Aircraft Maintenance Scheduling Using Engine Sensor Data

An aircraft maintenance scheduling optimization model using data-driven turbofan engine RUL prognostics

Arthur Pieter Reijns

TU Delft

Aircraft Maintenance Scheduling Using Engine Sensor Data

by

Arthur Pieter Reijns

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on 2 July 2021 .

Student number:	4556100
Project duration:	01-09-2020 – 02-07-2021
Thesis committee:	Prof. Dr. G. de Croon
	Dr. R. Ferrari
	Dr. M. Mitici
	M.Sc. I. de Pater

An electronic version of this thesis is available at http://repository.tudelft.nl/.

TU Delft

Acknowledgements

With this thesis I will conclude my MSc in Aerospace Engineering at Delft University of Technology. Writing this thesis took me 9 months, during which I have learned a lot. First of all, I would like to sincerely thank my thesis supervisor dr. Mihaela Mici for her continuous efforts and feedback. Furthermore, I would like to thank M.Sc. Ingeborg de Pater for being present during the thesis progress meetings. I consider your presence and feedback to be very valuable for the establishment of this work.

With the thesis being written in the midst of the Covid-19 pandemic, it was a time during which I spent a lot of hours in my room at home. Even though this complicated certain aspects of the research and made the thesis work less enjoyable at times, I still enjoyed doing the thesis for the most part as I was able to obtain new knowledge and gain valuable experience in the field of prognostics and maintenance scheduling. I would like to thank my family, friends and girlfriend for their continuous support throughout my studies in Delft and while working on my final thesis.

A. P. Reijns
Delft, June 2021

Contents

List of Figures

List of Tables

Nomenclature

Abbreviations

1D	1-Dimensional
2D	2-Dimensional
ADDP	Advanced Data-Driven Policy
ADD	Advanced Data-Driven
ANN	Artificial Neural Network
C-MAPSS	Commercial Modular Aero-Propulsion System Simulation
CDF	Cumulative Distribution Function
CNN	Convolutional Neural Network
CRA	Cumulative Relative Accuracy
CRPS	Continuous Ranked Probability Score
DDP	Data-Driven Policy
DD	Data-Driven
DE	Differential Evolution
EoL	End of Life
FC	Flight Cycles
FN	False Negative
FP	False Positive
GA	Genetic Algorithm
GDDP	Gap Data-Driven Policy
GIBP	Gap Inspection Based Policy
G	Gapped
HI	Health Indicator
IBP	Inspection Based Policy
IB	Inspection Based
ILP	Integer Linear Programming
KLM	Koninklijke Luchtvaart Maatschappij
KPI	Key Performance Indicator
LR	Linear Regression
LSTM	Long Short-Term Memory

MAE	Mean Absolute Error
MCD	Monte Carlo Dropout
MC	Monte Carlo
MDN	Mixed Density Network
MLP	Multi-Layer Perceptron
ML	Machine Learning
MODBNE	Multi-Objective Deep Belief Network Ensemble
MO	Maintenance Opportunity
MPI	Monetary Performance Indicator
MSET	Multivariate State Estimation Technique
MSE	Mean Squared Error
NASA	National Aeronautics and Space Administration
NGDDP	No Gap Data-Driven Policy
NGIBP	No Gap Inspection Based Policy
NG	No Gapped
NN	Neural Network
OPI	Operational Performance Indicator
PA	Prognostic Algorihtm
PDF	Probability Density Function
PHM	Prognostics and Health Management
RA	Relative Accuracy
RBM	Restricted Boltzmann Machine
RMSE	Root Mean Squared Error
RNN	Recurrent Neural Network
RUL	Remaining Useful Life
RVM	Relevance Vector Machine
SSR	Sampling Rate Robustness
SVM	Support Vector Machine
TH	Threshold
TN	True Negative
TP	True Positive
tRUL	Target RUL
WCRA	Weighted Cumulative Relative Accuracy
WCRPS	Weighted Continuous Ranked Probability Score

Symbols

α	Acceptable accuracy error bounds
β	Probability mass required to be between α accuracy bounds
$\beta_{\alpha-\lambda}$	Probability mass required to be the between α error bounds in order for a prediction to be acceptable
β_{FN}	Probability mass required to be below the acceptable early prediction error bound in order to count a prediction as a false negative
β_{FP}	Probability mass required to be above the acceptable late prediction error bound in order to count a prediction as a false positive
β_{PH}	Probability mass required to be the between prognostic horizon error bounds in order for a prediction to be acceptable
β_{TN}	Probability mass required to be between the acceptable early prediction error bound and the true RUL value in order to count a prediction as a true negative
β_{TP}	Probability mass required to be between the acceptable late prediction error bound and the true RUL value in order to count a prediction as a true positive
$\epsilon(i)$	Prediction error at time i
γ_e	Input parameter to the score function that determines how heavily an early prediction is penalized
γ_l	Input parameter to the score function that determines how heavily a late prediction is penalized
λ	A time-window modifier that can be chosen such that a specific accuracy α is achieved at a specific time instance
$\mathbb{1}$	Heaviside step function
$\mu^{j,k}$	Mean value of all recordings of the j-th sensor used under operating condition k
$\overline{L}$	Mean life of an engine
$\overline{RUL}_{k,i}^{\text{pred},nA_s}$	Mean of the predicted RUL distribution of engine i located on aircraft k at the moment the final alert for maintenance scheduling is generated
$\overline{WL}$	The mean wasted life of an engine after a replacement
ϕ	Activation function used at convolutional layer
$\pi[r(i_\lambda)]_{\alpha-}^{\alpha+}$	Probability mass between α accuracy bounds for the RUL prediction at time i
ψ	Optimization period
$\sigma^{j,k}$	Standard deviation of all recordings of the j-th sensor used under operating condition k
τ	Realization period
$\mathbf{w}$	Convolutional filter
$\mathbf{x}$	1D-input array for CNN
A	Number of agents in a generation
A^*	Number of agents with the highest fitness that go again in the next generation
b	Bias value used at convolutional layer

C_I	Cost of an inspection
C_R	Cost of a replacement
c_{ag}	Number of copies an agent receives in the genepool
c^*_{ag}	Number of copies the agent with the highest fitness receives in the genepool
$C^{p,k}_{C,\text{tot}}$	Total cancellation cost for aircraft k in realization period p
C_H	Cost of a hangar visit
$c_{i,k,t}$	Penalty cost parameter of assigning engine i of aircraft k to maintenance slot t
C_{LC}	Cost of a long term cancellation
C_{MC}	Cost of a mid term cancellation
C_{MHR}	Cost per employee per hour
C_O	Cost of engine overhaul
C_{SC}	Cost of a short term cancellation
C_{SI}	Cost of a safety incident
C_{TO}	Cost of tools
C_{TW}	Cost of towing per hour
C_T	Cost of flying 1 FC after the target RUL
Can_L	The number of long term cancellations per aircraft per week
Can_M	The number of mid term cancellations per aircraft per week
Can_S	The number of short term cancellations per aircraft per week
D_I	Duration of an inspection
D_R	Duration of a replacement
D_{TW}	Duration of towing
E_I	Employees required to perform an inspection
E_R	Employees required for a replacement
F	Cumulative distribution function of the prediction
f	Fitness of an agent in the genetic algorithm
f_1	Sampling frequency 1
f_2	Sampling frequency 2
f_C	Factor the cancellation cost is multiplied with in the sensitivity analysis
f_I	Factor the inspection cost is multiplied with in the sensitivity analysis
F_L	Length of the convolutional filter
F_N	Number of feature maps generated at a convolutional layer
f_s	Multiplication factor used to compute the target RUL
$F_{L,\text{final}}$	Length of the convolutional filter in the final convolutional layer

$FC_{>RUL}$	The number of FC flown after the target RUL per aircraft per week
FC_{flown}	The total number of flight cycles flown per aircraft per week
H	Number of available hangars
I	The number of inspections per aircraft per week
I_K	Set of all engines i located on aircraft k
IBP^*	The optimized policy vector for the inspection based policy
Int_I	Inspection interval for the inspection based maintenance polices
K	Set of all aircraft k
L_C	Number of convolutional layers
L_{TO}	Life of tools
LF_l	Long-term load factor
LF_m	Mid-term load factor
LF_s	Short-term load factor
lr_f	Factor with which the learning rate is multiplied after lr_p consecutive epochs without improvement in loss
lr_p	Number of consecutive epochs without improvement is loss required to lower the learning rate
M_{pax}	Maximum number of passengers for an aircraft
MC	Number of Monte Carlo simulations
N_E	The number of engines per aircraft in the simulation model
N_K	The number of aircraft in the simulation model
n_{FCL}	Number of neurons in the fully-connected layer of the CNN
N_{ft}	Number of features included as input to the CNN
N_{gen}	Number of generations for the genetic algorithm
N_{tw}	Number of historical flight cycles included as input to the CNN
nA_s	Number of consecutive alerts required for maintenance scheduling
nA_{mo}	Number of consecutive alerts required for maintenance opportunity creation
nMO	Number of maintenance opportunities created
P	The number of realization periods to run the simulation model for
p_λ	Set of all time indexes at which a prediction is made
p_d	Dropout probability
p_s	Probability mass required to be below threshold in order to generate an alert for maintenance scheduling
p_{mo}	Probability mass required to be below threshold in order to generate an alert for maintenance opportunity creation
p_{mut}	Probability of mutation

P_{new}	Percentage of total number of agents in a generation that are assigned a random policy vector in the next generation
R	The number of engine replacements per aircraft per week
$r_{\text{true}}(i)$	True RUL value at time i
R_{early}	Early constant target RUL value used for labelling
$RUL_{\text{insp}}^{\text{pred}}$	The predicted RUL value that is obtained after an inspection
$RUL_{\text{insp}}^{\text{true}}$	The true RUL value of an engine at the moment of an inspection
s	Score value
SI	The number of safety incidents per aircraft per week
sm_s	Safety margin used to compute the target RUL
T	The ticket price to book one flight cycle
t	The time in FC of the maintenance slot t
T_K	Set of all maintenance opportunities t available to aircraft k
t_{FN}	Acceptable late prediction error bound
t_{FP}	Acceptable early prediction error bound
TH_I	Threshold for maintenance scheduling for the inspection based maintenance polices
TH_s	Threshold for maintenance scheduling
TH_{mo}	Threshold for maintenance opportunity creation
$tRUL_{i,k}$	Target RUL value for engine i located on aircraft k
u_{insp}	Parameter that determines the standard deviation of normal distribution the error value for the estimated RUL at inspection is taken from
u_E	Parameter that determines the uniform distribution the error value for the estimated RUL at inspection is taken from
$w(r(i))$	Weighting function that gives more weight to engines with a lower value of $r(i)$
w_s	Weight factor that is used to give engines that are closer to failure a higher weight than engines that are healthy
x	True value that is aimed to be predicted
$x^{i,j,k}$	Input value of the i-th data point of the j-th sensor used under operating condition k
$x_{\text{max}}^{j,k}$	Maximum value of all recordings of the j-th sensor used under operating condition k
$x_{\text{min}}^{j,k}$	Minimum value of all recordings of the j-th sensor used under operating condition k
$x_{\text{norm}}^{i,j,k}$	Normalized value of the i-th data point of the j-th sensor used under operating condition k
$x_{i,k,t}$	Decision variable that equals 1 if engine i of aircraft k is assigned to maintenance slot t, 0 otherwise
z_i	Feature map i generated after a convolutional layer

Introduction

This thesis covers three topics that are highly relevant in the current aviation industry: data-driven remaining useful life (RUL) prognostics, prognostic algorithm performance evaluation and the use of RUL prognostics in maintenance scheduling. The main goal of this thesis is to further contribute to the available body of knowledge on those subjects by developing novel theories and methods.

This thesis aims to solve the following problems that are identified from literature: first, the accuracy of prognostic algorithms in terms of RUL prediction error has gone up over the past years. However, using novel computational techniques it is believed to be possible to further improve the prediction results compared to recent literature. Furthermore, current literature usually does not provide an extensive analysis on the obtained prediction results. Lastly, very limited research has been done into how probabilistic data-driven RUL predictions can be used to schedule maintenance.

These observations lead to the following condensed research questions: How can sensor recordings be used to predict a remaining useful life distribution of an aircraft engine at any moment in time? How can the performance of a prognostic algorithm be optimally assessed such all aspects are taken into consideration? How can probabilistic data-driven RUL predictions be integrated in the maintenance scheduling routine of airlines? These questions will be answered in this thesis alongside smaller sub-questions.

This research is relevant to any airline that wants to innovate its maintenance scheduling process by making use of data-driven techniques. Shifting towards a data-driven organization holds the potential benefits of increased scheduling efficiency and profits, while decreasing the time required to schedule maintenance on a daily basis. Optimizing the airline maintenance process will lead to lower aircraft downtime and less wasted life of aircraft components. This will result in a higher passenger satisfactory, a more sustainable way of operation and a higher profit for the airline. All of these aspects stress the importance of optimizing the airline maintenance scheduling process, both from a economic and societal point of view.

This thesis report is organized as follows : In Part I, the scientific paper is presented. Part II contains the relevant Literature Study that supports the research. Finally, in Part III, a chapter on how verification and validation are performed in this research is included.

