performance for the last $N_e = 50$ on the training dataset *(blue & green)* and the last $N_e = 50$ on the test dataset *(orange & red)*.

$\gamma = 0.999$ the agent takes into account all future rewards, while with $\gamma = 0.7$ the agent will adopt a more myopic policy and pay more attention to short-term rewards. The first scenario is strongly focused on achieving the lowest cost, while the second scenario shows the influence of relaxing the cost objective to achieve better performance in the other KPIs.

A. Definitions Comparison Maintenance Policies

The performance of the proposed PdM framework will be compared with three other maintenance policies to evaluate its effectiveness. Namely, Ideal maintenance, Time-Based maintenance, and Corrective maintenance. The long-term maintenance costs for these policies can be calculated as follows:

1) **Ideal Maintenance (IM)**: It is assumed that the true lifetime of each engine is known and can therefore be accurately predicted. Each engine is replaced exactly one timestep before the maximum true lifetime of that particular engine, so that no usage is wasted. The ideal maintenance strategy is chosen to compare the proposed framework with optimal maintenance decisions. There is no policy that can have a lower cost than Equation 44 and is therefore interesting to compare with. The IM policy costs can be computed as follows:

$$C_{\text{im}} = \frac{N_{pm} \cdot R_{pm}}{E_l} \tag{44}$$

Where N_{pm} is the number of replaced components, R_{pm} is the preventive maintenance reward, and E_l is the episode length.

2) **Time-Based Maintenance (TBM)**: The time between each maintenance action is determined by a value based on the average lifetime of the selected components, T_a. When an engine reaches T_a, it is replaced with a new component and considered as preventive maintenance. If the failure occurs before T_a, meaning that the true component lifetime was shorter than T_a, it is considered as corrective maintenance. The TBM policy costs can be determined as follows:

$$C_{\text{tbm}} = \frac{N_{pm} \cdot R_{pm} + N_{cm} \cdot \frac{1}{2} R_{cm}}{E_l} \tag{45}$$

Where N_{pm} is the number of preventive replaced components, R_{pm} is the preventive maintenance reward, N_{cm} is the number of corrective replaced components, R_{cm} is the corrective maintenance reward, and E_l is the episode length. R_{cm} is multiplied by half since only one of the two engines is replaced.

3) **Corrective Maintenance (CM)**: The aircraft remains in operation until an engine fails. When an engine fails, it is replaced with a new one. Therefore, the component utilization is high, but so is the cost of corrective maintenance. The long-term cost of the CM policy can be calculated as follows:

$$C_{\text{cm}} = \frac{N_{cm} \cdot \frac{1}{2} R_{cm}}{E_l} \tag{46}$$

Where N_{cm} is the number of replaced components, R_{cm} is the corrective maintenance reward, and E_l is the episode length.

B. Scenario 1: Performance of the PdM Framework Compared with other Maintenance Policies ($\gamma = 0.999$)

The first scenario are the experiments where we set $\gamma = 0.999$ to evaluate the performance of the PdM framework in terms of obtaining the lowest long-term maintenance costs. As described in Subsection V-F, the method used to evaluate the performance of the different maintenance policies is to run ten simulations, each time splitting the dataset randomly into an 80% train set and 20% test set. We train the agent for 200 episodes, followed by 100 episodes in the test part. In the test phase, we let the agent only act on the "held-out" engines to observe the performance of the agent's final policy. The other maintenance policies are evaluated on the same engines as the PdM framework. A comprehensive overview of the results, averaged over 100 episodes, is given in Table 15. The results in Table 15 include the average values as well as the minimum and maximum values to identify the varying performance. The average long-term maintenance cost for the PdM policy is 0.046 and for the IM policy 0.036. The cost for the TBM policy is 0.68 and 0.97 for the CM policy. Comparing the decisions of the proposed PdM framework with other maintenance policies yields the long-term maintenance cost results shown in Figure 28. It can be observed that the PdM policy produces superior results compared to TBM and CM policies in terms of long-term maintenance costs, and is close to the IM policy. The shaded area indicates the min-max values of the ten different simulations, whose exact values are given in Table 15.

A detailed plot of the 100 episodes and a comparison between the proposed PdM framework and IM policy is depicted in Figure 29. It shows that the PdM framework is close to the performance of the IM policy, which is expected since it takes into account the number of preventive, opportunistic, and corrective maintenance actions. The difference in cost is caused by the number of actions performed in an episode, since the PdM replaces the components earlier and thus uses more of them. However, the IM policy is an imaginary hypothesis that cannot be realized in reality. Nevertheless, the

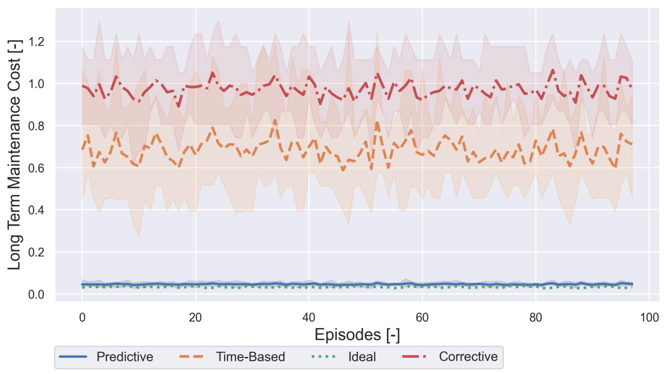


Fig. 28: Scenario 1: Long-term maintenance cost for $N_e = 100$ and $N_s = 10$ during testing.

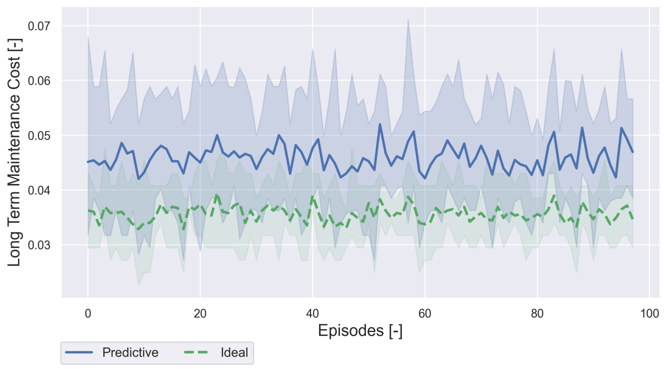


Fig. 29: Scenario 1: Long-term maintenance cost (PdM & IM) for $N_e = 100$ and $N_s = 10$ during testing.

proposed PdM framework performs well and is able to reduce operational costs compared to other policies.

The next KPI, the number of components, is shown in Figure 30. The average number of components used by the PdM framework in 100 episodes is 20.55 units, while the TBM policy requires 18.76 units. The difference is caused by the fact that the PdM framework replaces the components earlier, while the TBM policy replaces them either when they have failed or at the predetermined interval. Note that T_a is based on the mean time between failure (MTBF) of the components used in the simulation. Setting T_a to a more conservative value would increase the number of components for the TBM policy while keeping the PdM number the same. An extensive analysis on the influence of T_a is given in Subsubsection VI-B1. As expected, the IM and CM policies require the fewest components, namely 15.71 units.

Figure 31 depicts the average wasted component lifetime and shows that the TBM policy outperforms the PdM framework. The TBM policy replaces the components as corrective maintenance if they fail before T_a and thus do not waste component lifetime but incur higher costs. When the component lifetime reaches T_a, it is considered preventive maintenance and the wasted lifetime is defined as the difference between T_a and the true lifetime. The objective of the PdM framework is to minimize costs and therefore replace components before they reach their maximum lifetime, leading to a higher wasted

Maintenance Policy	*Long-Term Maintenance Cost*	*Number Components*	*Mean Wasted Lifetime*	*Mean Component Utilization*	*Mean Replacement Time*
Predictive	0.0459 (0.0271 - 0.0711)	20.55 (12 - 35)	46.87 (32.92 - 57.55)	0.664 (0.575 - 0.808)	115.35 (76.37 - 220.0)
Time-Based	0.6834 (0.2722 - 1.2503)	18.76 (12 - 26)	27.80 (9.32 - 44.29)	0.888 (0.841 - 0.959)	133.09 (94.83 - 218.96)
Ideal	0.0356 (0.0226 - 0.0475)	15.71 (10 - 21)	1.0 (1.0 - 1.0)	0.993 (0.992 - 0.996)	160.13 (113.47 - 243.1)
Corrective	0.9720 (0.6184 - 1.2986)	15.71 (10 - 21)	0.0 (0.0 - 0.0)	1.0 (1.0 - 1.0)	161.13 (114.47 - 244.1)

Table 15: Scenario 1: Mean long-term maintenance costs [-], mean number of replaced components [-], mean wasted component lifetime [t], mean component utilization [-], and mean replacement time [t] for $N_e = 100$ and $N_s = 10$. Values are indicated as: $Mean\,(Min - Max)$.

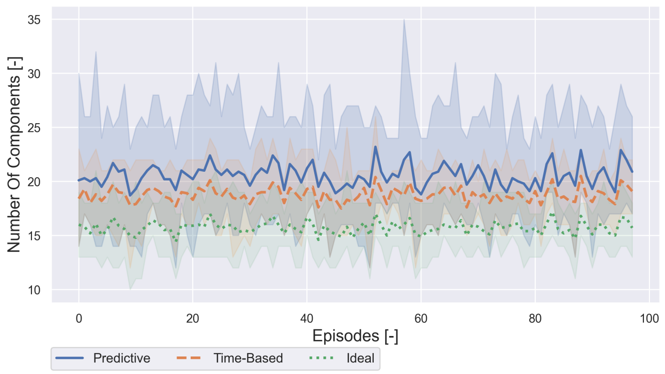


Fig. 30: Scenario 1: Number of components for $N_e = 100$ and $N_s = 10$ during testing. Note: CM is omitted because it has the same values as the IM policy.

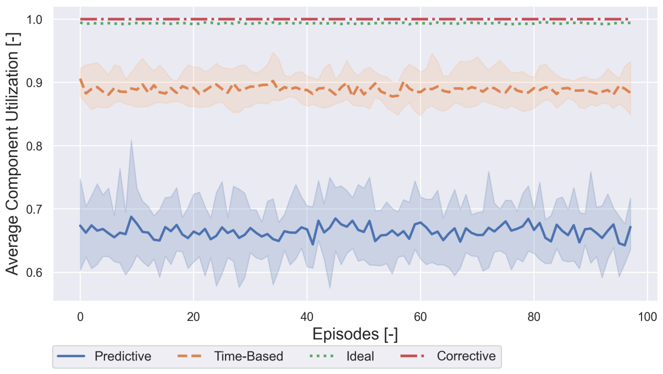


Fig. 32: Scenario 1: Average component utilization for $N_e = 100$ and $N_s = 10$ during testing.

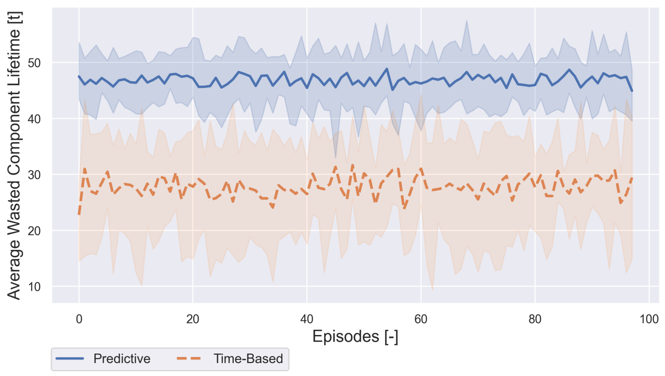


Fig. 31: Scenario 1: Average wasted component lifetime for $N_e = 100$ and $N_s = 10$ during testing. Note: CM and IM are omitted because the values for all episodes would be 0 and 1 respectively.

component lifetime. The wasted component lifetime is a trade-off between maintenance cost and component utilization. However, the spread of wasted component lifetime for the PdM framework is lower compared to the TBM policy. The average wasted component lifetime for the PdM framework is 46.87 cycles, while the average wasted component lifetime for the TBM policy is 27.80 cycles. The average wasted component lifetime for the IM and CM policies is 1 and 0, respectively. Since the IM strategy is to replace the component exactly one cycle before its maximum lifetime, and the CM strategy is to run it to its maximum lifetime.

Figure 32 depicts the average component utilization and shows that the TBM policy has a higher value than the PdM framework. The TBM policy uses 88.8% of the component lifetime while the PdM framework only uses 66.4%. The difference is caused by the fact that the PdM framework is conservative and replaces components earlier than the TBM policy. The IM and CM policies have a component utilization of 99.3% and 100%, respectively.

The average replacement age, depicted in Figure 33, shows that the PdM framework replaces components around 115.4 cycles, while the TBM does so about 133.1 cycles. The maximum values of the policies are not identical as T_a is calculated each episode and is based on the selected engines and not on the average of the total pool of components. The IM policy has an average replacement time of 160.1 cycles and the CM policy of 161.1 cycles. This phenomenon is explainable because these policies run the components to their maximum lifetime.

The results of the first scenario show that the PdM framework reduces long-term maintenance costs compared to a CM and a TBM strategy, by 95.2% and 93.2%, respectively. However, as expected, costs increase by 28.9% compared to the IM policy. Moreover, the PdM strategy uses 9.5%, 30.8%, and 30.8% more components than the TBM, IM, and CM policies, respectively.

1) Time-Based Average (T_a) Thresholds: The average time limit (T_a) reported in the above results strongly influences the results of the TBM policy. Therefore, we perform an analysis in which we set T_a to different thresholds, the results

Threshold	*LTMC: C_{tbm} [-]*	*NC: T_c [-]*	*MWL: W_t [t]*	*MCU: C_u [-]*	*MRT: R_t [t]*
T_a = Mean	0.664 (0.332 - 1.182)	18.48 (12 - 24)	26.76 (9.32 - 43.35)	0.892 (0.846 - 0.959)	135.26 (104.00 - 218.96)
$T_a = 50$	0.109 (0.109 - 0.109)	48.0 (48 - 48)	112.36 (72.85 - 188.83)	0.361 (0.217 - 0.456)	50.0 (50.0 - 50.0)
$T_a = 75$	0.072 (0.072 - 0.072)	32.0 (32 - 32)	86.76 (46.37 - 165.93)	0.544 (0.323 - 0.685)	75.0 (75.0 - 75.0)
$T_a = 100$	0.306 (0.054 - 0.833)	25.05 (24 - 26)	64.73 (27.24 - 140.5)	0.695 (0.432 - 0.842)	97.71 (93.34 - 100.0)
$T_a = 125$	0.621 (0.045 - 1.067)	21.35 (20 - 24)	47.18 (12.68 - 115.1)	0.797 (0.54 - 0.948)	115.28 (103.95 - 125.0)
$T_a = 150$	0.625 (0.036 - 1.122)	19.30 (16 - 23)	34.14 (7.15 - 88.0)	0.86 (0.655 - 0.968)	128.11 (109.56 - 150.0)
$T_a = 175$	0.672 (0.091 - 1.239)	17.86 (14 - 22)	22.76 (2.11 - 71.25)	0.913 (0.751 - 0.989)	139.17 (113.22 - 174.78)
$T_a = 200$	0.807 (0.327 - 1.241)	16.97 (13 - 22)	14.67 (0.0 - 51.92)	0.947 (0.825 - 1.0)	147.11 (116.18 - 194.84)
$T_a = 225$	0.832 (0.265 - 1.239)	16.43 (12 - 21)	8.93 (0.0 - 38.76)	0.969 (0.872 - 1.0)	152.76 (115.04 - 212.08)

Table 16: Key Performance Indicators for the Time-Based maintenance policy for different T_a values in the range $[50 - 225]$. Values are indicated as: $Mean\,(Min - Max)$.

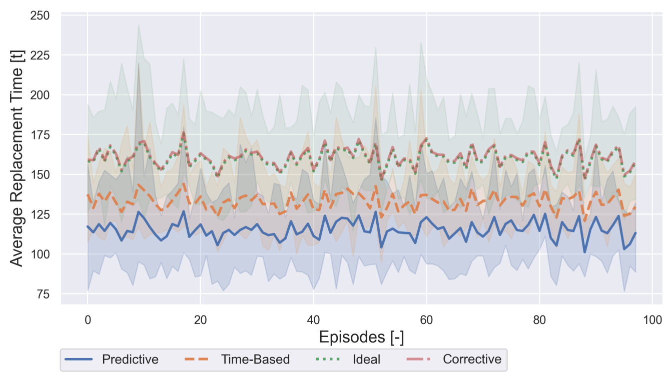


Fig. 33: Scenario 1: Average replacement time for $N_e = 100$ and $N_s = 10$ during testing.

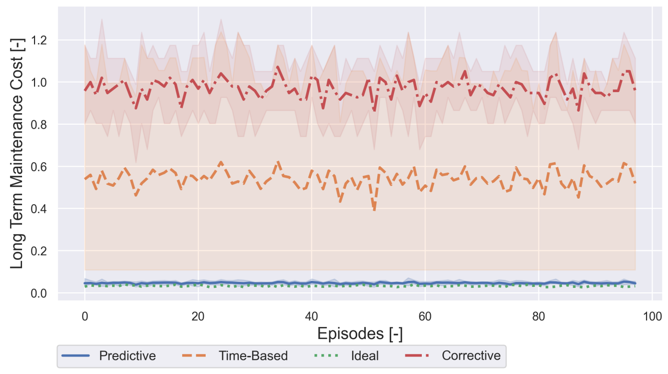


Fig. 34: Long-term maintenance cost for several T_a values, indicating the *(Min-Max)* values.

of which are shown in Table 16. The data in the table show that the Time-Based policy is sensitive to the value of T_a and never achieves lower average maintenance costs than the value of 0.046 for the PdM strategy. When we set $T_a = 50$, the long-term maintenance cost is only 0.109 versus 0.832 when $T_a = 225$. Figure 34 shows the spread of the long-term maintenance cost for the various T_a values. Consequently, the number of components drops from 48.0 to 16.43 units. As expected, the average wasted lifetime and the average replacement lifetime are inversely proportional. The component utilization rate increases from 0.361 when $T_a = 50$ to 0.969 when $T_a = 225$.

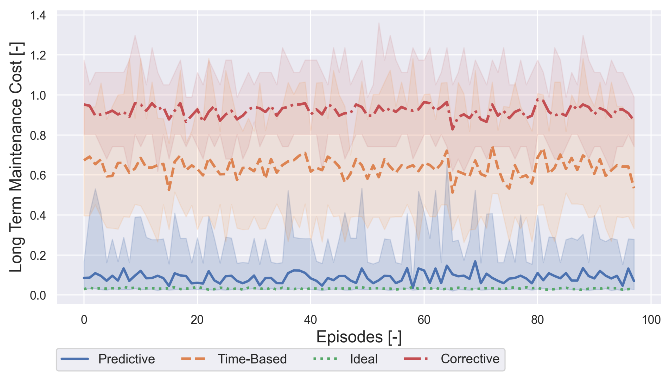


Fig. 35: Scenario 2: Long-term maintenance cost for $N_e = 100$ and $N_s = 10$ during testing.

C. Scenario 2: Performance of the PdM Framework Compared with other Maintenance Policies ($\gamma = 0.7$)

The second scenario are the experiments where we set $\gamma = 0.7$ to evaluate whether the performance of the PdM framework would increase in terms of the KPIs except long-term maintenance cost. In the previous results, we saw that the agent is relatively conservative because it replaces components early and avoids the corrective maintenance action. When we reduce the discount factor to 0.7, we expect the agent to run the components longer and thus utilize more component lifetime. The results for the second scenario can be found in Table 17. The maintenance-related costs, depicted in Figure 35, show that the PdM framework is experiencing the corrective maintenance action more frequently, as indicated by the blue peaks. The average long-term maintenance cost increases to 0.088, which is still 86.2% lower than the 0.64 from the TBM policy and 90.5% lower than the 0.91 of the CM policy. However, the cost increases by 59.9% compared to the 0.034 of the IM strategy.

As the agent makes the decision to replace an engine later, the number of components, shown in Figure 36, decreases compared to scenario 1. The PdM framework uses only 16.27 components, compared to 17.92 components for the TBM policy. The IM and CM policies are using the least components, namely 14.87 units. The component usage of the PdM framework decreases by 9.2% compared to the TBM policy and increases by 9.4% compared to the IM and CM strategies. Lowering the discount factor to 0.7 results in the

Maintenance Policy	*Long-Term Maintenance Cost*	*Number Components*	*Mean Wasted Lifetime*	*Mean Component Utilization*	*Mean Replacement Time*
Predictive	0.0876 (0.0203 - 0.6523)	16.27 (9 - 31)	23.54 (10.46 - 48.61)	0.844 (0.718 - 0.943)	147.54 (93.14 - 246.63)
Time-Based	0.6370 (0.2609 - 1.1840)	17.92 (10 - 25)	28.61 (7.54 - 51.88)	0.888 (0.817 - 0.951)	140.83 (100.29 - 240.68)
Ideal	0.0337 (0.0204 - 0.0498)	14.87 (9 - 22)	1.0 (1.0 - 1.0)	0.994 (0.990 - 0.996)	170.08 (112.41 - 265.88)
Corrective	0.9193 (0.5565 - 1.360)	14.87 (9 - 22)	0.0 (0.0 - 0.0)	1.0 (1.0 - 1.0)	171.08 (113.41 - 266.88)

Table 17: Scenario 2: Mean long-term maintenance costs [-], mean number of replaced components [-], mean wasted component lifetime [t], mean component utilization [-], and mean replacement time [t] for $N_e = 100$ and $N_s = 10$. Values are indicated as: $Mean\,(Min - Max)$.

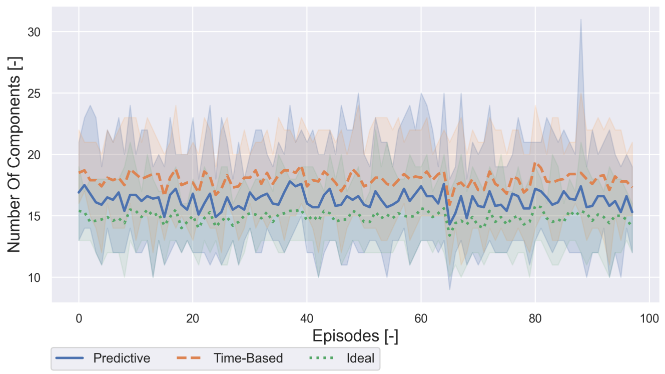


Fig. 36: Scenario 2: Number of components for $N_e = 100$ and $N_s = 10$ during testing. Note: CM is omitted because it has the same values as the IM policy.

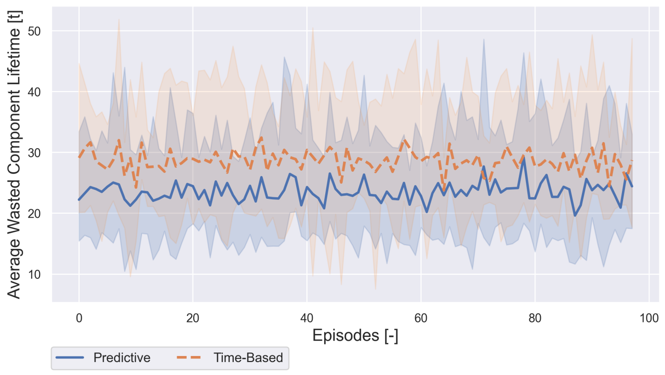


Fig. 37: Scenario 2: Average wasted component lifetime for $N_e = 100$ and $N_s = 10$ during testing. Note: CM and IM are omitted because the values for all episodes would be 0 and 1 respectively.

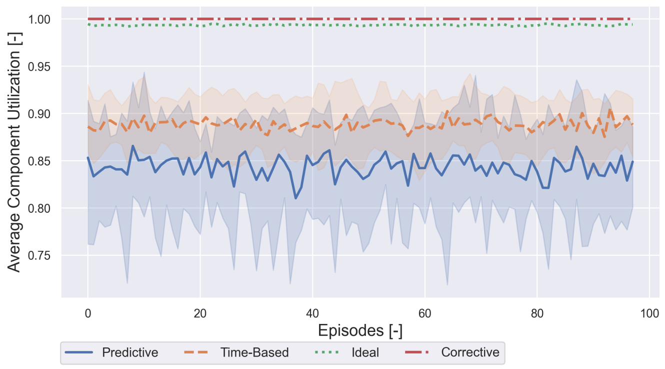


Fig. 38: Scenario 2: Average component utilization for $N_e = 100$ and $N_s = 10$ during testing.

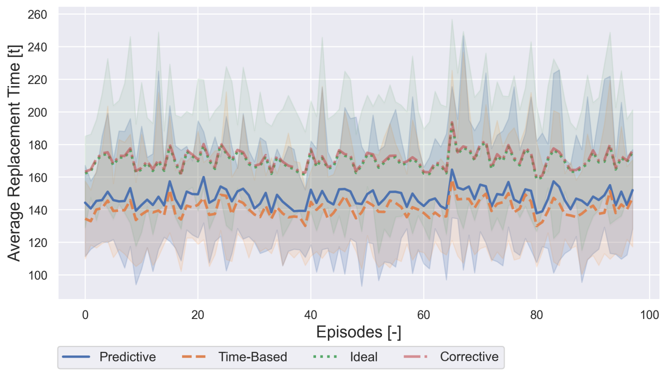


Fig. 39: Scenario 2: Average replacement time for $N_e = 100$ and $N_s = 10$ during testing.

PdM framework using fewer components than the TBM policy and being closer to the IM and CM policies.

The average wasted component lifetime, shown in Figure 37, of the PdM policy is lower than that of the TBM policy. While the PdM policy wastes an average of 23.54 cycles, the TBM policy wastes 28.61 cycles. Use of the PdM strategy instead of the TBM policy would reduce the wasted component lifetime by about 17.7%. The IM and CM policies waste an average of 1.0 and 0.0 cycles, respectively.

Figure 38 shows the average component utilization. In this scenario, the component utilization is 84.4% for the PdM policy, compared to 88.8% for the TBM policy. However, it can be seen that the dispersion of the *min-max* values of the PdM has increased compared to scenario 1, due to the corrective maintenance occurrences. The component utilization for the IM and CM policies is 0.994 and 1.0, respectively.

The average replacement time, shown in Figure 39, for the PdM policy increases to 147.54 cycles compared to 140.83 cycles for the TBM policy. Using a PdM policy over a TBM policy would increase the average replacement time by about 5%. The IM and CM policies have a replacement time of 170.1 and 171.1 cycles, respectively.

The results from scenario 2 show that the PdM framework reduces long-term maintenance costs compared to a CM and a TBM strategy, by 90.5% and 86.2%, respectively. However, as

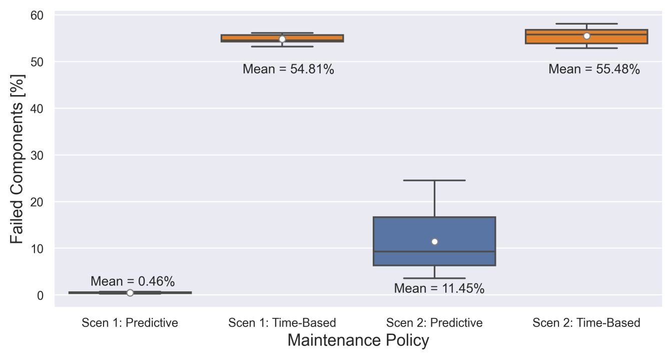


Fig. 40: Scenario 1 and 2: Percentage of failed components for the Predictive and Time-Based maintenance policies. Corrective and Ideal maintenance were omitted because they would represent 100% and 0% failed components, respectively.

expected, costs increase by 59.9% compared to the IM policy. However, the PdM strategy now uses 9.2% fewer components than the TBM policy and is competitive in terms of wasted component lifetime and replacement time. The second scenario shows that reducing the discount factor to $\gamma = 0.7$ produces comparable results in terms of cost, but better performance for the other KPIs compared to scenario 1.

D. Failed Components % Scenario 1 and Scenario 2

Figure 40 shows the percentage of failed components for the PdM framework and TBM policy for both scenarios. The Corrective and Ideal policies were omitted because they would represent 100% and 0% failed components, respectively. A comparison of the PdM framework with the TBM policy shows that the former has fewer failed components in both situations. Figure 40 indicates that the percentage of failed components, considered as corrective maintenance, is 0.46% for the PdM framework compared to 54.81% for the TBM policy for the first scenario. For the second scenario, the percentage of failed components within the PdM policy is 11.5%, while the percentage of failed components within the TBM policy is 55.5%. The increase in the failure rate for the PdM policy is caused by the decrease in the discount factor.

E. Statistical Significance

Statistical tests are conducted to better compare the performance of the PdM framework with the benchmark policies. The statistical tests are based on the differences between the values of the long-term maintenance costs and the other KPIs. Comparing two policies is done by means of a paired t-test, as the sample populations are equal for both policies. The t-test was used to test the significance of the difference between the values of the average long-term maintenance costs and other KPIs. The null hypothesis of the t-test was that there would be no difference in the respective average values. The alternative hypothesis was that the PdM values differ from the values of the comparison policy. All the statistical tests were performed with respect to a 5% level of significance. The p-values for each combination of maintenance policies are shown in Table 18. A p-value close to zero indicates that the null hypothesis is rejected. As can be seen in Table 18, all p-values are less than 0.05, i.e., $p < 0.05$. This means that for all KPIs there is a difference in the average values for the compared policies. The p-values in Table 18 are related to scenario 1, while the values for scenario 2 can be found in the appendix [50]. All values are also close to zero, with the largest value of $2.06e^{-38}$ for the PdM long-term maintenance cost compared with the IM policy.

Maintenance Policy	*Time-Based*	*Corrective*	*Ideal*
PdM C_m [-]	$9.55e^{-111}$	$9.21e^{-144}$	$2.58e^{-89}$
PdM T_c [-]	$9.94e^{-56}$	$1.47e^{-90}$	$1.47e^{-90}$
PdM W_t [t]	$4.22e^{-96}$	$1.11e^{-171}$	$9.19e^{-171}$
PdM C_u [-]	$3.58e^{-127}$	$3.23e^{-153}$	$2.54e^{-152}$
PdM R_t [t]	$3.77e^{-96}$	$3.29e^{-127}$	$3.81e^{-128}$

Table 18: Scenario 1: P-values of the statistical t-test for the long-term maintenance cost (C_m), number of replaced components (T_c), wasted component lifetime (W_t), component utilization (C_u), and component replacement time (R_t).

F. Computational Time

This section provides information on the computational time for the numerical experiments performed. All experiments were performed on an M1 processor with 8GB of RAM. The computational time for one experiment, consisting of 20 epochs, for the component prognostics was approximately 130 seconds. The prediction of the LSTM classifier on the test set took about 3 seconds. The second part of the experiments, in which the agent was trained on the component prognostics for maintenance decisions, took noticeably longer. The training part for the MDP, consisting of 200 train episodes, lasted approximately 630 seconds. The test part for the MDP, which included 100 test episodes, took approximately 285 seconds. The reduction in computational time for the test part comes from the fact that the agent only observes and acts, and no longer learns by updating the network parameters. When the model is trained and run for a couple of episodes as in the test section, it has a good computation time for practical applications. The computation time for industrial applications could be reduced by using high-performance clusters.

VII. Discussion

This section describes the strengths and limitations of the proposed PdM framework, as well as a general picture of its broader application.

A. Strengths

In this paper, we proposed a new PdM framework which is capable of mapping independent component degradation levels to maintenance decisions for a multi-component aircraft with a cost minimization objective. The numerical experiments showed that the PdM framework is able to reduce long-term maintenance costs compared to the traditional maintenance policies. This finding is highly relevant because it demonstrates a cost-effective, viable approach to making maintenance decisions based on data-driven prognostics.

Another important finding is the effect of the discount factor on the results of the PdM framework. When this observation is put into perspective at a higher level, it appears that in a real-world application, it could be a trade-off between component utilization and maintenance costs. In the first scenario ($\gamma = 0.999$), the agent is relatively conservative and components are replaced early in anticipation of failure. The results show that the goal of the agent is to replace the components before they fail and when the state space shows an increased probability for degradation level 2. In contrast, the findings of scenario 2 show that a reduction in the discount factor ($\gamma = 0.7$) results in higher maintenance costs, but also higher component utilization and replacement time. The large difference in the percentage of failed components, 0.5% versus 11.5%, highlights the effect of the discount factor on the agent's maintenance policy. In a realistic application, the different scenarios could be used when focusing on long-term maintenance costs or component utilization. Furthermore, another observation is that the agent is able to adapt to different initial data distributions, demonstrated by the same convergence behavior of the reward. From a broader perspective, this indicates that the developed PdM framework can actually be used in a real-life scenario for unknown and varying prognostics.