I

Scientific Paper

Aircraft Maintenance Scheduling Using Engine Sensor Data

Arthur Pieter Reijns*

Delft University of Technology, Delft, The Netherlands

Abstract

Over the past years, prognostic algorithms that aim to predict the remaining useful life (RUL) of aircraft engines have seen an increase in prediction accuracy. However, novel computational methods indicate that there is still room for improvement in the accuracy of those prognostic algorithms, especially when these algorithms are desired to be used to schedule maintenance in practice. Furthermore, only limited research has been done on how probabilistic data-driven RUL predictions can be used to schedule maintenance for aircraft engines. This paper proposes a convolutional neural network (CNN) for predicting aircraft engine's remaining useful life. This CNN takes recordings from sensors placed inside the engine as input and returns a probabilistic prediction of the engine's RUL. This paper also introduces a large framework of prognostic algorithm performance assessment in order to obtain a full picture of the strengths and weaknesses of the proposed prognostic model. It was shown that by using optimal feature selection, data normalization, Monte Carlo dropout and a tuned CNN model, RUL prediction performance was improved significantly in comparison to recent literature when testing the model on the popular C-MAPSS dataset. This paper then shows how probabilistic data-driven RUL predictions can be used in a maintenance scheduling simulation model. In this model, the RUL predictions are used to determine a maintenance window using several maintenance policies. These data-driven maintenance policies are optimized using a genetic algorithm (GA). An inspection based policy is included to be able to compare the data-driven maintenance policies to current practices. The optimal maintenance opportunity in the maintenance window for the engine is then selected using an integer linear programming (ILP) model. The interaction of all maintenance polices with two types of flight schedules is investigated in order to find the optimal maintenance strategy for an airline such that profit is maximized. The performance of all maintenance strategies is evaluated using both operational and monetary performance indicators. It was found that the data-driven maintenance strategies outperform inspection based strategies in terms of airline profit. A sensitivity analysis on the optimal maintenance strategy revealed that it is very sensitive, indicating that appropriate safety margins should be applied when adopting the data-driven strategies in practice. Nonetheless, the developed data-driven maintenance scheduling strategies hold great potential when it comes to adopting data-driven RUL prognostics in the daily scheduling routine of an airline.

1 Introduction

Repair and maintenance costs are among the top expenses of big aviation companies. According to the annual reports of the Royal Dutch Airlines KLM, the aircraft maintenance costs were around €882 million in 2019, equivalent to $\sim$ 14.5% of their total yearly expenses [1]. Furthermore, not being able to fly an aircraft is also very costly. DHL has estimated that if an aircraft has to remain on ground due to technical issues, this can cost an airline up to €925.000 per day [2]. This shows the importance of being able to monitor the health of the aircraft and its components, as well as predicting when specific parts will fail. Over the last few decades, Prognostics and Health Management (PHM) has received increasing attention both from a practical and scientific perspective. PHM combines real-time and historical information of the system to improve the decision-making in terms of maintenance operations. It is considered to be an engineering disciple that has as a goal to minimize maintenance costs while ensuring adequate levels of safety and operationally. Is does so by assessing the health state of a system using available information from for example sensors, and tries to make predictions on the remaining useful life (RUL) accordingly. Over the past years, RUL prediction has received increasing attention in scientific literature, partially due to the availability of an open source turbofan engine degradation dataset provided by NASA [3]. A high variety of prognostic algorithms is developed using this dataset with the goal of making the most accurate RUL prediction model. The problem is that even though current prognostic algorithms achieve acceptable prediction accuracy, there is still room for improvement in order to make the prognostic algorithm more suited for use in practice. Furthermore, there is a clear gap in literature regarding how of RUL predictions can be incorporated into airline daily maintenance scheduling

*Msc Student, Air Transport and Operations, Faculty of Aerospace Engineering, Delft University of Technology

operations. This limits the usefulness of the prognostic models, as even though accurate RUL predictions can be made, the predictions can not be used in a profitable manner for an airline.

This work will focus on 3 main aspects: (i) developing a prognostic model that is able to predict engine RUL using engine sensor data more accurately than is done in current literature, (ii) introducing an extensive performance evaluation framework for probabilistic RUL estimations and (iii) developing a maintenance scheduling simulation model that is able to simulate maintenance scheduling for an airline using data-driven RUL predictions obtained using the prognostic model. The prognostic model that is designed is a convolutional neural network (CNN). Li et al. and Babu et al. have shown that CNN is a promising method for RUL estimation, and therefore this method is further explored in this work [4, 5]. It is believed that by optimizing the input sensor selection, the inclusion of more input parameters and the application of Monte Carlo dropout sampling, the prediction results achieved by Li et al. can be significantly improved. Furthermore, most current literature that performs RUL prediction only includes a limited model performance evaluation. In this work, both developed and novel metrics are used to present a holistic overview of the prognostic model's performance. Lastly, a variety of maintenance scheduling strategies will be developed in order to investigate how RUL prognostics can be adopted into maintenance scheduling. This will be done by developing scheduling policies that take a time series of probabilistic RUL predictions as input and return the desired moment of maintenance. In addition, the interaction of those policies with the flight schedule is investigated. Inspection based maintenance policies are also introduced in order to compare the data-driven maintenance strategies to strategies that are similar to how maintenance scheduling is performed nowadays. A genetic algorithm is used to optimize the data-driven maintenance scheduling policies. The results of all scheduling strategies are then compared using both operational and monetary performance indicators to see which strategy performs best and is most suitable for adoption in practice.

This paper is structured as follows: first, section 2 presents a literature study on current prognostic algorithms and maintenance scheduling models. After that, section 3 covers the first part of this work, which is data-driven RUL prediction using a CNN. Section 3.1 describes all the methods used to predict RUL using a CNN, subsection 3.2 elaborates on the experimental set-up of the test case and subsection 3.3 introduces all the performance evaluation metrics used on the prognostic model outcome. Lastly, subsection 3.4 presents the prediction results obtained using the CNN and compares the results to relevant literature. How the obtained data-driven RUL predictions can be used for maintenance scheduling is explained in the second part of this work in section 4. First, subsection 4.1 introduces the methodology used, which includes explanations on the scheduling policies, the interaction with the flight schedule, the Integer Linear Programming (ILP) model that picks the optimal maintenance slots and the genetic algorithm (GA) used for policy optimization. After this, subsection 4.2 states the experimental set-up for the maintenance scheduling test case and subsection 4.3 describes how the performance of a maintenance scheduling strategy can be evaluated. Section 4.4 then discusses the results of the optimized policies and the maintenance strategies. Lastly, sensitivity analysis on the optimal policies and the cost input values is presented in subsection 4.5. Conclusions and recommendations for further work are stated in section 5.

2 Literature Review

This section will discuss the most prominent current literature on RUL prediction using machine learning algorithms and on maintenance scheduling. This literature provides insight into relevant methods and ideas that can be adopted and adjusted in this work. First, literature on data-driven RUL prognostics is discussed, with a focus on papers that use a CNN for RUL prediction. After that, literature on prognostic algorithm performance evaluation is introduced. Lastly, literature on maintenance scheduling is highlighted.

2.1 Data-driven RUL prognostics

Over the past years, RUL prediction using sensor data has received increasing attention. This is partially due to the availability of the open source turbofan engine degradation dataset called C-MAPSS provided by NASA [3, 6]. Many recent works develop prognostic algorithms to predict RUL and test their models on the C-MAPSS dataset afterwards. Heimes is the first author to attempt to predict RUL using machine learning methods as part of the PHM008-Challenge [7]. He first introduces a Multi-Layer Perceptron (MLP) and is able to successfully differentiate between engines in a healthy and unhealthy state. He furthermore introduces a Recurrent Neural Network (RNN) for RUL prediction, which achieves satisfactory prediction performance. In [8], Zheng et al. use a Long Short-Term Memory (LSTM) network for RUL prediction. The objective is to minimize the mean squared error between the true RUL and the predicted RUL. Compared to a MLP and a CNN, this LSTM model obtained lower error values. Ellefsen et al. also use a LSTM model with unsupervised pre-training and supervised learning to improve RUL prediction quality [9]. Furthermore, a Genetic Algorithm

(GA) is used to tune the hyperparameters of the deep architecture. Aremu et al. use a machine learning data dimension reduction framework after which low-dimension representations of each cluster are learned using Laplacian eigenmaps embedding [10]. Yu et al. use a RNN autoencoder scheme and obtain the lowest RMSE values on the C-MAPSS dataset found in literature [11]. More recently, TV. et al. use a variety of LSTM extensions to perform RUL prediction [12]. Still, they fail to beat the lowest RMSE values obtained by Ellefsen et al. and Aremu et al. on the C-MAPSS dataset.

The first attempt at RUL prediction using a CNN is done by Babu et al. in 2016 [4]. At the time, the lowest RMSE values in literature were obtained by passing a 2D-input array into a CNN with 1D-filters. The filters slide along the temporal dimensions and extract features that are eventually used by a fully connected layer and a regression node to predict the RUL value. More recently, Li et al. have shown that by tuning the CNN hyperparameters a CNN network with regression node is well able to predict engine RUL [5]. At the time, the tuned deep CNN architecture trained using dropout was able to obtain significantly lower error values than previous attempts. The work of Li et al. will form the basis of the CNN developed in this paper as this CNN method holds great potential even though limited research into the use of a CNN for RUL prediction has been performed. For an extensive literature review on data-driven RUL prediction the reader is referred to part 2 of this thesis.

2.2 Prognostic algorithm performance evaluation

Only limited literature on prognostic algorithm performance evaluation can be found. In 2008, Saxena et al. developed a framework of prognostic metrics to evaluate the performance of prognostic algorithms [13]. Saxena et al. continued by writing a paper in 2010 describing a large variety of metrics that can be used for the offline evaluation of prognostic algorithms [14]. Both of these works are still considered to be the most relevant literature available nowadays, as is also stated by Baur et al in 2020: "*..., the development of improved standardized metrics, suitable even for on-line applications, still represents a stimulating topic for the research community*" [15]. This work will make use of the evaluation framework developed by Saxena et al. and will furthermore develop novel metrics suitable to evaluate the performance of prognostic models. Lastly, a metric that is interesting to include is the Continuous Ranked Probability Score (CRPS) introduced by Gneiting and Raftering [16]. This metric is able to compare a probabilistic prediction to a single true value by returning a single error value. For an extensive literature review on prognostic algorithm performance evaluation the reader is referred to part 2 of this thesis.

2.3 Maintenance scheduling

Schneider and Cassady introduce a maintenance scheduling model in which the probability that all future missions of aircraft in a set are completed successfully is maximized. A cost-based optimization model is used that minimizes the cost and maximizes reliability [17]. Lam and Banjanic develop a policy that is able to determine the optimal moments of inspection by monitoring the condition of the component. At each decision point, a choice is made to either inspect of replace the component, and the inspection interval for the next time period is determined [18]. Vu et al. introduce a maintenance optimization framework that makes use of a rolling horizon method that continuously optimizes the maintenance decisions for a certain period of time. Using a cost model and a heuristic optimization scheme, a maintenance grouping strategy is developed for a multi-component system [19]. In [20], Wu and Castro develop a maintenance policy for a system of which the condition is continuously monitored. If a combination of degradation processes has reached a specified threshold, the system is considered to have failed. Preventive maintenance is used in this model, in which the system is fully replaced after several preventive maintenance actions.

In [21], Zhang and Zhang first develop a prognostic model that is able to predict probabilistic RUL distributions for aircraft engines by making use of a stacked autoencoder long short-term memory network. The obtained predictions are then used to create a condition-based maintenance optimization framework. This framework makes use of several threshold values that determine the moment in time maintenance is required. Both a periodic and an ideal maintenance policy are developed in order to compare the scheduling results in terms of cost. A flaw of this work is that it only outputs the moment in time preventive maintenance should occur, and thus does not consider the interaction with the aircraft's flight schedule. More recently, de Pater and Mitici show how RUL prognostics can be used to perform predictive maintenance for a multi-component system [22]. Over time, RUL prognostics are updated as new sensor data becomes available. The developed maintenance planning model combines continuously updated RUL prognostics with available maintenance slots in the flight schedule. The predictive maintenance strategy was shown to outperform both a corrective and a preventive strategy in terms of costs.

3 Predicting RUL Using A Convolutional Neural Network

This section covers the first part of this research: predicting turbofan engine RUL using a machine learning algorithm. In this part, a machine learning algorithm will be developed that is able to take as input raw sensor data collected during flight and return as output a probabilistic distribution of the engine's RUL. First, the methods used in this section will elaborated on. Next, the experimental set-up of the test case is explained. The data that is used is highlighted and the pre-processing steps are elaborated on. After that, the performance evaluation of prognostic algorithms is explained. Both current and novel prognostic metrics will be introduced. Lastly, the results of the developed model are shown using the metrics that were introduced prior. The results obtained in this work are also compared to related recent publications to present a full picture of the performance of the model.

3.1 Methods

This section aims to explain the methods that were used to develop the prognostic machine learning model. First, the convolutional neural network method used in this paper is discussed. After that, Monte Carlo dropout sampling is explained. Lastly, the network structure used in this paper is stated.

3.1.1 Convolutional neural network

This paper will develop a deep convolutional neural network (CNN) with regression node at the end for RUL estimation. CNN's were first introduced by LeCun for image processing as they posses beneficial properties to deal with 2D inputs that have a grid like topology [23]. Using spatially shared weights and pooling, CNN's are able to identify hidden features in the input data that become more complex as more layers are added. The convolutional layers convolve the input array with several learned filters to obtain features. Pooling layers can then be used to ensure that only local features are kept that posses the most information. For example, when using a CNN for image recognition, the output of the first layer might be abstract features such as lines and shapes, but after multiple layers the features might be more detailed, such as faces or objects. A CNN performs particularly well on data that has spatially neighbouring features in the input data, meaning that the input values close to each other are similar and that there is a relation between them. For example, the pixels in an image are spatially neighbouring features as pixels with similar RGB-values are likely to be part of the same object. In this paper, the CNN will be used to find a pattern along the temporal dimension of the sensor input data.

In this paper, the input to the CNN will be signals of a variety of sensors that are placed inside the aircraft turbofan engines that record a single value per flight cycle. An example might be the average pressure in the low pressure turbine during a single flight. The format of the input array is 2D, where one dimension is the number of features (sensors) included and the other is the number of historical flight cycles. Note that the data is only spatially neighbouring across the temporal dimension and not across the feature dimension. Even though the input is 2D, the effective CNN that is developed in this work can be regarded as a 1D CNN as the filter width is kept at 1 and the filters only slide along the temporal dimension.

Suppose x_1 represents the vector containing all measured sensor values after the first flight cycle. Combining all vectors for all the N recorded flight cycles leads to the following sequential 1D representation of the complete input array: $\mathbf{x} = [x_1, x_2, \ldots, x_N]$. The convolution operation can now be defined as the multiplication between a filter $\mathbf{w}$ with length F_L and the array obtained by concatenating F_L vectors from x. Note that a filter $\mathbf{w}$ is a 1D array of weights that can be learned by the model. This concatenation operation leads to the input array $\mathbf{x}_{i:i+F_L-1}$ and is obtained as shown in Equation 1. The array $\mathbf{x}_{i:i+F_L-1}$ now represents an input array of F_L consecutive signals starting at point i and ending at point $i + F_L - 1$.

$$\mathbf{x}_{i:i+F_L-1} = x_i \oplus x_{i+1} \oplus \cdots \oplus x_{i+F_L-1} \tag{1}$$

The convolution operation of filter $\mathbf{w}$ on subset $\mathbf{x}_{i:i+F_L-1}$ is defined in Equation 2. In this equation, $\mathbf{w}^T$ is the transpose of the filter $\mathbf{w}$, b is a bias value and ϕ is the activation function that is applied. Furthermore, z_i can be seen as the learned feature of the filter $\mathbf{w}$ on a subset of the full input array, namely $\mathbf{x}_{i:i+F_L-1}$. While sliding the filter over the complete input x, the value of i keeps increasing until the edge of the input array is reached. When doing this, one filter $\mathbf{w}$ will generate F_L values of z_i, which can be denoted by $\mathbf{z}_i$ as shown in Equation 3. $\mathbf{z}_i$ now represents a single feature map that is obtained using one filter.

$$z_i = \phi(\mathbf{w}^T \mathbf{x}_{i:i+F_L-1} + b) \tag{2}$$

$$\mathbf{z}_i = [z_i^1, z_i^2, \ldots, z_i^{N-F_L+1}] \tag{3}$$

In a CNN, each layer can have many filters each generating its own feature map. Figure 1 shows how multiple filters can generate feature maps from a single 1D input array. The output of this is then a 3D array with depth n, as the depth is determined by the number of feature maps generated.

The number of feature maps generated per layer F_N and the length of the filters F_L are amongst the most important model hyperparameters. While tuning the model hyperparameters, trial and error methods will be applied to optimize these values in order to maximize model performance. It should be noted that F_L has a clear upper limit as it is not possible to make the filter longer than the available history of flight cycles for each engine. Lastly, the application of pooling layers is common when designing CNN's. However, as the input data size is relatively small due to the low number of input features, it is chosen to neglect pooling in this paper as some useful information might be filtered out. This likely does not outweigh the potential benefit of increased computational time that can be achieved due to the decrease in feature dimensionality.

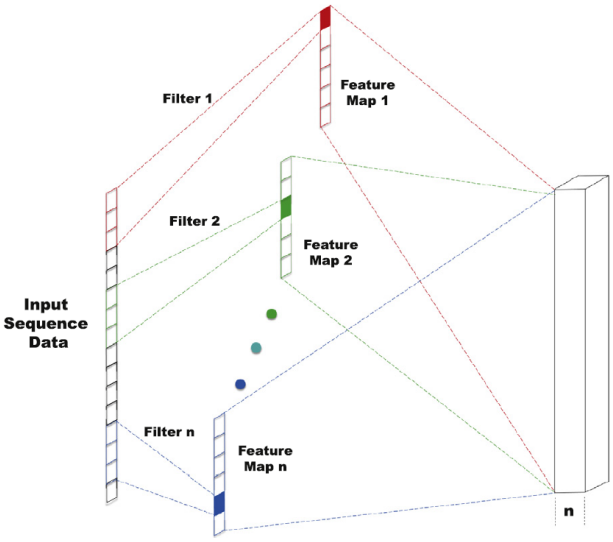


Figure 1: CNN feature map generation using multiple filters on a 1D input [5]

3.1.2 Network architecture

The network architecture that will be used can be seen in Figure 2. On the left, the input array can be seen. This input array has size $N_{tw} \times N_{ft}$, where N_{tw} equals the number of historical flight cycles included and N_{ft} equals the number of features (sensor signals) included. The convolution operation is then applied to this 2D input array using the 1D filter with size $F_L \times 1$, making the convolution operation 1D as well. In the image shown in Figure 2, 10 feature maps are generated after each convolutional layer and a hyperbolic tangent activation function is applied. Note that the number of feature maps generated after each layer F_N and the activation function at each layer are model hyperparameters that need to be carefully tuned. After L_C convolutional layers, a final convolutional layer is placed that only generates a single feature map using a smaller filter size $F_{L,\text{final}}$, making the output of this layer 2D. This 2D output is then flattened and connected to a single fully connected layer that consists of n_{FCL} neurons. The final layer is a single neuron with linear activation that acts as a regression node. The output of the final layer is directly used as the predicted RUL value.

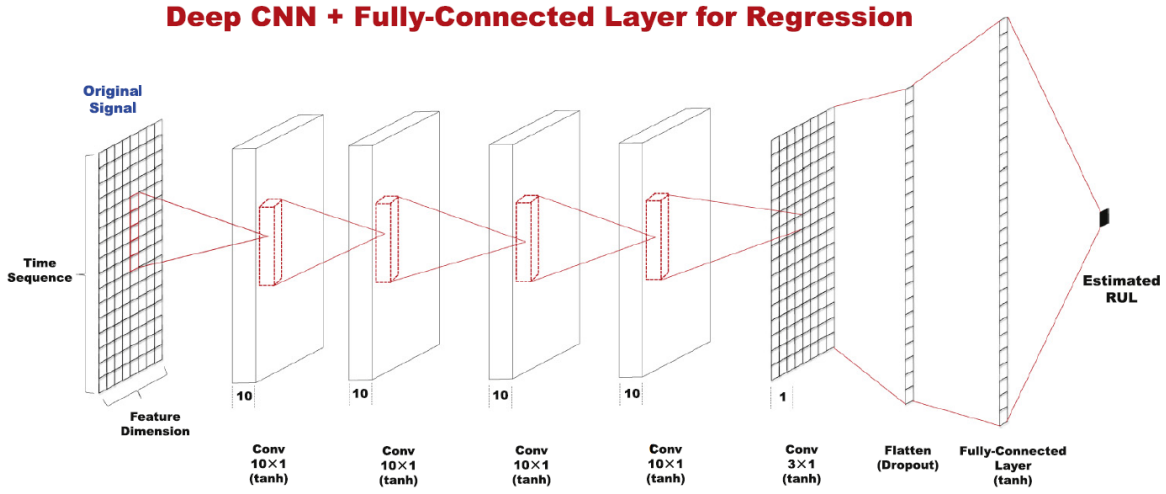


Figure 2: Architecture of the deep convolutional neural network that is used to predict RUL [5]

During model training, the default weight initialization is Xavier normal. The optimizer used during training the the Adam optimizer, which is the fastest and best suited optimizer available nowadays. The loss function that is used to train the model is the mean squared error. The model learning rate is automatically multiplied by a factor lr_f after lr_p consecutive epochs without improvement in the verification loss. Every time the model

finds a set of weight with lower loss, this set of weights is automatically saved such that the best version of the model at any time can be used for testing. Regarding the training data, 80% of the training samples is used for training each epoch, while the other 20% of training data is used for computing the validation loss. The weights can then be trained in the right direction using back-propagation.

3.1.3 Monte Carlo dropout

In recent years, dropout has emerged as a promising technique to combat neural network overfitting. When using dropout in a regular fully-connected neural network, the weights connected to a random subset of neurons in each layer are set to 0 with a specified probability during training. This is done such that the model does not become overly dependent on a small number of neurons and forces the model to be flexible and robust. Monte Carlo dropout is a variant during which dropout is kept on during model testing as well. This causes the model output to be slightly different each time the same input array is passed. In this way, the same input can be passed many times such that a sampled output distribution is obtained for that input. This paper will makes use of this method to sample output RUL distributions for each input by using MC forward passes per input. Furthermore, Jospin et al. found that by making use of dropout and averaging the results of the approximated ensemble network, superior prediction results can be obtained [24]. Gal et al. found that Monte Carlo dropout in CNN's approximates a Gaussian process and showed that models that kept dropout on during test time outperformed models that only use dropout during training [25]. Monte Carlo dropout thus not only allows CNN's to obtain output distributions, but will also improve the prediction performance of the model. Note that in this work Monte Carlo dropout is only applied at the fully connected layer due to the fact that there still remains a big knowledge gap in literature about the effect of dropout in convolutional layers. Furthermore, during model development it was found that incorporating dropout in the convolutional layers would not increase model performance, both when dropping out entire feature maps and when dropping out individual neurons across all filters. The dropout will be applied at the fully-connected layer with the dropout probability being denoted as p_d.

3.2 Experimental set-up

This section will describe the experimental set-up of the test case. First, the data that will be used to test the developed method will be explained. After that, data pre-processing and sensor selection will be elaborated on. Then, the creation of input samples for the CNN will be explained. Lastly, an overview of the tuned CNN hyperparameters will be presented. In this work, all experiments that are conducted are ran on the HP ZBook Studio G3 with an Intel Core i7-6700HQ CPU. All the coding work is done in Python 4.0 where use is made of the Tensorflow keras API for creating the CNN architecture.

3.2.1 C-MAPSS dataset

The publicly available NASA turbofan engine degradation dataset is a popular benchmarking problem for comparing the performance of prognostic algorithms [3, 6]. This dataset is developed using a model-based simulation program called C-MAPSS (Commercial Modular Aero-Propulsion System Simulation), which is a tool made by NASA that is able to simulate degradation in turbofan engines in a Matlab and Simulink environment. This dataset is commonly referred to as the C-MAPSS dataset. This section will explain the structure of the dataset, including the use of the training and testing datasets. For a more in depth explanation of the dataset including all 21 sensors that the dataset consists of the reader if referred to Appendix B.

The complete C-MAPSS data set consists of 4 sub-data sets, named FD001, FD002, FD003 and FD004, where the difference between them is the number of operating conditions and fault modes. Each data set is made up of a training data set and a testing data set. The run-to-failure training data is simulated for multiple engines that start with varying degrees of wear and is stored as time series data for all 21 sensors. After some time, a fault develops in an engine and grows until the engine can no longer fulfil its intended function, after which the engine is declared unhealthy. The final recorded time step for each engine is the time at which the engine has failed. The training data set is run-to-failure and can be used to train the model. For the testing data set, the time series is terminated at some unknown point prior to failure, meaning that the engines in this dataset are thus not run-to-failure. The goal is to predict the remaining cycles until failure would occur given the time series sensor data of that engine up to that point. For each engine in the test dataset, a single prediction should thus be made at the moment the recorded sensor data is stopped. The correct RUL values of the test data are also provided. The data set is structured as a n-by-26 matrix, where n is the number of time cycles until failure for each engine.

For the 26 columns, the first column is the engine number, the second column is the operational cycle number, the third to fifth columns are the operational settings, and the remaining 21 columns represent the sensor measurement values. Table 1 gives a comprehensive overview of the number of engines in the C-MAPSS sub-data sets. Here, it can be seen that the engines in FD002 and FD004 contain multiple operating conditions, while the engines in FD001 and FD003 only have a single operating condition.

Table 1: Overview of the C-MAPSS dataset [3, 6]

C-MAPPS instance	**FD001**	**FD002**	**FD003**	**FD004**
Engines for training	100	260	100	249
Engines for testing	100	259	100	248
Operating conditions	1	6	1	6
Fault modes	1	1	2	2
Training samples	14184	36890	17456	53852
Validation samples	3574	9223	4364	10771

3.2.2 Data normalization

The C-MAPSS dataset contains time series recordings from a total of 21 different sensors [3]. The values recorded for all sensors are from different ranges and therefore is it beneficial to the training process to first normalize the input such that all sensor have the same range of values. In this paper, multiple ways of data normalization have been tried in order to see which one leads to be best model performance. The following data normalization and standardization methods were attempted in this work: normalization in range [-1,1] as shown in Equation 4, normalization in range [0,1] and standardization as shown in Equation 5.

$$x_{\text{norm}}^{i,j,k} = \frac{2(x^{i,j,k} - x_{\text{min}}^{j,k})}{x_{\text{max}}^{j,k} - x_{\text{min}}^{j,k}} - 1, \quad \forall i, j, k \tag{4}$$

$$x_{\text{norm}}^{i,j,k} = \frac{x^{i,j,k} - \mu^{j,k}}{\sigma^{j,k}}, \quad \forall i, j, k \tag{5}$$

In these equations, $x^{i,j,k}$ indicates the input value of the i-th data point of the j-th sensor used under operating condition k, and $x_{\text{norm}}^{i,j,k}$ indicates the normalized input value. $x_{\text{min}}^{j,k}$ and $x_{\text{max}}^{j,k}$ indicate the minimum and maximum value of sensor j under operating condition k respectively. $\mu^{j,k}$ and $\sigma^{j,k}$ indicate the mean and standard deviation of all recordings of sensor j under operating condition k respectively. This thus means that normalization is applied with respect to the current operating condition the engine is in, which makes sense as the ranges of sensor values might be vastly different when used in a different operational setting. In this paper, it was found that normalization in range [-1,1] obtained superior model results compared to the other normalization and standardization techniques attempted. Therefore, normalization as shown in Equation 4 will be the method used in the remainder of this paper.