Moreover, this paper illustrated the PdM framework for a two-component system. Theoretically, the model could even be further extended to include a third or more components. The addition of a third component would increase the state space by *4* states and the action space by *4* actions. Therefore, the PdM framework can be scaled up relatively easily for an N-component system by increasing the state space and the action space. In addition, the PdM framework can handle different qualities of prognostic information. However, the component utilization efficiency of the framework depends on the prognostics in the state space. Finally, since the trained DQN has a strong computational performance, it can be used in practice by maintenance engineers as the maintenance decisions are made almost instantly. Nevertheless, if the PdM framework is applied in a different environment, the DQN network must be re-trained.

B. Limitations

The PdM framework also has its limitations because several assumptions and simplifications were made. The maintenance environment is a simplified representation of a real-life environment. For example, we have assumed that maintenance slots and replacement components are always available. A more realistic representation would include specified slots and a limited number of components. In reality, maintenance opportunities are highly dependent on the availability of components and manpower. Including resource availability could influence the PdM framework and lead to different results. Furthermore, we assumed an immediate replacement of the engines, whereas in real life the replacement of an engine would take several days or weeks and could affect the decisions of the PdM framework. An additional limitation in this research is the sample size of the dataset, which could be mitigated by using a larger dataset or one that is based on real data. Moreover, the rewards for maintenance are a function of e^f. For a more realistic model, the costs should be based on real costs reported by the industry. The use of real costs could influence the agents' policies in a way that leads to different maintenance decisions.

In addition, some unexpected findings were noted. Contrary to our expectations, the variation of the opportunistic maintenance factor (ϵ_{om}) was found to have no significant effect on the performance of the model, which could be a consequence of the magnitude of ϵ_{om}. As described above, this limitation point can be mitigated by using realistic costs and adjusting the acceptable risk of component failure. Finally, the PdM framework could not be compared to existing literature, as to date there is no study that reflects the exact same problem statement. An interesting experiment would be to apply the PdM framework in practice and analyze how it relates to a real maintenance policy.

C. Broader Application

Apart from its performance and flexibility in terms of the number of components, the PdM framework could also be used for other types of components. In this paper, we have demonstrated the PdM framework on a turbofan engine. However, in theory, any sensor-based component could be used, as long as it collects sensor measurements and sufficient fault information is available. For example, the PdM framework could be used for aircraft brakes, cooling units, and auxiliary power units. At a general level, if a homogeneous degradation formula can be computed for different parts, the PdM framework could be extended for a complete aircraft. In order for the PdM framework to be of practical use to an airline, it first must be analyzed in terms of efficiency and reliability in a real-world environment. Based on the results of the numerical experiments conducted, the PdM framework is likely to be of great value to an airline's maintenance operations because it can identify and respond to component deterioration behavior.

VIII. Conclusion and Recommendations

This section contains the conclusion of this research, followed by the recommendations for future work.

A. Conclusion

Aircraft maintenance is a challenging branch of engineering that constantly seeks to improve maintenance decision-making based on sensor data. In this paper, we proposed a novel end-to-end framework for maintenance decisions based on component sensor data for a multi-component system. The Predictive Maintenance (PdM) framework covers the entire process, from computing component prognostics to making maintenance decisions. The first part of the proposed method uses a Long-Short Term Memory (LSTM) network to classify to which degradation levels a component belongs. The LSTM network computes the probability that a component belongs to a given degradation level and estimates the failure within a given time window. The bounds of the time window can be chosen based on operational requirements. We evaluated the LSTM network on the C-MAPPS dataset, and the results show that the model is able to calculate accurate prognostics for several configurations.

In the second part of the PdM framework, the maintenance environment is modeled as a Markov Decision Process (MDP). A Deep Q-Learning (DQL) approach is used to obtain the optimal policy for decision-making. The objective of the PdM framework is to reduce maintenance-related costs. Maintenance decisions are made based on the probability of degradation levels, which are represented in the MDP state space. The Deep Q-Network (DQN) agent can take several actions, including do nothing, preventive, or opportunistic maintenance. The MDP rewards are based on maintenance costs and encourage the agent to replace components only when truly necessary. We evaluated the effectiveness of the PdM framework through numerical experiments. The results showed that the proposed framework outperforms traditional maintenance policies. Several Key Performance Indicators (KPIs) were used to assess the performance of the different policies, including the long-term maintenance cost, number of replaced components, component utilization, and the wasted component lifetime. The first experiment focused on obtaining the lowest long-term maintenance cost, while the second experiment focused on obtaining improved results in the other KPIs.

Specifically, the first numerical experiment, where the discount factor of the agent is equal to 0.999, showed that the PdM framework reduces long-term maintenance costs by 95.2% compared to a Corrective maintenance policy, by 93.2% compared to a Time-Based policy, but increases long-term maintenance cost by 28.9% compared to an Ideal policy. Moreover, the PdM policy uses 9.5% more components than the Time-Based policy, and 30.8% more than the Corrective and Ideal policies. It should be noted that in the proposed PdM framework, only 0.5% of the components failed during the decision period, compared to 54.8% in the Time-Based policy. The Corrective and Ideal policies represent 100% and 0% failed components, respectively. The combination of low maintenance costs and percentile improvements suggests that the proposed PdM framework is superior to using traditional approaches.

The second numerical experiment, where the discount factor of the agent is equal to 0.7, shows that long-term maintenance costs are reduced by 90.5% and 86.2% relative to the Corrective and Time-Based policies, respectively. In contrast, the long-term maintenance cost increased by 59.9% relative to the Ideal policy. However, the PdM framework now uses 9.2% fewer components than the Time-Based policy, and 9.4% more than the Corrective and Ideal policies. In this scenario, about 11.5% of the components in the PdM framework failed, compared to 55.5% of the components in the Time-Based policy. Reducing the discount factor yields better performance for the other KPIs, while keeping maintenance costs low compared to other traditional maintenance policies.

Finally, our proposed PdM framework shows how component prognostics can be successfully integrated into a maintenance decision framework. The approach taken is able to deal with an aircraft system consisting of multiple components. In summary, this paper evaluated a novel predictive maintenance framework capable of mapping degradation levels of independent components to make maintenance decisions for a multi-component aircraft with a cost minimization object.

B. Recommendations

Based on the results of this study, several directions for further research are identified. In future work, it would be interesting to look at multi-objective optimization to address other objectives. Addressing multiple objectives can be done by including other variables in the reward function in addition to maintenance costs (e.g., operating costs or downtime costs). To provide a more realistic picture, a time penalty for repairs can also be included. In addition, the state space could be expanded in future work to include component availability and maintenance slots. The inclusion of maintenance opportunities would provide a more realistic view of a maintenance environment. Furthermore, the possibility of "imperfect" maintenance could be added to the model by a different initial distribution of the degradation levels. Flight schedules and crew assignment could be taken into account, which would result in additional mathematical modeling. Finally, to better understand the performance of the model, benchmarking with a real maintenance policy can be performed and a larger number of key performance indicators could be used.

Acknowledgment

The author is sincerely grateful for the consistent and helpful guidance provided by Dr. M. Mitici (Delft University of Technology) and I.I. De Pater (Delft University of Technology) during this project.

References

[1] D. Dinis, A. Barbosa-Póvoa, and P. Teixeira, “Valuing data in aircraft maintenance through big data analytics: A probabilistic approach for capacity planning using Bayesian networks,” *Computers and Industrial Engineering*, vol. 128, no. October 2018, pp. 920–936, 2019. [Online]. Available: https://doi.org/10.1016/j.cie.2018.10.015

[2] D. Dinis and A. P. Barbosa-Póvoa, “On the optimization of aircraft maintenance management,” *Operations Research and Big Data: IO2015-XVII Congress of Portuguese Association of Operational Research (APDIO)*, vol. 15, no. June 2020, pp. 49–57, 2015.

[3] R. Pick, “Predictive maintenance: taking better care of fleets with OMAHA.” [Online]. Available: https://www.lufthansa-industry-solutions.com/de-en/solutions-products/aviation/advancing-the-future-of-predictive-maintenance

[4] J. Daily and J. Peterson, “Predictive Maintenance: How Big Data Analysis Can Improve Maintenance,” in *Supply Chain Integration Challenges in Commercial Aerospace*. Cham: Springer International Publishing, 2017, pp. 267–278. [Online]. Available: http://link.springer.com/10.1007/978-3-319-46155-7_18

[5] J. Ibarz, J. Tan, C. Finn, M. Kalakrishnan, P. Pastor, and S. Levine, “How to train your robot with deep reinforcement learning: lessons we have learned,” *The International Journal of Robotics Research*, vol. 40, no. 4-5, pp. 698–721, 4 2021. [Online]. Available: http://journals.sagepub.com/doi/10.1177/0278364920987859

[6] M. W. Brittain, X. Yang, and P. Wei, “Autonomous Separation Assurance with Deep Multi-Agent Reinforcement Learning,” *Journal of Aerospace Information Systems*, pp. 1–16, 2021.

[7] S. Navathe, W. Wu, S. Shekhar, X. Du, X. Sean Wang, and H. Xiong, "Database Systems for Advanced Applications: 21st International Conference, DASFAA 2016 Dallas, TX, USA, April 16–19, 2016 Proceedings, Part I," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9642, pp. 214–228, 2016.

[8] X. Li, Q. Ding, and J. Q. Sun, "Remaining useful life estimation in prognostics using deep convolution neural networks," *Reliability Engineering and System Safety*, vol. 172, no. November 2017, pp. 1–11, 2018. [Online]. Available: https://doi.org/10.1016/j.ress.2017.11.021

[9] P. R. d. O. da Costa, A. Akçay, Y. Zhang, and U. Kaymak, "Remaining useful lifetime prediction via deep domain adaptation," *Reliability Engineering and System Safety*, vol. 195, no. August 2019, p. 106682, 2020. [Online]. Available: https://doi.org/10.1016/j.ress.2019.106682

[10] C. Che, H. Wang, Q. Fu, and X. Ni, "Combining multiple deep learning algorithms for prognostic and health management of aircraft," *Aerospace Science and Technology*, vol. 94, p. 105423, 2019. [Online]. Available: https://doi.org/10.1016/j.ast.2019.105423

[11] O. Bektas, J. A. Jones, S. Sankararaman, I. Roychoudhury, and K. Goebel, "A neural network filtering approach for similarity-based remaining useful life estimation," *International Journal of Advanced Manufacturing Technology*, vol. 101, no. 1-4, pp. 87–103, 2019.

[12] A. Delmas, M. Sallak, W. Schon, and L. Zhao, "Remaining useful life estimation methods for predictive maintenance models: defining intervals and strategies for incomplete data," 2018.

[13] D. Laredo, Z. Chen, O. Schütze, and J. Q. Sun, "A neural network-evolutionary computational framework for remaining useful life estimation of mechanical systems," *Neural Networks*, vol. 116, pp. 178–187, 2019. [Online]. Available: https://doi.org/10.1016/j.neunet.2019.04.016

[14] M. Sri, K. Kopuru, S. Rahimi, and K. T. Baghaei, "Recent Approaches in Prognostics: State of the Art," *International Conference Artificial Intelligence*, pp. 358–365, 2019.

[15] L. Zhang and J. Zhang, "A Data-Driven Maintenance Framework under Imperfect Inspections for Deteriorating Systems Using Multitask Learning-Based Status Prognostics," *IEEE Access*, vol. 9, no. Ddm, pp. 3616–3629, 2021.

[16] K. T. Nguyen and K. Medjaher, "A new dynamic predictive maintenance framework using deep learning for failure prognostics," *Reliability Engineering and System Safety*, vol. 188, no. February, pp. 251–262, 2019. [Online]. Available: https://doi.org/10.1016/j.ress.2019.03.018

[17] C. Chen, C. Wang, N. Lu, B. Jiang, and Y. Xing, "A data-driven predictive maintenance strategy based on accurate failure prognostics," *Eksploatacja i Niezawodnosc*, vol. 23, no. 2, pp. 387–394, 2021.

[18] X. Ma, B. Liu, L. Yang, R. Peng, and X. Zhang, "Reliability analysis and condition-based maintenance optimization for a warm standby cooling system," *Reliability Engineering and System Safety*, vol. 193, no. July 2019, p. 106588, 2020. [Online]. Available: https://doi.org/10.1016/j.ress.2019.106588

[19] M. Knowles, D. Baglee, and S. Wermter, "Reinforcement learning for scheduling of maintenance," *Res. and Dev. in Intelligent Syst. XXVII: Incorporating Applications and Innovations in Intel. Sys. XVIII - AI 2010, 30th SGAI Int. Conf. on Innovative Techniques and Applications of Artificial Intel.*, pp. 409–422, 2011.

[20] Y. Hu, X. Miao, J. Zhang, J. Liu, and E. Pan, "Reinforcement learning-driven maintenance strategy: A novel solution for long-term aircraft maintenance decision optimization," *Computers and Industrial Engineering*, vol. 153, no. October 2020, p. 107056, 2021. [Online]. Available: https://doi.org/10.1016/j.cie.2020.107056

[21] K. S. Hoong Ong, D. Niyato, and C. Yuen, "Predictive Maintenance for Edge-Based Sensor Networks: A Deep Reinforcement Learning Approach," *IEEE World Forum on Internet of Things, WF-IoT 2020 - Symposium Proceedings*, 2020.

[22] E. Skordilis and R. Moghaddass, "A deep reinforcement learning approach for real-time sensor-driven decision making and predictive analytics," *Computers and Industrial Engineering*, vol. 147, no. June, p. 106600, 2020. [Online]. Available: https://doi.org/10.1016/j.cie.2020.106600

[23] N. Zhang and W. Si, "Deep reinforcement learning for condition-based maintenance planning of multi-component systems under dependent competing risks," *Reliability Engineering and System Safety*, vol. 203, no. June, p. 107094, 2020. [Online]. Available: https://doi.org/10.1016/j.ress.2020.107094

[24] A. Kuhnle, J. Jakubik, and G. Lanza, "Reinforcement learning for opportunistic maintenance optimization," *Production Engineering*, vol. 13, no. 1, pp. 33–41, 2019. [Online]. Available: http://dx.doi.org/10.1007/s11740-018-0855-7

[25] M. Kraus, S. Feuerriegel, and A. Oztekin, "Deep learning in business analytics and operations research: Models, applications and managerial implications," *European Journal of Operational Research*, vol. 281, no. 3, pp. 628–641, 2020. [Online]. Available: https://doi.org/10.1016/j.ejor.2019.09.018

[26] Y. Lei, N. Li, L. Guo, N. Li, T. Yan, and J. Lin, "Machinery health prognostics: A systematic review from data acquisition to RUL prediction," *Mechanical Systems and Signal Processing*, vol. 104, pp. 799–834, 2018. [Online]. Available: https://doi.org/10.1016/j.ymssp.2017.11.016

[27] G. Pasa, I. Medeiros, and T. Yoneyama, "Operating condition-invariant neural network-based prognostics methods applied on turbofan aircraft engines," *Proceedings of the Annual Conference of the Prognostics and Health Management Society, PHM*, vol. 11, no. 1, pp. 1–10, 2019.

[28] A. Listou Ellefsen, E. Bjørlykhaug, V. Æsøy, S. Ushakov, and H. Zhang, "Remaining useful life predictions for turbofan engine degradation using semi-supervised deep architecture," *Reliability Engineering and System Safety*, vol. 183, no. November 2018, pp. 240–251, 2019. [Online]. Available: https://doi.org/10.1016/j.ress.2018.11.027

[29] A. Elsheikh, S. Yacout, and M. S. Ouali, "Bidirectional handshaking LSTM for remaining useful life prediction," *Neurocomputing*, vol. 323, pp. 148–156, 2019. [Online]. Available: https://doi.org/10.1016/j.neucom.2018.09.076

[30] A. Saxena, K. Goebel, D. Simon, and N. Eklund, "Damage Propagation Modeling for Aircraft Engine Run-to-Failure Simulation," pp. 1–9, 2008. [Online]. Available: https://ti.arc.nasa.gov/tech/dash/groups/pcoe/prognostic-data-repository/

[31] E. Ramasso and A. Saxena, "Review and analysis of algorithmic approaches developed for prognostics on CMAPSS dataset," *PHM 2014 - Proceedings of the Annual Conference of the Prognostics and Health Management Society 2014*, pp. 612–622, 2014.

[32] P. Andrade, C. Silva, B. Ribeiro, and B. F. Santos, "Aircraft Maintenance Check Scheduling Using Reinforcement Learning," *Aerospace*, vol. 8, no. 4, p. 113, 4 2021. [Online]. Available: https://www.mdpi.com/2226-4310/8/4/113

[33] C. Kjelgaard, "GE Delivering New GEnx Durability Upgrade," 2018. [Online]. Available: https://www.ainonline.com/aviation-news/air-transport/2018-07-12/ge-delivering-new-genx-durability-upgrade

[34] S. Ackert, "Keeping Score : Analysis of an Engine's Shop Visit Rate," pp. 0–15, 2015.

[35] A. Stetco, F. Dinmohammadi, X. Zhao, V. Robu, D. Flynn, M. Barnes, J. Keane, and G. Nenadic, "Machine learning methods for wind turbine condition monitoring: A review," *Renewable Energy*, vol. 133, pp. 620–635, 2019. [Online]. Available: https://doi.org/10.1016/j.renene.2018.10.047

[36] Z. Chen, H. Zhao, X. Shu, Y. Zhang, J. Shen, and Y. Liu, "Synthetic state of charge estimation for lithium-ion batteries based on long short-term memory network modeling and adaptive H-Infinity filter," *Energy*, vol. 228, p. 120630, 2021. [Online]. Available: https://doi.org/10.1016/j.energy.2021.120630

[37] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*. Elsevier, 2012. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/C20090618195

[38] P. D. Suleyman, "Short term predictive demand model based on transport times for the reverse supply chain."

[39] H. Yin, X. Zhang, F. Wang, Y. Zhang, R. Xia, and J. Jin, "Rainfall-runoff modeling using LSTM-based multi-state-vector sequence-to-sequence model," *Journal of Hydrology*, vol. 598, no. February, p. 126378, 2021. [Online]. Available: https://doi.org/10.1016/j.jhydrol.2021.126378

[40] E. Ramasso and A. Saxena, "Performance benchmarking and analysis of prognostic methods for CMAPSS datasets," *International Journal of Prognostics and Health Management*, vol. 5, no. 2, pp. 1–15, 2014.

[41] F. O. Ozkok and M. Celik, "A hybrid CNN-LSTM model for high resolution melting curve classification," *Biomedical Signal Processing and Control*, vol. 71, 2022.

[42] A. Luque, A. Carrasco, A. Martín, and A. de las Heras, "The impact of class imbalance in classification performance metrics based on the binary confusion matrix," *Pattern Recognition*, vol. 91, pp. 216–231, 7 2019. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0031320319300950

[43] W. Yu, I. Y. Kim, and C. Mechefske, "An improved similarity-based prognostic algorithm for RUL estimation using an RNN autoencoder scheme," *Reliability Engineering and System Safety*, vol. 199, no. February, p. 106926, 2020. [Online]. Available: https://doi.org/10.1016/j.ress.2020.106926

[44] Z. Shi and A. Chehade, "A dual-LSTM framework combining change

point detection and remaining useful life prediction," *Reliability Engineering and System Safety*, vol. 205, 2021.

[45] S. Fu, Y. Zhang, L. Lin, M. Zhao, and S. s. Zhong, "Deep residual LSTM with domain-invariance for remaining useful life prediction across domains," *Reliability Engineering and System Safety*, vol. 216, no. February, p. 108012, 2021. [Online]. Available: https://doi.org/10.1016/j.ress.2021.108012

[46] Y. Wu, M. Yuan, S. Dong, L. Lin, and Y. Liu, "Remaining useful life estimation of engineered systems using vanilla LSTM neural networks," *Neurocomputing*, vol. 275, pp. 167–179, 2018. [Online]. Available: https://doi.org/10.1016/j.neucom.2017.05.063

[47] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pp. 1–15, 2015.

[48] S. R. Saufi, Z. A. B. Ahmad, M. S. Leong, and M. H. Lim, "Challenges and opportunities of deep learning models for machinery fault detection and diagnosis: A review," *IEEE Access*, vol. 7, no. Dl, pp. 122 644–122 662, 2019.

[49] B. K. Iwana and S. Uchida, "An empirical survey of data augmentation for time series classification with neural networks," *PLoS ONE*, vol. 16, no. 7 July, pp. 1–43, 2021.

[50] L. Posthuma, "Predictive Maintenance Decisions for a Multi-Component Aircraft based on Prognostics," Delft University of Technology (TU Delft), Tech. Rep., 2022.

[51] E. F. Morales and J. H. Zaragoza, "An introduction to reinforcement learning," *Decision Theory Models for Applications in Artificial Intelligence: Concepts and Solutions*, pp. 63–80, 2011.

[52] D. Xu, F. Zhu, Q. Liu, and P. Zhao, "Improving exploration efficiency of deep reinforcement learning through samples produced by generative model[Formula presented]," *Expert Systems with Applications*, vol. 185, 2021.

[53] A. D. Tijsma, M. M. Drugan, and M. A. Wiering, "Comparing exploration strategies for Q-learning in random stochastic mazes," *2016 IEEE Symposium Series on Computational Intelligence, SSCI 2016*, 2017.

[54] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015. [Online]. Available: http://dx.doi.org/10.1038/nature14236

[55] M. Laskin, K. Lee, P. Abbeel, A. Stooke, A. Srinivas, and L. G. Nov, "Reinforcement Learning with Augmented Data," no. NeurIPS, 2020.

[56] W. Cheng, S. Deng, L. Zeng, Y. Wang, and A. Brinkmann, "AIOC2: A deep Q-learning approach to autonomic I/O congestion control in Lustre," *Parallel Computing*, vol. 108, no. August, 2021.

[57] P. Yan and S. Choudhury, "Deep Q-learning enabled joint optimization of mobile edge computing multi-level task offloading," *Computer Communications*, vol. 180, no. August 2020, pp. 271–283, 2021. [Online]. Available: https://doi.org/10.1016/j.comcom.2021.09.028

[58] H. Park, M. K. Sim, and D. G. Choi, "An intelligent financial portfolio trading strategy using deep Q-learning," *Expert Systems with Applications*, vol. 158, p. 113573, 2020. [Online]. Available: https://doi.org/10.1016/j.eswa.2020.113573

II

Literature Study

Previously graded under AE4020

1 Introduction

A large percentage of an airline's operation costs are caused by the maintenance of airplanes. Therefore, airlines try to reduce the expenditures, which are caused by maintenance costs as much as possible (IATA [2019]). One of the top three savings for airlines comes from 'Health monitoring and predictive maintenance driven by improved reliability'. An effective application of health monitoring and predictive maintenance results in reduced maintenance costs and improved aircraft availability. Within the field of engineering, the above described method is known as 'Prognostic Health and Management' (PHM). The development and improvement of Predictive Maintenance (PdM) is an important branch in this field because it may result in lower operational costs. PdM is focusing on using intelligent sensors to provide reliable prognostics about the remaining useful life (RUL) of a component or system. These prognostics can be used to efficiently adjust the required maintenance activities to the system's properties.

The PdM framework consists of two sub components. The first component focuses on computing prognostics for components or systems while the second part focuses on maintenance optimization. Little research has been done in the field of a complete PdM framework. Therefore, this literature review focuses on integrating the prognostics into aircraft maintenance planning. The following topics are discussed within this literature review:

- Relevant literature
- Problem formulation
    - Research objective
    - Research questions
    - Research methodology
- Project planning

The report is structured as follows: in Chapter 2, the theoretical background of aircraft maintenance is discussed. Among others, the planning of maintenance and existing constraints are described in detail. Chapter 3 contains a review on the development of prognostics and different methods to compute them. Afterwards, in Chapter 4 the combination of maintenance and prognostics is reviewed, which covers the implementation of prognostics into maintenance planning. Subsequently, within Chapter 5, the problem formulation is defined. This includes the research objective, questions, scope and methodology. Finally, in Chapter 6 the conclusion is drawn. After a brief recap of the literature on 'integrating prognostics into aircraft maintenance planning' the conclusions are discussed. Additionally, in Chapter 7 a Gantt Chart is provided for the research project planning.

2 Aircraft Maintenance

Aircraft maintenance is a vital part in the operations of an airline. Maintenance tasks are required actions, which have to be performed to ensure airworthiness for an aircraft or aircraft part. The following chapter provides information about the maintenance strategies of airlines. According to Man [2010], maintenance can be defined as follows: "The combination of all technical, administrative and managerial actions during the life cycle of an item intended to retain it in, or restore it to, a state in which it can perform the required function". The differences between various existing maintenance policies regarding daily operations will be described in Section 2.1. Subsequently, Section 2.2 describes the scheduling of maintenance policies.It will be distinguished between scheduled and unscheduled maintenance. In Section 2.3 the real life constraints with regards to maintenance planning will be presented. Finally, in Section 2.4 recent studies on aircraft maintenance planning are reviewed.

2.1. Aircraft Maintenance Planning

In order to minimize maintenance costs and maximize aircraft availability, it is from high importance for airlines to schedule maintenance in an efficient manner. Thus, optimizing the scheduled checks (A/B/C/D checks) while respecting the safety regulations is of special interest for an airline. Generally, maintenance operations are re required for three principal reasons: operational reasons, value retention reasons and regulatory requirements (Ackert [2010]). Operational reasons mean, that an aircraft needs to function in a reliable condition to generate revenue for an airline. Furthermore, to retain the value of an aircraft it needs to be maintained properly to reduce the physical deterioration. Typically, aircraft, which are maintained accordingly have a higher value for a longer time in comparison to aircraft which do not get maintained appropriately. Finally, maintenance planning is required to meet the regulatory requirements. For each aircraft there are certain requirements for repair and overhauls specified by aviation authorities. These requirements must be met to retain the airworthiness.

In general, an airline has to minimise the costs and maximize the efficiency for keeping the airline fleet in an airworthiness state. Optimizing the utilization involves many stakeholders. These stakeholders include different parties, such as original equipment manufacturer (OEM), operators, airports as well as regulatory bodies. Therefore, maintenance planning is a complex process with various steps and procedures (Papakostas et al. [2010]). Typically, the first step in an aircraft maintenance framework is programme development and implementation. In this step new maintenance frameworks and methodologies are developed and integrated. Afterwards the maintenance is planned and prepared, which is followed by the execution. The final steps are reporting and programme improvement.

In Figure 2.1 different maintenance policies are displayed, such as reactive, proactive and aggressive maintenance. Swanson [2001] defines reactive maintenance by saying, that equipment is allowed to be used until failure. Once a part is broken it is either repaired or replaced. In proactive maintenance strategies, breakdowns are avoided and actions are taken to restore equipment to an appropriate condition. The proactive maintenance approach reduces the probability of unexpected equipment failures. The final strategy, known as aggressive maintenance, aims to improve the functioning of the complete equipment (Swanson [2001]). One example for the aggressive approach is to improve the design of new and existing equipment.

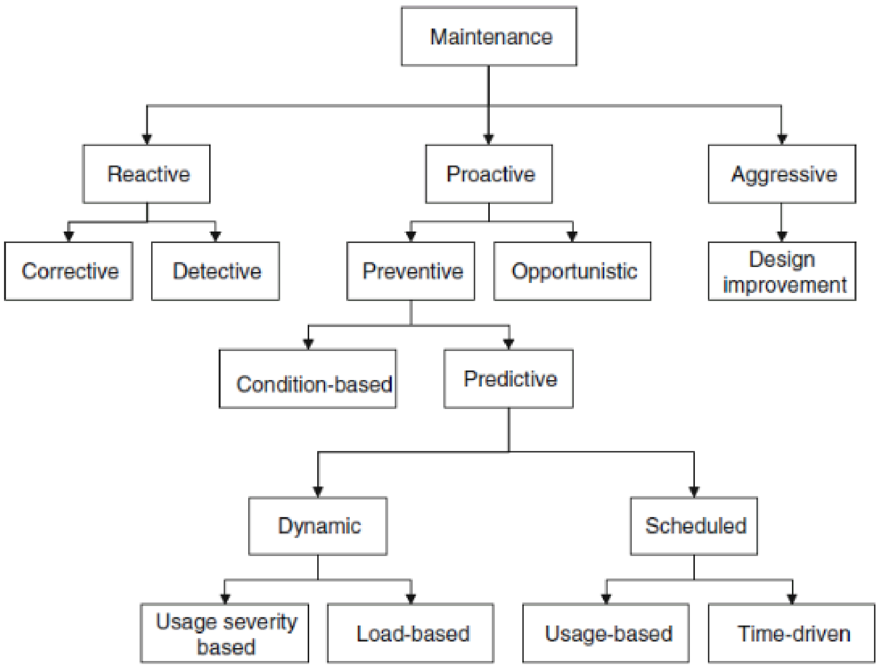


Figure 2.1: Lit. Study: Maintenance policies. Figure taken from Tingo [2013].

2.1.1. Reactive Maintenance (RM)

In the past, aircraft maintenance mainly consisted of reactive maintenance strategies (Ackert [2010]). In reactive maintenance the components were replaced or fixed either when they have failed or when the components were about to fail. There were no scheduled interventions before a failure occurred. The advantage of reactive maintenance is that the component lifetimes are fully utilized. However, the reactive maintenance approach has multiple disadvantages: inefficient grouping of maintenance, no possibility to forecast supportability requirements and an high amount of downtime. Besides these disadvantages, it can also be very dangerous when a component or system unexpectedly breaks down. From an airline's perspective, the reactive maintenance strategy could result in high costs caused by unexpected failures and corresponding delays in flight schedules.

2.1.2. Preventive Maintenance (PM)

Preventive maintenance was developed to prevent unexpected high costs and possible dangerous situations caused by the application of reactive maintenance (Vilarinho et al. [2017]). It is assumed that mechanical parts wear out and show degradation behaviour. The wear out of components cause failures and affects the safety of a system. Identifying different intervals of a hazard function results in three consequences: early failures, random failures and wear out failures. This particular form of the hazard function is described as the 'Bathtub' Curve (Klutke et al. [2003]).

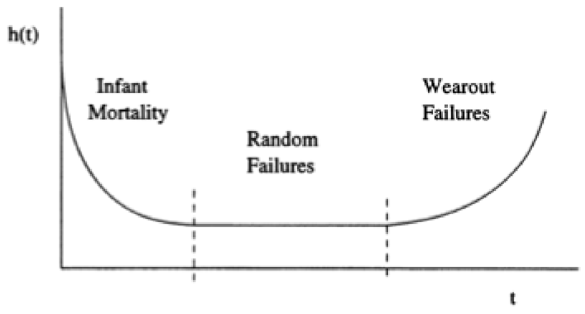


Figure 2.2: Lit. Study: Hazard function over time, classical bathtub curve. Figure taken from Klutke et al. [2003].

The idea, which is presented within the bathtub curve, is to replace the components before they reach the wear-out failure phase and therefore before an actual failure occurs. The hazard function of a component can be based on statistical and reliability analysis. The maintenance schedule, based on this analysis, is created with fixed time intervals. These intervals are specified in parameters such as flight time, flight cycles or calender days. During these inspections, the condition of the component is checked and compared with a given threshold to determine whether the component must be replaced or not. The preventive maintenance framework increased the safety level of an aircraft, as it prevents in-flight failures. However, the preventive maintenance strategy still fails to successfully predict an accurate remaining useful lifetime. Therefore, the preventive maintenance approach is not perfect, as some components are replaced too early, which results in spilling costs. The idea of replacing or repairing components before failure occurs prevents critical systems to fail. Moreover, consequential damage can be reduced when using the preventive maintenance strategy. The major disadvantage of this approach is, that it is difficult to determine the optimal moment of replacement as safety considerations often dictates early replacements.

2.1.3. Predictive Maintenance (PdM)

The preventive maintenance strategy is limited because it only describes a small part of system failures (Klutke et al. [2003]). Therefore, researchers improved the maintenance strategy by combining the condition-based method with a predictive aspect. This new maintenance method is called predictive maintenance and aims to predict the time point of a component's failure. With this method, unexpected failures can be prevented and repairs as well as replacements can be scheduled efficiently. In comparison to the preventive maintenance, predictive maintenance is using the data of the equipment to update the mean time of regular maintenance (Einabadi et al. [2019]). The predictive maintenance strategy uses monitoring information to predict the next failure of the system and perform maintenance before this predicted time. The main challenge of predictive maintenance is to correctly predict the remaining useful life, which is described later in Chapter 3.