3.2.3 Target labelling using piece-wise linear function

Each training sample is given a single output target label during training. This target label is equal to the true RUL of the engine that corresponds to the input sensor data. The CNN model can be seen as a regression model that attempts to learn the true RUL value based on recorded historical sensor data. When an engine is still healthy and has a high true RUL value, no clear degradation pattern can yet be seen in the sensor recordings. For example, both an engine with a true RUL of 400 FC and an engine with a true RUL of 200 FC might not yet show any signs of degradation, making it very hard for the model to predict a RUL value of 400 FC for the first engine, and 200 FC for the second one. For this reason, it is assumed that the RUL label during training for healthy engines can be set to a constant value. This approach can be seen in many other papers that use the C-MAPSS dataset, such as in the works of Li et al., Ellefsen et al. and TV. et al. [4, 9, 12]. The value for a constant early RUL R_{early} of 125 FC is used in this paper, which is a common value in literature. This thus means that for any engine that has a true RUL higher than 125 FC, the model will be trained to predict a RUL value of 125 FC. Because the target RUL is constant in the early phase and linearly decreasing when the true RUL becomes smaller than R_{early}, the target function used is piece-wise linear.

3.2.4 Selecting sensors to include as model inputs

Not all 21 sensors in the C-MAPSS dataset are equally useful. Some sensors only output a constant value or jump between stationary levels. Therefore, it is important to first determine which sensors should be used as input to the CNN for optimal prediction performance. An overview the sensor outputs over time for FD001 can be found in Appendix C. Most works such as the paper by Li et al. only use the sensors that are non-constant and continuous, those being sensors 2, 3, 4, 7, 8, 9, 11, 12, 13, 14, 15, 17, 20 and 21 [5]. In the recent work of Zhang et al., a full methodology on sensor selection is provided. In this paper, 4 metrics are developed that can be applied to time series sensor measurements in order to find which sensors reveal most information. The sensors are evaluated on correlation, monotonicity, robustness and predictability. Application of this methodology to the sensors in the C-MAPSS dataset causes only 9 sensors that show the clearest and

most consistent trend over time to be kept. For example, sensors number 8 and 13 are not used because they have high variance in correlation, and low predictability. Furthermore, sensors 9 and 14 are dropped due to the high variation in monotonicity and correlation, with even lower values of predictability [26].

To test the methods other works have used, the CNN was trained using several combinations of sensors inputs. The results are evaluated on the FD001 test dataset and are shown in Table 2. In Table 2 a very interesting phenomenon can be observed, namely that the model performance increases with more sensors added as model input. From this, it can be concluded that the CNN is able to extract useful information from all sensors that are not constant, even if a sensor only attains stationary values which is the case for sensor 17. For this reason, the CNN developed in this work will have input from a total of the 14 sensors corresponding to the top row of Table 2.

Table 2: Sensors included as model input versus model performance. The RMSE values are obtained on the FD001 test dataset

Sensors included as input	**Description**	**RMSE**
2,3,4,7,8,9,11,12,13,14,15,17,20,21	All except constant	12.16
2,3,4,7,8,9,11,12,13,14,15,20,21	All except constant and stationary	12.91
2,3,4,7,8,11,12,13,15,20,21	All except constant, stationary and sensors with inconsistent trend	14.02
2,3,4,7,11,12,15,20,21	9 sensors with most information [26]	14.61
7,12,15	3 sensors with clearest trend [27]	18.50

3.2.5 Creating convolutional neural network input samples

The model input is a 2D array of size $N_{tw} \times N_{ft}$, where N_{tw} equals the number of historical flight cycles included and N_{ft} equals the number of features (sensor signals) included. As explained in the previous subsection, the number of sensors included in the input N_{ft} is 14. To find the optimal number of historical flight cycles to include as model input, the work of Li et al. can be consulted [5]. In their sensitivity analysis, they find that up to a value of $N_{tw} = 30$ the model performance significantly increases. Adding more than 30 previous flights cycles as input does no longer improve the results while it increases computational time. Therefore, this work will use a default value of $N_{tw} = 30$. Some instances, such as FD002 and FD004, contain engines in the training set have have only been used for 21 FC and 19 FC respectively. If 30 historical FC are required as input, these engines can not be evaluated by the model. Therefore, additional models are trained for FD002 and FD004 with $N_{tw} = 21$ and $N_{tw} = 19$ respectively in order to be able to make comparisons with related works.

Furthermore, instances FD002 and FD004 are more complex due to the number of operating conditions the engines are used in. The first solution in dealing with this is explained in subsubsection 3.2.2, which is to apply data normalization with respect to the engine's current operating condition. Furthermore, the history of each engine in each of the 6 operating conditions can be added as additional inputs to the network. This method was also used by Babu et al. and was shown to yield better prediction results [4]. For instances FD002 and FD004 N_{ft} is thus equal to 20, where 14 features are the sensor time series recordings and the other 6 features are the number of flight cycles spend in each operating condition. These additional inputs are believed to improve the prediction performance on instances FD002 and FD004. A visual representation of the selected and normalized sensors as well as the normalized operating condition input for FD004 can be found in Appendix D.

3.2.6 Tuned CNN hyperparameters

The final step before the CNN can start making predictions is to tune the hyperparameters such that the performance is optimized. Neural network hyperparameter tuning is still an active area of research and trial and error is one of the methods that is commonly used. In this work, the most important hyperparameters were tuned by varying them along a specified range and checking the influence on the model performance. The works of Babu et al. and Li et al. were used as a starting point for the hyperparameter values because they achieved good prediction results with their CNN architectures [4, 5].

The CNN developed in this paper will have the architecture as shown in Figure 2. The network has 4 convolutional layers where the number of feature maps generated at each layer $F_N = 10$. The activation function used at these layers the hyperbolic tangent, or simply tanh. The filter size is set to 10×1, meaning that the filter length $F_L = 10$. After those 4 layers, a single convolutional layer is placed that generates only 1 feature map. The filter size used in this final convolutional layer is 3×1, meaning that $F_{L,\text{final}} = 3$. The dropout rate in the fully connected layer (FCL) is found to be optimal at at value of $p_d = 0.5$ and this FCL is made up of $n_{FCL} = 100$ neurons. At the FCL, the activation function is also tanh. At all layers, the padding

is kept at 'same', meaning that the size of the feature map that is generated has the same dimensions as the input from the previous layer. The network training is performed with a batch size of 256 and for 250 epochs per model. The initial learning rate is set at $lr = 0.001$ and is multiplied by a factor of $lr_f = 0.6$ after $lr_p = 10$ consecutive epochs without improvement. An overview of the tuned hyperparameters and other important model parameters can be seen in Table 3.

Table 3: Summary of tuned parameters and other inputs of the CNN model developed in this paper

Model parameter	**Symbol**	**Value**	**Model parameter**	**Symbol**	**Value**
Number of feature maps generated at conv. layer	F_N	10	Number of historical flight cycles as input for FD001 to FD004	N_{tw}	30/21/30/19
Filter length	F_L	10	Dropout rate	p_d	0.5
Filter length final conv. layer	$F_{L,\text{final}}$	3	Validation split	-	0.2
Number of input features for FD001 to FD004	N_{ft}	14/20/14/20	Initial learning rate	lr	0.001
Early constant target RUL	R_{early}	125	Learning rate reduction patience	lr_p	10
Convolutional layers	L_C	5	Learning rate reduction factor	lr_f	0.6
Neurons in fully-connected layer	n_{FCL}	100	Batch size	-	256
Input normalization range	-	[-1,1]	Number of epochs	-	250

3.3 Prognostic algorithm performance evaluation

This section will explain how the performance of a prognostic algorithm that predicts a component's RUL can be measured. This will be done by first analyzing why specific metrics are useful for performance evaluation and what the drawbacks are when using these metrics. Next, a variety of general error metrics and prognostic algorithm specific metrics will be discussed. Also, requirements that are specific to prognostic metrics are stated. Lastly, it is stated how probabilistic predictions and uncertainty estimations can be incorporated into the evaluation metrics. The metrics introduced in this section will be used in the next section to present the results on the developed CNN model.

3.3.1 Strengths and limitations of performance metrics

Performance metrics are used to reveal information on the performance of the model that is developed. Metrics can therefore be specifically designed to indicate the strengths and weaknesses of a model on a variety of test cases. Over the past years, attempts have been made to develop novel metrics that can be used to further analyze the performance of prognostic algorithms for adoption in practice. These metrics are excellent to compare different models and algorithms with one another, but lack to ability to reveal for a single model how well it can perform in a real world scenario. For example, if an airline wants to use data driven RUL predictions for maintenance scheduling in practice, it will first need to develop multiple prognostic models that can compute RUL predictions. Metrics can then be used to compare the different models and select a single prognostic model that outperforms the other competitors. However, these metrics do not yet reveal much information on how reliable and with what accuracy this prognostic model can be adopted into the daily scheduling routine. Therefore, prognostic performance metrics can mainly be used to compare different models and algorithms with each other. To test if prognostic models can be used in a daily scheduling routine, other methods such as Monte Carlo simulations can be used. This will be explored further in section 4.

3.3.2 General error metrics

The most widely used general error metrics for prediction models can be found in Appendix A. In this table, the definition of each metric is presented, as well as the range and further references. The most important metrics from this table will be highlighted in this section. The most commonly used error metric in prognostics is the root mean squared error (RMSE), which is computed by squaring the error for each prediction, then taking the mean over all samples and finally taking the square root of that. A true positive (TP) is defined as a prediction that has a positive error ($\epsilon(i) \geq 0$, so early prediction) that is smaller than the acceptable early prediction error bound t_{FP}, while a true negative (TN) is defined as a prediction that has a negative error ($\epsilon(i) < 0$, so late prediction) that is smaller than the acceptable late prediction error bound t_{FN}. A false positive (FP) and a false negative (FN) are defined as predictions that have a unacceptable early or positive error respectively. Note that in the remainder of this paper not all of the metrics included in Appendix A will be used. A selection of the most used general metrics will be made and used for the model performance evaluation.

Besides the general error metrics shown in Appendix A, other metrics exist that can be used to reveal information on the model performance. Three commonly used metrics for evaluating machine learning algorithms

are accuracy, precision and recall and are defined in Equation 6-4. These metrics make use of the previously defined metrics TP, TN, FP, FN and the number of predictions N. The final metric that can be used is called the F1-score and is defined as the harmonic mean of the precision and recall as shown in Equation 9.

$$\text{accuracy} = \frac{TP + TN}{N} \quad (6) \qquad \text{precision} = \frac{TP}{TP + TN} \quad (7) \qquad \text{recall} = \frac{TP}{TP + FN} \quad (8)$$

$$F1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} = \frac{TP}{TP + 0.5 \cdot (FP + FN)} \quad (9)$$

3.3.3 Metrics for prognostic algorithm performance assessment

In the field of prognostics it is important to evaluate models using metrics that are specific to the problem. Over the past years, metrics have been developed that can take into account requirements that are specific to prognostic predictions. The most important performance requirements for models that perform RUL prediction are as follows [14]:

1. In RUL prediction, a late prediction is usually considered to be worse than an early prediction. Late predictions may cause high safety risks because a component is used in highly deteriorated state, while an early prediction only leads to unnecessary maintenance causing higher costs.

2. The error of a prediction should decrease over time. If an error of for example 20 flight cycles (FC) is achieved when the true RUL is 200 FC, this is not a big problem because the engine is not near failure and maintenance is not needed for a long period of time. However, if the same error of 20 FC is achieved with a true RUL of 5 FC, then it might happen that maintenance is scheduled based on an incorrect prediction.

From these requirements specific metrics have been developed that can be used to present a better picture of the prognostic model performance. These metrics will be explained in the following paragraphs.

PHM008-challenge score

The first metric that is used in this work is the PHM008-challenge score and is defined in Equation 10. This is an asymmetric scoring function that penalizes a late prediction harder than an early prediction by setting the values of γ_l and γ_e. Note that $\epsilon(i) < 0$ indicates a late prediction and $\epsilon(i) \geq 0$ indicates an early prediction. The default values for γ_l and γ_e used in the PHM008-challenge are 13 and 10 respectively. Furthermore, one can extent this metric by weighting this obtained score value s using the true RUL value of the engine that the prediction is made on. The definition of the weighted score, abbreviated in this work as W. score, is shown in Equation 11. Engines that have a low true RUL and thus are closer to failure are given a higher weight than engines that are still healthy and have a high true RUL. The weighting factor of $e^{-w_s * r_{\text{true}}(i)}$ ensures that the second requirement posed on prognostic models is also incorporated. The degree to which engines with lower true RUL $r_{\text{true}}(i)$ are weighted heavier can be tuned using the parameter w_s, which is set at 0.025 as default.

$$s = \begin{cases} e^{-\frac{\epsilon(i)}{\gamma_l}} - 1 & \text{for} \quad \epsilon(i) < 0 \\ e^{-\frac{\epsilon(i)}{\gamma_e}} - 1 & \text{for} \quad \epsilon(i) \geq 0 \end{cases} \quad (10)$$

$$\text{W. score} = s \cdot e^{-w_s * r_{\text{true}}(i)} \quad (11)$$

Prognostic Horizon

The second metric is the Prognostic Horizon (PH) and is visualized in Figure 3. This metric is equal to the time difference between the time the RUL prediction enters a specified performance criteria and the component end of life (EoL). In Figure 3, the black line indicates the component true RUL. The grey areas around this line represent areas of constant error which are considered acceptable upper and lower bounds for the RUL prediction. As soon as a PA outputs a RUL estimation that is within the grey area, the prognostic horizon can be computed. Note that it may occur that a prediction at a certain time is within the grey zone, but a prediction made at a later time is no longer within the grey zone. The PH can then be defined as the time difference between the time the RUL prediction is within the grey area and no longer leaves this area at a later time, and the EoL. A high value for PH indicates that there is more time to act for an airline based on a prediction within a certain level of accuracy [13].

$\alpha - \lambda$ accuracy

The third metric of importance is the $\alpha - \lambda$ metric, and is shown in Figure 4. This metric is based on the point that the performance of the PA should increase over time, as predictions near component EoL are vital for airline operational planning. A specific form of this metric is $\alpha - \lambda$ accuracy, which indicates that the RUL prediction should fall between specific α accuracy bounds that get tighter around the true RUL when time progresses. In Figure 4, the blue line indicates the true RUL and the blue shaded area shows the α bounds that get closer to the true RUL line over time. λ is a time-window modifier that can be chosen such that a specific accuracy α is achieved at a specific time instance. For example, a requirement that can be set is that the accuracy halfway of the true RUL should be 20%. If the prediction at a certain time falls within the α accuracy bounds, the $\alpha - \lambda$ accuracy takes a value of 1, and takes value 0 if the prediction is outside the α bounds. This metric can be used to compute what percentage of all predictions meet the $\alpha - \lambda$ accuracy requirement, which reveals information on how useful and reliable the prognostic algorithm is [13].

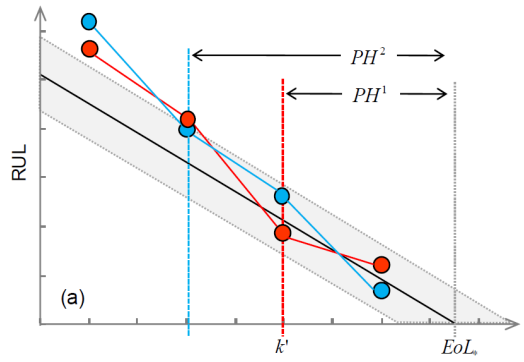


Figure 3: PH metric for point estimate [13]

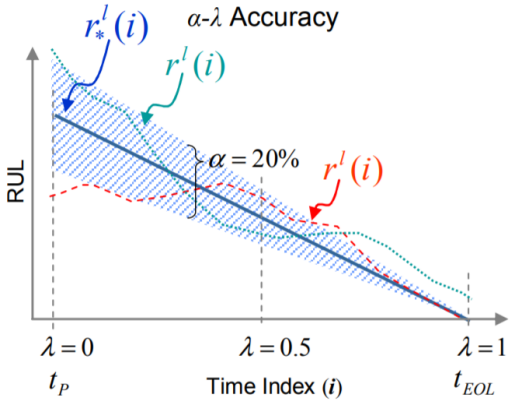


Figure 4: $\alpha - \lambda$ metric for point estimate [13]

Relative accuracy

The fourth metric used in this work is the Relative Accuracy (RA). This metric is defined as the measured error relative to the true RUL value at a specific moment in time i_λ. The RA is defined in Equation 12.This metric is similar to the $\alpha - \lambda$ metric, but the difference is that while the $\alpha - \lambda$ metric only tells whether a prediction in inside the α bounds or not, the RA gives a quantitative value that indicates how close a prediction is to the actual value. The RA thus reveals information on the accuracy at a specific time. The Cumulative Relative Accuracy (CRA) is defined as the weighted sum of several RA values at different time points, normalized by the number of RA values. If all weights are set equally, the CRA takes the form as defined in Equation 13. In this definition, p_λ is the set of all time indexes at which a prediction is made and $|p_\lambda|$ is length of the set. If the weighted variant of the CRA is applied, denoted in this work as WCRA, the values of $w(r(i))$ can be altered to give more weight to predictions on engines that are closer to failure as is shown in Equation 14 [13].

$$RA_\lambda = 1 - \frac{|r_{\text{true}}(i_\lambda) - r_{\text{pred}}(i_\lambda)|}{r_{\text{true}}(i_\lambda)} \tag{12}$$

$$CRA_\lambda = \frac{1}{|p_\lambda|} \sum_{i \in p_\lambda} RA_\lambda \tag{13}$$

$$WCRA_\lambda = \frac{\sum_{i \in p_\lambda} w(r(i)) \cdot RA_\lambda}{\sum_{i \in p_\lambda} w(r(i))} \tag{14}$$

Sampling rate robustness

The sampling rate robustness tests how sensitive a prognostic algorithm is to the frequency of evaluating new input. The definition of SSR is presented in Equation 15, where $f1$ and $f2$ represent two different sampling frequencies. For example, one can evaluate the MAE for a series of predictions sampled after each flight cycle and also evaluate the MAE for a series of predictions sampled only at each 5th flight cycle. If the absolute difference in MAE using the different sampling frequencies is large, the model has high sensitivity to sampling frequency which is undesired [13].

$$SSR = |MAE_{f1} - MAE_{f2}| \tag{15}$$

Convergence

The convergence metric is able to compute a single value that indicates how quickly any metric M improves with time. Figure 5 shows how 3 different arbitrary performance measures evolve over time, where it can be seen that the blue metric does not decrease significantly, while the red and green metrics end up at a value near 0 at component EoL. The convergence metric determines how quickly each metric reached 0 by computing the center of mass of the area below the curve of each metric. In Figure 5 it can be seen that the red metric has the fastest convergence as its center of mass is closest to point t_p. A fast convergence of any error metric is desired as this means that the initial prediction error decreases with decreasing engine true RUL [13].

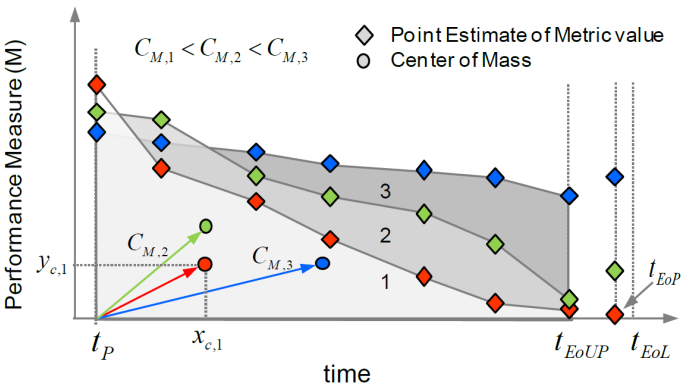


Figure 5: Convergence metric [13]

3.3.4 Incorporating uncertainty estimates

One of the novel contributions of this work is that RUL predictions are given in the form of a probabilistic distribution instead of just a single-valued prediction using the Monte Carlo dropout sampling method. This output requires some of the performance metrics to be adjusted such that they are able to evaluate probabilistic predictions. This subsection will elaborate on how the previously introduced metrics can be applied to this type of output. Furthermore, a novel error metric called the continuous ranked probability score will be introduced that can be applied specifically to predictions that have a probabilistic form.

Predictions in the form of a probability distribution can be evaluated by defining an upper and lower bound around the true RUL value, denoted by α^+ and α^- respectively. The probability mass of the prediction then has to be higher than a specified level β in order for the prediction to be considered to be within the α-bounds. This is mathematically denoted in Equation 16, where $\pi[r(i_\lambda)]_{\alpha-}^{\alpha+}$ denotes the probability mass between α^+ and α^- bounds for the RUL distribution prediction $r(i_\lambda)$ [13].

$$\pi[r(i_\lambda)]_{\alpha-}^{\alpha+} \geq \beta \tag{16}$$

Accuracy, prognostic horizon and $\alpha - \lambda$ accuracy for probability distributions

This concept can be used for all previously defined metrics that make use of acceptable error bounds, such as the TP, TN, FP and FN metrics. A predicted probability distribution for an engine can be considered a TP if at least a probability mass of for example $\beta_{TP} = 0.5$ is located between the acceptable early prediction error bound t_{TP} and the true RUL value at that moment and can be considered a TN if at least a probability mass of for example $\beta_{TN} = 0.5$ is located between the true RUL value at that moment and the acceptable late prediction error bound t_{TN}. The probabilistic form of the prognostic horizon metric can be observed in Figure 6. For this metric, a predicted distribution is considered to lie within the acceptable error bounds α^+ and α^- if at least a probability mass of β_{PH} is located between them. The same principle holds for the probabilistic form of the $\alpha - \lambda$ metric shown in Figure 7, where at least a probability mass of $\beta_{\alpha-\lambda}$ is required to lie between the upper and lower error bound in order for the prediction to be classified as acceptable [14].

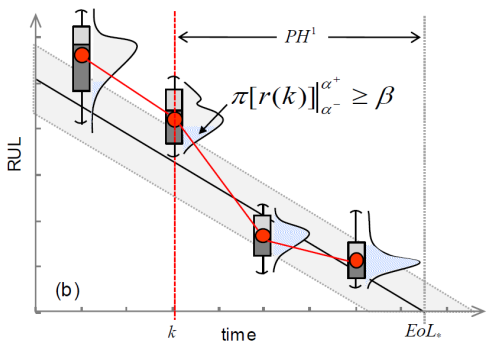


Figure 6: PH metric for PDF prediction [14]

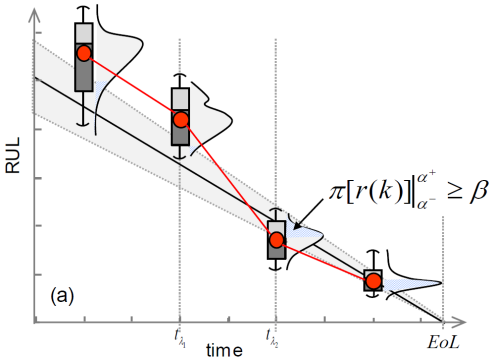


Figure 7: $\alpha - \lambda$ metric for PDF prediction [14]

Continuous ranked probability score

Another metric that can evaluate probabilistic predictions is the continuous ranked probability score (CRPS). This metric returns a single error value and requires as input the CDF of the predicted distribution and the true RUL value. The definition of the CRPS metric can be found in Equation 17, where F is the CDF of the predicted distribution and $\mathbb{1}$ is the Heaviside step function that equals 1 if $(y-x) \geq 0$ and equals 0 if $(y-x) < 0$, where x is the true RUL value [16]. This step function is thus 0 at all values smaller than the true RUL and 1 at all values larger than the true RUL. This function can be evaluated numerically by splitting the integral in two parts, one from minus infinity to x and one from x to plus infinity. The CRPS metric returns a single value that computes the area between the predicted CDF and the Heaviside step function and is thus able to evaluate the performance of algorithms that produce probabilistic output. For a single-valued prediction, the CRPS reduces to just the MAE.

$$\text{CRPS}(F, x) = \int_{-\infty}^{\infty} (F(y)^2 - \mathbb{1}(y - x))^2 dy \tag{17}$$

The final metric introduced in this paper is the weighted scored CRPS. This metric is similar to the weighted score metric introduced in subsubsection 3.3.3, but now the CRPS value computed using Equation 17 is used in Equation 11 in stead of the regular error ϵ. This metric now combines all important requirements that were determined in this section: it penalizes late predictions harder than early predictions due to the asymmetric scoring function. Moreover, it exponentially weighs engines that are near EoL (engines with low true RUL r_{true}) heavier because of the weighting factor w_s. Lastly, because the CRPS value is now used in stead of the error value, this metric also takes into account the predicted probability distribution in stead of just a single-valued prediction.

$$\text{W. CRPS}(F, x) = e^{-w_s * r_{\text{true}}(i)} \cdot \begin{cases} e^{-\frac{\text{CRPS}(F,x)}{\gamma_l}} - 1 & \text{for} \quad \epsilon(i) < 0 \\ e^{-\frac{\text{CRPS}(F,x)}{\gamma_e}} - 1 & \text{for} \quad \epsilon(i) \geq 0 \end{cases} \tag{18}$$

3.4 Experimental results and analysis

In this section, the results of the developed model on the C-MAPSS test dataset are shown. This will be done by making use of the metrics defined in subsection 3.3. First, the results on all 4 instances of the C-MAPSS test data set will be shown. Specific examples will be further analyzed to show what the model output looks like. Lastly, the prediction results obtained in this work will be compared to related previous works. The reasons as to why this paper obtains superior RMSE values will be briefly discussed.

3.4.1 Results on C-MAPSS test dataset

The results of the CNN model on the C-MAPSS test data set for all 4 instances are shown in Table 4. In this table all metrics introduced in subsection 3.3 are included, except the ones that are specific to making a series of predictions over time for a single engine. The C-MAPSS test dataset consists of a number of engines that are used for a certain number of flight cycles, and the sensor recordings are stopped at some unknown moment prior to failure. The question then is to predict for each engine the RUL at that moment in time. This thus means that for each engine in the test dataset, only one RUL estimation is made, meaning that metrics such as the prognostic horizon are not suitable. Only metrics are included that are able to evaluate predictions made at a single moment in time.

The default values of $\gamma_l = 13$ and $\gamma_e = 10$ are used for computing the score metric. For the $\alpha-\lambda$ metric, both the single-valued mean prediction variant and the distribution variant are used, indicated with by $\alpha - \lambda$(d) and $\alpha - \lambda$(p) respectively. The $\alpha - \lambda$(d) metric determines the percentage of predictions for which the distribution mean lies inside a margin of 30% from the true RUL. The $\alpha - \lambda$(p) metric determines the percentage of predictions for which at least a probability mass of $\beta_{\alpha-\lambda} = 0.75$ lies inside a margin of 30% from the true RUL. For computing the TP, TN, FP and FN, a required probability mass of $\beta_{TP} = \beta_{TN} = \beta_{FP} = \beta_{FN} = 0.5$ and an acceptable error margin of 30% are used. Lastly, the weighting factor used to compute the W. score and the W. CRPS is set to $w_s = 0.025$.