2.2. Airline Maintenance Scheduling

This section will elaborate on the airline maintenance scheduling methods. First the scheduled maintenance tasks will be covered. Afterwards the unscheduled maintenance tasks will be described.

2.2.1. Scheduled Maintenance (A,B,C,D Checks)

Aircraft maintenance checks are inspections performed after a certain time or performance threshold. These thresholds can be specified in calender days, flight cycles or flight hours. During the inspections the aircraft systems are, if necessary, restored to an airworthy condition. The regular maintenance checks are divided into A,B,C and D checks. Typically, within an A-check small operations, such as the replacement of filters and lubrication of systems are done, while in the larger checks (C and D) systems with higher structural importance will be checked. The intervals for the maintenance checks are heavily depending on the aircraft type. The inspections are divided into the following checks (Spreen [2019]):

Check	*Threshold*	*Duration [Man Hours]*	*Activities*
A-Check	80-100 FH / 7-9 DY	10-20	Filters etcetera
B-Check	500-600 FH / 60-90 DY	100-300	Lubrication
C-Check	7500 FH / 2 YR	10000 - 30000	Individual systems and components
D-Check	6 YR	30000 - 50000	Airframe and wings

Table 2.1: Lit. Study: Scheduled base maintenance checks. Taken from Spreen [2019].

Many airlines merge the D-check into the C-check, as it has to be grounded for multiple weeks and will not generate revenue for the airline (Deng et al. [2020]). Scheduled maintenance is often a complex procedure, especially for a large non-homogeneous fleet. If the planning of the checks is not done efficiently, the maintenance costs will increase. Often schedules could be improved when daily factors and progresses would be taken into account during the time planning. While creating a maintenance plan, multiple uncertain factors, such as flight planning, have to be considered, which makes the creation of the maintenance plan difficult.

2.2.2. Unscheduled Maintenance

In contrast to scheduled maintenance, which occurs after a specific threshold of flight hours or calendar time, unscheduled maintenance is required in case of malfunction or damage of aircraft components during operation. Unexpected maintenance matters are classified as unscheduled maintenance (Hinsch [2018]). Usually, unscheduled maintenance results in high expenses for airlines as the airline has to perform the reparation at a non-hub airport. Moreover, this can cause delays in the flight schedule. Sometimes, it requires a team of specialists to travel to the stranded aircraft and carry out a local repair (Spreen [2019]).

2.3. Planning Constraints

There are multiple constraints in regard to maintenance scheduling. The constraints can be classified into four different groups (Steiner [2006]). The first group are constraints defined by maintenance actions, such as inter-maintenance flying hours and sequences as well as duration of maintenance actions. The second group is related to manual operator settings, such as fixed special services, quarterly flying hour requirements and max flying hours. Another a very important constraint within this group are available maintenance capabilities. To perform certain maintenance actions, resources such as hangars, personnel and inventory must be available. The third group of constraints is related to ERP data up to the time of scheduling. This relates to constantly changing of aircraft specific information. The last group of constraints are general specifications, which include restrictions due to public holidays.

As described above, many factors influence the maintenance opportunities for an airline. For maintenance scheduling the following are considered to be important within this field of research:

- Flight schedule (determines the inspection moments)
- Components availability (determines if a component can be replaced or repaired)

- Manpower availability (determines if there is enough manpower available to conduct the maintenance)
- Costs of missed / unplanned flights (determines the decision if the components need repair or do nothing)

The above-stated constraints will be combined into maintenance slots. If a maintenance slot is available the components of an aircraft can be repaired or replaced. This means that an assumption will made that if there is a slot available, maintenance can be done for one aircraft.

2.4. Recent Studies on Maintenance Scheduling

There are two types of maintenance, namely line and base maintenance. Line maintenance can be described as 'unexpected maintenance', which is performed within the operating environment. Contrarily, base maintenance requires to remove the aircraft from the service, which results in a longer period of unavailability.

2.4.1. Line Maintenance

Line maintenance can be performed on the apron, while the aircraft remains in the operating environment. The tasks related to line maintenance are relatively easy and contains often Line Replaceable Units (LRU). These are aircraft components which are designed to be replaced quickly at an operating location (Sahay [2012]).

To avoid costs of delay and cancellations, unscheduled maintenance must be planned and carried out in a quick manner. Normally, line maintenance is done during the turn around of an aircraft and involves a routine check, post-flight inspection and malfunction rectification (Papakostas et al. [2010]). During this process, a GO or NOGO decision is made for the next flight. The article by Papakostas et al. [2010] describes an short-term planning methodology of line maintenance activities and supports decision making for maintenance actions. The decisions are made on 'Cost' and 'Operational risk' criterion, but also takes 'Flight delay' into consideration. Furthermore, Sarac et al. [2006] describes a brand-and-price algorithm for an aircraft maintenance routing problem. The algorithm takes maintenance resource availabilities constrains into consideration. The article provides an operational view instead of a long term view, which makes it useful in regard to the present project because of the used constraints. The article by Muchiri and Smit [2011] presents a method, which clusters certain maintenance actions together to reduce costs and increase operational availability.

Moreover, stochastic optimization of maintenance schedules under unexpected failures is researched in the article of Basciftci et al. [2018]. The authors create a framework for integrated condition-based maintenance for a fleet of generators. The framework uses the remaining useful life of a generator to compute maintenance costs and failure probabilities. These factors are then integrated into a stochastic mixed-integer optimization model that determines optimal maintenance and operational decisions.

2.4.2. Base Maintenance

Base maintenance is different from line maintenance as it requires to move the aircraft into a hangar. The aircraft is separated from the operating environment and involves larger tasks, such as A,B,C and D checks. The checks are often scheduled with the operational schedule in mind. A method to come up with an optimized long-term maintenance schedule is proposed by Deng et al. [2020]. The article by Deng et al. [2020] describes a methodology which aims at minimizing the wasted interval between the checks. The methodology takes the following things into consideration: status, maintenance capacity, operational constraints and aircraft type. The article provides a Dynamic Programming method to solve the long term base maintenance problem.

Steiner [2006] provides a heuristic aircraft maintenance scheduling method under various constraints using a linear optimization method and includes constraints, such as maximum flying hours per week. Likewise, Sriram and Haghani [2003] use a heuristic procedure in combination with linear programming problem. The algorithm uses a combination of random search and depth first search. The accuracy of the solution depends on the number of aircraft combinations and explored nodes. In other research, Senturk and Ozkol [2018] are examining how aircraft utility can be maximized and maintenance cost minimized. Senturk and Ozkol [2018] propose a method which is called 'Single-task oriented maintenance' in which they aim to minimize the time spend on the ground.

Maintenance scheduling can also be done by Monte-Carlo Tree search. In the article of Shang et al. [2020], the researchers describe how Monte Carlo tree search can be used to schedule maintenance of a distributed network. Monte-Carlo tree search is a reinforcement learning approach, which is able to provide a solution in combination with a superior performance over other approaches. The model is developed to minimize the maintenance costs considering real life constraints and reliability indices.

In addition, a review on maintenance optimization is provided by de Jonge and Scarf [2020]. The article provides a concise overview of the research done in the topic of maintenance activities. There is a distinguish between single-unit and multi-unit systems, followed by a separation based on deterioration processes. de Jonge and Scarf [2020] describe the research done about continuous deterioration state space with required inspections to obtain condition information. This article might be relevant for the project as it will consider a continuous degradation but with fixed inspection intervals. Furthermore, the article provides a section about multi-unit systems, which has various types of dependencies: economic dependence, structural dependence and stochastic dependence. The following definitions are retrieved from Laggoune et al. [2010]:

- Economic dependence: Concerns the influence of component operation and maintenance actions on the overall system costs. For example, the overall system costs can be lower if multiple components are maintained at the same time as less maintenance or downtime costs are induced (Laggoune et al. [2010]).
- Structural dependence: Concerns components which structurally form a part. Therefore maintenance on failed components implies actions on other components. For example, if a component in an engine must be replaced the whole engine must be disassembled (Laggoune et al. [2010]).
- Stochastic dependence: Concerns when the state of a component influences the lifetime distributions of other components or when components are subjected to common-cause failures. For example, when a degraded component leads to an internal force and causes overload on other components (Laggoune et al. [2010]).

Within Chapter 4 further existing literature on maintenance policies for multiple component systems is reviewed.

3
Aircraft Component Prognostics

Recently, the field of component prognostics and predictive maintenance receives increasing attention from researchers. Companies have a big interest in optimizing their maintenance planning and repairs in order to reduce unexpected high costs. This chapter will provide an overview of the relevant literature regarding the modelling of failure prognostics of aircraft components. First, in Section 3.1, an overview of different prognostics forms is given. The following different types of prognostics will be described: Model-Based, Data-Driven or the Hybrid approach. Furthermore, within Section 3.2, an overview of multiple component prognostics is provided. Finally, in Section 3.3, the most recent studies on prognostics are provided.

3.1. Prognostics

Prognostics is known as an technical field which focuses on predicting the time at which a system or component will no longer perform its intended function (Vachtsevanos et al. [2006]). This can be in the form that the system is not longer able to meet its original or desired performance. An example could be a bearing failure within an engine, which impedes the aircraft from flying. The engine failure can result in high costs for the airline because the incident can cause delay. Moreover, in case of a non-functioning aircraft, the airline has to provide a replacement, which can cause additional costs. To avoid unexpected failures, prognostics are used to determine the remaining useful life (RUL). RUL gives a prediction of time, which a component can be expected to operate within its stated specifications (Sri et al. [2019]). Hence, the RUL can be used to decide whether a component needs service or replacement. The component's predicted RUL is very useful, if estimated correctly, because it can be used to prevent the whole system from failing. However, to be able to predict various modes of failures, useful condition indicators must be identified.

A field that makes use of RUL prognostics, which could be used for preventive maintenance, is known as Prognostics and Health Management (PHM; Tsui et al. [2015]). There are several methods to create RUL prognostics, namely Model-Based, Data-Driven and a Hybrid approach (Zhao et al. [2017]).

3.1.1. Model-Based Prognostics

The first method to predict failure probabilities and RUL is to use a Model-Based approach. This is done by representing the system's underlying physics in an appropriate model. Likewise, Rezvanizaniani et al. [2014] created a computer simulated model of a battery from an electric vehicle to identify battery health issues. A major advantage of Model-Based prognostics is that accurate and precise data can be acquired, if the models are developed correct (Daigle and Goebel [2011]). A disadvantage of Model-Based prognostics is that complex systems are almost impossible to represent in a simulation model. Usually, the underlying physics are too complex to reproduce. Also from a practical point of view, the theoretical model are not always able to adapt to various operational variables, such as temperature or loads over time. Usage of the unsuitable models can lead to wrong maintenance decisions and will result in a sub optimal solution.

The Model-Based prognostics are based on stochastically modelling the system degradation evolution. The degradation evolution can be based on Markov processes or variants for which the transition probabilities and system states are known. These transition probabilities and system states can be obtained from historical reliability data (Papakonstantinou and Shinozuka [2014]). Other forms of Model-Based prognostics can be based on the assumption that a stochastic process characterizes the degradation mechanism of a system (You et al. [2010]). The continuous degradation of an aircraft brake can be modelled by using a Gamma Process (Lee and Mitici [2020]).

3.1.2. Data-Driven Prognostics

In contrast to Model-Based prognostics, Data-Driven prognostics are using historical data to predict the RUL without using the nature of the degradation mechanism (Si et al. [2011]). Hence, Data-Driven models can be used for more complex systems. The Data-Driven models are based on sensor information of the system's state. Therefore, the quality of the prognostics is heavily depending on the data processing and used techniques. In the past, the majority of the processes were done manually. Nowadays the manual processes are replaced with statistical or Machine learning (ML) techniques.

Statistical Methods

The RUL of a component or asset is a random variable (X_t), depending on the operation environment, current age and the health information (Si et al. [2011]). Within this article, the RUL (X_t) is described as a probability density function (PDF), which is depending on the history of operational profiles (Y_t):

$$f(X_t \mid Y_t) = f(X_t) = \frac{f(t+X_t)}{R(t)} \tag{3.1}$$

Where $f(t+X_t)$ is the PDF of the life at $t+X_t$ and $R(t)$ is the survival function at t. The novelty of using statistical methods in order to determine the RUL is to estimate $f(t+X_t)$ based on historical data. The data used to determine the RUL of a component can be divided into two main categories: event data and condition monitoring (CM) data (Jardine et al. [2006]). Event data can be defined as recorded failure data, while CM data is 'any' data about the system. CM data includes data which might be relevant for the estimation of RUL, such as environmental, operational and performance data. The idea is to link the CM data to the RUL, such that the CM data can be categorized as direct CM and indirect CM data. The direct CM data, such as wear and cracks, can be directly linked to the prediction of RUL. Indirect data, as the name indicates, can only be used to indirectly refer to the system's condition. Thus, the CM data can be linked directly or indirectly to the recorded failure data from the past to detect anomalies. The anomalies can be used to compute realistic confidence intervals, which compares the measured data to historical data. In statistical methods it is essential to compute a realistic PDF because if the PDF is inappropriate the comparison with new measured data could produce wrong estimations and consequently evoke false alarms. Figure 3.1 represents an overview of the different statistical driven approaches for Direct CM and Indirect CM data.

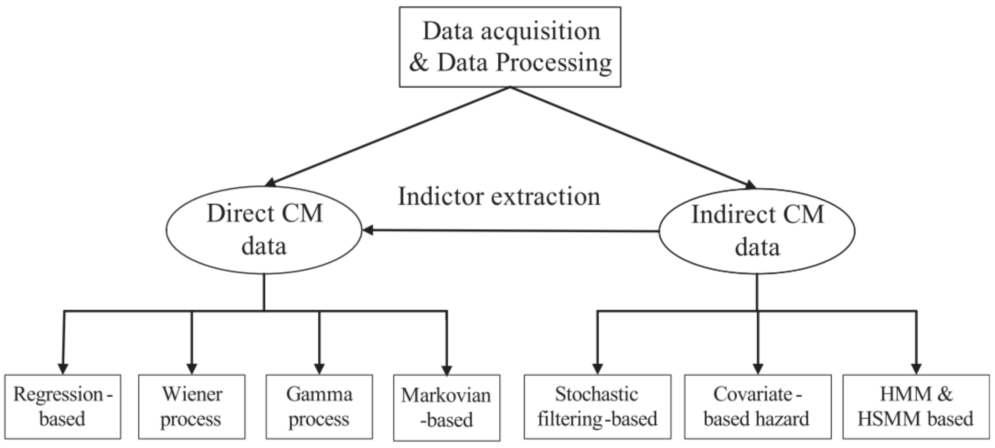


Figure 3.1: Lit. Study: Statistical data driven approaches for RUL estimation. Figure taken from Si et al. [2011].

Machine Learning

In the past decade, many studies have examined machine learning (ML) techniques to improve the performance of RUL prognostics. The supposed function of ML is that these methods can automatically extract and construct useful information. Machine learning methods are more effective than statistical methods in processing raw data and extracting useful information. ML techniques are trained on a part of the whole data set and identifies important features which might not be predefined. For example, Lin et al. [2018], provide a integrated hierarchical learning framework based on both diagnostics and prognostics.

ML is a very broad field of topics. Within the present article only the relevant parts regarding the research aim will be discussed. In the articles of Zhang et al. [2018] and Wu and Castro [2020], a Long Short-Term Memory (LSTM) network is used to predict the RUL of a system. LSTM networks are useful to learn both long and short-term dependencies (da Costa et al. [2020]). The usage of LSTM networks occurs frequently in recent published articles and shows a promising performance in the field of RUL prognostics. However, many of the articles solely focus on the prognostics, while ignoring the corresponding maintenance decisions. Therefore, it would be interesting to investigate the computed prognostics in light of the corresponding maintenance decisions (Nguyen and Medjaher [2019]).

3.1.3. Hybrid Prognostics

Another form of prognostics is called 'Hybrid Prognostics'. Hybrid Prognostics combine Model-Based and Data-Driven prognostics. The combining approach is making use of the advantages of model- and data-driven techniques to improve the prediction performance (Assaf [2004]). In their article, Celaya et al. [2014], a combination of Data-Driven and Model-Based methodology is used to make prognostics for power devices under thermal stress.

3.2. Prognostics for Multiple Components

The previously discussed techniques are demonstrated on single components. As described in subsection 2.4.2 between several system components there can be multiple kinds of dependencies: economic, structural and stochastic dependence. These dependencies can make it complex to predict the RUL if there is no prior knowledge about the influence from one component on another (Hafsa et al. [2016]). The prediction of RUL for multi-components systems must be handled differently with respect to single component systems, because a full understanding of the interactions is needed. Hafsa et al. [2016], describes a stochastic dependency model for degradation rate interactions and a prognostics method, which calculates the RUL considering the degradation rate interaction. The prognostics for multi-components within this article is done by computing a Weibull probability density function for each component and then transform the density function into a cumulative distribution function (CDF) for the whole system. The CDF is then related to system degradation states and used for computing the RUL. In this article by Arts and Basten [2018], the authors use different Weibull distributions for different components. Afterwards a periodic usage based maintenance (PUBM) policy or periodic condition based maintenance (PCBM) is applied on each component. Combining the policies and distributions, a non-linear non-convex programming problem is solved by using a greedy search algorithm. The important parameters in the optimization problem are the costs of the policies, down-times and components.

Assaf [2004] uses a generic degradation formula which incorporates the effect of other components. The effect of component i on component j is defined in a matrix, where the values are computed by historical experience. Consequently, the degradation of the system is modelled as a normal degradation function.

Furthermore, a review about condition-based maintenance policies for multiple dependent components was published by Olde Keizer et al. [2017]. The author provides a comprehensive overview for maintenance policies for multiple components. The article is divided by different dependencies: Economic, Structural, Stochastic and Resource dependencies. The costs related to maintenance can be divided into negative and positive economic dependence. Negative economic dependencies cause more costs when maintaining components individually, while positive dependencies cause lower costs when maintaining multiple components. Most articles discussed in Olde Keizer et al. [2017] use a preventive replacement strategy based on a certain threshold in combination with an optimized inspection interval. Structural dependency can be split into technical or performance dependence, where the latter is divided into series or parallel systems. Most of the reviewed articles are describing a performance (series) dependency and are using a threshold for preventive maintenance. Stochastic dependence means that the degradation of a component partially depends on another component. Examples for stochastic dependencies are: load sharing, failure-induced damage and common-mode deterioration. Many researchers use a threshold for preventive replacement as a CBM policy structure. Lastly, a novel classification is Resource dependency. The condition based maintenance model is optimized regarding the following restrictions in mind: maintenance workers, tools, spares, transport or budget restrictions. For instance, maintenance worker restrictions are introduced to cope with a certain amount of resources available.

3.3. Recent Studies on Prognostics

As described in subsection 3.1.2, much research has been done using Long Short-term Networks for prognostics. Other studies on prognostics use Convolutional Neural networks, Recurrent Neural Networks, Hidden Markov Models or alternative approaches (Sri et al. [2019]).

4 Integrating Prognostics in Planning

The previous chapters elaborated on aircraft maintenance planning and prognostics separately. Within the present chapter methods to integrate prognostics in maintenance planning will be discussed. Much research is done either investigating prognostics or maintenance scheduling independently from one another. Contrary, this chapter covers the optimization of maintenance scheduling of the complete framework from data-driven prognostics to maintenance decisions. Section 4.1 provides a general overview of the usage of prognostics in maintenance planning. Subsequently, Section 4.2 elaborates on State-of-the-art techniques in this field.

4.1. Prognostics in Maintenance Planning

The maintenance concept has advanced significantly over the last time, it uses historical data, models, simulations and failure probabilities to predict fault and system deterioration for the remaining useful life (Sakib and Wuest [2018]). This method is called predictive maintenance or condition based maintenance and is using this information to compute maintenance schedules accordingly. Scheduling has been incorporated in the policies by various researchers. A scheduling policy for predictive maintenance based on the least flexible job (LFJ) and the longest processing time (LPT) has been reported in Paprocka and Skołud [2017]. They have used a Hybrid Multi-Objective Immune Algorithm (HMOIA), supported by minimal impact of the disrupted operation on the schedule (MIDOS). The advantage of this method is that the predicted time of failure is taken into account for maintenance work. The suggested method reduces the frequency of unpredicted downtime due to system failure. Paprocka and Skołud [2017] uses three stages in their research to come up with the final schedule: generating the basic schedule, followed by predictive scheduling and lastly reactive scheduling.

Furthermore, Zhu et al. [2017], presents a periodic maintenance policy for a single component which is part of a complex system. The component follows a stochastic degradation process (Gamma Process), which is monitored continuously and assumes a Random Coefficient Model (RCM). The authors use both scheduled and unscheduled downs for the system. The scheduled downs follow a fixed interval, while the unscheduled downs follow a Poisson process. The component is replaced during a scheduled or unscheduled down when it reaches the control limit. The authors extend the model into a multi-component system in which 20 components are modelled by a random coefficient model. Subsequently, the total cost is computed and optimized via an iterative procedure.

Another article, described in Liu et al. [2019], presents an integrated decision model that coordinates predictive maintenance decisions based on prognostics information, It considers a single-system and optimizes to minimize the total expected cost. Research about a dynamic predictive maintenance policy for multi-components which minimizes the long-term mean maintenance cost per unit time is described by Van Horenbeek and Pintelon [2013]. It uses a Gamma process for the degradation with a certain threshold for failure. Afterwards the method groups maintenance actions according to the remaining useful life and corresponding costs. It decomposes the multi-component maintenance problem into n single-component models considering an age-replacement policy and optimizes for long-term mean maintenance costs.

The research performed by Poppe et al. [2018], is about a condition-based maintenance policy for continuously monitored components with two degradation thresholds. When the degradation level of the monitored component surpasses a first 'opportunistic' threshold, it will be serviced together with the other components. This happens for instance during a planned intervention, breakdown or when another component requires maintenance. Another threshold is implemented in order to prevent a failure, such that both thresholds can be optimized to minimise the total expected maintenance costs or downtime of the system.

4.2. Research Studies on Integrating Prognostics into Maintenance Scheduling

In the following section articles, which are from special importance in this research field will be presented. Parts of the information discussed in subsection 4.2.1 may be used in further research.

4.2.1. A new dynamic predictive maintenance framework using deep learning for failure prognostics

The article by Nguyen and Medjaher [2019] focuses on a dynamic predictive maintenance framework, which considers a complete process from data-driven prognostics to maintenance decisions. The title of this article is "A new dynamic predictive maintenance framework using deep learning for failure prognostics" (Nguyen and Medjaher [2019]). An overview of the dynamic predictive maintenance process is shown in Figure 4.1. The framework is applicable on various complex systems and allows fast decisions for multiple options by rapidly evaluating their costs.

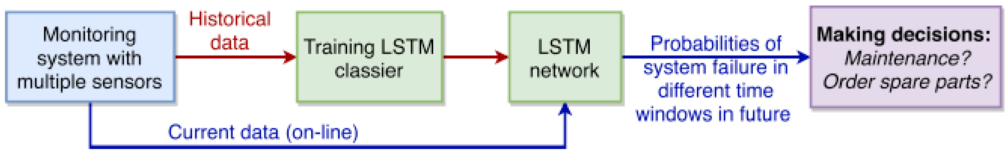

Monitoring system with multiple sensors
Historical data
Training LSTM classier
LSTM network
Probabilities of system failure in different time windows in future
Making decisions: Maintenance? Order spare parts?
Current data (on-line)

Figure 4.1: Lit. Study: Dynamic predictive maintenance process. Figure taken from Nguyen and Medjaher [2019].

As described in subsection 3.1.2, Long Short-Term Memory (LSTM) networks are used to estimate the remaining useful life of a system. Nguyen and Medjaher [2019] also make use of the LSTM network, but one which indicates the probabilities, that a system will fail into different time intervals instead of predicting a single RUL value. The adaption of the LSTM network by Nguyen and Medjaher [2019] has two advantages, namely their approach does not require a Piece-wise linear (PWL) RUL target function compared to previous research. The value of a single RUL estimation depends solely on the prediction horizon and therefore using this value at the first stage of the system can lead to a wrong decision. The second advantage is that failure intervals adapt better to practical demands because they can be specified by the operation planner.

As first the failure probabilities for different time windows are estimated, the process is shown in Figure 4.1. After the probabilities are determined, the maintenance decisions rules are made.

The prognostics, as described before, are computed with use of LSTM network. These networks are good in learning over long time sequences and retaining memory. Therefore, applying LSTM for prognostics allows to look back to the system degradation and uses this information for the prediction (Nguyen and Medjaher [2019]). To train the model, each component has it's own sensor measurements and own true RUL. The LSTM classifier will take the sensor measurements and attempts to output the corresponding RUL window probability. The data which is used for the LSTM network has to be preprocessed first:

- Normalisation for heterogeneous data for training LSTM classifier for each feature (mean & variance). Every feature has the same range, between zero and one.
- Data labeling to perform classification. The first class is that the system residual time is larger than the time window: Deg 0 = $RUL > w_1$. The second class describes the RUL laying in the estimated period: Deg 1 = $w_0 \leq RUL < w_1$. The third class is that the RUL will not exceed the time period: Deg 2 = $RUL \leq w_0$.
- Formalisation, create different features from sensor output. Each sensor output is one feature.

After the data is processed, the LSTM network acts as classifier, it seeks the connection between the inputs (sensor data) and outputs (true RUL values). It also uses a 'Dropout' function to prevent over fitting by probabilistically removing inputs to the system. The output then is a probability distribution over the three classes: Deg 0, Deg 1 or Deg 2. The objective function is called caterogical crossentropy which is specially used to solve the multiple mutually-exclusive class problem. This function returns the cross-entropy $H(p,q)$ between a predicted probability distribution ($p(x)$) and a true probability distribution ($q(x)$): $H(p,q) = -\sum_x q(x)\log(p(x))$. Then, ADAM is used as the optimization algorithm, it is an extended version of the stochastic gradient descend.

A probability confusion matrix is used to evaluate the performance of the LSTM classifier. It shows the true labels and predicted labels, where the values in the matrix corresponds to the mean probability that the predicted value is correct. An example is shown in Figure 4.2.

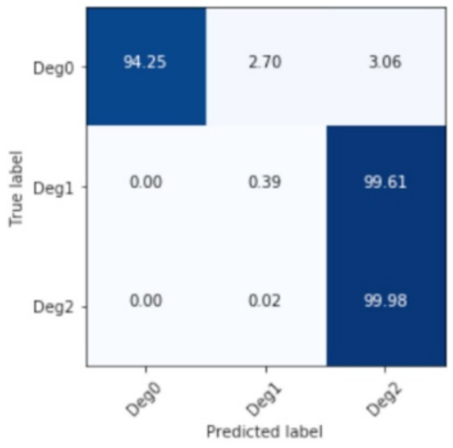
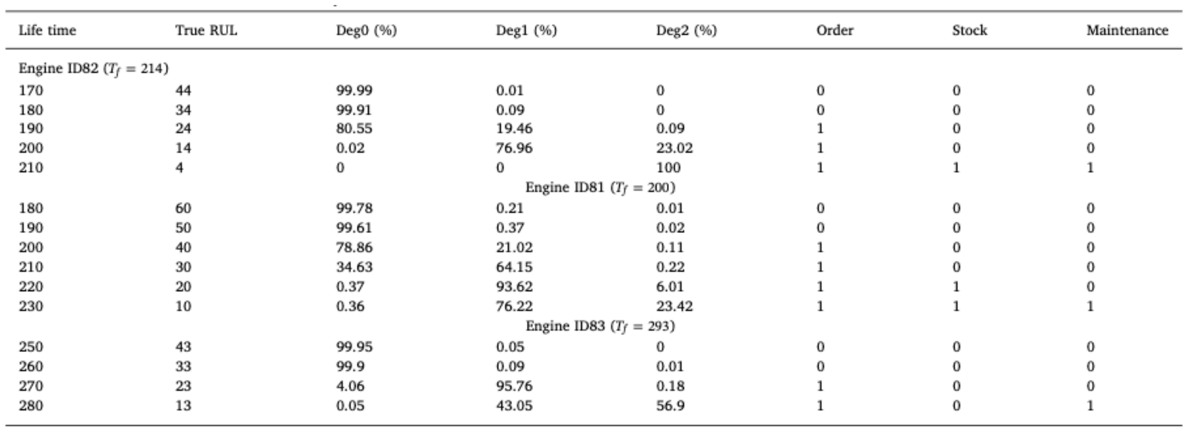


Figure 4.2: Lit. Study: Probability confusion matrix (%) on a test set. Figure taken from Nguyen and Medjaher [2019].

The decision rules based on the prognostic information are executed at the beginning of each inspection period. The inspection periods are equally distributed and have the same interval. Additionally the spare part inventory is included within the article ofNguyen and Medjaher [2019] by minimizing the holding inventory cost. At every inspection period, the failure probabilities are updated according to the monitored data. The following decisions are made each inspection, based on the prognostics information:

1. To order (O) or no-order the (N) spare part. This decision is based comparing the cost of the two options. NO option is specified as: $NO = C_{os} \cdot P(w_1 < RUL \leq w_1 + \Delta T)$. While the O option is linked to the estimated holding cost and specified as: $O = P(RUL > w_1) \cdot C_i \left[\frac{\max(T_F - h\Delta T - w_1, \quad \Delta T)}{\Delta T} \right]^+ \Delta T$.
2. The second decision made at an inspection is to replace (R) or do-nothing (DN). This decision is based on the cost-rate of each option, where the lower cost rate will be chosen. The R cost rate is given by: $R = \frac{C_p + C_{os} \cdot \delta(S_h = 0)}{h\Delta T}$, which is based on the cost of preventive maintenance and the cost of an spare part being out of stock. On the other hand, the DN cost rate is specified as: $DN = \frac{P(RUL \leq w_0) \cdot (C_c + C_i \cdot \delta(S_h = 1)\Delta T + C_{os} \cdot \delta(S_{h+1} = 0))}{(h+1)\Delta T}$. This cost rate is based on the probability the component will fail in the next period in combination with corrective maintenance (including downtime cost).

At each inspection the probabilities the system will fail (Deg0,Deg1,Deg2) are updated and a decision is made to order, not order, repair or do nothing. To evaluate the performance of the framework, two different maintenance policies are introduced to be compared with. The first one is called Period Maintenance Policy, in which the mean time to failure is used to determine the predictive maintenance cost. The second policy is the Ideal Predicted Maintenance Policy and assumes that the residual life time is correctly determined. A real application case study has been performed in the article, based on the data set: Turbofan Engine Degradation Simulation provided by NASA Ames Prognostics Data Repository. The data set contains multiple sub data sets with different conditions, fault numbers and sensor measurements. As described previously, the dynamic predictive framework uses RUL intervals instead of RUL values which makes it impossible to use general evaluation criteria such as MAE, MAPS and MSE. Therefore, the authors use confusion probability matrices to evaluate the accuracy of the prognostics information.



Life time	True RUL	Deg0 (%)	Deg1 (%)	Deg2 (%)	Order	Stock	Maintenance
Engine ID82 (T_f = 214)							
170	44	99.99	0.01	0	0	0	0
180	34	99.91	0.09	0	0	0	0
190	24	80.55	19.46	0.09	1	0	0
200	14	0.02	76.96	23.02	1	0	0
210	4	0	0	100	1	1	1
			Engine ID81 (T_f = 200)				
180	60	99.78	0.21	0.01	0	0	0
190	50	99.61	0.37	0.02	0	0	0
200	40	78.86	21.02	0.11	1	0	0
210	30	34.63	64.15	0.22	1	0	0
220	20	0.37	93.62	6.01	1	1	0
230	10	0.36	76.22	23.42	1	1	1
			Engine ID83 (T_f = 293)				
250	43	99.95	0.05	0	0	0	0
260	33	99.9	0.09	0.01	0	0	0
270	23	4.06	95.76	0.18	1	0	0
280	13	0.05	43.05	56.9	1	0	1

Figure 4.3: Lit. Study: Dynamic framework. Figure taken from Nguyen and Medjaher [2019].

Figure 4.3 shows the life cycle of three different engines and the framework in action. It shows the life time at each inspection and corresponding True RUL. Then the probabilities of the three classes are visible in the next three columns. The parts in order and in stock are given in column 6 and 7. The last column shows the maintenance actions performed. The advantage of the framework is that the decisions are simply and quick made on the prognostics information. In comparison with the other two policies, this framework performs almost as good as the Ideal Predicted Maintenance Policy and outperforms the Period Maintenance Policy.

This article is relevant for the research project as it provides a complete framework, where the prognostics of the components are used for maintenance decisions.

4.2.2. Modelling and application of condition-based maintenance for a two-component system with stochastic and economic dependencies

The following article, Do et al. [2019], describes a condition-based maintenance policy for a two-component system with both stochastic and economic dependencies. The state dependence modelling is done by $\Delta X_t^i = \Delta X_t^{ii} + X_t^{ji}$. The first element is the intrinsic effect of itself. The second term is the interaction effect between the two components. The two components are stochastically dependent and the degradation level of component i depends on its own state and the state of the other component. In an example given within the article, the random intrinsic effect is following a Gamma probability density function and the interaction effect follows a non-linear function: $\Delta X_i^{ji} = \mu^j \cdot \left(X_i^j\right)^{\sigma'}$. Another example is given in which the intrinsic effect is described by a Brownian motion process.

Economic dependence modelling is done by considering that replacements may be corrective(that is on failure of the system) or preventive (prior to system failure). All necessary maintenance resources (spare parts, maintenance tools, repairman) are assumed to be available at planned inspection time. The individual maintenance costs are modelled by the standard costs (spares, labour, set-up) and downtime cost $C_c^i = c_c^i + c_d \cdot d_i$. Also cost saving is implemented in the way that replacing two components at the same time, total maintenance cost can be reduced. Joint replacement is modelled as: $CS_{-,-} = a \cdot \left(c_-^1 + c_-^2\right) + b \cdot (d_1 + d_2) \cdot c_d$.

For the Maintenance policy is assumed that the system downtime due to failure is known and the two components of the system are inspected at regular time intervals with inter-inspection intervalT, a decision variable which has to be optimized. According to the maintenance policy, a component will be replaced if it reaches the failure threshold between Tk1and Tk. Or if the component is still functioning, but has to be replaced according to the rules for individual preventive replacement or opportunistic replacement. The main idea of the proposed opportunistic replacement model is to capitalize on both the economic dependence and the stochastic dependence. Three policies are defined: V = Opportunistic replacement policy, V1 = Classical condition-based maintenance policy, V2 = Joint replacement policy. The optimization of the maintenance policy is done by a cost model, based on the long-run expected cost per unit of time (cost-rate) including re-placement and inspection costs. The article of Do et al. [2019] refers to a method based on semi-regenerative theory and Markov decision processes are introduced to obtain a closed-form expression for the cost-rate of a two-component system with independent degradation behaviour. However, this cannot be applied when there is interaction between components as the degradation process are dependent and non-longer time-homogeneous. Therefore, the cost rate is evaluated by the use of a Monte Carlo Simulation. The values of the decision variables are varied and the minimum cost-rate can be identified.

5 Research Proposal

To analyse the conditions of aircraft components the usage of sensor data became more important in the field of aircraft maintenance during the last decades (Assaf [2004]). The prognostics of the components can be used to improve the quality of maintenance scheduling, in terms of time and cost efficiency. To optimize the scheduling of maintenance, failure prognostics of components are used to determine the remaining useful life of a component. The failure prognostics of components can be used for the planning of maintenance tasks. The aim of this research is to develop a framework for maintenance schedules, that take both the component prognostics and constraints into consideration. The constraints, which have an influence on maintenance schedules are for example the availability of resources and available maintenance slots. The main challenges to address are the modelling of constraints related to the component prognostics as well as the integration of this model into maintenance planning models.

The problem can be divided into two parts:

1. The modelling of the degradation of aircraft components and the corresponding remaining useful life time windows. Currently, this is done by a model based or data-driven approach. For instance, random failures are represented by an Exponential probability distribution, while deteriorating failures are represented by a Weibull or Normal probability distribution.
2. The implementation of prognostics models for multiple aircraft and multiple identical parts into one maintenance planning framework. This can be done for instance by a Monte Carlo Tree search, as probability measures are incorporated. The goal of this research is to implement the degradation models for components into one maintenance planning framework. A maintenance planning framework needs to take the following parameters into consideration: component prognostics, maintenance resources, maintenance slots and maintenance costs.

The aim of the research is to create a stochastic optimisation maintenance planning model for aircraft components with failure prognostics and maintenance resources availability constraints. Another objective of the research is to formulate implementation requirements for the above-mentioned model in practice.

5.1. Research Objective

The problem formulated in the previous paragraph can be translated into a specific research objective. The research objective for this project is as follows:

To achieve a validated stochastic optimisation maintenance planning model for multiple identical aircraft components with a specific remaining useful time window for component failure and maintenance resources availability constrains by means of a Monte Carlo Tree Search algorithm combined with Reinforcement Learning.

5.2. Research Questions

To fully meet the goal of this research project, the above-stated objective is divided into multiple research questions. The main research questions which will be answered in this project is:

How can prognostics be integrated into aircraft maintenance planning for multiple identical components to obtain a complete dynamic framework?

The following sub-questions will be answered to obtain a thorough answer of the overall research question:

1. How is aircraft maintenance scheduled?
   (a) Which forms of aircraft maintenance are currently used? Such as: Reactive, Preventive or Predictive Maintenance
   (b) How are these maintenance methods scheduled?
   (c) What are real life constraints for maintenance planning?
   (d) How is opportunistic maintenance scheduled? What are the objective and cost functions used in these models?
   (e) How to model realistic slot data and determine available maintenance slots?
2. How are aircraft component prognostics modelled?
   (a) How are prognostics modelled for multiple components which are identical or dependent?
   (b) Which models can deal with real life constrains and are suitable for multiple components?
3. How can prognostics be implemented into a planning framework?
4. Which measures are suitable to evaluate the model in terms of accuracy?

5.2.1. Hypotheses

The research questions form a framework for the project and will help to answer the main question. The theoretical research questions will be answered in this literature review, while the rest of the research questions will be answered in the final thesis, including results and conclusions. In order to formulate concrete answers for the research questions, quantifiable hypotheses are formulated in this section. The verification and validation of the model is done by an analysis with different forms of maintenance: perfect maintenance, block-based maintenance or age-based maintenance.

5.3. Research Scope and Assumptions

The research scope is important to specify in order to define the boundaries of this project. The number of aircraft will vary between 1 and 10, with increments of 2. Also the number of identical parts will vary between 1, 2 and 3. The identical components might have the same but also different remaining useful life time windows because the distribution will be randomized. Furthermore, the maintenance slots will be specified on the basis of a realistic example. The minimum time for the next schedule opportunity is one day, which means, that the maintenance can be scheduled for the next day. To compare and evaluate the proposed maintenance framework, the performance will be compared with a periodic maintenance policy and an ideal predicted maintenance policy.

5.4. Methodology

This section contains an illustrative example of the problem which is aimed to be solved, shown in Figure 5.1. The figure illustrates the problem for one aircraft with two identical components but with different remaining life time window probabilities. The probability distribution is defined into windows instead of a single value. The boxes for each component provides the probabilities that the component fails in a certain time window. Using a time window instead of a single value has a practical importance and suits real life applications better because the intervals can be defined according to the requirements of an operation planner.

In addition, real life constraints such as spare parts inventory, maintenance slots and manpower availability are shown in Figure 5.1. In the rolling horizon principle every inspection includes a decision to either repair or do nothing depending on the remaining useful life probabilities. An extensive dynamic maintenance framework was discussed and explained throughout Chapter 4.

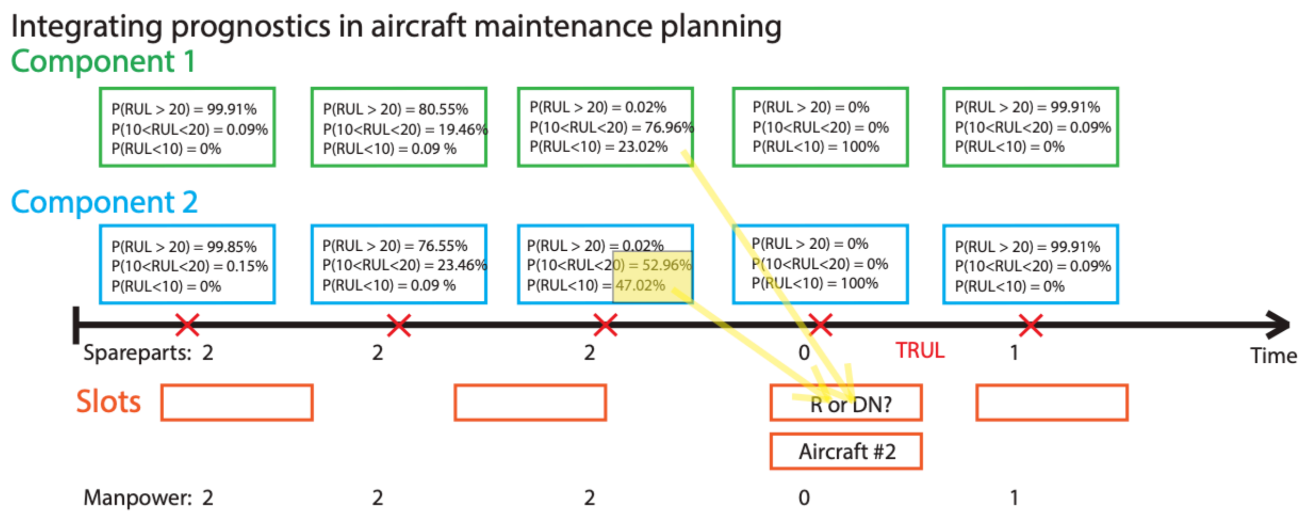


Figure 5.1: Lit. Study: Illustrative example predictive maintenance framework

6 Conclusion

This literature review illustrated the importance of making use of prognostics into maintenance planning. The first component of the Prognostic Health and Management (PHM) framework is computing prognostics for components or systems. The second part of the framework focuses on the maintenance optimization. Within this field of research, Predictive Maintenance (PdM) is receiving increasing importance as it can result in lower operational costs. PdM focuses on using sensor data to provide reliable prognostics for the remaining useful life (RUL) of an individual component or complete system. These prognostics can be used to improve the scheduling of maintenance activities, which eventually results in lower costs and higher utilization.

The aim of the research project will be to integrate these prognostics into maintenance planning. As there is still a lack of research which investigates the complete framework, this project will be of high relevance within the research field. An elaboration of the research sub-questions can be found in Chapter 5. The main research question which will be answered in the research project is:

How can prognostics be integrated into aircraft maintenance planning for multiple identical components to obtain a complete dynamic framework?

The goal of the present research is to create a stochastic optimisation maintenance planning model for multiple identical aircraft components which includes failure prognostics and resource availability constraints. The methodology for the component prognostics is to use a probability distribution for specific time intervals instead of a single RUL number. This is out of practical importance and suits real life applications better due to the fact it can be used by an operation planner. Afterwards, these prognostics will be used to make the maintenance decision between Repairing or Not Repairing, dependent on the evaluation of costs and availability of maintenance slots.

Within the literature review of this research field, different existing aircraft maintenance methodologies and recent studies are described. Generally, maintenance planning is divided into scheduled maintenance and unscheduled maintenance. Scheduled maintenance consists of maintenance checks which have to be performed after a certain threshold, such as flight hours, flight cycles or calendar days. Unscheduled maintenance is required in case of a malfunction during operation and has to be performed on the apron. Maintenance scheduling is subject to important constraints such as flight schedules, components availability, manpower availability and the costs of missed as well as delayed flights. These constrains have a significant influence on the availability of maintenance opportunities.

The prognostics of the remaining useful life of aircraft components can be computed by three different categories: model-based, data-driven and the hybrid approach. Model-based prognostics use a model which represents the underlying physics of a system or a individual component. Data-driven prognostics use historical data to predict the RUL which can then be split into statistical or machine learning methods. Hybrid prognostics combine the two previously described methods. The major challenge of computing prognostics is to estimate the RUL correctly, which is especially difficult for multiple dependent components that interact with each other. In order to obtain an useful framework for airlines, maintenance planning and component prognostics should be combined.

7 Research Planning

This chapter provides the research planning according to the general planning of the Faculty of Aerospace Engineering. The department of Air Transport Operations predefined the important deliverables and corresponding time windows. The enumerated list below shows the key review points, milestones and deliverables. The deadline of the first draft of the Literature review report is on the 24th of July 2020. Afterwards, a break is planned until the 31st of August 2020. After the summer break an internship of six months is scheduled from the 7th of September 2020 until the 5th of March 2021. Therefore, the Kick-off meeting will be planned at 22nd of March 2021. The starting dates are indicated in blue. The following topics are incorporated into the Gantt Chart:

1. Kick-off meeting (`22/03/2021`)
    - Report on the literature study
    - Problem formulation
    - Methodology
    - Project planning
2. Initial phase (3.5 Months) (`16/03/2021-02/07/2021`)
    - Initiate research
    - Model development
    - Data analysis
3. Mid-term meeting (`05/07/2021`)
    - Assess progress and project management
    - Prepare presentation
    - Concise report
    - Model, results and analysis
4. Final phase (2.5 Months) (`06/07/2021-15/11/2021, including vacation`)
    - Incorporate mid-term feedback
    - Develop case studies
    - Validation and verification
    - Report on research
    - Hand in draft thesis

5. Green light meeting (18/11/2021)
    - Present final results
    - Hand in final thesis (01/12/2021)
    - Establish exam date and committee

III

Supporting Work

1 Predictive Maintenance Framework

This chapter contains additional information about the predictive maintenance framework. First, additional information is provided on Reinforcement Learning and Deep Q-Learning. Second, there is a section on reward analysis where the overall reward and individual episode reward are discussed for Scenario 1. Followed by information about the agent actions and illustration of convergence for Scenario 2. Finally, the results for the t-test for Scenario 2 are provided.

1.1. Reinforcement Learning Elaboration

Recently, Reinforcement Learning (RL) has grown in popularity and has shown promising results in the field of predictive and condition-based maintenance [15]. The idea behind RL is to create an agent that receives a reward after performing an action. The agent learns how to act in a specific way based on an environmental state. The goal is to maximize the total reward based on the policy between a state and an action. Therefore, the probability of taking specific actions will increase or decrease based on previous experiences. The long-term goal in RL is to maximize the cumulative reward based on a sequence of actions. An overview of the interaction within an RL model is shown in Figure 1.1. An agent receives a representation of an environment in a state description. Based on this state description, the agent takes an action to maximize the reward received. The goal of the agent is to interact with the environment and obtain maximum rewards based on state-action pairs.

The maintenance environment is modelled as a discrete time Markov Decision Process (MDP). An MDP provides a mathematical formulation for decision-making under uncertainties. Moreover, the state transitions satisfy the Markov property, which implies that the future states are independent of the past states, given the current state. The MDP is represented as a five item tuple

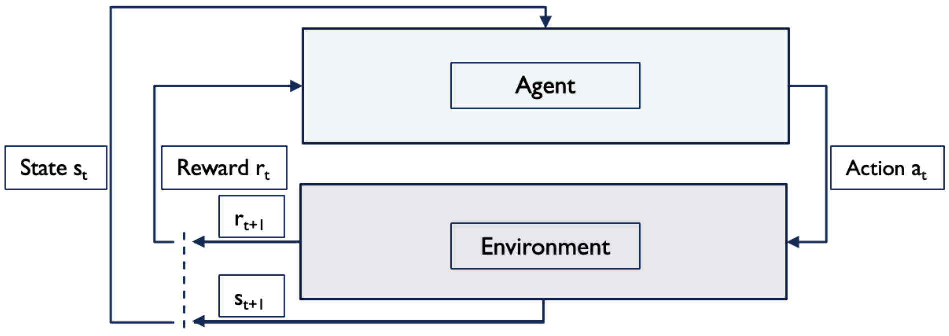


Figure 1.1: The Agent-Environment interaction.

(S, A, P_a, R_a, γ):

1. S **is a set of states.** At every timestep t, a new state $s_t \in S$ takes place.
2. A **is a set of actions.** At every timestep t, a new action $a_t \in A$ is performed.
3. P_a **is a probabilistic distribution of state transitions.** Which is the probability of being in s_{t+1} after being in s_t after taking action a_t.
4. $R_a(s_t, s_{t+1})$ **is the reward function.** Which is the immediate reward, received after transitioning from state s_t to s_{t+1} due to an action a_t.
5. γ **is the discount factor**. Which implies the importance of the future rewards considered in state s_t.

The goal is to find an optimal policy that maximizes the total expected return R. The total return is defined as the sum of discounted rewards over a given time horizon T and is defined in Equation 1.1.

$$R = \sum_{t=0}^{T} \gamma^t r_t \tag{1.1}$$

Where γ is the discount factor $(0 \leq \gamma \leq 1)$, and r_t is the instantaneous reward. When γ is large, future rewards are considered more than when γ is small.

1.1.1. Q-Learning

The objective in the MDP is to find an optimal policy π^* that maximizes the total reward. The policy π defines the agent's actions in different states, and is represented as an action a_t taken in state s_t. Formally, the policy π is defined as a probability $\pi(a|s)$ of taking an action $a \in A$ in state $s \in S$ [25].

$$\begin{aligned} &\pi : A \times S \rightarrow [0,1] \\ &\pi(a,s) = \mathbb{P}\,(a_t = a \mid s_t = s) \end{aligned} \tag{1.2}$$

The state-value function $V^\pi(s_t)$ describes how good it is to be in state s_t. It represents the value of a state given a policy π. The state-value function is defined as the expected cumulative reward of following the policy π from state s_t at timestep t.

$$V^\pi(s_t) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_t, \pi\right] \tag{1.3}$$

Q-Learning is a model-free reinforcement learning algorithm that aims to learn the quality value of an action in a given state [15]. Q-Learning uses an state-action value function $Q^\pi(s_t, a_t)$ that specifies the quality of taking a specific action in a state s_t. The Q-values represent the expected reward of a state-action pair.

$$Q : S \times A \rightarrow \mathbb{R} \tag{1.4}$$

The Q-value function describes how good the state-action pair is and focuses on a specific action in a specific state. The Q-value function is defined as the expected cumulative reward of taking action a_t in state s_t and following the policy π. The Q-value function Q^π is described as:

$$Q^\pi(s_t, a_t) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_t, a_t, \pi\right] \tag{1.5}$$

The main difference is that $V^\pi(s_t)$ considers only the expected discounted rewards in a given state s_t over all actions according to the policy π. While $Q^\pi(s_t, a_t)$ considers the expected discounted rewards based on state s_t, following policy π and taking action a_t.

The Q-Learning agent initializes a table consisting of the Q-values $Q(s_t, a_t)$, which are state-action quality values and are used for selecting the optimal actions based on a state. The agent aims to learn the optimal policy π^*, which yields sequential decisions with the highest expected discounted reward. Consequently, the optimal Q-value function $Q^{\pi^*}(s_t, a_t$ is the maximum expected cumulative reward achievable from a given (s_t, a_t) pair. The optimal policy π^*, is described as:

$$\pi^*(s_t) = \arg\max_{a_t} Q^{\pi^*}(s_t, a_t) \tag{1.6}$$

In Q-Learning, Q^{π^*} satisfies the Bellman optimality equation, which is used to find the optimal policy. Ensuring that for each (s_t, a_t) the $Q^*(s_t, a_t)$ values are known, then the optimal strategy is to take the action that maximizes the expected value of the consecutive total rewards. The optimal action-value function, obeying the Bellman equation, is defined in Equation 1.7.

$$Q^*(s_t, a_t) = \mathbb{E}\left[r + \gamma \max_{a_{t+1}} Q^*(s_{t+1}, a_{t+1}) \mid s_t, a_t\right] \tag{1.7}$$

During training, at each timestep t, the agent takes an action a_t in state s_t and observes the reward r_t. The quality value of this state-action pair is computed and the Q-table is updated as follows:

$$Q(s_t, a_t) \leftarrow (1-\alpha)Q(s_t, a_t) + \alpha\left(r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1})\right) \tag{1.8}$$

Where $(1-\alpha)Q(s_t, a_t)$ is the old value, weighted by the learning rate α. αr_t is the reward obtained by taking an action weighted by the learning rate α. And $\alpha\gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1})$ is the maximum reward obtained from the state s_{t+1}, weighted by the learning rate α and discount factor γ.

Algorithm 1 Q-Learning Algorithm for the Predictive Maintenance Framework

```
Input: State Space $S$, Action Space $A$, Reward Function $R$.
Output: Maintenance Decisions.
Initialization: $Q(s_t, a_t) = 0, \forall s_t \in S, a_t \in A$.
Settings: Learning rate $\alpha$, Discount rate $\gamma$.
for episode $e \in \{1, \dots, E\}$ do
    Initialize state $s_t$
    for timestep $t \in \{1, \dots, T\}$ in the episode do
        Observe the state of the system $s_t$
        Select action $a_t$ using $\varepsilon$-greedy strategy, otherwise select $a_t = \max_{a_t} Q^{\pi^*}(s_t, a_t)$
        Execute $a_t$ in the simulation
        Observe the reward $r_t$ and the new state $s_{t+1}$
        Update Q-Table: $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left[r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\right]$
    end
end
```

Exploration vs Exploitation

A well-known dilemma in training an agent is the exploration versus exploitation trade-off [53]. The agent must choose between exploration (making new decisions) or exploitation (repeating experienced decisions). At the beginning of the training phase, the agent must perform random actions because it does not have complete knowledge of the environment. As the episodes progress, the agent must choose the most favorable action over exploring new actions. An exponentially decaying epsilon parameter $(0 \le \epsilon < 1)$, based on the ratio of the initial and final epsilon, is chosen. The agent chooses the action with the highest Q-value with $P(1-\epsilon)$ and a random action otherwise [46]. The exploration probability decays over the total amount of episodes from an initial value of 1 to a final value of 0.01.

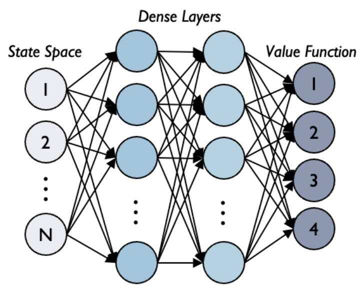


Figure 1.2: Neural network used to estimate the value function in the Deep Q-Learning Network.

1.1.2. Deep Q-Learning

The disadvantage of using traditional Q-Learning is the increase in size of the Q-table when the action- or state- space increases in size. The original Q-Learning algorithm is not scalable because it must compute $Q(s_t, a_t)$ for each pair. In this research, the state space is large because of the many values it can hold. Each entry in the state space can have any value between [0,1]. And since we consider a long-term horizon that includes two components, the unpredictability caused by replacing a component at any given point also contributes to the size of the Q-table. An improvement to traditional Q-Learning is Deep Q-Learning (DQL), which uses a neural network to estimate the Q-value, developed by Mnih et al. in 2015 [24]. DQL uses a function approximator to estimate the action-value function. This is done by utilizing a neural network (Figure 1.2) with network parameters θ, also known as weights. DQL approximates the Q-values by a function: $Q(s,a;\theta) \approx Q^*(s,a)$. The essence is that two similar states $(s_t^1 \approx s_t^2)$, are about equally good or bad to be in and therefore also have a similar Q-value: $Q^\pi(s_t^1, a_t) \approx Q^\pi\left(s_t^2, a_t\right)$ for a specific a_t.

The used Deep Q-Network (DQN), shown in Figure 1.2, consists of an input layer, dense layers and an output layer. The input layer of the neural network receives the environment state space (s_t). The input is passed through the fully connected dense layers in the network, where the computational processes are performed. The nodes within the dense layer consist of non-linear transformation functions with network parameters θ. The output layer produces the Q-values for each possible action (a_t) for the input state (s_t). The output layer has a dimension of 4, represented by $Q(s_t, a_0)$, $Q(s_t, a_1)$, $Q(s_t, a_2)$, $Q(s_t, a_3)$.

Training Strategy

The DQN can be trained by iteratively minimizing the loss function $L_i(\theta_i)$ formulated in Equation 1.9. The network parameters θ_i are updated with a stochastic gradient descent algorithm by minimizing the mean squared error between the target Q-values and the predicted Q-values.

$$L_i(\theta_i) = \left(y - Q(s_t, a_t; \theta_i)\right)^2, \text{ where } y = r_t + \gamma \max_{a_{t+1}} Q\left(s_{t+1}, a_{t+1}; \theta_i^-\right) \tag{1.9}$$

Where the first term, y, are the Q-values of the target network, and the second term are the Q-values of the online network. The architecture of the target network is identical to that of the online network, except for the network parameters θ. The target network parameters (θ_i^-) are kept constant when calculating the loss function of the online network parameters (θ_i). The target network parameters are updated only every N_θ steps, which prevents the network from slipping into a worse policy. In addition, an experience sampling method is used to select a random batch of transitions to avoid correlated experiences. Each timestep, the transition s_t, a_t, r_t, s_{t+1} is stored in the replay memory buffer D. Instead of updating the network parameters θ based on only the last transition, a batch of experiences from D is used. Experience replay is used to ensure that the Deep Q-Network is trained not only on consecutive samples but on random mini-batches and prevents the DQN from learning the correlation between transitions.

The algorithm for Deep-Q Learning used by the PdM framework is given in algorithm 2. The core idea is the same as for ordinary Q-Learning. However, the value function is now approximated by a neural network. In addition, a target network is used whose network weights are updated only every N_θ time steps.

Algorithm 2 Deep Q-Learning Algorithm for the Predictive Maintenance Framework

```
Input: State Space $S$, Action Space $A$, Reward Function $R$.
Output: Maintenance Decisions.
Initialization: $Q(s_t, a_t) = 0, \forall s_t \in S, a_t \in A$. Initialize replay memory $D$. Initialize network $Q_\theta$ and
  target network $\hat{Q}_\theta$.
Settings: Learning rate $\alpha$, Discount rate $\gamma$.
for episode $e \in \{1, \dots, E\}$ do
    Initialize state $s_t$
    for timestep $t \in \{1, \dots, T\}$ in the episode do
        Observe the state of the system $s_t$
        Select action $a_t$ using $\varepsilon$-greedy strategy, otherwise select $a_t = \max_{a_t} Q^{\pi^*}(s_t, a_t; \theta)$
        Execute $a_t$ in the simulation
        Observe the reward $r_t$ and the new state $s_{t+1}$
        Store transition $\{s_t, a_t, r_t, s_{t+1}\}$ in $D$
        Sample random minibatch of transitions from $D$
        Compute $y_i = r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}; \theta)$ if state $s_{t+1}$ is not terminal. Else: $y_i = r_t$
        Calculate loss by gradient descent step on $\left(y_i - Q(s_t, a_t; \theta)\right)^2$ with respect to network pa-
        rameters $\theta$.
        Update target network weights $\hat{Q}_\theta$ every $N_\theta$ timesteps
    end
end
```

1.1.3. Action Example Algorithm

An arbitrary example for the algorithm of action a_t = Repair $En_{1,t}$ (PM) is given in algorithm 3. The algorithm describes the steps taken in a simulation and shows the operating principle of the agent and the calculations of the KPIs. For example, if the agent decides to replace an engine, the component utilization C_u is calculated as En_1^l / En_1^a, where $En_{1,t}^l$ is the engine lifetime at timestep t and $En_{1,t}^a$ is the actual lifetime of the engine at timestep t. In addition, if the engine is repaired, it will be replaced with a new one from the pool P of components, indicated by $En_{1,t=0}^{p+1}$.

1.1.4. Motivation Shaded Area

The figures in the scientific paper use the shaded area, which is defined by the *minimum-maximum* values. The standard deviation does not give a good indication of the *minimum-maximum* performance, since it can be caused by a single outlier run or by large variability between runs.

1.2. Scenario 1: PdM Framework with $\gamma = 0.999$

This section contains additional information on Scenario 1, in which the PdM framework is used with $\gamma = 0.999$. Figure 1.3 shows the reward evolution for one simulation and illustrates the convergence of the agent's policy. The figure shows the cumulative reward for each episode. In contrast, Figure 1.4 shows the reward given per timestep, which clearly shows that the agent experiences corrective maintenance at the beginning of the episodes. An example of an episode in which the agent experiences corrective maintenance ($t = 393$) is shown in Figure 1.5. The corresponding component utilization and reward are depicted in Figure 1.6 and Figure 1.7, respectively.

Algorithm 3 Perform Maintenance Action a_t: Repair $En_{1,t}$

```
Input: State space: $s_t(AC_t)$, including the engine states and degradation level probabilities
Output: State space: $s_{t+1}(AC_{t+1})$
for timestep t in the episode do
    Observe the state of the system $s_t$
    Select action $a_t$ = Repair $En_{1,t}$
    if $En^s_{i,t+1} = 0$ (e.g., $En^l_{i,t+1} = En^a_{i,t+1}$) then
        Maintenance Activity = Corrective Maintenance
        Set Maintenance Reward: $r_t = -R_{cm}$
        $N_{cm} \leftarrow N_{cm} + 1$
        $C_u \leftarrow 1.0$
        $En^p_{i,t} \leftarrow En^{p+1}_{i,t=0}$
    else if $En^s_{1,t+1} = 1$ (e.g., $En^l_{1,t+1} < En^a_{1,t+1}$) then
        Maintenance Activity = Preventive Maintenance
        Set Maintenance Reward: $r_t = -R_{pm}$
        $N_{pm} \leftarrow N_{pm} + 1$
        $C_u \leftarrow En^l_1 / En^a_1$
        $En^p_{1,t} \leftarrow En^{p+1}_{1,t=0}$
    return $s_{t+1}, r_t, N_{cm}, N_{pm}, C_u$
end
```

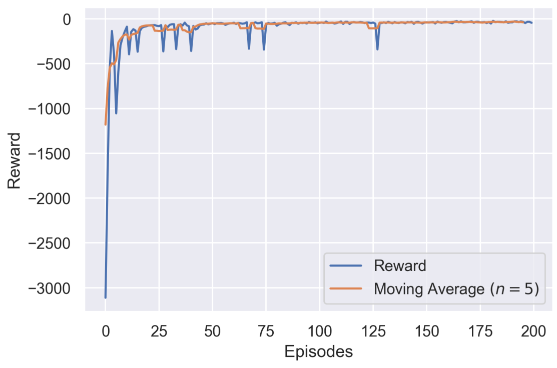


Figure 1.3: The reward development for $N_e = 200$, $N_s = 1$, and $N_{ts} = 1200$. Illustrating the convergence of the DQN agent's policy.

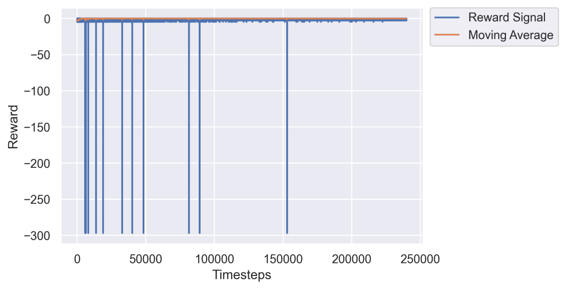


Figure 1.4: The reward stimulants for $N_e = 200$, $N_s = 1$, and $N_{ts} = 1200$. Illustrating the received rewards per timestep.

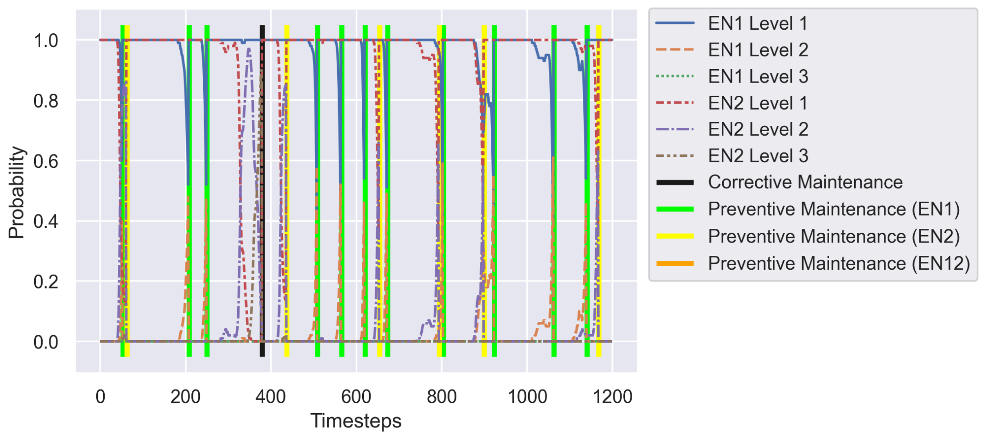


Figure 1.5: The working principle of the proposed maintenance decision framework for $N_e = 1$ and $N_{ts} = 1200$. Illustrating a corrective maintenance action at $t = 393$.