Furthermore, for FD002 and FD004 two different series of results can be observed, one without an * and one with an *. The * indicates that the results are shown for the model that is trained using a input time window size of 30 previous flight cycles of sensor data. As described in subsubsection 3.1.2, this time window size was found to lead to lowest RMSE as shown by Li et at. and is therefore used as the default time window size in this paper [5]. However, instances FD002 and FD004 contain test engines that have not been used for 30 flight cycles yet and therefore the model can not make predictions on all test engines. For this reason, two additional models are trained for FD002 and FD004 with an input time window size of the engine that is used for the

fewest number of flight cycles in these instances, being 21 and 19 flight cycles respectively. These models are now able to make predictions on all engines included in the test dataset and are shown in Table 4 as FD002 and FD004 without the *. These models will also be used when comparing the results obtained in this paper to related works.

Regarding the results of the metrics shown in Table 4, one can see that the developed model performed well. Across all instances, the MAE is between 8.36 FC and 12.06 FC, which can be considered low. Another interesting observation is that the MAE is lowest for FD003, while FD003 does not obtain the lowest score value. This indicates that even though FD003 has the lowest absolute error in its predictions, the predictions are often late ($\epsilon(i) < 0$) in stead of early ($\epsilon(i) \geq 0$). This is undesired and reflected upon in the higher score value. This example shows the added benefit of evaluating the model output on multiple metrics in stead of just a single one such as the RMSE.

Table 4: Results on the 4 instances of the C-MAPSS test data set shown using the most important prognostic performance metrics. The * after FD002 and FD004 indicates that the ideal input time window size of 30 is used and thus not all engines in the test data set could be evaluated. Network training time for 250 epochs is included in the bottom-right column in seconds

	Prognostic metric									
Instance	**MAE**	**MAPE**	**RMSE**	**ESTD**	**FP**	**FN**	**TP**	**TN**	**Score**	**W. score**
FD001	8.99	15.08	12.16	8.2	0.07	0.08	0.33	0.43	2.33	1.32
FD002	10.46	19.28	14.71	10.34	0.07	0.13	0.39	0.26	4.05	2.48
FD002*	9.88	17.56	13.44	9.11	0.07	0.13	0.4	0.27	3.12	1.84
FD003	8.36	13.58	12.01	8.62	0.01	0.08	0.35	0.48	3.39	2.22
FD004	12.06	26.53	16.57	11.37	0.04	0.26	0.39	0.26	7.4	6.43
FD004*	10.22	19.55	14.79	10.69	0.03	0.18	0.43	0.27	5.45	3.62

Instance	**Acc.**	**Prec.**	**Recall**	**F1**	$\alpha - \lambda$**(d)**	$\alpha - \lambda$**(p)**	**CRA**	**CRPS**	**W.CRPS**	**T. time [s]**
FD001	0.84	0.43	0.8	0.81	0.84	0.73	0.85	10.1	1.81	1923
FD002	0.76	0.6	0.75	0.79	0.8	0.59	0.81	11.52	3.57	5583
FD002*	0.77	0.6	0.75	0.8	0.8	0.62	0.82	10.74	2.63	7131
FD003	0.9	0.42	0.81	0.89	0.9	0.78	0.86	9.75	3.13	2402
FD004	0.69	0.6	0.6	0.73	0.71	0.6	0.73	13.2	9.02	7198
FD004*	0.77	0.61	0.71	0.8	0.78	0.67	0.8	11.23	4.83	8576

The results can be further observed by plotting the predicted RUL distributions and true RUL values for all engines in instance FD001 as can be seen in Figure 8. This figure shows all test engines included in FD001 as a single point, sorted by true RUL. The value on the x-axis indicates the index of that test engine in the FD001 test dataset. Every blue dot thus shows the true RUL of an engine at the moment the sensor recordings have terminated, and the red violin shows the predicted probability distribution generated using the CNN. The red dot inside the violin indicates the mean of the predicted distribution. As can be seen, the predictions on FD001 are very close to the true RUL values. Especially for the engines that have a true RUL smaller than 40 FC, the prediction means have an error of just a few flight cycles, indicating a very accurate prediction. This observation might be useful when using the CNN for maintenance scheduling, as one knows that if the models predicts low RUL values for an engine, it is likely that the true RUL is also low for that engine. This observation will be tested later in this paper when maintenance scheduling using CNN predictions is considered in more detail. The same figures for the other 3 instances can be found in Appendix E.

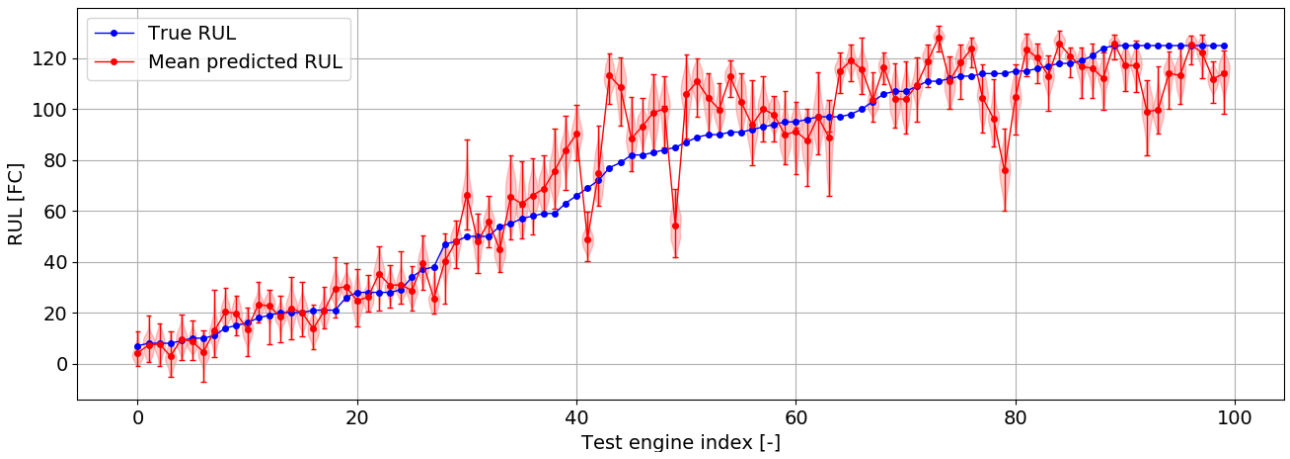
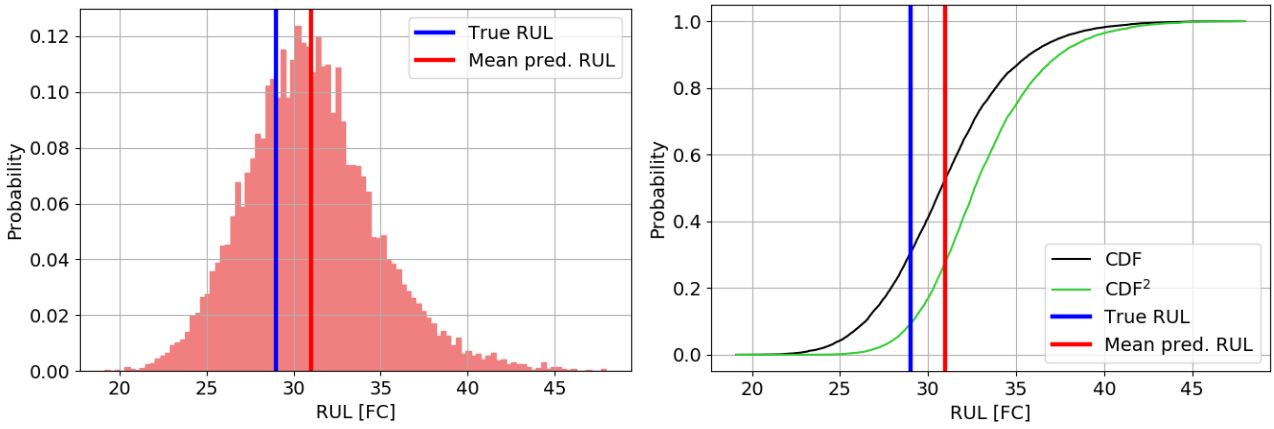


Figure 8: True RUL and predicted RUL distributions for all 100 engines in FD001 sorted by true RUL

Lastly, the predicted RUL distribution for a single test engine is shown in 9(a). This figure shows in detail the distribution that is obtained after making 10,000 predictions using Monte Carlo dropout sampling for test engine 24 from instance FD001. The mean of the distribution and the true RUL of that engine at that moment in time are also shown using a red and blue vertical line respectively. It can be seen that after a large number of MC predictions, the output distribution highly resembles a normal distribution. In addition, the CDF and CDF^2 of the predicted RUL distribution are shown in 9(b). The CDF and CDF^2 are important because they are needed to compute the CRPS metric and they are therefore included here. The same figures for engines 1, 56 and 80 can be found in Appendix F to also show the distributions of engines at different true RUL values.



(a) Histogram of 10,000 MC predictions

(b) CDF and CDF^2 of 10,000 MC predictions

Figure 9: MC prediction results on test engine 24 from FD001

3.4.2 Comparison to related works

This work is not the first to make use of the C-MAPSS dataset. Over the past years, other attempts have been made to develop prognostic algorithms that predict the RUL of the engines included in the test dataset. Table 5 shows the most relevant and recent papers that have attempted to develop prognostic algorithms and tested them using the C-MAPSS dataset. The performance of the methods used in these works is shown using the RMSE on all 4 instances of the C-MAPSS dataset in order to present a complete picture of the methods performance. This single metric is chosen because most related works only show a small number of performance metrics and the RMSE is most widely used amongst them. Note that for the RMSE values obtained in this work, the values are used that are obtained when including all the test engines for evaluation. This means that the values are taken from Table 4 where the instance without * is shown.

It can be observed in Table 5 that the method used in this paper outperforms all related works in terms of RMSE across all instances. This can first of all be attributed to the fact that a CNN is proven to be an excellent method for RUL prediction by Li et al. [5]. Furthermore, this work used extensive experimentation on data pre-processing and normalization, sensor selection and input sample generation. This lead to the combination of an optimized CNN architecture and a careful training sample generation process, resulting in lower prediction error across all dataset instances. Furthermore, the use of Monte Carlo dropout sampling also may have caused the RMSE values obtained in this paper to be lower. The final row of the table shows the improvement in RMSE when comparing the results obtained in this work to the lowest RMSE value obtained in any of the previous related works. It can be seen that the improvement is highest for instances FD002 and FD004, being 24.91% and 25.19% respectively. This can be attributed to the normalization of the data with respect to the current operating condition and to the fact that the operating condition history is added as additional model input. Many other works do not make use of this operating condition partitioning and thus achieve higher prediction errors on instances FD002 and FD004. Finally it must be noted that not all papers make use of an early constant RUL value. This does have an effect on the model RMSE values as models that do not use this early constant RUL value suffer an increase in RMSE. Despite this, the values shown in Table 5 are still competitive and can be used for a good comparison.

Table 5: Overview of most relevant and recent RUL prediction attempts on the C-MAPSS test dataset (ordered by year of publication). The lowest RMSE values from other works are shown in blue, while the RMSE values from this paper are shown in green. The final row shows the improvement in RMSE when comparing the blue value and the green value, where the improvement is equal to $1 - Blue/Green$

Authors	**Paper**	**Year**	**Method**	**RMSE FD001**	**RMSE FD002**	**RMSE FD003**	**RMSE FD004**	$\mathbf{R}_{early}$
Ramasso	[28]	2014	RULCLIPPER	13.27	22.89	16.00	24.33	135
Babu et al.	[4]	2016	Deep CNN	18.45	30.29	19.82	29.16	Not provided
Malhotra et al.	[29]	2016	LSTM-ED	12.81	-	-	-	125
Zhang et al.	[30]	2016	MODBNE, no TW	17.96	28.06	19.41	29.45	Not applied
Zhang et al.	[30]	2016	MODBNE, with TW=30	15.04	25.05	12.51	28.66	Not applied
Zhang et al.	[30]	2016	DBN	15.21	27.12	14.71	29.88	Not applied
Zheng et al.	[8]	2017	LSTM	16.14	24.49	16.18	28.17	130
Li et al.	[5]	2018	Deep CNN with R_{early}	12.61	22.36	12.64	23.31	125
Li et al.	[5]	2018	Deep CNN, no R_{early}	13.32	24.86	14.02	29.44	Not applied
Ellefsen et al.	[9]	2019	RBM + LSTM + GA	12.56	22.73	12.10	22.66	130
Aremu et al.	[10]	2020	MLDR + classifier	-	35.27	-	51.58	Not applied
Yu et al.	[11]	2020	SBI EN	13.58	19.59	19.16	22.15	150/200
TV. et al.	[12]	2021	LSTM-ORCE	14.62	-	-	27.74	130
TV. et al.	[12]	2021	LSTM-MR	15.62	-	-	26.88	130
This work	-	2021	Deep CNN	12.16	14.71	12.01	16.57	125
			Improvement	3.18%	24.91%	0.74%	25.19%	

4 Maintenance Scheduling Using Data-driven RUL Prognostics

In current literature, there still is a knowledge gap on how a time series of RUL estimations for a component can be used in an actual scheduling scenario. This section will explore how the data-driven probabilistic RUL predictions obtained using the CNN model can be used in a realistic scheduling simulation. The goal is to develop novel methods and policies that are able to use RUL estimations in daily maintenance scheduling operations in order to maximize profit and minimize safety risks. First, the methods used in this section are described. This includes the maintenance policies that are developed, as well as the methods used for maintenance scheduling optimization and simulation. A genetic algorithm that is used to optimize the maintenance policies is also presented. Next, the set-up of the experimental test case is described. After this, the performance evaluation of the scheduling simulation is presented. Lastly, the results of both the optimized policies and the collected performance indicators are presented and discussed. Also, a sensitivity analysis is included.