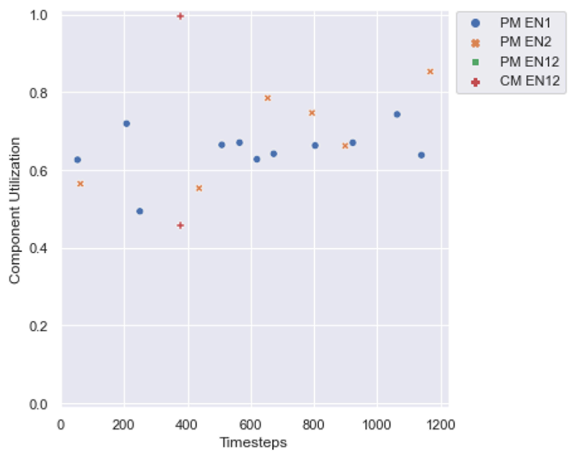


Figure 1.6: Corresponding component utilization values for En_1 and En_2 and for $N_e = 1$ and $N_{ts} = 1200$. Illustrating a corrective maintenance action at $t = 393$.

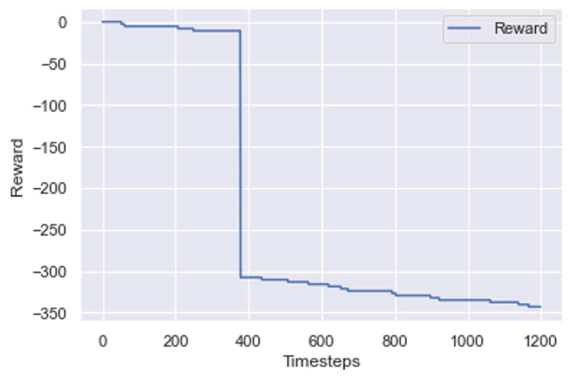


Figure 1.7: Corresponding reward for $N_e = 1$ and $N_{ts} = 1200$. Illustrating a corrective maintenance action at $t = 393$.

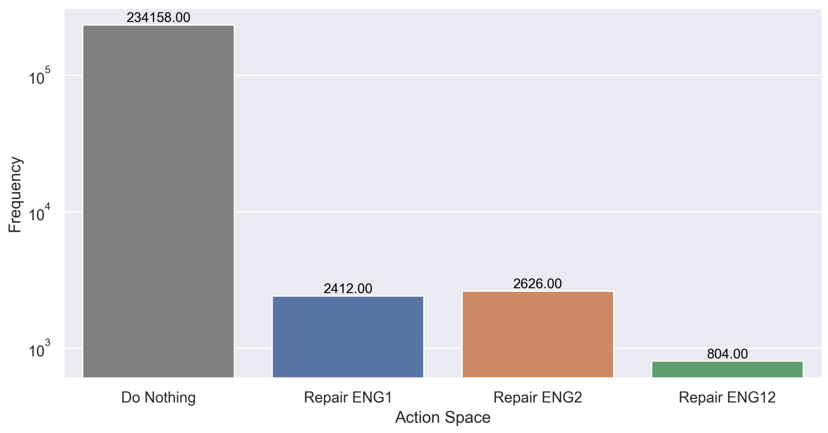


Figure 1.8: Action space analysis for $N_e = 200$ and $N_s = 1$ during training (Note: y-axis uses log scale).

1.2.1. Detailed Action Analysis

This section contains a detailed analysis of the agent's actions for the first scenario. Figure 1.8 shows the analysis of the action space for one episode in which the agent has to take 240000 actions. The agent takes the action Do Nothing most of the time, namely 234158 times. The actions Repair Engine 1 and Repair Engine 2 are almost equally distributed with about 2500 occurrences. The action to Repair Engine 1 & 2 at the same time is performed the least often, only 804 times. Figure 1.9 shows the distribution of the maintenance activities for one episode. The corrective maintenance activity is appended, which is not an action of the agent but a consequence of the late execution of a random action. However, it is executed only 17 times, which is very few compared to the other maintenance activities. In addition, Figure 1.10 shows the action space for 10 simulations to analyze the distributions. It can be seen that the boxplots show the same distribution for several simulations, indicating that the agent is able to learn the optimal policy for various initial parameters. Figure 1.11 shows the utilization of the components for a specific simulation, in which the agent performs the preventive maintenance action 2269 times. It can be seen that at the beginning of the actions, the utilization is close to zero. As the timesteps pass, the agent learns to use the components more and the utilization rate begins to increase up to a stable level. Figure 1.12 shows the same components as Figure 1.11 only with the replacement time in blue and the true component lifetime in orange.

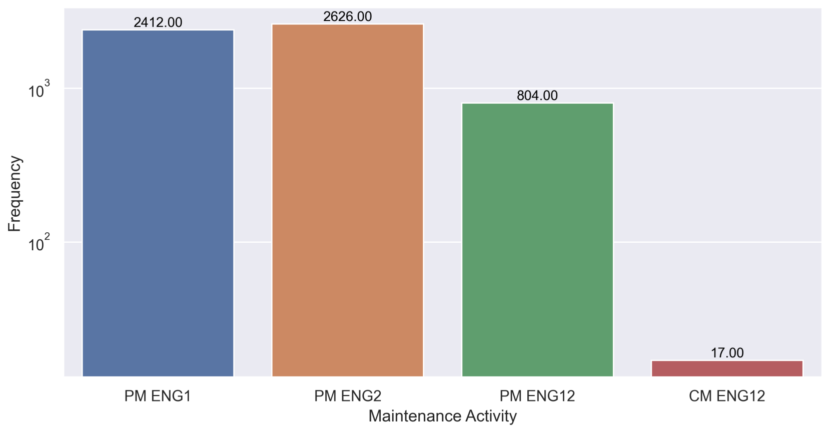


Figure 1.9: Maintenance activity analysis for $N_e = 200$ and $N_s = 1$ during training (Note: y-axis uses log scale).

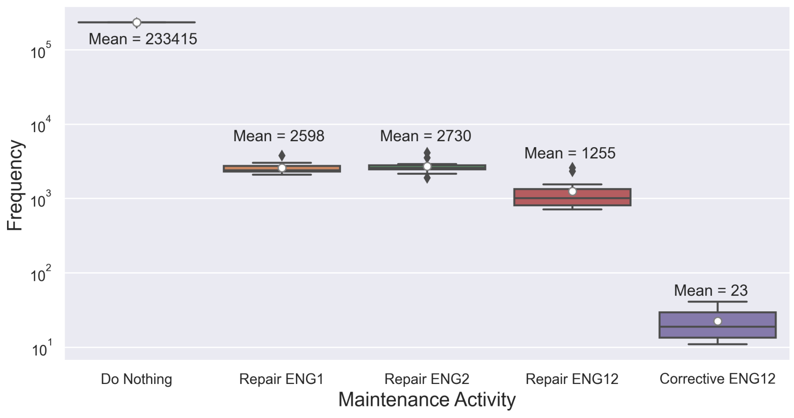


Figure 1.10: Maintenance activity analysis for $N_e = 200$ and $N_s = 10$ during training (Note: y-axis uses log scale).

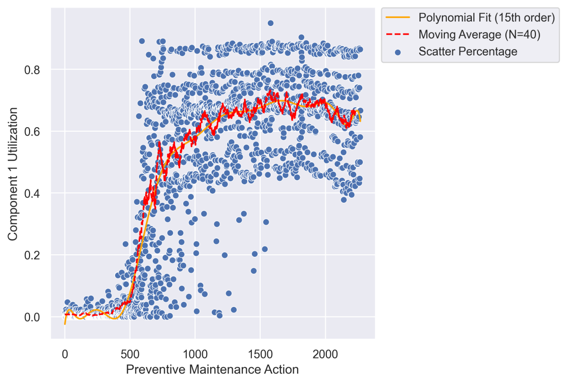


Figure 1.11: Illustration of Component 1 utilization rate for the preventive maintenance action. Where $N_e = 200$, $N_s = 1$, and $N_{pm} = 2269$.

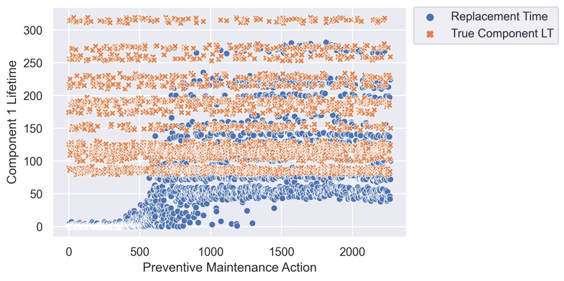


Figure 1.12: Illustration of Component 1 replacement time and true component lifetime. Where $N_e = 200$, $N_s = 1$, and $N_{pm} = 2269$.

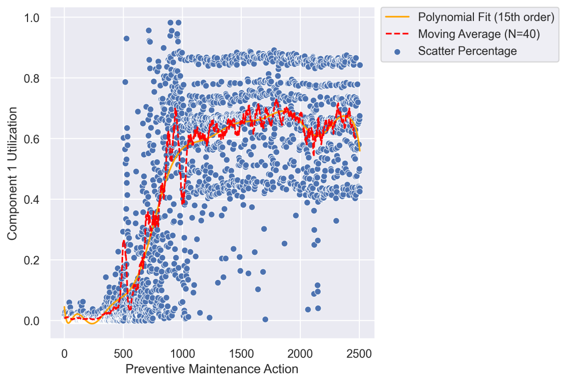


Figure 1.13: Illustration 2 of Component 1 utilization rate for the preventive maintenance action. Where $N_e = 200$, $N_s = 1$, and $N_{pm} = 2525$.

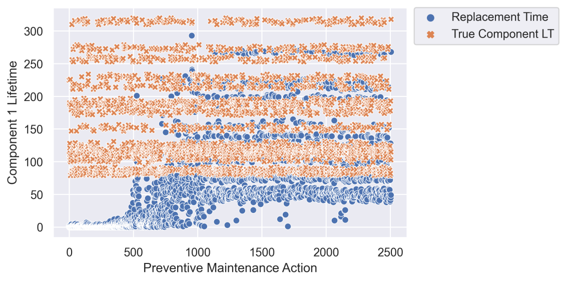


Figure 1.14: Illustration 2 of Component 1 replacement time and true component lifetime. Where $N_e = 200$, $N_s = 1$, and $N_{pm} = 2525$.

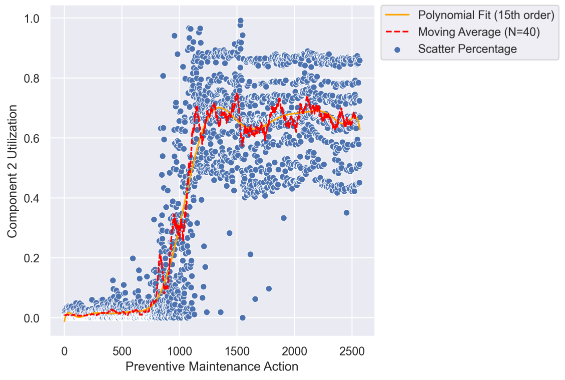


Figure 1.15: Illustration of Component 2 utilization rate for the preventive maintenance action. Where $N_e = 200$, $N_s = 1$, and $N_{pm} = 2569$.

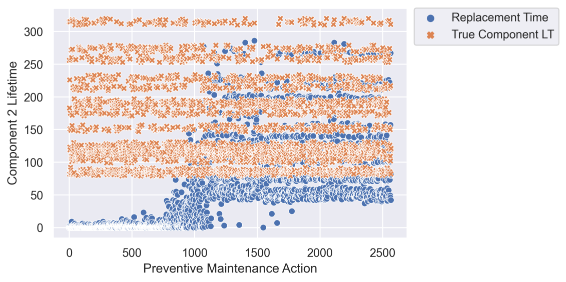


Figure 1.16: Illustration of Component 2 replacement time and true component lifetime. Where $N_e = 200$, $N_s = 1$, and $N_{pm} = 2569$.

Another utilization rate of a component is depicted in Figure 1.13. Now, however, it can be seen that the agent experiences high utilization rates before stable behavior occurs. This is the point at which he has experienced corrective maintenance a few times and therefore runs the components no longer. Figure 1.14 shows the replacement time and true component lifetime from the same components as in Figure 1.13.

As expected, component 2 exhibits the same behavior, since the engine data is not linked to the component number. Figure 1.15 shows the component utilization for engine number 2. Figure 1.16 shows the corresponding replacement time and true component lifetimes.

For the opportunistic maintenance action, depicted in Figure 1.17, we see the same behavior as for the individual maintenance actions. At the beginning of the episodes, the component usage is low. As the agent begins to recognize the state spaces, the component usage increases. For another simulation, shown in Figure 1.18, we can identify the same behavior.

Analyzing the corrective maintenance actions, whose utilization rates are given in Figure 1.19, we see each time that one of the two components reaches the value 1.0. Figure 1.20 shows the engine lifetime counter for one simulation and 240000 timesteps. In which is visible that the lifetimes are increasing up to their maximum lifetime.

Figure 1.21 shows an example of a proposed replacement decision for one specific engine. The agent decides to replace the component when the second degradation level is at its highest point. Another example, including the representation of the individual data points, is depicted in Figure 1.22.

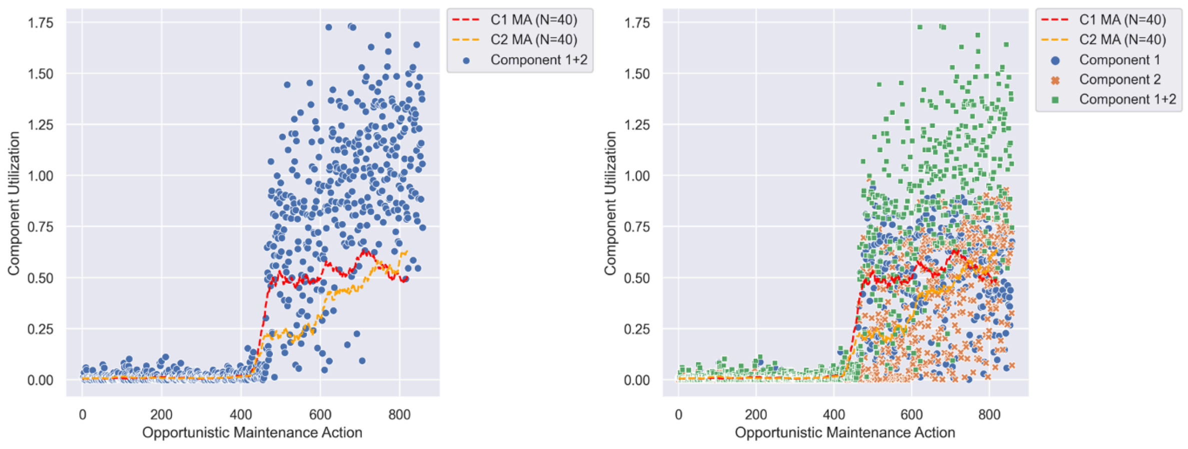


Figure 1.17: Illustration of the component utilization rates for the opportunistic maintenance action. Where $N_e = 200$, $N_s = 1$, and $N_{om} = 866$.

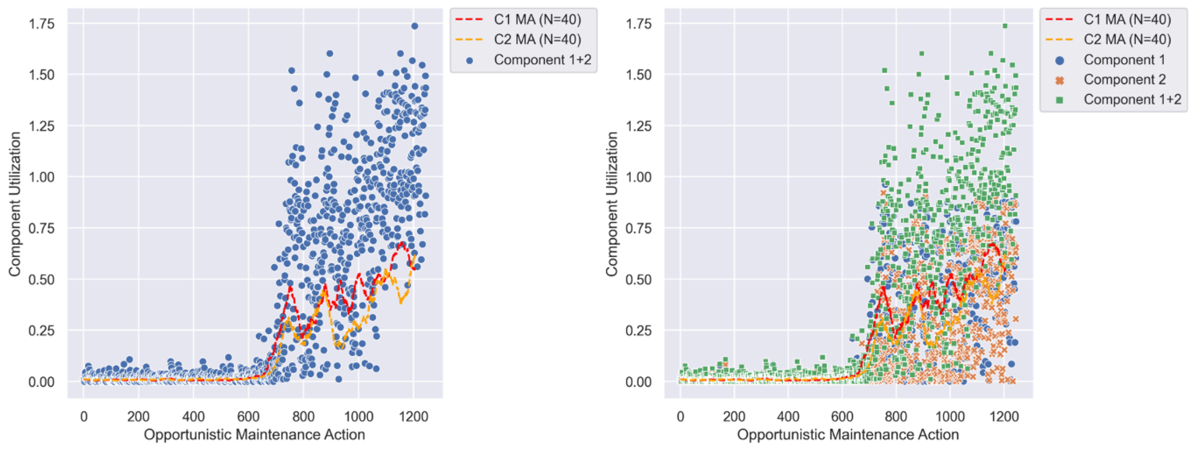


Figure 1.18: Illustration 2 of the component utilization rates for the opportunistic maintenance action. Where $N_e = 200$, $N_s = 1$, and $N_{om} = 1235$.

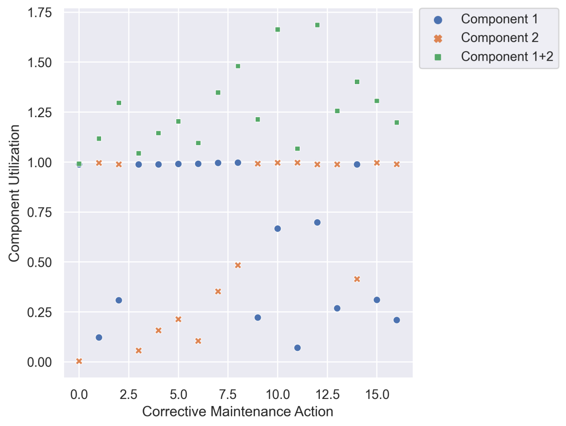


Figure 1.19: Component utilization values for En_1 and En_2 and for $N_e = 1$, $N_s = 1$, and $N_{ts} = 1200$. Illustrating the corrective maintenance actions.

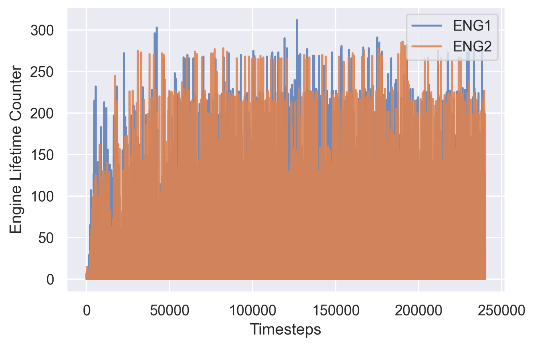


Figure 1.20: The working principle illustrated by the engine lifetime counter for $N_e = 200$, $N_s = 1$, and $N_{ts} = 240000$.

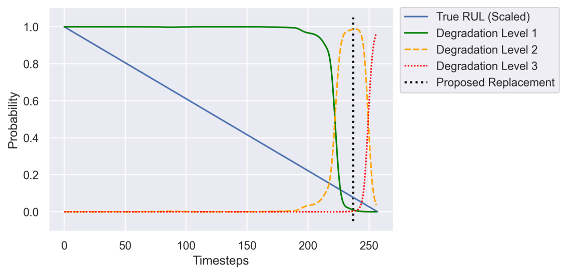


Figure 1.21: Degradation level probabilities for one specific UnitID. Including proposed replacement time.

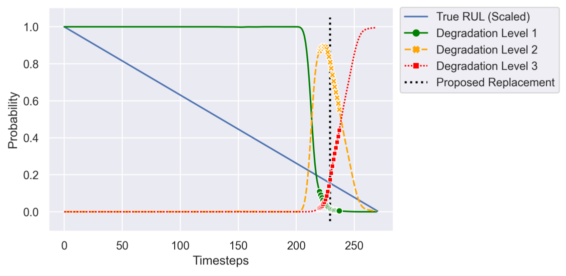


Figure 1.22: Degradation level probabilities for one specific UnitID. The proposed replacement is at $t = 229$ with state space: $s_t = [1, 0.022, 0.839, 0.139]$.

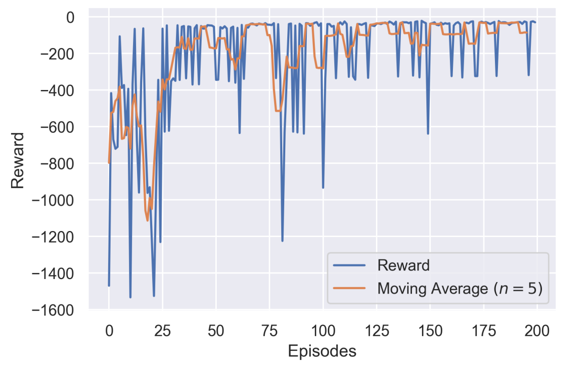


Figure 1.23: Scenario 2: The reward development for $N_e = 200$, $N_s = 1$, and $N_{ts} = 1200$. Illustrating the convergence of the DQN agent's policy.

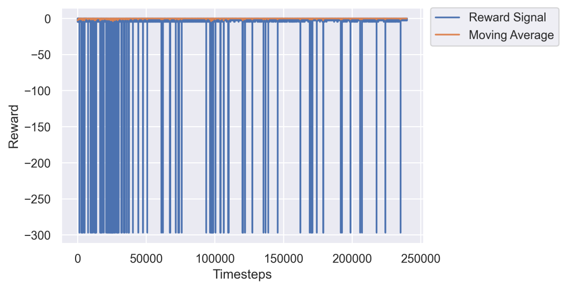


Figure 1.24: Scenario 2: The reward stimulants for $N_e = 200$, $N_s = 1$, and $N_{ts} = 1200$. Illustrating the received rewards per timestep.

1.3. Scenario 2: PdM Framework with $\gamma = 0.7$

A relaxation of the discount factor from $\gamma = 0.999$ to $\gamma = 0.7$ produces a different behavior in the development of the reward, as shown in Figure 1.23. As the agent has a more myopic policy, he experiences the corrective maintenance action more often. This is indicated by the peaks in Figure 1.23, representing the large negative corrective maintenance reward. In addition, the reward incentives in Figure 1.24 now occur more frequently and show that the corrective maintenance action occurs more often.

The working principle for the second scenario is depicted in Figure 1.25. A clear difference can be seen at the moment when the agents decide to take an action, compared to the first scenario. Now the probabilities for dl_2 and dl_3 show higher values. The component utilization rate, as shown in Figure 1.26, is now significantly higher than in Scenario 1. Where Scenario 1 had an average component utilization of about 0.7, Scenario 2 reaches a value of almost 0.92.

Also, the component utilization for the preventive maintenance action for engine 1 shows the expected behavior. As depicted in Figure 1.28, the blue dots are now closer to one, indicating a higher utilization rate. In addition, Figure 1.29 shows the same components as Figure 1.28, but now the replacement time in blue and the true component lifetime in orange. As expected, the same behavior is visible for the second component, shown in Figure 1.30 and Figure 1.31.

The opportunistic maintenance action, depicted in Figure 1.32, sometimes shows a total component utilization rate close to 2, indicating that both individual engines are being utilized to their maximum.

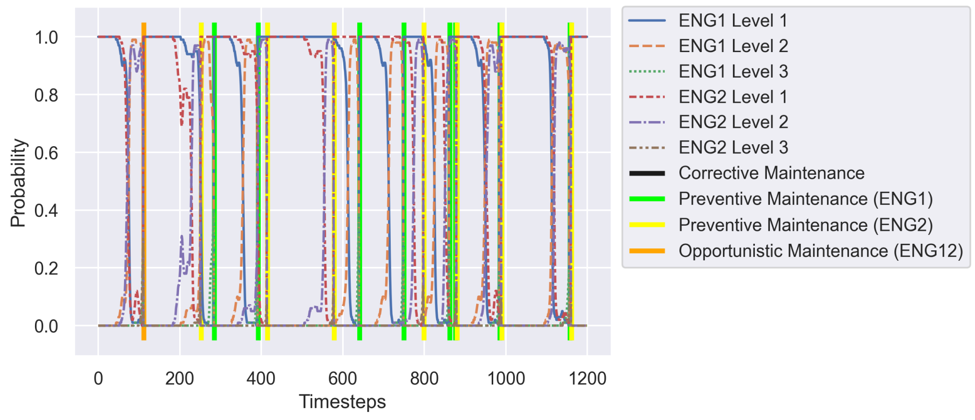


Figure 1.25: Scenario 2: The working principle of the proposed maintenance decision framework for $N_e = 1$ and $N_{ts} = 1200$. Illustrating the influence of the discount factor.

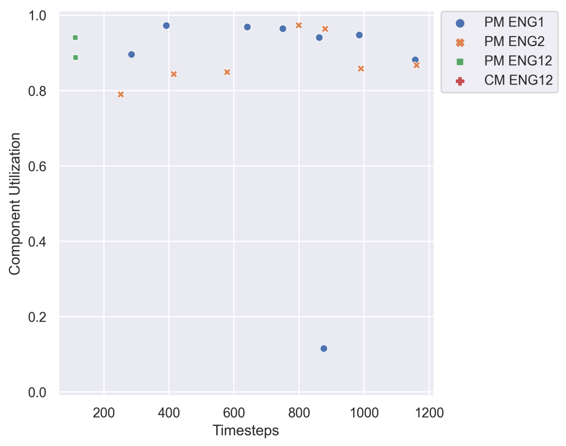


Figure 1.26: Scenario 2: Corresponding component utilization values for En_1 and En_2 and for $N_e = 1$ and $N_{ts} = 1200$.

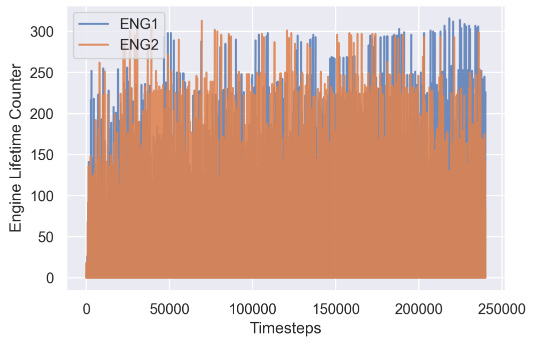


Figure 1.27: Scenario 2: The working principle illustrated by the engine lifetime counter for $N_e = 200$, $N_s = 1$, and $N_{ts} = 240000$.

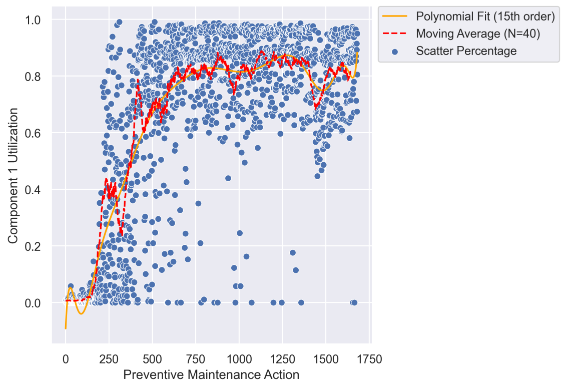


Figure 1.28: Scenario 2: Illustration of Component 1 utilization rate for the preventive maintenance action. Where $N_e = 200$, $N_s = 1$, and $N_{pm} = 1711$.

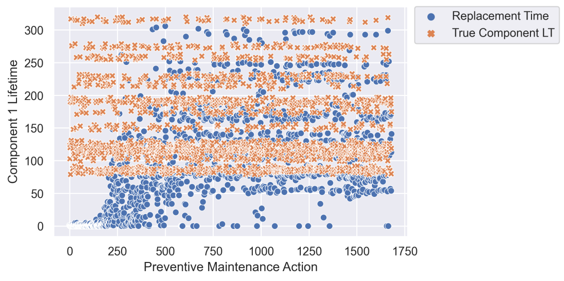


Figure 1.29: Scenario 2: Illustration of Component 1 replacement time and true component lifetime. Where $N_e = 200$, $N_s = 1$, and $N_{pm} = 1711$.

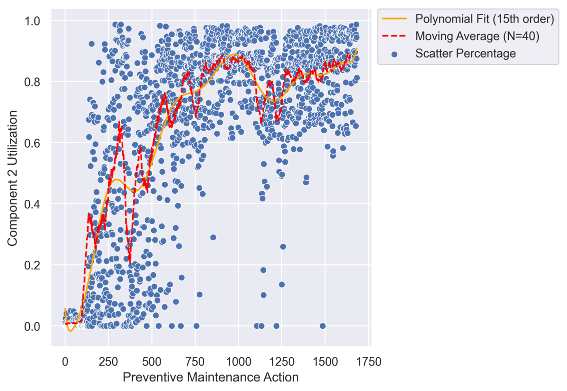


Figure 1.30: Scenario 2: Illustration of Component 2 utilization rate for the preventive maintenance action. Where $N_e = 200$, $N_s = 1$, and $N_{pm} = 1692$.

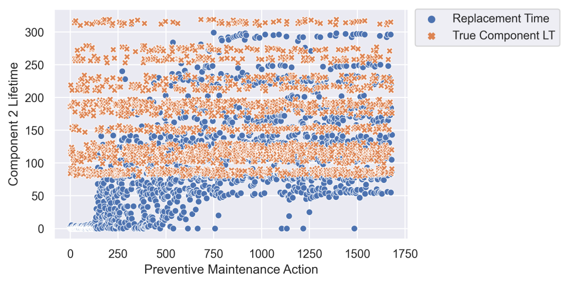


Figure 1.31: Scenario 2: Illustration of Component 2 replacement time and true component lifetime. Where $N_e = 200$, $N_s = 1$, and $N_{pm} = 1692$.

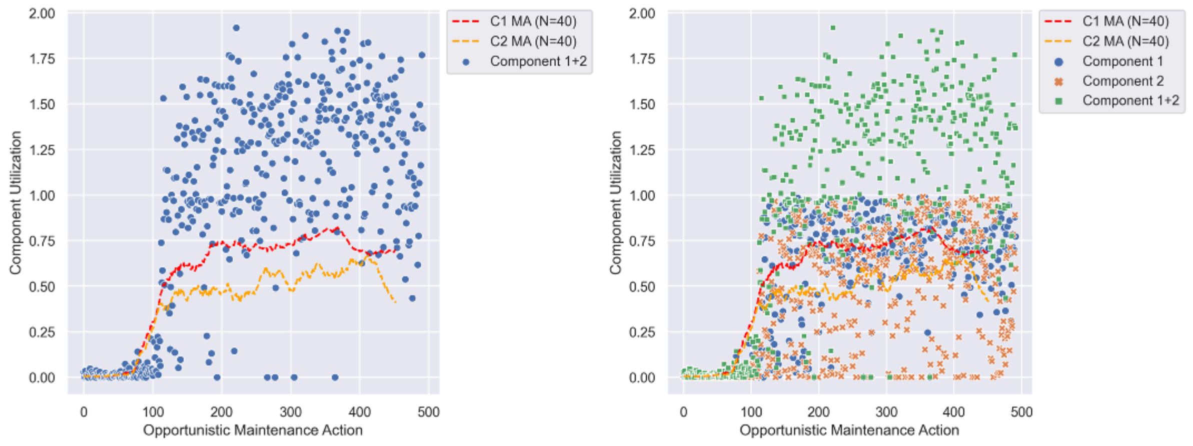


Figure 1.32: Scenario 2: Illustration of the component utilization rates for the opportunistic maintenance action. Where $N_e = 200$, $N_s = 1$, and $N_{om} = 492$.

1.3.1. Results Statistical T-Test

The results for the statistical t-test for Scenario 2 are given in Table 1.1, which shows that all results are significant ($p < 0.05$).

Maintenance Policy	*Time-Based*	*Corrective*	*Ideal*
PdM C_m [-]	$2.73e^{-109}$	$1.71e^{-139}$	$2.06e^{-38}$
PdM T_c [-]	$2.35e^{-61}$	$1.78e^{-52}$	$1.78e^{-52}$
PdM W_t [t]	$5.26e^{-40}$	$2.40e^{-116}$	$1.66e^{-114}$
PdM C_u [-]	$2.02e^{-62}$	$2.00e^{-116}$	$1.75e^{-114}$
PdM R_t [t]	$4.06e^{-52}$	$1.52e^{-97}$	$2.26e^{-99}$

Table 1.1: Scenario 2: P-values of the statistical t-test for the long-term maintenance cost (C_m), number of replaced components (T_c), wasted component lifetime (W_t), component utilization (C_u), and component replacement time (R_t).