4.1 Methods

This section will cover the methods developed to simulate maintenance scheduling using data-driven RUL predictions. First, several maintenance scheduling policies that determine when maintenance should be scheduled will be highlighted. Next, an integer linear programming (ILP) model is developed that is able to allocate an aircraft that requires maintenance to an optimal opportunity in the flight schedule. Simulating this process in the long run is done using a rolling horizon method that is explained in the next section. After that, an overview of the scheduling simulation is presented. Finally, a genetic algorithm (GA) is presented that is used to optimize the parameters of the data-driven maintenance policies.

4.1.1 Determining when to schedule maintenance

The first aspect that will be elaborated on is how a series of RUL predictions for an aircraft engine can be used to determine when maintenance should be scheduled. After each flight cycle an aircraft has flown, new sensor data is recorded for all of its engines. The CNN developed in section 3 can then be used to predict a RUL probability distribution for all of the aircraft engines after that flight cycle. Over time, a series of probabilistic RUL predictions is formed that can be used to predict when the engine is required to undergo maintenance. Determining how to go from this series of predicted RUL distributions to a window in which maintenance should be scheduled is done using a maintenance policy. This paper will investigate a total of 3 different maintenance policies. These 3 maintenance policies are denoted as follows: data-driven (DD), advanced data-driven (ADD) and inspection based (IB). A maintenance policy can be denoted as a vector of parameters that takes as input the predicted RUL distributions and then answers the following questions: (i) does maintenance need to be scheduled for this engine, and if so (ii) at what moment in time should maintenance be scheduled. The 3 policies

that are investigated in this work consist of a subset of the parameters shown in Table 6. This table forms the basis of the policies that will be elaborated on in the next paragraphs as it explains how each parameter contributes to determining the maintenance window. Note that all of the 3 policies are engine-specific, meaning that a policy will determine for each engine individually the moment in time maintenance is required. From this it follows that if one of the engines of an aircraft requires maintenance, the aircraft can not fly and has to visit a hangar.

Table 6: Parameters of the developed data-driven and inspection based scheduling policies and their eludication

Policy parameter	Eludication
TH_{mo}	Threshold below which the RUL prediction must lie with at least a probability mass of p_{mo} to generate an alert to create MO's
nA_{mo}	Number of alerts needed to create MO's in the far future using the mean of the latest RUL prediction
p_{mo}	Probability mass required to be below TH_{mo} in order to generate an alert
nMO	Number of MO's created around the early estimated RUL when at least nA_{mo} alerts are generated
TH_s	Threshold below which the RUL prediction must lie with at least a probability mass of p_s to generate an alert to schedule maintenance
nA_s	Number of alerts needed to schedule maintenance using the mean of the latest RUL prediction
p_s	Probability mass required to be below TH_s in order to generate an alert
f_s	Factor by which the latest RUL prediction is multiplied if at least nA_s alerts are generated. Used to compute $tRUL$
sm_s	Safety margin subtracted from the latest RUL prediction if at least nA_s alerts are generated. Used to compute $tRUL$
Int_I	Inspection interval for inspection based policies
TH_I	Threshold used for inspection based policies. Maintenance is scheduled as soon as the estimated RUL obtained via inspection is lower than TH_I

Besides these 3 maintenance policies, 2 types of flight schedules will be considered in this paper. The first type is a flight schedule that contains a standard maintenance opportunity during which maintenance can be carried out every week. During this period the aircraft is located at the base airport for sufficient time for maintenance to be scheduled and can thus not be utilized to generate revenue. The second type is a flight schedule that is completely filled with flights, maximizing the aircraft utilization and revenue. The downside of this type of flight schedule is that if maintenance is required, a flight pair of 2 flight cycles is required to be cancelled to create a maintenance opportunity. These 2 types of flight schedules used in this work are denoted as gap (G) or no gap (NG). This work will investigate a total of 5 different maintenance strategies that are obtained by combining a maintenance policy and a type of flight schedule. Note that a maintenance policy thus determines the maintenance window during which maintenance should occur for an engine based on RUL estimations, while a maintenance strategy covers both the maintenance policy that is applied and the type of flight schedule that is used. The next paragraphs will explain the maintenance policies that are developed in this work, as well as the type of flight schedule they can be combined with to form a maintenance strategy.

Data-driven maintenance scheduling policy for a fixed flight schedule

The first policy that is investigated in this paper is the data-driven policy (DDP). First, it will be explained how this policy uses a series of data-driven probabilistic RUL predictions to determine if and when maintenance should be scheduled. After that, it will be explained how this policy can be used in combination with both the gap and no gap flight schedule.

How the DDP uses RUL predictions to estimate the window in which maintenance can be scheduled will be explained using Figure 10. In this figure, a series of probabilistic RUL predictions over time shown as red violins can be seen for a single engine. In blue, the true RUL values of the engine are shown. The DDP considers an engine for maintenance scheduling if there are nA_s consecutive predictions with a probability mass of at least p_s

below the scheduling threshold value TH_s. A single prediction with a probability mass of p_s below the threshold TH_s is referred to as an alert. In Figure 10 example values of $TH_s = 25, p_s = 0.7$ and $nA_s = 3$ are used to explain the procedure. It can be observed that at the moment this specific engine is used for 146 FC, there are 3 consecutive violins with a probability mass of more than 0.7 below the RUL threshold of 25 FC. This means that this engine is sufficiently near failure that maintenance should be scheduled for this engine. The DDP now computes the target RUL of engine i located on aircraft k, denoted as $tRUL_{k,i}$, using the prediction that was required to generate the final alert. The target RUL corresponds to the latest moment in time maintenance can be performed on this engine and is computed in Equation 19.

$$tRUL_{k,i} = f_s \cdot \overline{RUL}_{k,i}^{\text{pred},nA_s} - sm_s \tag{19}$$

In this equation, f_s and sm_s are the final two parameters that the DDP consists of and are used to compute the target RUL $tRUL_{k,i}$ using the RUL prediction of the CNN model at the time the final required alert is generated, denoted by $\overline{RUL}_{k,i}^{\text{pred},nA_s}$. The output of this DDP is now a window in which maintenance can be scheduled as is shown in Figure 10 by the green curly bracket. This maintenance window forces maintenance to be scheduled between the current time (the time of the final required alert) and the target RUL of the engine. Note that if the DPP parameters take values such that the target RUL falls before the current time or after the true moment of failure of the engine, the policy becomes infeasible. The DDP can be denoted as a vector of the parameters that this policy consists of as follows: DDP$= [TH_s, nA_s, p_s, f_s, sm_s]$.

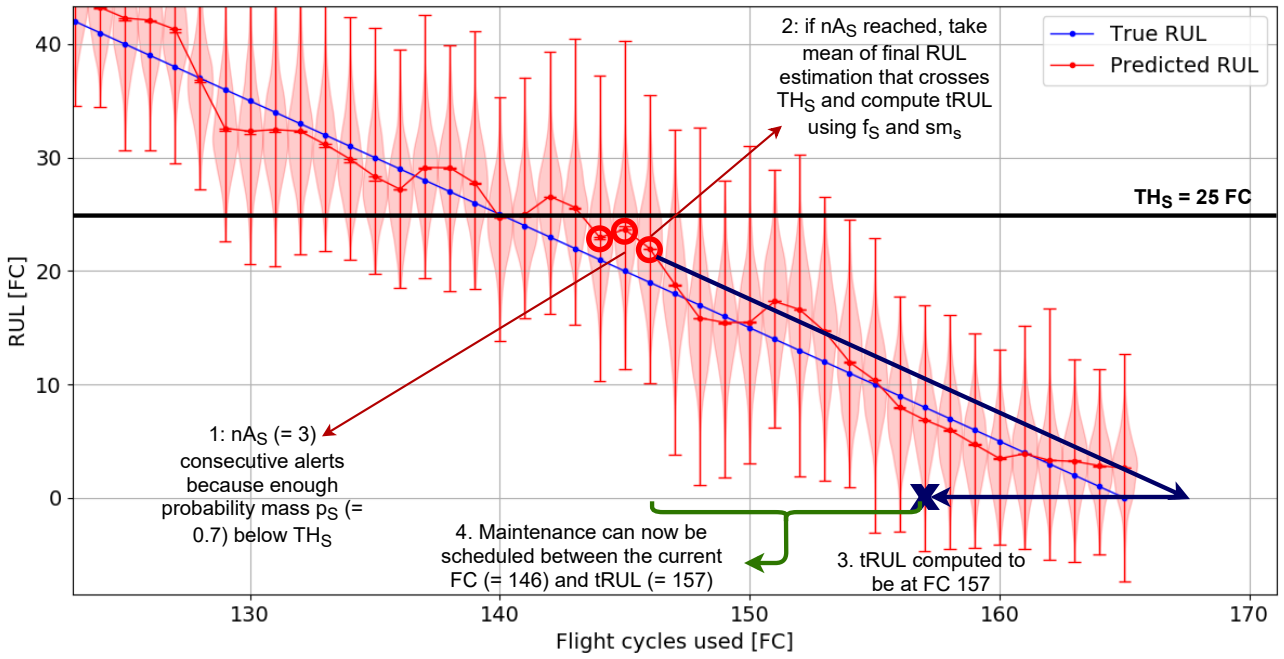


Figure 10: Visual explanation of the data-driven policy (DDP) that determines an engine's target RUL using a series of RUL predictions. Values shown in this figure are used as example

The DDP can be used on both the flight schedule that has a weekly maintenance opportunity and the flight schedule that has no standard opportunities. Both of these flight schedules are referred to as fixed flight schedules, meaning that the flight schedule is repeating weekly and is always the same. If the DDP is used on the schedule with weekly maintenance opportunities, the gap data-driven policy (GDDP) strategy is created. This maintenance strategy is visualized in Figure 11. In this figure, a blue block indicates 2 flight cycles and thus the time an aircraft is away from the base airport and a red diamond shows the weekly maintenance opportunity. The rectangle surrounding 4 blue blocks and 1 red diamond represents the flight schedule for one week, which is repeated to form the long term flight schedule. In this figure, it can be observed that after the engine has been used for some number of flight cycles, the threshold for maintenance scheduling is reached. In this paper, it is assumed that the sensor recordings only become known at the end of each week. At the end of the week the RUL predictions are thus made using all the data obtained during the flights that happened the previous week. It then becomes known that the scheduling threshold is reached during that week and the target RUL is computed. As can be seen in Figure 11, the target RUL lies a certain number of weeks into the future. The aircraft that the engine is placed on will then continue flying for some time until a maintenance opportunity is selected that is before and as close to the target RUL as possible to minimize the engine's wasted life. How the optimal maintenance opportunity is selected will be further explained in subsubsection 4.1.2. Assuming that

the optimal opportunity is picked in Figure 11, the engine has wasted a total of 10 FC of useful life due to the fact that the true moment of engine failure lies after the selected moment of maintenance.

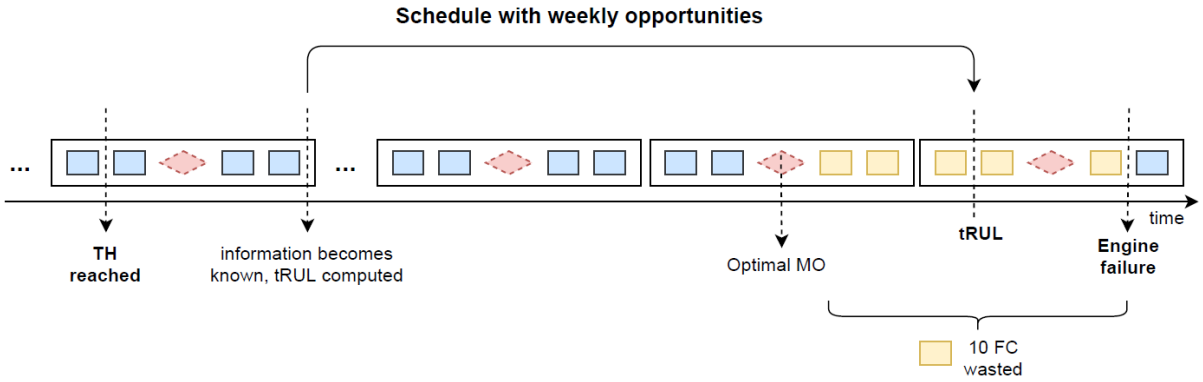


Figure 11: The GDDP strategy of scheduling maintenance using a target RUL with weekly maintenance opportunities. Each blue block indicates a flight pair (2 FC) and a red diamond indicates a maintenance opportunity

The second maintenance strategy is created by combining the DDP and the schedule with no standard gaps. Due to the full flight schedule this strategy is able to generate maximum revenue. This strategy will be denoted as the no gap data-driven policy (NGDDP). This scheduling strategy is similar to the GDDP, but now a flight pair of 2 flight cycles needs to be cancelled in order to create a gap in the schedule during which maintenance can be performed. This process is visualized in Figure 12. Again, at some moment in time the scheduling threshold is reached. In this case, the flight pair right before the predicted target RUL is cancelled to create a maintenance opportunity. Using this strategy, only 6 FC are wasted in this example due to the fact that maintenance can be performed as close as possible to the target RUL. The NGDDP thus has as an advantage that the number of wasted flight cycles is lower compared to the GDDP, but it comes at a cost of cancelling a flight pair that passengers might have already bought tickets for, which is especially expensive if the cancellation occurs shortly before the flight was planned.