2
Component Prognostics

This chapter contains a sector that further explores the structure of the LSTM cell, followed by a pseudocode representation of the LSTM algorithm. This is followed by a section describing the C-MAPSS dataset. This chapter concludes with the results and illustrations of various window configurations, including the training and validation loss.

2.1. LSTM Cell Explanation

A general LSTM cell structure is shown in Figure 2.1. x_t is the cell input and h_t is the cell output value, which in this research represents the sensor data and the degradation level classification. Each LSTM cell contains a cell state (C_t) that depends on the previous cell state (C_{t-1}), characterized by Equation 2.1. Which information is retained depends on the internal cell gates, which are able to learn which information should be retained and which should be omitted.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{2.1}$$

Where C_t is the new cell state, f_t is the forget layer, C_{t-1} is the old cell state, i_t is the input layer, and $\tilde{C}_t$ are the candidate values. The information can be altered by the gates, which are deciding which information is added or removed. The three gates consists of a sigmoid function, which outputs a value between 0 and 1. If the value is zero, no information is added, while if the value is one, all information is passed through. For all below stated equations, we consider W as the trainable

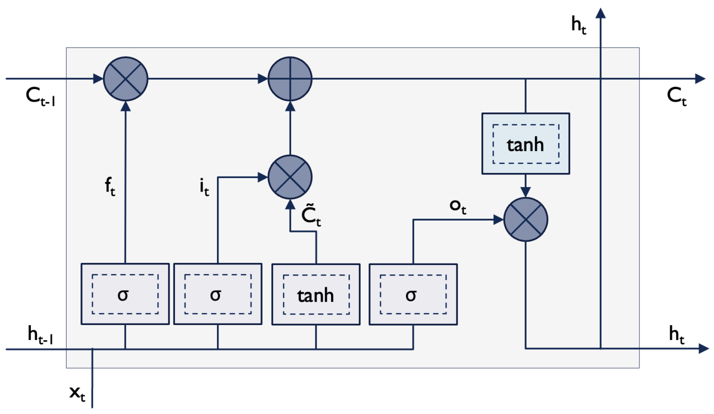


Figure 2.1: Long-Short Term Memory cell structure.

weight matrix and b as the bias matrix.

$$W = \begin{bmatrix} W_f \\ W_i \\ W_c \\ W_o \end{bmatrix}, b = \begin{bmatrix} b_f \\ b_i \\ b_c \\ b_o \end{bmatrix} \tag{2.2}$$

Looking closer at the LSTM cell structure, the following three gate layers can be identified:

1. **Forget Gate Layer**: considers h_{t-1} and x_t and outputs a value between 0 and 1 to decide what information is carried along and what information is omitted. Characterized by the equation:

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \tag{2.3}$$

   Where f_t is the forget gate, σ is the sigmoid activation function, h_{t-1} is the output at timestep $t-1$ and x_t is the input at timestep t.

2. **Input Gate Layer**: consists of two layers, where the sigmoid layer considers h_{t-1} and x_t and decides which values are updated. The next layer, based on a tanh activation function, computes a vector of values ($\tilde{C}_t$) which could be added to the state. These two layers are then multiplied to generate an update to the current cell state and are defined by the following equations:

$$\begin{aligned} i_t &= \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right) \\ \tilde{C}_t &= \tanh\left(W_C \cdot [h_{t-1}, x_t] + b_C\right) \end{aligned} \tag{2.4}$$

   Where i_t is the input gate and $\tilde{C}_t$ is the vector of new possible information.

3. **Output Layer**: the last layer determines what the output of the cell will be. The output is a filtered version of the cell state. The signal is passed through a sigmoid layer to determine which part will be the output and then multiplied by a tanh function to obtain values in the range $[-1, 1]$ and subsequently provides the desired output values. The output layer is defined as follows:

$$\begin{aligned} o_t &= \sigma\left(W_o\,[h_{t-1}, x_t] + b_o\right) \\ h_t &= o_t * \tanh\left(C_t\right) \end{aligned} \tag{2.5}$$

   Where o_t is the output gate and h_t is the output value of the current cell and input for the next cell.

In conclusion, the forget gate layer determines what is important to retain from previous timesteps. The input gate layer determines what information from the current timestep should be added. The output gate layer determines the next cell state and output. In the LSTM network (Figure 2.1), various sigmoid and tanh functions (Figure 2.2) are used to specify which information is transmitted and which is not, and to regulate the network. The sigmoid function is characterized by Equation 2.6 and has a range of $[0, 1]$. The tanh function, defined in Equation 2.7, has a range of $[-1, 1]$.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.6}$$

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.7}$$

A softmax function is used as the activation function in the output layer, as the PdM framework requires failure probabilities between [0,1]. The softmax function converts the real values of the LSTM layer into a vector of probabilities, summing to 1, characterized by Equation 2.8. The LSTM

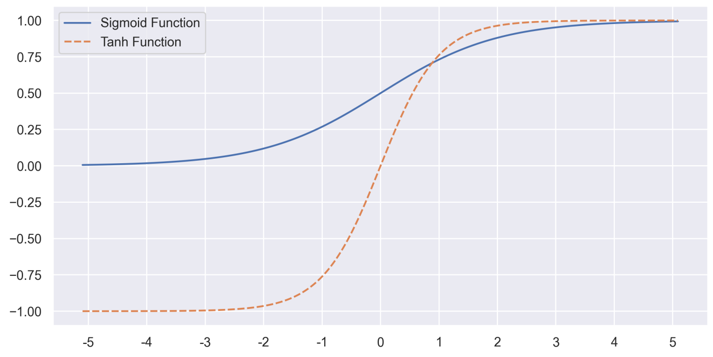


Figure 2.2: The sigmoid (blue) and tanh (orange) functions used in the LSTM network.

architecture used has 3 output nodes, where each node gives the probability that the component belongs to that particular degradation class. In addition, dropout layers are used in each LSTM layer to avoid over-fitting of the training data and to improve the performance of the model [21]. The dropout layers act as a regularization method by probabilistically excluding input and recurrent connections.

$$S(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{J} e^{x_j}} \quad \text{for } i = \{1, \dots, J\} \tag{2.8}$$

2.1.1. LSTM Algorithm

The pseudocode for the LSTM network classifier is given in algorithm 4.

Algorithm 4 LSTM Network Classifier for Degradation Level Probabilities

```
Input: Preprocessed sensor data: $X_t^i = [x_t^1, x_t^2, x_t^3, ..., x_t^N]$. Relevant features selected, normalized,
       exponential smoothed, labeled and sequenced.
Output: Probabilities that Engine $i$ belongs to degradation level $j$ at timestep $t$: $En_{i,t}^{dl_j}$.
Notation: $X$ = Sensor data (Input)
          $y$ = Degradation classes (Output)
          $\hat{y}$ = Predicted Degradation classes (Output)
Initialization: Split data in train, test, and validation subsets.
Settings: Number of epochs = 20, Number of layers = 3, Nodes per layer = 256, Batch size = 64.
while max number of epochs not reached do
    Set model: sequential LSTM
    Set loss function = categorical crossentropy
    Set optimizer = Adam
    Compile LSTM model output: $\hat{y} = f(\text{LSTM}(X))$ based on train set
    Compile loss: $L_C = -\sum_{i=1}^{N} y_{ij} * \log \hat{y_{ij}}$
    Update LSTM network weights
end
Predict probabilities $\hat{y}$ on test set by using LSTM classifier
```

2.2. Turbofan Engine Degradation Simulation Data Set

Description from NASA [37]: *"Engine degradation simulation was carried out using C-MAPSS. Four different were sets simulated under different combinations of operational conditions and fault modes. Records several sensor channels to characterize fault evolution. The data set was provided by the Prognostics CoE at NASA Ames."* The system model is representing an aircraft engine in C-MAPSS

Data Set	*Train Units*	*Test Units*	*Operating Conditions*	*Fault Modes*
FD001	100	100	1	1
FD002	260	259	6	1
FD003	100	100	1	2
FD004	249	248	6	2

Table 2.1: Overview of the different subsets (FD001-FD004) within the C-MAPSS dataset.

Sensor Number	*Description*	*Unit*
1	Total temperature at fan inlet	°R
2	Total temperature at low pressure compressor outlet	°R
3	Total temperature at high pressure compressor outlet	°R
4	Total temperature at low pressure turbine outlet	°R
5	Pressure at fan inlet	psia
6	Total pressure in bypass-duct	psia
7	Total pressure at high pressure compressor outlet	psia
8	Physical fan speed	rpm
9	Physical core speed	rpm
10	Engine pressure ratio	–
11	Static pressure at high pressure compressor outlet(Ps30)	psia
12	Ratio of fuel flow to Ps30	pps/psi
13	Corrected fan speed	rpm
14	Corrected core speed	rpm
15	Bypass ratio	–
16	Burner fuel-air ratio	–
17	Bleed enthalpy	–
18	Demanded fan speed	rpm
19	Demanded corrected fan speed	rpm
20	High pressure turbine coolant bleed	lbm/s
21	Low pressure turbine coolant bleed	lbm/s

Table 2.2: Overview of the 21 different sensors within the C-MAPSS dataset.

Saxena et al. [2008]. The data set contains four different scenarios, described in Table 2.1.

The two fault modes are high-pressure degradation and fan degradation. The input for the operating conditions are altitude, Mach number and throttle resolver angle, which combined result in six different operating conditions. Each data set contains the following variables: Engine Unit ID, Time [Cycles], Operational Setting [1,2,3], Sensor Readings [1-21]. Where the sensors details are given in Table 2.2.

2.2.1. Dataset Exploration

Various descriptive statistics are used to gain a better understanding of the various data sets. The values are computed for the first subset, 'FD001', and shown in Table 2.3. Table 2.3 shows that there are 100 engines with various maximum life cycles. The mean and intervals are not exactly lining up due to the fact that each unit can have different maximum life cycles and therefore a different amount of rows in the data set. Furthermore, the *Time [Cycles]* column shows that a unit broke down at first at 128 cycles and at last at 362 cycles. The distribution of the true engine lifetimes within the training dataset is shown in Figure 2.3. Moreover, the statistical values of the Operational Settings are shown in Table 2.4. In addition, the statistical values of the sensor values are given in Table 2.7.

2.2.2. Sensor Overview and Feature Selection

Only the useful features are selected by statistical analysis to reduce the feature space of Table 2.7. The feature selection method is described in Part I. Moreover, the sensor trends are indicated in

Variable	*UnitID [-]*	*Time [Cycles]*
count	20631	100
mean	51.51	206.31
std	29.23	46.34
min	1.00	128.00
25%	26.00	177.00
50%	52.00	199.00
75%	77.00	229.25
max	100.00	362.00

Table 2.3: Descriptive statistics of subset FD001 (C-MAPSS dataset).

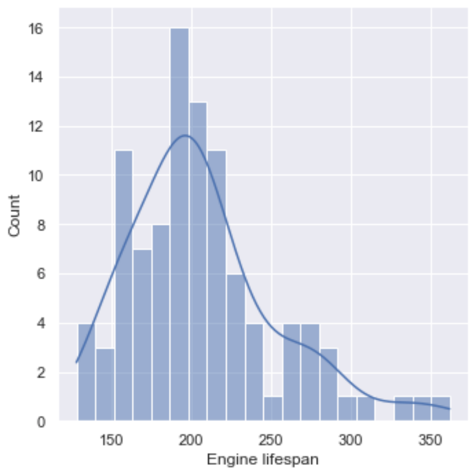


Figure 2.3: Histogram showing the distribution of the true engine lifetimes of subset FD001 (C-MAPSS dataset).

Variable	*Operational Setting 1*	*Operational Setting 2*	*Operational Setting 3*
count	20631	20631	20631
mean	-0.000009	0.000002	100.0
std	0.002187	0.000293	0.0
min	-0.008700	-0.000600	100.0
25%	-0.001500	-0.000200	100.0
50%	0.000000	0.000000	100.0
75%	0.001500	0.000300	100.0
max	0.008700	0.000600	100.0

Table 2.4: Operational settings statistics of subset FD001 (C-MAPSS dataset).

Trend	*Sensor Number*
Ascending	7, 12, 20, 21
Descending	2, 3, 4, 8, 11, 13, 15, 17
Irregular / Constant	1, 5, 6, 9, 10, 14, 16, 18, 19

Table 2.5: Sensor data trends of subset FD001 (C-MAPSS dataset).

Variable	*mean*	*std*	*min*	*25%*	*50%*	*75%*	*max*
LT	166.6	69.9	79.0	110.7	152.0	217.0	320.0

Table 2.6: Descriptive statistics used components in the PdM framework.

Sensor	*count*	*mean*	*std*	*min*	*25%*	*50%*	*75%*	*max*
Sensor 1	20631	518.67000	0.00000	518.67000	518.67000	518.67000	518.67000	518.67000
Sensor 2	20631	642.68093	0.50005	641.21000	642.32500	642.64000	643.00000	644.53000
Sensor 3	20631	1590.52312	6.13115	1571.04000	1586.26000	1590.10000	1594.38000	1616.91000
Sensor 4	20631	1408.93378	9.00060	1382.25000	1402.36000	1408.04000	1414.55500	1441.49000
Sensor 5	20631	14.62000	0.00000	14.62000	14.62000	14.62000	14.62000	14.62000
Sensor 6	20631	21.60980	0.00139	21.60000	21.61000	21.61000	21.61000	21.61000
Sensor 7	20631	553.36771	0.88509	549.85000	552.81000	553.44000	554.01000	556.06000
Sensor 8	20631	2388.09665	0.07099	2387.90000	2388.05000	2388.09000	2388.14000	2388.56000
Sensor 9	20631	9065.24294	22.08288	9021.73000	9053.10000	9060.66000	9069.42000	9244.59000
Sensor 10	20631	1.30000	0.00000	1.30000	1.30000	1.30000	1.30000	1.30000
Sensor 11	20631	47.54117	0.26709	46.85000	47.35000	47.51000	47.70000	48.53000
Sensor 12	20631	521.41347	0.73755	518.69000	520.96000	521.48000	521.95000	523.38000
Sensor 13	20631	2388.09615	0.07192	2387.88000	2388.04000	2388.09000	2388.14000	2388.56000
Sensor 14	20631	8143.75272	19.07618	8099.94000	8133.24500	8140.54000	8148.31000	8293.72000
Sensor 15	20631	8.44215	0.03751	8.32490	8.41490	8.43890	8.46560	8.58480
Sensor 16	20631	0.03000	0.00000	0.03000	0.03000	0.03000	0.03000	0.03000
Sensor 17	20631	393.21065	1.54876	388.00000	392.00000	393.00000	394.00000	400.00000
Sensor 18	20631	2388.00000	0.00000	2388.00000	2388.00000	2388.00000	2388.00000	2388.00000
Sensor 19	20631	100.00000	0.00000	100.00000	100.00000	100.00000	100.00000	100.00000
Sensor 20	20631	38.81627	0.18075	38.14000	38.70000	38.83000	38.95000	39.43000
Sensor 21	20631	23.28971	0.10825	22.89420	23.22180	23.29790	23.36680	23.61840

Table 2.7: Descriptive statistics sensor values FD001 (C-MAPSS dataset).

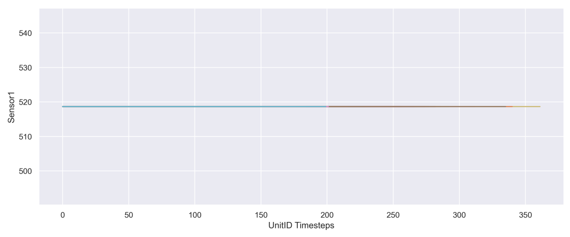


Figure 2.4: Overview of Sensor 1 for the complete FD001 C-MAPSS dataset.

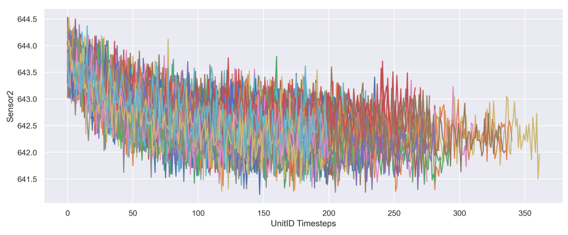


Figure 2.5: Overview of Sensor 2 for the complete FD001 C-MAPSS dataset.

Table 2.5. The selected sensors are: Sensor 2, 3, 4, 7, 8, 9, 11, 12, 13, 14, 15, 17, 20, and 21. The first sensor is depicted in Figure 2.4, up to the last sensor in Figure 2.24. The descriptive statistics for the data used in the numerical experiments are shown in Table 2.6.

2.3. Additional Results

This section contains the results of the different window configurations. Followed by an overview of degradation levels for several components. It concludes with a section on training and validation losses.

2.3.1. Results Window Bounds

This section illustrates the consequence of varying the window bounds on the probabilities for a specific engine. The degradation level probabilities for the first configuration, where $T_0 = 10, T_1 = 20$, is depicted in Figure 2.25. The last configuration, where $T_0 = 10, T_1 = 80$, is depicted in Figure 2.31.

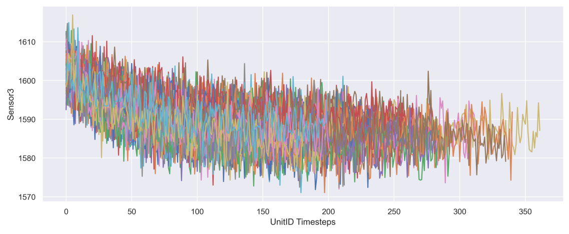


Figure 2.6: Overview of Sensor 3 for the complete FD001 C-MAPSS dataset.

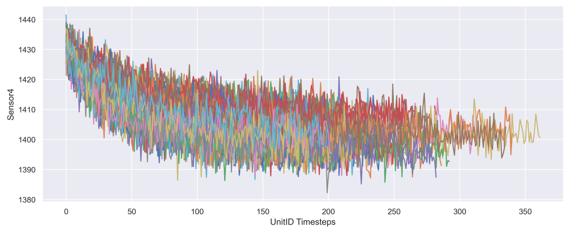


Figure 2.7: Overview of Sensor 4 for the complete FD001 C-MAPSS dataset.

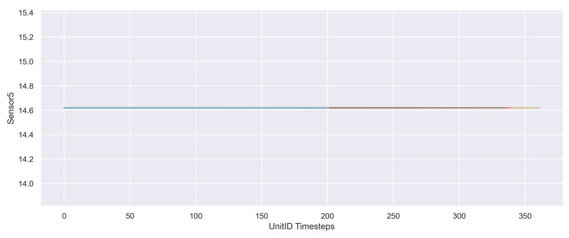


Figure 2.8: Overview of Sensor 5 for the complete FD001 C-MAPSS dataset.

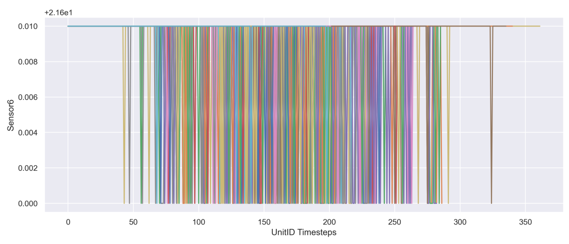


Figure 2.9: Overview of Sensor 6 for the complete FD001 C-MAPSS dataset.

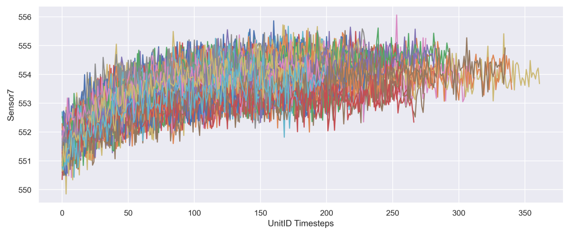


Figure 2.10: Overview of Sensor 7 for the complete FD001 C-MAPSS dataset.

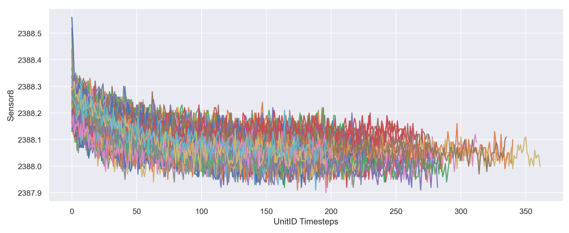


Figure 2.11: Overview of Sensor 8 for the complete FD001 C-MAPSS dataset.

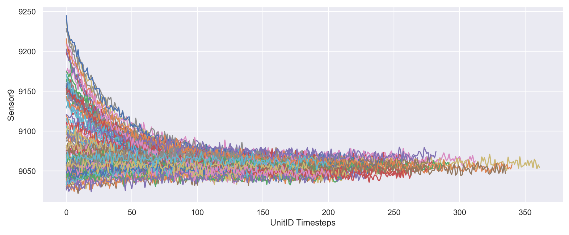


Figure 2.12: Overview of Sensor 9 for the complete FD001 C-MAPSS dataset.

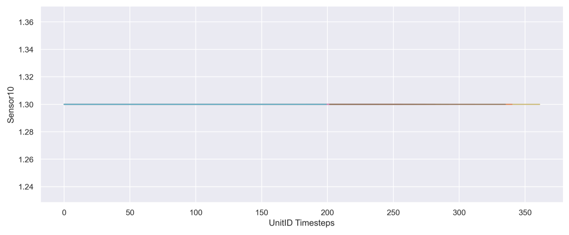


Figure 2.13: Overview of Sensor 10 for the complete FD001 C-MAPSS dataset.

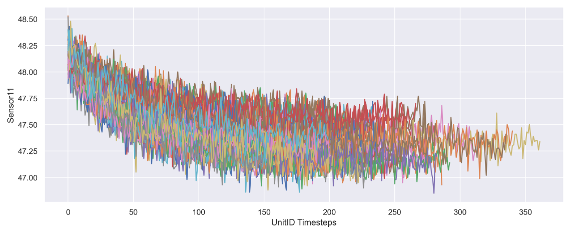


Figure 2.14: Overview of Sensor 11 for the complete FD001 C-MAPSS dataset.

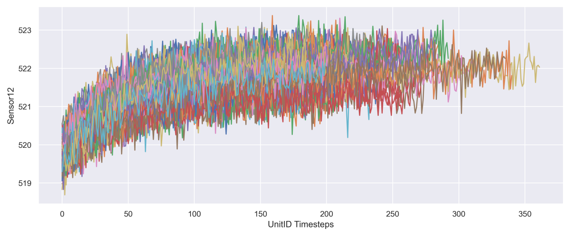


Figure 2.15: Overview of Sensor 12 for the complete FD001 C-MAPSS dataset.

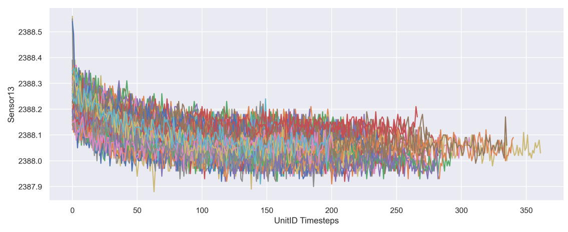


Figure 2.16: Overview of Sensor 13 for the complete FD001 C-MAPSS dataset.

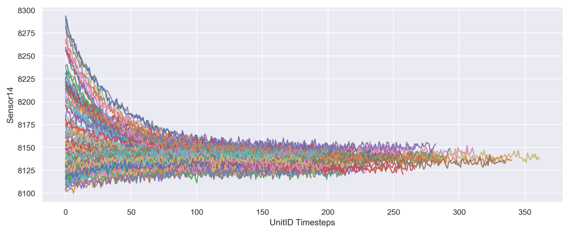


Figure 2.17: Overview of Sensor 14 for the complete FD001 C-MAPSS dataset.

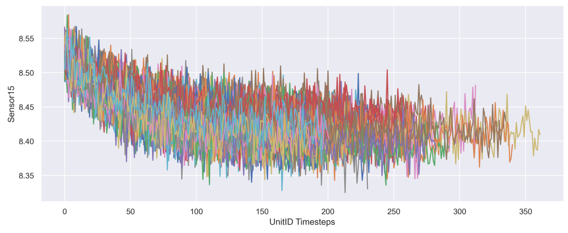


Figure 2.18: Overview of Sensor 15 for the complete FD001 C-MAPSS dataset.

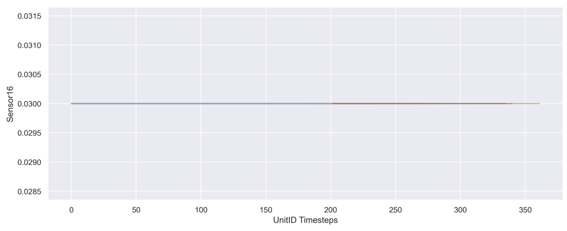


Figure 2.19: Overview of Sensor 16 for the complete FD001 C-MAPSS dataset.

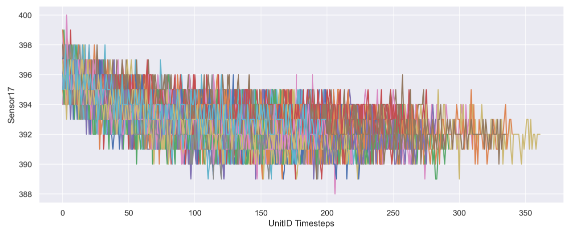


Figure 2.20: Overview of Sensor 17 for the complete FD001 C-MAPSS dataset.

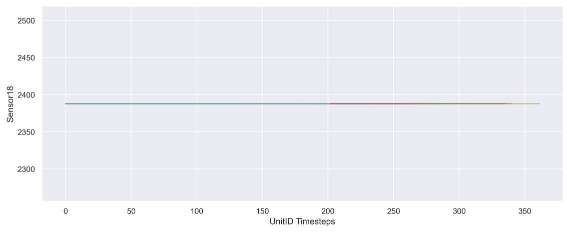


Figure 2.21: Overview of Sensor 18 for the complete FD001 C-MAPSS dataset.

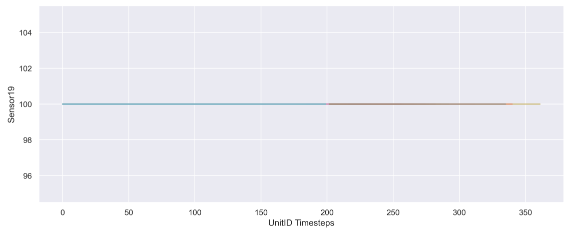


Figure 2.22: Overview of Sensor 19 for the complete FD001 C-MAPSS dataset.

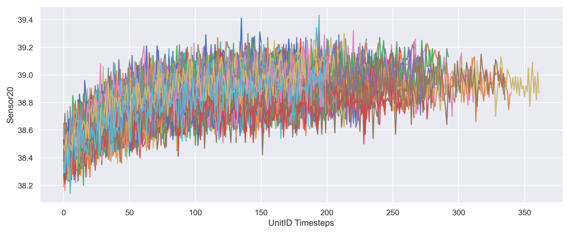


Figure 2.23: Overview of Sensor 20 for the complete FD001 C-MAPSS dataset.

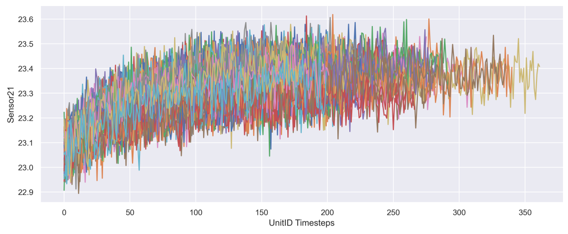


Figure 2.24: Overview of Sensor 21 for the complete FD001 C-MAPSS dataset.

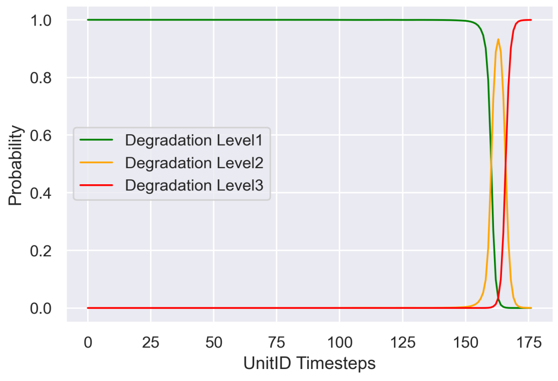


Figure 2.25: Degradation level probabilities UnitID for window configuration: $T_0 = 10, T_1 = 20$.

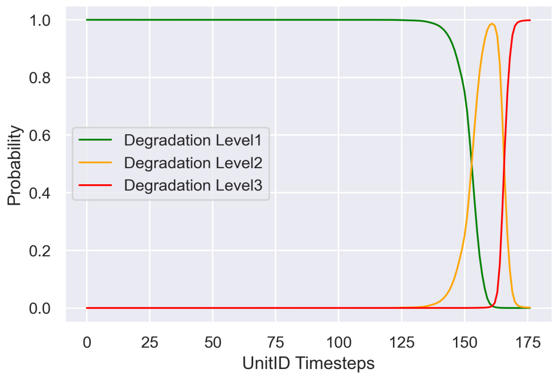


Figure 2.26: Degradation level probabilities UnitID for window configuration: $T_0 = 10, T_1 = 30$.

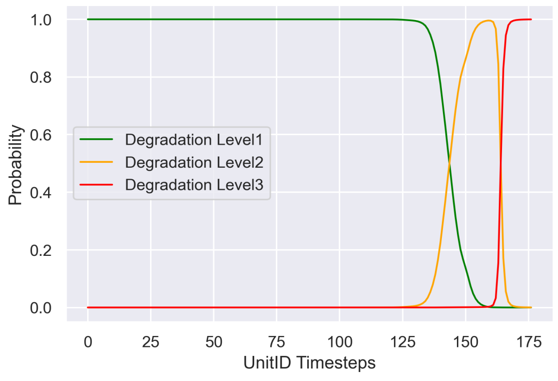


Figure 2.27: Degradation level probabilities UnitID for window configuration: $T_0 = 10, T_1 = 40$.

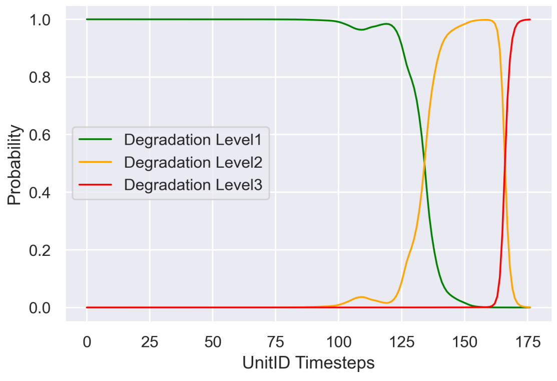


Figure 2.28: Degradation level probabilities UnitID for window configuration: $T_0 = 10, T_1 = 50$.

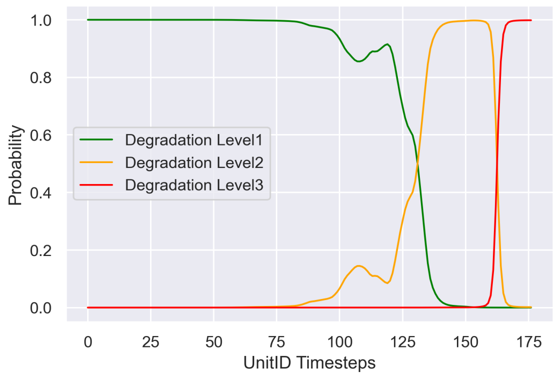


Figure 2.29: Degradation level probabilities UnitID for window configuration: $T_0 = 10, T_1 = 60$.

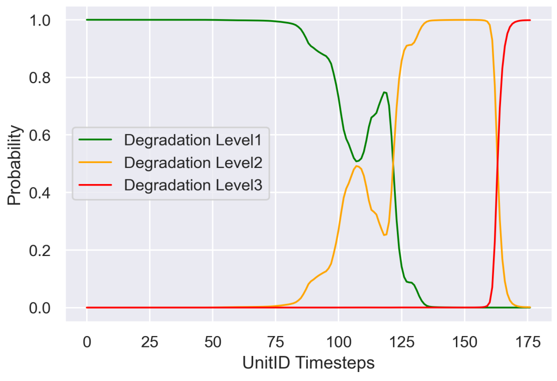


Figure 2.30: Degradation level probabilities UnitID for window configuration: $T_0 = 10, T_1 = 70$.

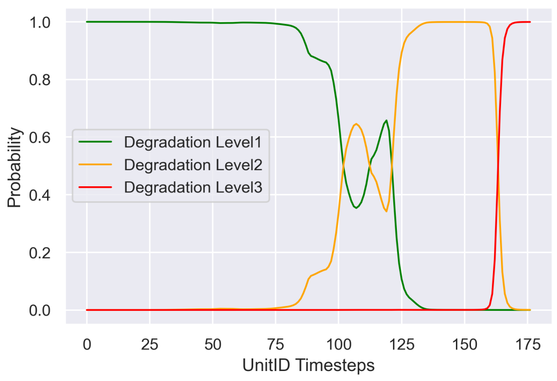


Figure 2.31: Degradation level probabilities UnitID for window configuration: $T_0 = 10, T_1 = 80$.

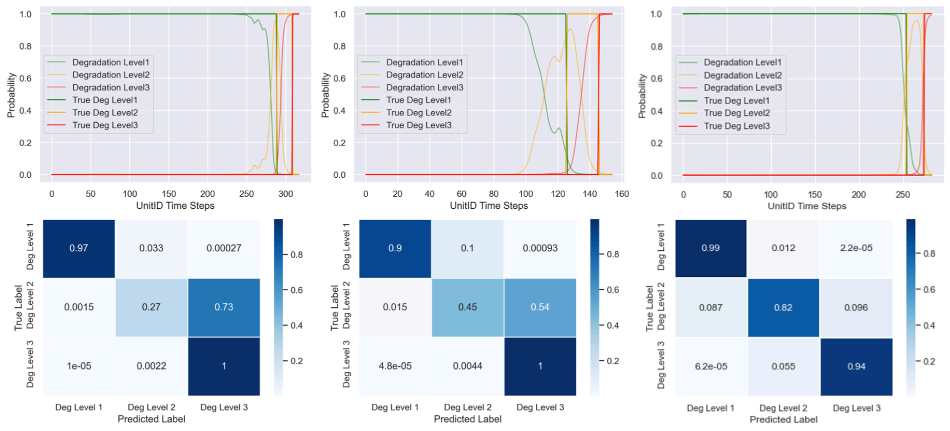


Figure 2.32: Illustrative example of moderately predicted prognostics, window configuration: $T_0 = 10, T_1 = 30$.

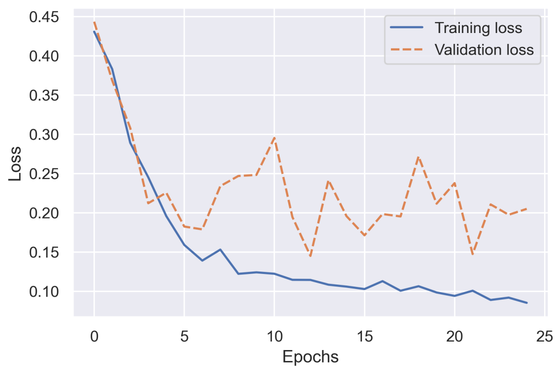


Figure 2.33: Training and validation loss graph for window configuration: $T_0 = 10, T_1 = 30$.

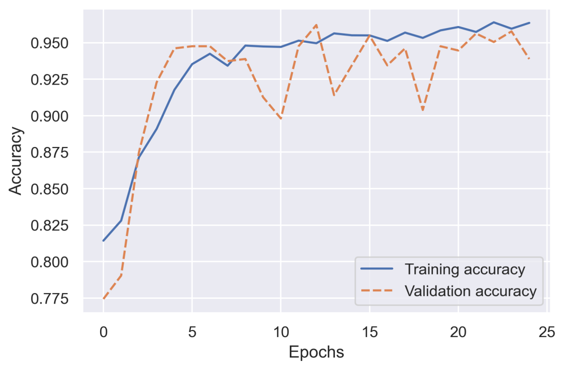


Figure 2.34: Training and validation accuracy graph for window configuration: $T_0 = 10, T_1 = 30$.

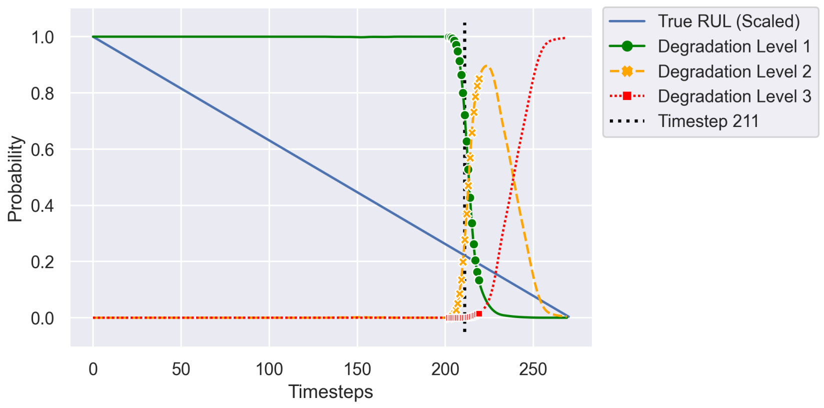


Figure 2.35: Degradation level probabilities for one specific UnitID. Visible is that the data points are coarse for all degradation levels.

2.3.2. Training and Validation

The training and validation loss for configuration 2 is depicted in Figure 2.33. The loss in training and validation indicates how well the model fits the training and the new data. The training and validation accuracy for configuration 2 is depicted in Figure 2.34.

2.3.3. Component Overview

An overview of different components can be seen in Figure 2.36, which shows a clear difference in prognostics and true engine lifetime. An example of the degradation level probabilities for a given UnitID is shown in Figure 2.35, which shows that the data points are relatively coarse.

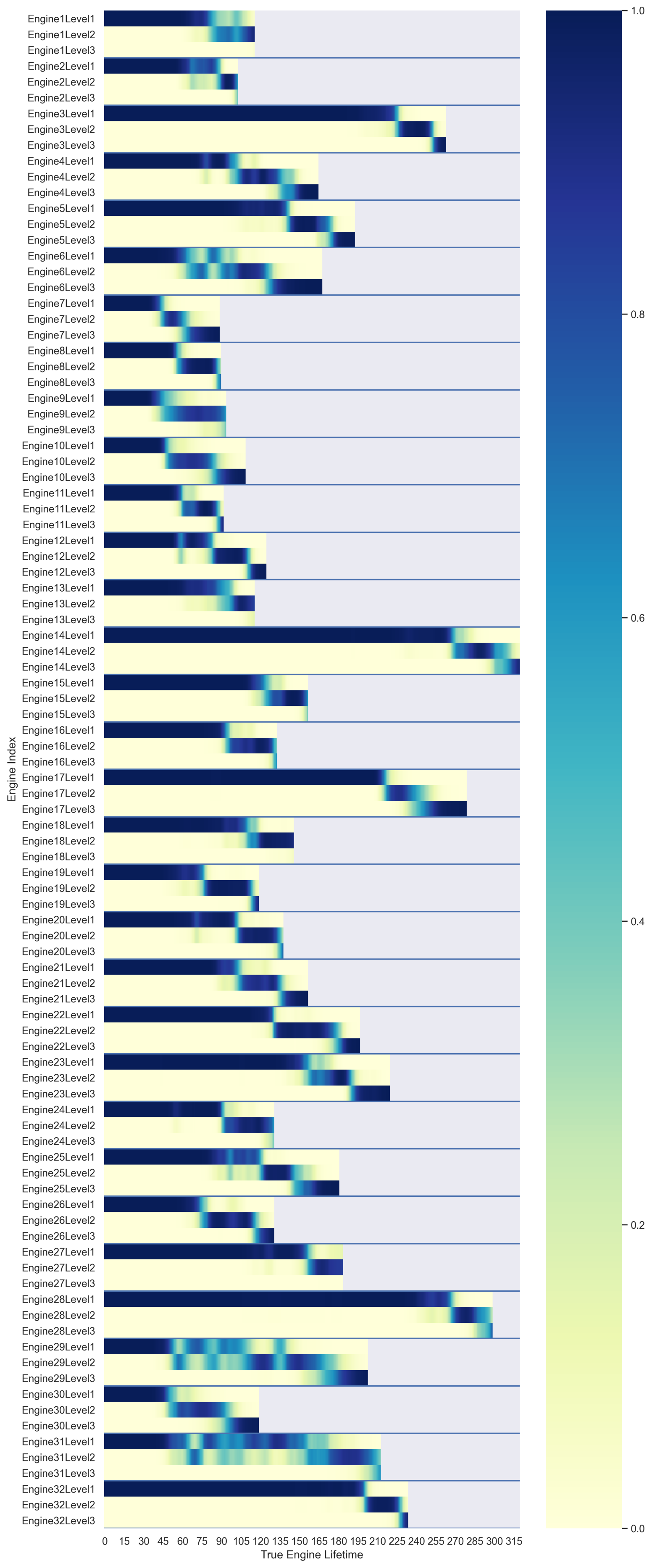


Figure 2.36: Overview of degradation level probabilities for several components.

3 Sensitivity Analysis

This chapter provides a brief description of the experiments conducted on local sensitivity. As discussed in the scientific paper, the parameters have a strong influence on the performance of the PdM framework. The conducted experiments are presented in Table 3.1. The separate sections for each varied parameter contain the reward development for a simulation of 200 episodes and the extended results table.

Experiment	*Analysis*
Exp 1	The effect of using different learning rates (α)
Exp 2	The effect of using different discount factors (γ)
Exp 3	The effect of using different corrective maintenance factors (θ_{cm})
Exp 4	The effect of using different opportunistic maintenance factors (ϵ_{om})
Exp 5	The effect of using different smoothing factors (α_s)
Exp 6	The effect of varying the Time-Based threshold (T_a)

Table 3.1: Overview of experiments performed within scientific paper.

3.1. Learning Rate (α)

Figure 3.1 shows the reward development for all varied learning rates α. The learning rate determines how much the network weights are updated and thus affects the convergence behavior. It can be seen that $\alpha = 0.001$, indicated in orange, obtains the best performance. The simulation with $\alpha = 0.01$, indicated in blue, shows an unstable slow convergence. While the simulation with $\alpha = 0.0001$, indicated in green, shows a slow learning behavior in the first 25 episodes. The results of the relevant Key Performance Indicators (KPIs) are given in Figure 3.2. The main conclusion is that the average reward for both 50 and 200 episodes is the lowest for $\alpha = 0.001$, the final chosen value for both scenarios in the scientific paper.

3.2. Discount Factor (γ)

Figure 3.3 shows the reward development for all varied discount factors γ. The discount factor determines how much the agent takes into account immediate and future rewards. It can be seen low discount factors trigger unstable behavior, indicated by the large negative reward values at the beginning of the episodes. Moreover, higher discount values show more stable results with relatively smaller peaks. The difference in agents' behavior can be explained by the fact that when the discount factor is high, the agent takes more account of future rewards and therefore replaces the engines sooner. The results of the relevant KPIs are given in Figure 3.4. The most important conclusion is the trend indicated by the corrective maintenance actions, component utilization, and the

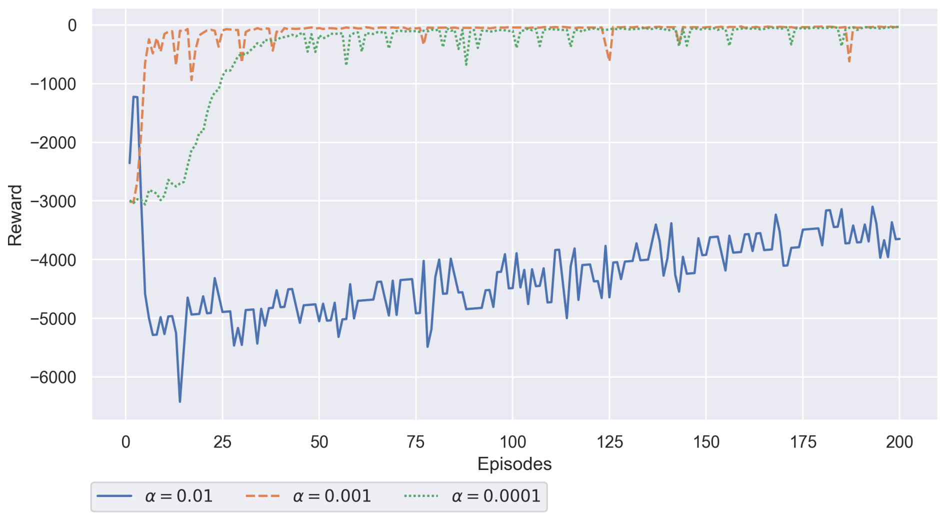
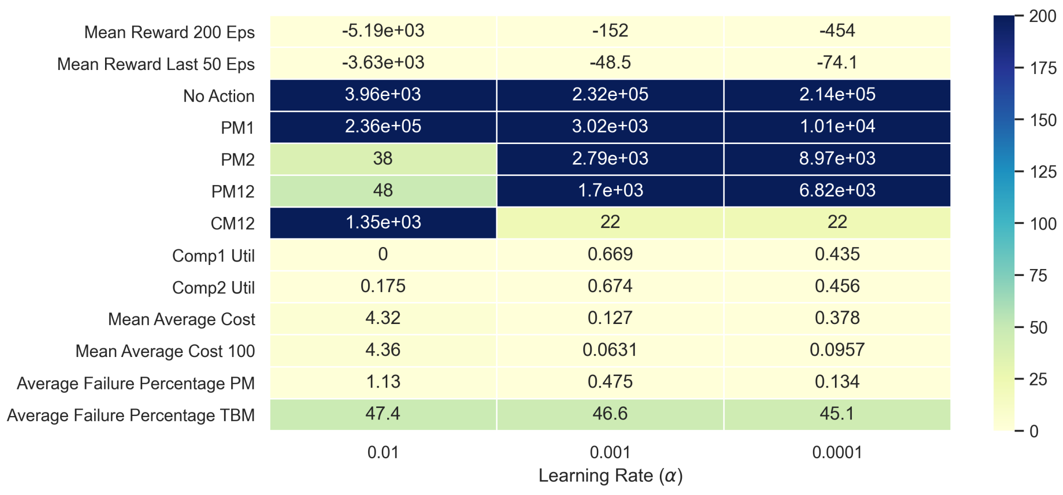


Figure 3.1: Reward development for $N_e = 200$ and $N_s = 1$ during local sensitivity analysis for the learning rate (α).



Learning Rate (α)	0.01	0.001	0.0001
Mean Reward 200 Eps	-5.19e+03	-152	-454
Mean Reward Last 50 Eps	-3.63e+03	-48.5	-74.1
No Action	3.96e+03	2.32e+05	2.14e+05
PM1	2.36e+05	3.02e+03	1.01e+04
PM2	38	2.79e+03	8.97e+03
PM12	48	1.7e+03	6.82e+03
CM12	1.35e+03	22	22
Comp1 Util	0	0.669	0.435
Comp2 Util	0.175	0.674	0.456
Mean Average Cost	4.32	0.127	0.378
Mean Average Cost 100	4.36	0.0631	0.0957
Average Failure Percentage PM	1.13	0.475	0.134
Average Failure Percentage TBM	47.4	46.6	45.1

Figure 3.2: Overview of relevant results for the local sensitivity analysis for the learning rate (α). $N_{eps} = 200$ and $N_s = 1$ during testing for every single learning rate (α).

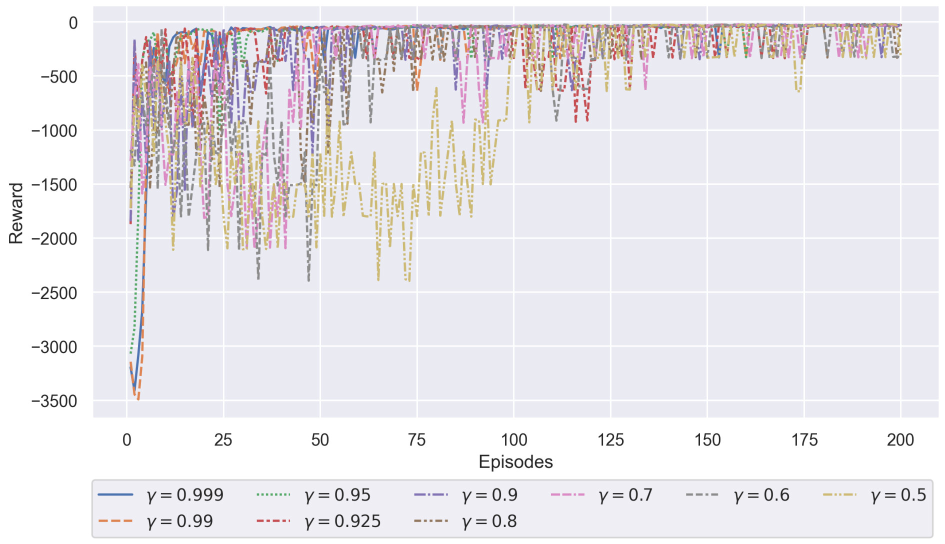
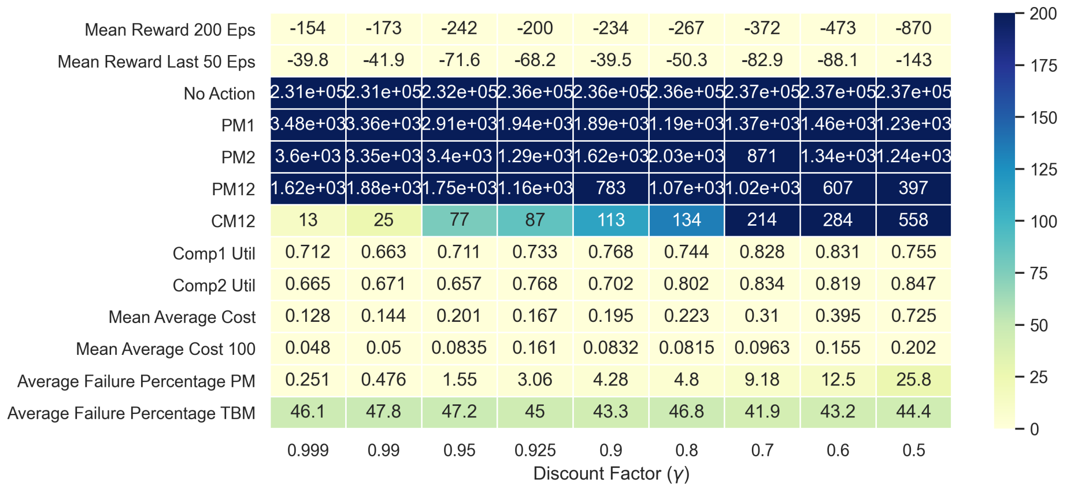


Figure 3.3: Reward development for $N_e = 200$ and $N_s = 1$ during local sensitivity analysis for the discount factor (γ).



Discount Factor (γ)	0.999	0.99	0.95	0.925	0.9	0.8	0.7	0.6	0.5
Mean Reward 200 Eps	-154	-173	-242	-200	-234	-267	-372	-473	-870
Mean Reward Last 50 Eps	-39.8	-41.9	-71.6	-68.2	-39.5	-50.3	-82.9	-88.1	-143
No Action	2.31e+05	2.31e+05	2.32e+05	2.36e+05	2.36e+05	2.36e+05	2.37e+05	2.37e+05	2.37e+05
PM1	3.48e+03	3.36e+03	2.91e+03	1.94e+03	1.89e+03	1.19e+03	1.37e+03	1.46e+03	1.23e+03
PM2	3.6e+03	3.35e+03	3.4e+03	1.29e+03	1.62e+03	2.03e+03	871	1.34e+03	1.24e+03
PM12	1.62e+03	1.88e+03	1.75e+03	1.16e+03	783	1.07e+03	1.02e+03	607	397
CM12	13	25	77	87	113	134	214	284	558
Comp1 Util	0.712	0.663	0.711	0.733	0.768	0.744	0.828	0.831	0.755
Comp2 Util	0.665	0.671	0.657	0.768	0.702	0.802	0.834	0.819	0.847
Mean Average Cost	0.128	0.144	0.201	0.167	0.195	0.223	0.31	0.395	0.725
Mean Average Cost 100	0.048	0.05	0.0835	0.161	0.0832	0.0815	0.0963	0.155	0.202
Average Failure Percentage PM	0.251	0.476	1.55	3.06	4.28	4.8	9.18	12.5	25.8
Average Failure Percentage TBM	46.1	47.8	47.2	45	43.3	46.8	41.9	43.2	44.4

Figure 3.4: Overview of relevant results for the local sensitivity analysis for the discount factor (γ). $N_{eps} = 200$ and $N_s = 1$ during testing for every single discount factor (γ).

average reward. When $\gamma = 0.999$, the agent is considering future rewards and replaces the engines in anticipation of failure. When $\gamma = 0.5$, the agent runs the engines longer and obtains a higher utilization rate of the components, but also higher average maintenance costs. The value $\gamma = 0.999$ is used in the first scenario in the scientific paper to obtain the lowest average long-term maintenance cost. In the second scenario, $\gamma = 0.7$ is used to improve the performance of other KPIs while keeping maintenance costs low.

3.3. Corrective Maintenance Factor (θ_{cm})

Figure 3.5 shows the reward development for all varied corrective maintenance factors θ_{cm}. The corrective maintenance factor determines how heavily the corrective maintenance activity is penalized. The greater the value of θ_{cm}, the larger the penalty becomes. In Figure 3.5 can be seen that high corrective maintenance factors trigger large negative rewards, shown by the brown dotted line. The results of the relevant KPIs are given in Figure 3.6. The main conclusion is that the average reward is more negative for larger θ_{cm}. However, only slightly penalizing corrective maintenance,

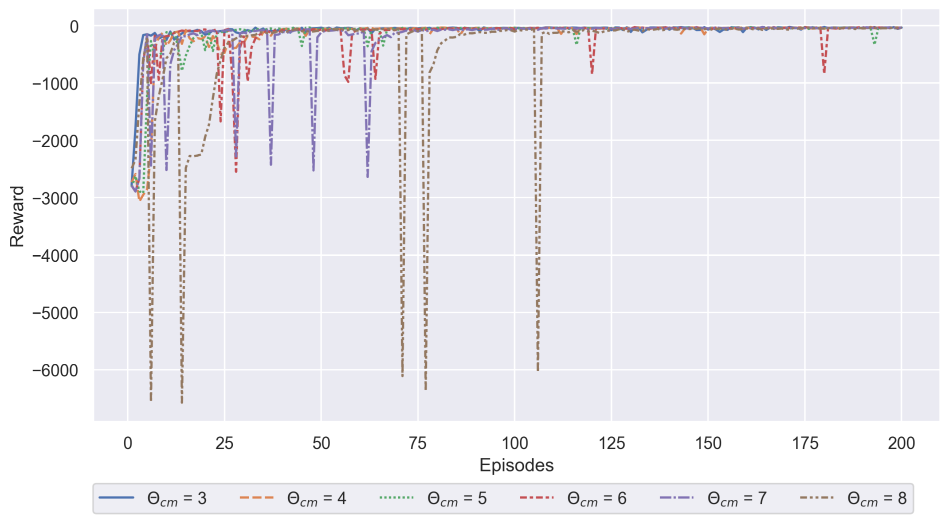
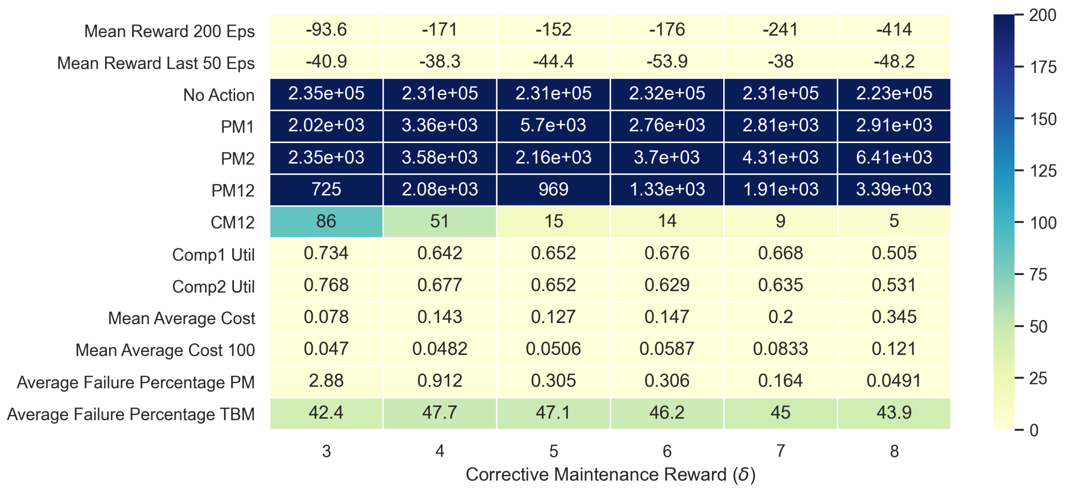


Figure 3.5: Reward development for $N_e = 200$ and $N_s = 1$ during local sensitivity analysis for the corrective maintenance factor (θ_{cm}).



Corrective Maintenance Reward (δ)	3	4	5	6	7	8
Mean Reward 200 Eps	-93.6	-171	-152	-176	-241	-414
Mean Reward Last 50 Eps	-40.9	-38.3	-44.4	-53.9	-38	-48.2
No Action	2.35e+05	2.31e+05	2.31e+05	2.32e+05	2.31e+05	2.23e+05
PM1	2.02e+03	3.36e+03	5.7e+03	2.76e+03	2.81e+03	2.91e+03
PM2	2.35e+03	3.58e+03	2.16e+03	3.7e+03	4.31e+03	6.41e+03
PM12	725	2.08e+03	969	1.33e+03	1.91e+03	3.39e+03
CM12	86	51	15	14	9	5
Comp1 Util	0.734	0.642	0.652	0.676	0.668	0.505
Comp2 Util	0.768	0.677	0.652	0.629	0.635	0.531
Mean Average Cost	0.078	0.143	0.127	0.147	0.2	0.345
Mean Average Cost 100	0.047	0.0482	0.0506	0.0587	0.0833	0.121
Average Failure Percentage PM	2.88	0.912	0.305	0.306	0.164	0.0491
Average Failure Percentage TBM	42.4	47.7	47.1	46.2	45	43.9

Figure 3.6: Overview of relevant results for the local sensitivity analysis for the corrective maintenance factor (θ_{cm}). $N_{eps} = 200$ and $N_s = 1$ during testing for every single corrective maintenance factor (θ_{cm}).

with $\theta_{cm} = 3$, yields a higher number of corrective maintenance actions.

3.4. Opportunistic Maintenance Factor (ϵ_{om})

Figure 3.7 shows the reward development for all varied opportunistic maintenance factors ϵ_{om}. The opportunistic maintenance factor determines how much economic benefit the opportunistic maintenance action provides. It can be seen that the opportunistic maintenance does not have a significant impact on the reward development. The results of the relevant KPIs are given in Figure 3.8. The main conclusion is that the average reward remains about the same, but the maintenance cost and the number of opportunistic maintenance actions increase when $\epsilon_{om} = 0.8$.

3.5. Smoothing Factor (α_s)

Figure 3.9 shows the results of varying the smoothing factor α_s. When $\alpha_s = 1.0$, no sensor smoothing is applied, which is visible in the degradation level prognostics. The probabilities of the degradation

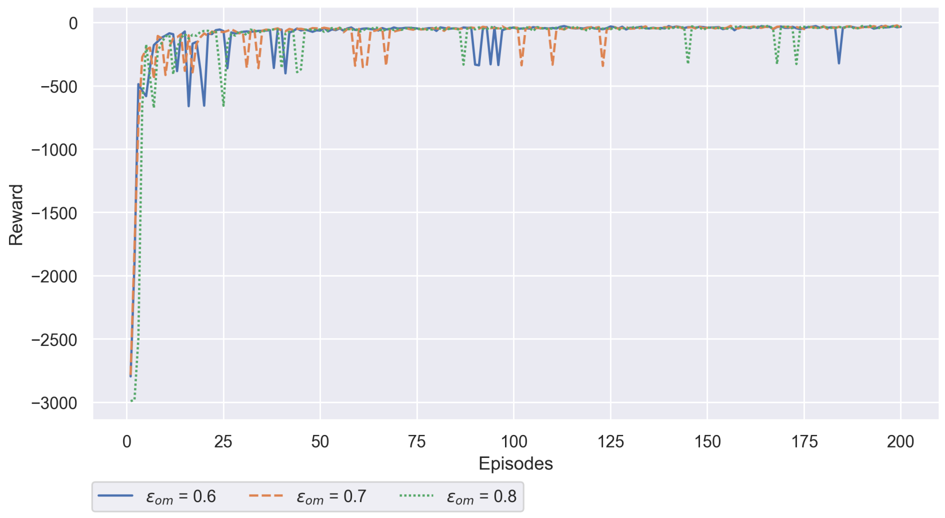
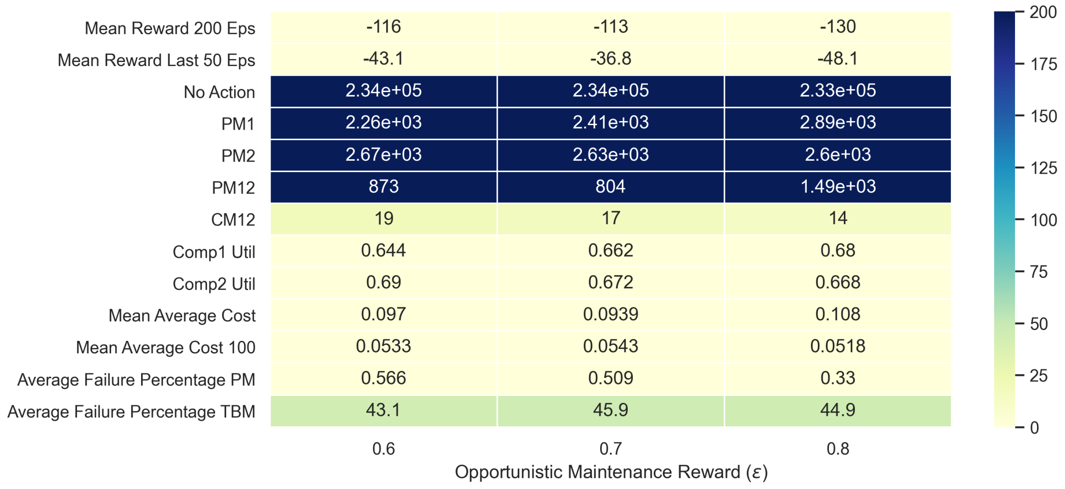


Figure 3.7: Reward development for $N_e = 200$ and $N_s = 1$ during local sensitivity analysis for the opportunistic maintenance factor (ϵ_{om}).



	0.6	0.7	0.8
Mean Reward 200 Eps	-116	-113	-130
Mean Reward Last 50 Eps	-43.1	-36.8	-48.1
No Action	2.34e+05	2.34e+05	2.33e+05
PM1	2.26e+03	2.41e+03	2.89e+03
PM2	2.67e+03	2.63e+03	2.6e+03
PM12	873	804	1.49e+03
CM12	19	17	14
Comp1 Util	0.644	0.662	0.68
Comp2 Util	0.69	0.672	0.668
Mean Average Cost	0.097	0.0939	0.108
Mean Average Cost 100	0.0533	0.0543	0.0518
Average Failure Percentage PM	0.566	0.509	0.33
Average Failure Percentage TBM	43.1	45.9	44.9

Figure 3.8: Overview of relevant results for the local sensitivity analysis for the opportunistic maintenance factor (ϵ_{om}). $N_{eps} = 200$ and $N_s = 1$ during testing for every single opportunistic maintenance factor (ϵ_{om}).

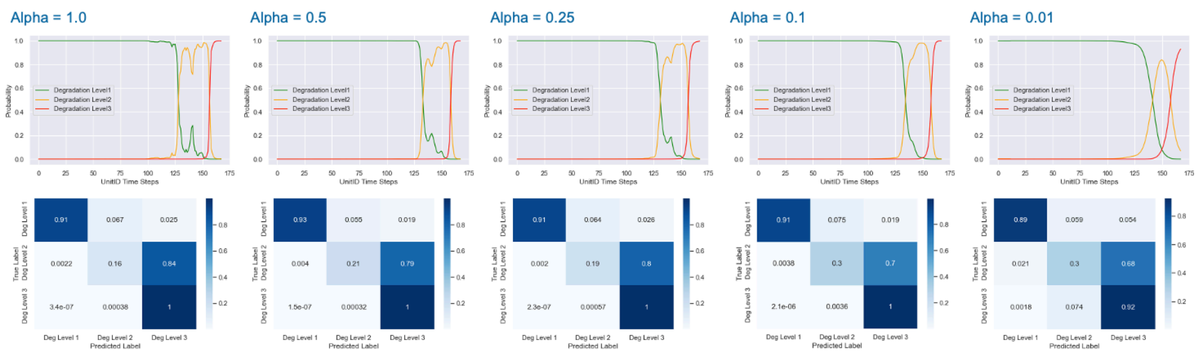
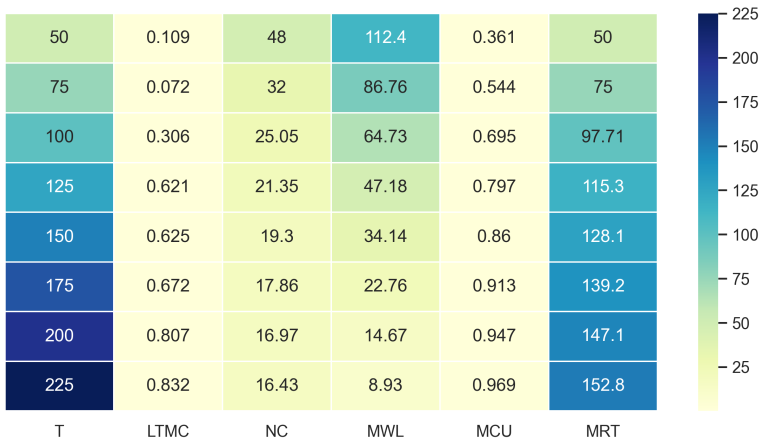


Figure 3.9: Overview of the local sensitivity analysis of the smoothing factor (α_s), including the accuracy.

levels contain noise and are not characterized by smooth lines. Decreasing the factor α_s shows that the signal becomes smoother. For example, when $\alpha_s = 0.1$, one can observe that the degradation levels reflect the degradation behavior and consist of smooth lines. However, $\alpha_s = 0.01$ shows that the prognostics are not representing the original degradation behavior anymore and should therefore be avoided. The peak of degradation level 2 becomes low and the predictions for degradation level 3 are inaccurate. The final chosen parameter for both scenarios is $\alpha_s = 0.1$.

3.6. Time-Based Threshold (T_a)

As described in the scientific paper, the threshold T_a has a strong influence on the results for the time-based maintenance policy. Figure 3.10 shows the relevant results for varying the time-based threshold T_a. The acronyms in the heatmap represent the following terms: T = time-based threshold T_a, LTMC = long-term maintenance cost, NC = number of replaced components, MWL = mean wasted component lifetime, MCU = mean component utilization, and MRT = mean replacement time. As can be seen in Figure 3.10, long-term maintenance costs increase as T_a becomes larger, which is caused by the fact the policy experiences corrective maintenance more frequent. As expected, the number of components and the average wasted component lifetime decrease as T_a increases. The component utilization and replacement time are increasing, which is caused by the time-based threshold becoming larger. Figure 3.11 depicts the number of components used per episode. The shaded area indicates the *(minimum-maximum)* values from Figure 3.10 and shows a large spread. The shaded area is the difference between the lowest and highest value for the TBM policy. Figure 3.12 depicts the average component utilization, Figure 3.13 shows the average wasted component lifetime, and Figure 3.14 indicates the average replacement time.



T	LTMC	NC	MWL	MCU	MRT
50	0.109	48	112.4	0.361	50
75	0.072	32	86.76	0.544	75
100	0.306	25.05	64.73	0.695	97.71
125	0.621	21.35	47.18	0.797	115.3
150	0.625	19.3	34.14	0.86	128.1
175	0.672	17.86	22.76	0.913	139.2
200	0.807	16.97	14.67	0.947	147.1
225	0.832	16.43	8.93	0.969	152.8

Figure 3.10: Heatmap for the relevant KPIs in the T_a analysis, where $T_a \in \{50-225\}$.

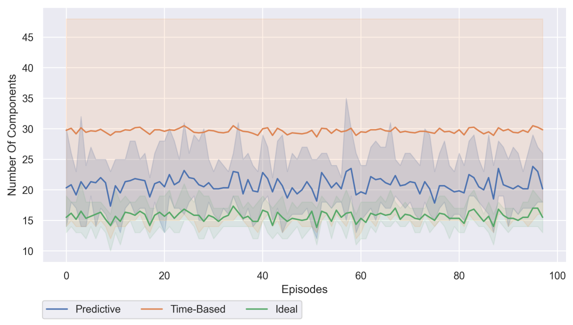


Figure 3.11: Number of components for several T_a values, indicating the *(Min-Max)* values.

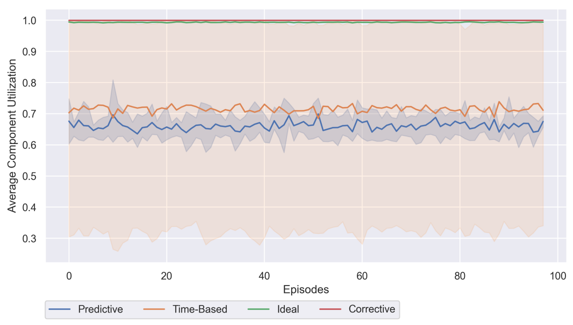


Figure 3.12: Average component utilization for several T_a values, indicating the *(Min-Max)* values.

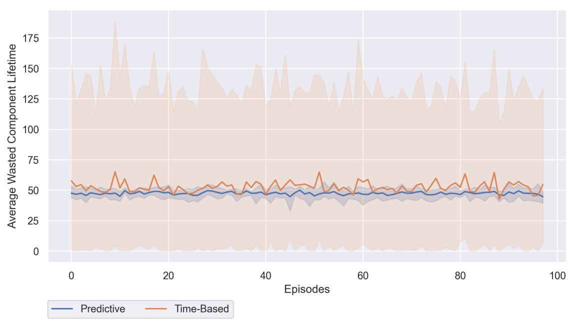


Figure 3.13: Average wasted component lifetime for several T_a values, indicating the *(Min-Max)* values.

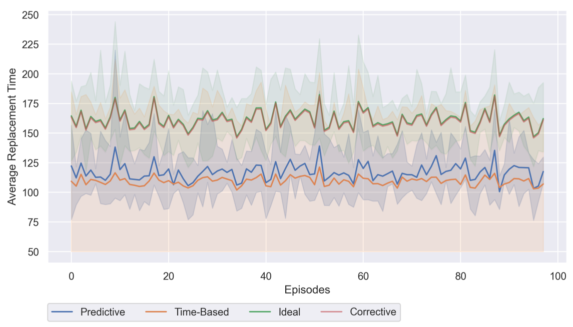


Figure 3.14: Average replacement lifetime for several T_a values, indicating the *(Min-Max)* values.

4 Verification and Validation

This chapter contains the verification and validation of this study. The verification part verifies that the model is implemented correctly and does the right things. Then the validation part verifies that the model is consistent and works within a satisfactory accuracy range. We occasionally refer to the figures in the previous chapters to avoid reproducing the same figure twice in this report. The model verification is described in Section 4.1, followed by the model validation in Section 4.2.

4.1. Model Verification

The verification approach is done by analyzing the agent's behavior. We compared the agent's decisions with common knowledge and determined that the agent is following a feasible approach. Model verification is described by comparing the model's behavior to "common knowledge" behavior and answering the question: does it work *as expected*?

Our approach is to analyze different episodes to see if the agent performs the expected maintenance actions based on the state space. We analyzed several episodes, as shown in Figure 1.5 and Figure 1.25. During the episode, the agent makes the appropriate decisions based on the probabilities of the degradation levels. If degradation level 2 increases, the agent replaces the engines preventively. However, if the engine reaches its maximum lifetime, the corrective maintenance activity is performed correctly, as can be seen in Figure 1.7. The component utilization rates are depicted in Figure 1.6. The results shown verify the correct working principle of the framework as it does what is expected of the agent. Not only does the agent make the decisions correctly, but the framework works well in selecting random engines, as shown by the different distributions in various figures, such as Figure 1.11. In addition, the framework exhibits the expected behavior in terms of corrective maintenance: when the engine reaches its maximum lifetime, it is replaced with a new one. A second verification of the model decisions is given in Figure 1.25, applicable to scenario 2. It can be verified that the agent behaves differently depending on the setting of the discount factor. When the discount factor is reduced, from $\gamma = 0.999$ to $\gamma = 0.7$, it was expected and verified that the agent obtains a higher component utilization, but also higher maintenance costs. Another example is provided in Part I, in which the opportunistic maintenance actions are shown. The model also behaves as expected regarding the component utilization development given in Figure 1.15 and Figure 1.17. Moreover, varying the discount factor exhibits the expected behavior for the opportunistic maintenance action as it almost reaches the value 2, shown in Figure 1.32.

Moreover, the approach to verify the implementation of the code is performed by unit testing, debugging, and total testing. These different aspects are done during the research by separating a

KPI	*Unit 1*	*Unit 2*	*Unit 3*	*Unit 4*	***Average***
True Lifetime	100	140	200	300	**Average: 185 LT**
Number Components	1	2	3	4	**Total: 4 Components**
Maintenance Action	Cor	Cor	Prev	Prev	**Cost: 153/El**
Component Utilization	1	1	0.75	0.5	**0.8125**
Wasted Lifetime	0	0	50	150	**50**
Replacement Time	100	140	150	150	**135**

Table 4.1: Example of calculation of key performance indicators (KPIs) for the Time-Based maintenance policy.

piece of code and experiments to check if the code works as expected. In addition, analyzing the inputs and outputs, in this case the agent's behavior, ensures the correctness of the code. An example is the analysis of the discount factor γ, shown in Figure 3.3 and Figure 3.4, where we have seen that the model exhibits the expected behavior. When the discount factor is reduced, the agents' policy becomes more myopic and higher component utilization and replacement time are achieved. Moreover, the verification of the learning rate α, shown in Figure 3.1 and Figure 3.2, guarantees that the Deep Q-Learning algorithm has been correctly implemented. The results of all the sensitivity analyses, including the maintenance factors θ_{cm} & ϵ_{om}, can be found in Chapter 3.
Finally, the results of varying the window boundaries, described in subsection 2.3.1, show the expected model behavior. As the window bounds are increased, the probabilities of degradation for the middle window become higher and wider. In addition, an example of a proposed replacement action and corresponding degradation level probabilities is given in Figure 1.22, which shows expected behavior by replacing the component at a high degradation level 2.

4.1.1. Key Performance Indicator (KPI) Example

Table 4.1 contains an example calculation for the Time-Based maintenance policy assuming an average of $T_a = 150$ cycles. In this example, we show the calculation of the defined KPIs.

4.2. Model Validation

We validated the model by running multiple simulations and verifying that the agent exhibited consistent behavior. As expected, the agent's behavior is consistent across multiple runs, as evidenced by similar experimental results. To analyze the model validation, we conducted several simulations with different initial parameters. The results for 10 simulations are shown in Figure 4.1. The figure shows the minimum, maximum, and average values for the reward obtained by the agent. The distribution of values shows that the model is accurate and can handle different initial parameters. Moreover, the number of components for 10 simulations is depicted in Figure 4.2, which exhibits the same converging behavior. The distribution of the agents' actions is given in Figure 4.3. The figure shows that the actions are distributed in a consistent manner and thus indicates high accuracy of the model during various simulations.

Overall, the results obtained are logical when compared to the other maintenance policies. The original purpose of the framework is to replace components before they fail, which is done by using the probability of their degradation levels. Compared to the Time-Based maintenance policy, the PdM framework has proven to work and be useful, serving the original purpose of cost reduction. Compared to the Ideal maintenance policy, the agent achieves competitive costs in the first scenario. This validates the model in terms of purpose and accuracy. Since the PdM framework only considers engines and does not follow a true maintenance policy, which includes multiple components, it is difficult to compare with actual real-world performance.

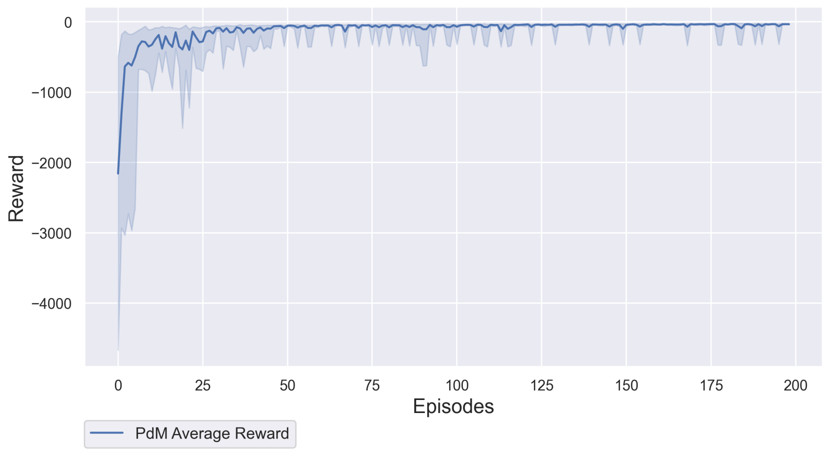


Figure 4.1: Reward development for $N_e = 200$ and $N_s = 10$ during training.

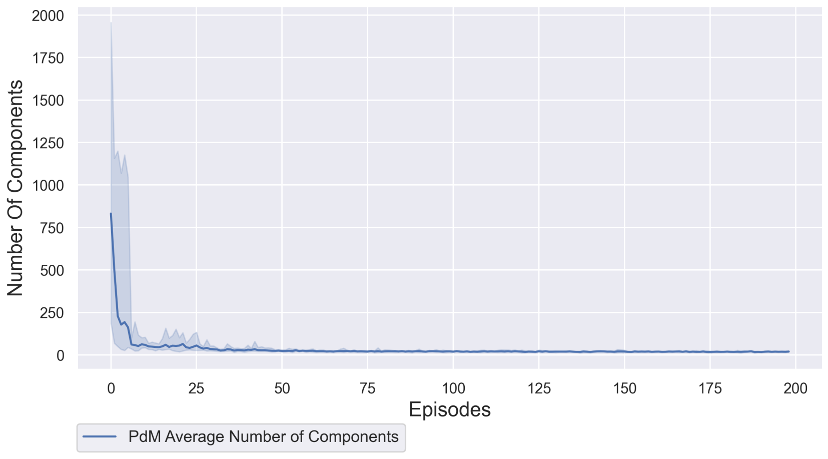


Figure 4.2: Number of components for $N_e = 200$ and $N_s = 10$ during training.

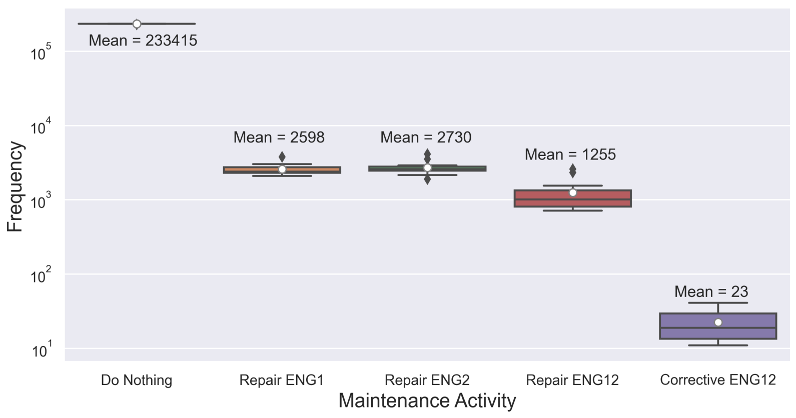


Figure 4.3: Maintenance activity analysis for $N_e = 200$ and $N_s = 10$ during training. Note: y-axis uses log scale.

Bibliography

[1] Introduction to Maintenance in Production Systems. In *Maintenance for Industrial Systems*, pages 65–85. Springer London, London, 2010. ISBN 978-1-84882-575-8. doi: 10.1007/978-1-84882-575-8{_}4. URL https://doi.org/10.1007/978-1-84882-575-8_4.

[2] Shannon P Ackert. *Basics of Aircraft Maintenance Programs for Financiers*. 2010. URL http://aircraftmonitor.com/uploads/1/5/9/9/15993320/basics_of_aircraft_maintenance_programs_for_financiers___v1.pdf.

[3] Joachim Arts and Rob Basten. Design of multi-component periodic maintenance programs with single-component models. *IISE Transactions*, 50(7):606–615, 2018. ISSN 24725862. doi: 10.1080/24725854.2018.1437301.

[4] Roy Assaf. Prognostics and health management for JSF program. *Sound and Vibration*, 38(5): 8, 2004. ISSN 15410161.

[5] Beste Basciftci, Shabbir Ahmed, Nagi Z. Gebraeel, and Murat Yildirim. Stochastic optimization of maintenance and operations schedules under unexpected failures. *IEEE Transactions on Power Systems*, 33(6):6755–6765, 2018. ISSN 08858950. doi: 10.1109/TPWRS.2018.2829175.

[6] Jose R. Celaya, Abhinav Saxena, Sankalita Saha, and Kai F. Goebel. Prognostics of power mosfets under thermal stress accelerated aging using data-driven and model-based methodologies. *Proceedings of the Annual Conference of the Prognostics and Health Management Society 2011, PHM 2011*, (July 2016):443–452, 2014.

[7] Paulo Roberto de Oliveira da Costa, Alp Akçay, Yingqian Zhang, and Uzay Kaymak. Remaining useful lifetime prediction via deep domain adaptation. *Reliability Engineering and System Safety*, 195(August 2019):106682, 2020. ISSN 09518320. doi: 10.1016/j.ress.2019.106682. URL https://doi.org/10.1016/j.ress.2019.106682.

[8] Matthew J. Daigle and Kai Goebel. A model-based prognostics approach applied to pneumatic valves. *International Journal of Prognostics and Health Management*, 2(2), 2011. ISSN 21532648.

[9] Bram de Jonge and Philip A. Scarf. A review on maintenance optimization. *European Journal of Operational Research*, 285(3):805–824, 2020. ISSN 03772217. doi: 10.1016/j.ejor.2019.09.047.

[10] Qichen Deng, Bruno F. Santos, and Richard Curran. A practical dynamic programming based methodology for aircraft maintenance check scheduling optimization. *European Journal of Operational Research*, 281(2):256–273, 2020. ISSN 03772217. doi: 10.1016/j.ejor.2019.08.025.

[11] Phuc Do, Roy Assaf, Phil Scarf, and Benoit Iung. Modelling and application of condition-based maintenance for a two-component system with stochastic and economic dependencies. *Reliability Engineering and System Safety*, 182(January 2018):86–97, 2019. ISSN 09518320. doi: 10.1016/j.ress.2018.10.007. URL https://doi.org/10.1016/j.ress.2018.10.007.

[12] B. Einabadi, A. Baboli, and M. Ebrahimi. Dynamic Predictive Maintenance in industry 4.0 based on real time information: Case study in automotive industries. *IFAC-PapersOnLine*,

52(13):1069–1074, 2019. ISSN 24058963. doi: 10.1016/j.ifacol.2019.11.337. URL https://doi.org/10.1016/j.ifacol.2019.11.337.

[13] Wael Hafsa, Brigitte Chebel-Morello, Christophe Varnier, Kamal Medjaher, and Noureddine Zerhouni. Prognostics of health status of multi-component systems with degradation interactions. *Proceedings of 2015 International Conference on Industrial Engineering and Systems Management, IEEE IESM 2015*, (November):870–875, 2016. doi: 10.1109/IESM.2015.7380258.

[14] Martin Hinsch. *Industrial aviation management: A primer in European design, production and maintenance organisations*. 2018. ISBN 9783662547403. doi: 10.1007/978-3-662-54740-3.

[15] Yang Hu, Xuewen Miao, Jun Zhang, Jie Liu, and Ershun Pan. Reinforcement learning-driven maintenance strategy: A novel solution for long-term aircraft maintenance decision optimization. *Computers and Industrial Engineering*, 153(October 2020):107056, 2021. ISSN 03608352. doi: 10.1016/j.cie.2020.107056. URL https://doi.org/10.1016/j.cie.2020.107056.

[16] IATA. Airline Maintenance Cost: Executive Commentary Edition 2019 (FY2018 data). *URL: http://www. iata. org/whatwedo/ ...*, (January):1–16, 2019. URL http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Airline+Maintenance+Cost+Executive+Commentary#0.

[17] Andrew K.S. Jardine, Daming Lin, and Dragan Banjevic. A review on machinery diagnostics and prognostics implementing condition-based maintenance. *Mechanical Systems and Signal Processing*, 20(7):1483–1510, 2006. ISSN 08883270. doi: 10.1016/j.ymssp.2005.09.012.

[18] Georgia Ann Klutke, Peter C. Kiessler, and M. A. Wortman. A critical look at the bathtub curve. *IEEE Transactions on Reliability*, 52(1):125–129, 2003. ISSN 00189529. doi: 10.1109/TR.2002.804492.

[19] Radouane Laggoune, Alaa Chateauneuf, and Djamil Aissani. Impact of few failure data on the opportunistic replacement policy for multi-component systems. *Reliability Engineering and System Safety*, 95(2):108–119, 2010. ISSN 09518320. doi: 10.1016/j.ress.2009.08.007. URL http://dx.doi.org/10.1016/j.ress.2009.08.007.

[20] Juseong Lee and Mihaela Mitici. An integrated assessment of safety and efficiency of aircraft maintenance strategies using agent-based modelling and stochastic Petri nets. *Reliability Engineering and System Safety*, 202, 2020. ISSN 09518320. doi: 10.1016/j.ress.2020.107052.

[21] Xiang Li, Qian Ding, and Jian Qiao Sun. Remaining useful life estimation in prognostics using deep convolution neural networks. *Reliability Engineering and System Safety*, 172(November 2017):1–11, 2018. ISSN 09518320. doi: 10.1016/j.ress.2017.11.021. URL https://doi.org/10.1016/j.ress.2017.11.021.

[22] Yanhui Lin, Xudong Li, and Yang Hu. Deep diagnostics and prognostics: An integrated hierarchical learning framework in PHM applications. *Applied Soft Computing Journal*, 72:555–564, 2018. ISSN 15684946. doi: 10.1016/j.asoc.2018.01.036. URL https://doi.org/10.1016/j.asoc.2018.01.036.

[23] Qinming Liu, Ming Dong, F. F. Chen, Wenyuan Lv, and Chunming Ye. Single-machine-based joint optimization of predictive maintenance planning and production scheduling. *Robotics and Computer-Integrated Manufacturing*, 55(November 2017):173–182, 2019. ISSN 07365845. doi: 10.1016/j.rcim.2018.09.007.

[24] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015. ISSN 14764687. doi: 10.1038/nature14236. URL http://dx.doi.org/10.1038/nature14236.

[25] Eduardo F. Morales and Julio H. Zaragoza. An introduction to reinforcement learning. *Decision Theory Models for Applications in Artificial Intelligence: Concepts and Solutions*, pages 63–80, 2011. doi: 10.4018/978-1-60960-165-2.ch004.

[26] A K Muchiri and K Smit. Optimizing Aircraft Line Maintenance Through Task Re-Clustering and Interval De-Escalation. 2011.

[27] Khanh T.P. Nguyen and Kamal Medjaher. A new dynamic predictive maintenance framework using deep learning for failure prognostics. *Reliability Engineering and System Safety*, 188(February):251–262, 2019. ISSN 09518320. doi: 10.1016/j.ress.2019.03.018. URL https://doi.org/10.1016/j.ress.2019.03.018.

[28] Minou C.A. Olde Keizer, Simme Douwe P. Flapper, and Ruud H. Teunter. Condition-based maintenance policies for systems with multiple dependent components: A review. *European Journal of Operational Research*, 261(2):405–420, 2017. ISSN 03772217. doi: 10.1016/j.ejor.2017.02.044. URL http://dx.doi.org/10.1016/j.ejor.2017.02.044.

[29] K. G. Papakonstantinou and M. Shinozuka. Planning structural inspection and maintenance policies via dynamic programming and Markov processes. Part II: POMDP implementation. *Reliability Engineering and System Safety*, 130:214–224, 2014. ISSN 09518320. doi: 10.1016/j.ress.2014.04.006. URL http://dx.doi.org/10.1016/j.ress.2014.04.006.

[30] N. Papakostas, P. Papachatzakis, V. Xanthakis, D. Mourtzis, and G. Chryssolouris. An approach to operational aircraft maintenance planning. *Decision Support Systems*, 48(4):604–612, 2010. ISSN 01679236. doi: 10.1016/j.dss.2009.11.010. URL http://dx.doi.org/10.1016/j.dss.2009.11.010.

[31] Iwona Paprocka and Boena Skołud. A hybrid multi-objective immune algorithm for predictive and reactive scheduling. *Journal of Scheduling*, 20(2):165–182, 2017. ISSN 10946136. doi: 10.1007/s10951-016-0494-9.

[32] Joeri Poppe, Robert N. Boute, and Marc R. Lambrecht. A hybrid condition-based maintenance policy for continuously monitored components with two degradation thresholds. *European Journal of Operational Research*, 268(2):515–532, 2018. ISSN 03772217. doi: 10.1016/j.ejor.2018.01.039. URL https://doi.org/10.1016/j.ejor.2018.01.039.

[33] Seyed Mohammad Rezvanizaniani, Zongchang Liu, Yan Chen, and Jay Lee. Review and recent advances in battery health monitoring and prognostics technologies for electric vehicle (EV) safety and mobility. *Journal of Power Sources*, 256:110–124, 2014. ISSN 03787753. doi: 10.1016/j.jpowsour.2014.01.085. URL http://dx.doi.org/10.1016/j.jpowsour.2014.01.085.

[34] Anant Sahay. An overview of aircraft maintenance. *Leveraging Information Technology for Optimal Aircraft Maintenance, Repair and Overhaul (MRO)*, pages 1–230, 2012. doi: 10.1533/9780857091437.1.

[35] Nazmus Sakib and Thorsten Wuest. Challenges and Opportunities of Condition-based Predictive Maintenance: A Review. *Procedia CIRP*, 78:267–272, 2018. ISSN 22128271. doi: 10.1016/j.procir.2018.08.318. URL https://doi.org/10.1016/j.procir.2018.08.318.

[36] Abdulkadir Sarac, Rajan Batta, and Christopher M. Rump. A branch-and-price approach for operational aircraft maintenance routing. *European Journal of Operational Research*, 175(3): 1850–1869, 2006. ISSN 03772217. doi: 10.1016/j.ejor.2004.10.033.

[37] Abhinav Saxena, Kai Goebel, Don Simon, and Neil Eklund. Damage Propagation Modeling for Aircraft Engine Run-to-Failure Simulation, 2008. URL https://ti.arc.nasa.gov/tech/dash/groups/pcoe/prognostic-data-repository/.

[38] Caner Senturk and Ibrahim Ozkol. The effects of the use of single task-oriented maintenance concept and more accurate letter check alternatives on the reduction of scheduled maintenance downtime of aircraft. *International Journal of Mechanical Engineering and Robotics Research*, 7(2):189–196, 2018. ISSN 22780149. doi: 10.18178/ijmerr.7.2.189-196.

[39] Yuwei Shang, Wenchuan Wu, Jiawei Liao, Guo Jianbo, Jian Su, Wei Liu, and Yu Huang. Stochastic Maintenance Schedules of Active Distribution Networks based on Monte-Carlo Tree search. *IEEE Transactions on Power Systems*, 8950(c):1–1, 2020. ISSN 0885-8950. doi: 10.1109/tpwrs.2020.2973761.

[40] Xiao Sheng Si, Wenbin Wang, Chang Hua Hu, and Dong Hua Zhou. Remaining useful life estimation - A review on the statistical data driven approaches. *European Journal of Operational Research*, 213(1):1–14, 2011. ISSN 03772217. doi: 10.1016/j.ejor.2010.11.018. URL http://dx.doi.org/10.1016/j.ejor.2010.11.018.

[41] Wesley Spreen. Aerospace maintenance, repair, and overhaul. *The Aerospace Business*, (275): 312–324, 2019. doi: 10.4324/9780429299452-14.

[42] M Sri, Krishna Kopuru, S Rahimi, and K T Baghaei. Recent Approaches in Prognostics: State of the Art. *International Conference Artificial Intelligence*, pages 358–365, 2019.

[43] Chellappan Sriram and Ali Haghani. An optimization model for aircraft maintenance scheduling and re-assignment. *Transportation Research Part A: Policy and Practice*, 37(1):29–48, 2003. ISSN 09658564. doi: 10.1016/S0965-8564(02)00004-6.

[44] Albert Steiner. A heuristic method for aircraft maintenance scheduling under various constraints. *6th Swiss Transport Research Conference*, (December):28, 2006. URL http://www.strc.ch/conferences/2006/Steiner_STRC_2006.pdf.

[45] Laura Swanson. Linking maintenance strategies to performance. *International Journal of Production Economics*, 70(3):237–244, 2001. ISSN 09255273. doi: 10.1016/S0925-5273(00)00067-0.

[46] Arryon D. Tijsma, Madalina M. Drugan, and Marco A. Wiering. Comparing exploration strategies for Q-learning in random stochastic mazes. *2016 IEEE Symposium Series on Computational Intelligence, SSCI 2016*, 2017. doi: 10.1109/SSCI.2016.7849366.

[47] Tiedo Tingo. *Principles of loads and failure mechanisms. Applications in maintenance, reliability and design.* Springer, 2013. doi: 10.1007/978-1-4471-4917-0.

[48] Kwok L. Tsui, Nan Chen, Qiang Zhou, Yizhen Hai, and Wenbin Wang. Prognostics and health management: A review on data driven approaches. *Mathematical Problems in Engineering*, 2015, 2015. ISSN 15635147. doi: 10.1155/2015/793161.

[49] George Vachtsevanos, F L Lewis, M Roemer, Andrew Hess, and Blake wu. Intelligent fault diagnosis and prognosis for engineering systems: Methods and case studies. 13, 1 2006. doi: 10.1002/9780470117842.

[50] Adriaan Van Horenbeek and Liliane Pintelon. A dynamic predictive maintenance policy for complex multi-component systems. *Reliability Engineering and System Safety*, 120:39–50, 2013. ISSN 09518320. doi: 10.1016/j.ress.2013.02.029. URL http://dx.doi.org/10.1016/j.ress.2013.02.029.

[51] Sandrina Vilarinho, Isabel Lopes, and José A. Oliveira. Preventive Maintenance Decisions through Maintenance Optimization Models: A Case Study. *Procedia Manufacturing*, 11(June): 1170–1177, 2017. ISSN 23519789. doi: 10.1016/j.promfg.2017.07.241.

[52] Shaomin Wu and Inma T. Castro. Maintenance policy for a system with a weighted linear combination of degradation processes. *European Journal of Operational Research*, 280(1):124–133, 2020. ISSN 03772217. doi: 10.1016/j.ejor.2019.06.048.

[53] Dayong Xu, Fei Zhu, Quan Liu, and Peiyao Zhao. Improving exploration efficiency of deep reinforcement learning through samples produced by generative model[Formula presented]. *Expert Systems with Applications*, 185, 2021. ISSN 09574174. doi: 10.1016/j.eswa.2021.115680.

[54] Ming Yi You, Fang Liu, Wen Wang, and Guang Meng. Statistically planned and individually improved predictive maintenance management for continuously monitored degrading systems. *IEEE Transactions on Reliability*, 59(4):744–753, 2010. ISSN 00189529. doi: 10.1109/TR.2010.2085572.

[55] Jianjing Zhang, Peng Wang, Ruqiang Yan, and Robert X. Gao. Long short-term memory for machine remaining life prediction. *Journal of Manufacturing Systems*, 48(June):78–86, 2018. ISSN 02786125. doi: 10.1016/j.jmsy.2018.05.011. URL https://doi.org/10.1016/j.jmsy.2018.05.011.

[56] Zeqi Zhao, Bin Liang, Xueqian Wang, and Weining Lu. Remaining useful life prediction of aircraft engine based on degradation pattern learning. *Reliability Engineering and System Safety*, 164(457):74–83, 2017. ISSN 09518320. doi: 10.1016/j.ress.2017.02.007. URL http://dx.doi.org/10.1016/j.ress.2017.02.007.

[57] Qiushi Zhu, Hao Peng, Bas Timmermans, and Geert Jan van Houtum. A condition-based maintenance model for a single component in a system with scheduled and unscheduled downs. *International Journal of Production Economics*, 193(March 2016):365–380, 2017. ISSN 09255273. doi: 10.1016/j.ijpe.2017.07.014. URL https://doi.org/10.1016/j.ijpe.2017.07.014.