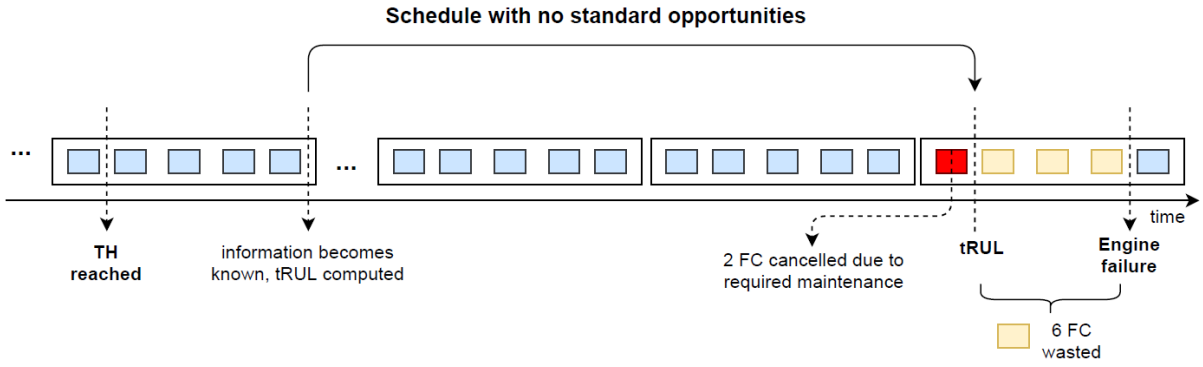


Figure 12: The strategy of scheduling maintenance using a target RUL with no fixed maintenance opportunities (NGDDP). Each blue block indicates a flight pair (2 FC). An opportunity is created by cancelling 2 FC as shown in red

Data-driven maintenance scheduling policy creating maintenance opportunities in flight schedule

The second maintenance policy investigated in this paper is the advanced data-driven policy (ADDP). This policy not only schedules maintenance using data-driven RUL predictions, but also uses those RUL predictions to create maintenance opportunities at an early stage. The ADDP is thus not used in combination with a fixed weekly repeating flight schedule. Due to the fact that the ADDP itself considers both creating maintenance opportunities in the flight schedule and scheduling maintenance, it is considered to be a maintenance strategy as well.

The ADDP nominally makes use of the full schedule with no maintenance opportunities. In order to avoid many short term flight cancellations that are needed to create maintenance opportunities, the ADDP uses the RUL predictions at an early stage to create maintenance opportunities in the far future. In this way, this strategy still relies on flight cancellations, but since those cancellations will occur a long time before the flights are planned to depart the cancellation cost is assumed to be low. How the ADDP uses the RUL predictions to create maintenance opportunities is explained using Figure 13. The ADDP consists of 3 parameters that

check if maintenance opportunity creation is required. It will create maintenance opportunities if there are nA_{MO} consecutive predictions with a probability mass of at least p_{MO} below the scheduling threshold value TH_{MO}. If this opportunity creation threshold is reached, the ADDP uses the RUL prediction at the final opportunity creation alert to estimate in what week the engine is expected to fail. The ADDP will then create nMO maintenance opportunities spread around the predicted week of failure. In this way, the ADDP can be used to create only the minimum amount of maintenance opportunities required such that aircraft utilization is kept at a high level while avoiding many short term cancellations. Once the maintenance opportunities are created, the aircraft is used for a certain amount of time until maintenance scheduling is required. For the ADDP, the process of maintenance scheduling is equivalent to that of the DDP as was shown in Figure 10 and thus also consists of the parameters used in the DDP. If it happens that maintenance is required to be scheduled in the ADDP but either no opportunities are available soon in the schedule or all the created opportunities have already past, a flight pair will have to be cancelled to create an opportunity. This comes as the cost of two undesired short term flight cancellations. The ADDP can be represented as a vector as follows: ADDP= $[TH_{MO}, nA_{MO}, p_{MO}, nMO, TH_s, nA_s, p_s, f_s, sm_s]$.

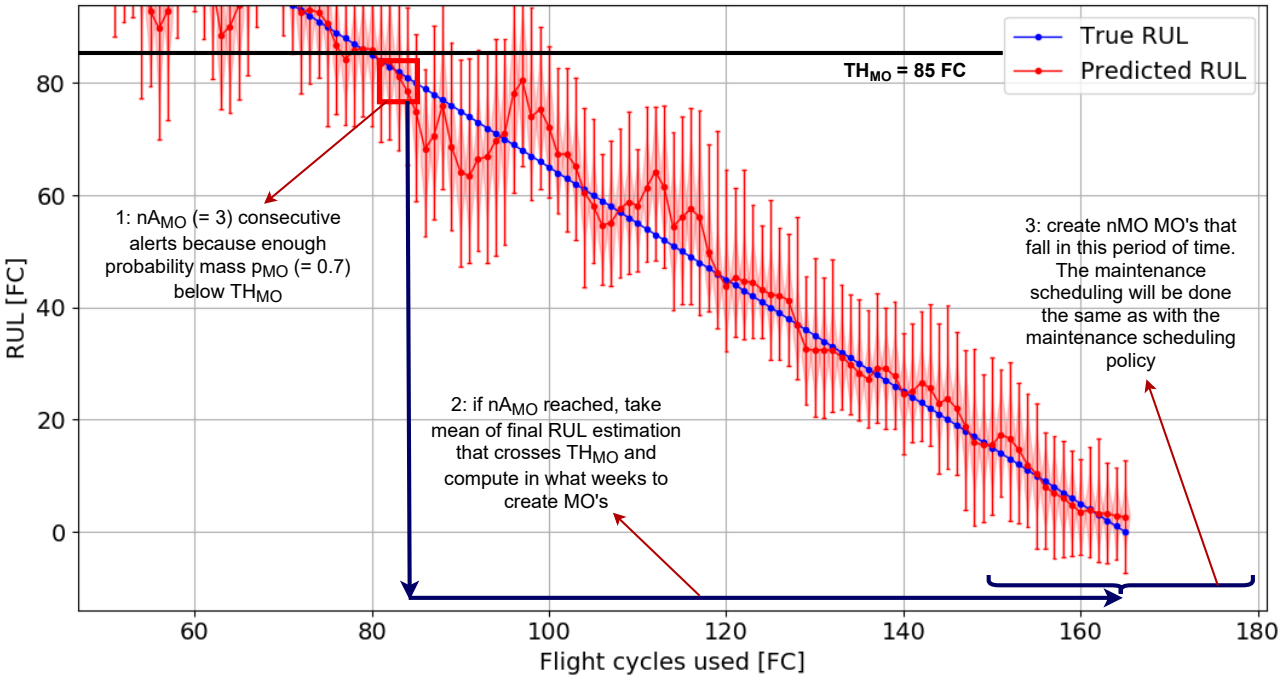


Figure 13: Visual explanation of the advanced data-driven policy (ADDP) that creates MO's and determines an engine's target RUL using a series of RUL predictions. Values shown in this figure are used as example

How the ADDP interacts with the flight schedule can be observed in Figure 14. Similar to Figure 10, a blue block indicates a flight pair of 2 flight cycles and a red diamond indicates a maintenance opportunity. At some moment in time, sufficient alerts are generated using the maintenance policy such that maintenance opportunities can be created using the early RUL estimation for one of the aircraft engines. In this example figure, $nMO = 3$, meaning that if the opportunity creation threshold is reached, 3 opportunities will be created. The black arrows on top of Figure 14 indicate the weeks in which the aircraft now flies the schedule with an opportunity. The aircraft then continues to fly for some time until the threshold for maintenance scheduling is reached for the engine. At this moment, the target RUL for the engine is computed. The optimal maintenance opportunity is the one closest to but before the target RUL. How the optimal maintenance opportunity is picked will be explained in subsubsection 4.1.2.

In Figure 14, maintenance opportunities are created because one of the aircraft engines crossed the opportunity creation threshold. In this work, aircraft do not have a single engine, but multiple. It can then happen that both engines cross the opportunity creation threshold around the same time. If for example 3 maintenance opportunities then have to be created for both engines, it often occurs that the weeks in which those opportunities are created overlap. If this is the case, only one opportunity will be created in that week. Furthermore, in Figure 14 one can observe that the final opportunity that is created is located after the target RUL and thus after the optimal opportunity that is likely picked. Therefore, this opportunity is redundant as the engine already underwent maintenance prior and can therefore be used to reschedule a flight pair in order to maximize revenue. Note that rescheduling flight pairs into redundant opportunities should be done with caution, as it might be the case that some opportunities that are located after an engine was scheduled for maintenance are

actually created for another engine of the aircraft. Therefore it should be checked to what engine the created opportunity belongs before it can be used the reschedule a flight pair.

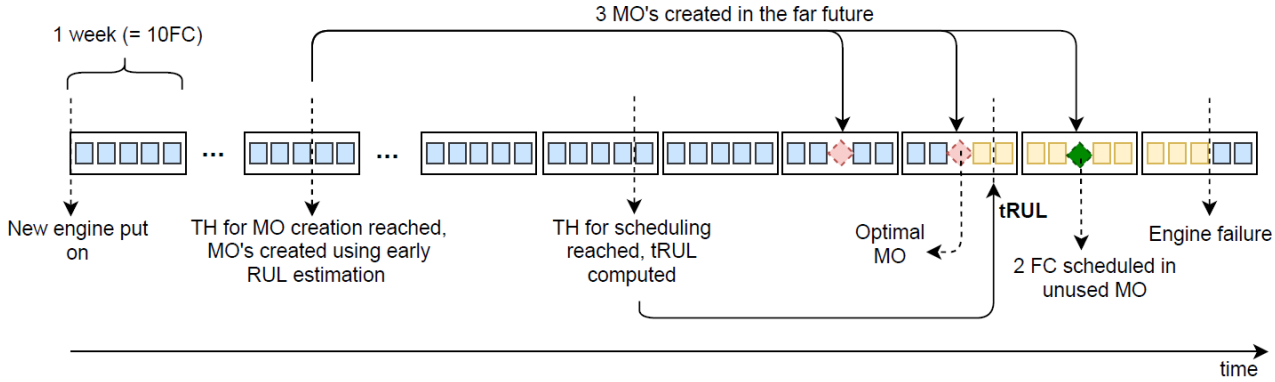


Figure 14: The advanced method of scheduling maintenance using a target RUL with maintenance opportunities that are created based on a RUL estimation early on (ADDP). Each blue block indicates a flight pair (2 FC) and a red diamond indicates a maintenance opportunity. In this figure, 3 MO's are created as an example. The green diamond shows a MO that is created but unused. This slot can therefore be used to reschedule 2 FC

Inspection based maintenance policy for a fixed flight schedule

The final maintenance scheduling policy considered in this paper is a policy that does not use data-driven RUL predictions to find the moment in time an engine requires maintenance. The inspection based policy (IBP) relies on inspecting the engine at a constant interval in order to obtain information on the engine's state. This policy is closely related to how maintenance is scheduled nowadays as it uses information about the engine's state to predict when maintenance is required, but the information that is used is obtained via inspection instead of advanced computational technology. This policy can not make use of a series of RUL predictions made by a prognostic algorithm and thus relies on the imperfect information that is obtained when a inspection is carried out. During an inspection, the estimated RUL of the engine is simulated using Equation 20.

$$RUL_{\text{insp}}^{\text{pred}} = RUL_{\text{insp}}^{\text{true}} + \mathcal{N}(0, u_{\text{insp}} * RUL_{\text{insp}}^{\text{true}}) + \mathcal{U}(-u_E, u_E) \tag{20}$$

In this equation, $RUL_{\text{insp}}^{\text{pred}}$ denotes the estimated RUL obtained after an inspection and $RUL_{\text{insp}}^{\text{true}}$ denotes the true RUL of the engine at the moment of inspection. The error in the RUL estimation when doing an inspection comes from 2 terms: an error term that follows a normal distribution with 0 mean and a standard deviation related to the true RUL denoted by $\mathcal{N}(0, u_{\text{insp}} * RUL_{\text{insp}}^{\text{true}})$ and a constant error term that follows a uniform distribution denoted by $\mathcal{U}(-u_E, u_E)$. The normal error term causes the prediction error to be higher when the engine still has a high true RUL, which makes sense as it is difficult to make an accurate RUL estimation for an engine that shows only limited signs of degradation. The values of u_{insp} and u_E can be set to change the distribution the random error values are drawn from in order to simulate scenarios with various levels of RUL estimation quality. In this paper, default values of $u_{\text{insp}} = 0.2$ and $u_E = 5$ are used.

The IBP is made up of two parameters. The first parameter is the inspection interval that determines the rate at which newly estimated RUL values for an engine are generated. The inspection interval of the IBP is denoted as Int_I. The second parameter is the threshold used to schedule maintenance, denoted as TH_I. As soon as a RUL estimation obtained via inspection is lower than TH_I, the engine is scheduled for maintenance. The IBP can now be represented by the following vector: IBP $= [Int_I, TH_I]$. The IBP can make use on both type of flight schedules, both gap and no gap. If the IBP is combined with the gap schedule, the gap inspection based policy (GIBP) maintenance strategy is created. This strategy works by scheduling maintenance at the first maintenance opportunity available to an aircraft that operates an engine that satisfies the condition $RUL_{\text{insp}}^{\text{pred}} < TH_I$. The IBP can also be combined with the no gap flight schedule, creating the no gap inspection based policy (NGIBP) maintenance strategy. If the NGIBP strategy is considered, maintenance is scheduled as soon as $RUL_{\text{insp}}^{\text{pred}} < TH_I$ by cancelling the flight pair right after the inspection. The NGIBP thus has as a disadvantage that many very short term cancellations are required to create maintenance opportunities, but has as advantage that revenue and aircraft utilization can be maximized due to the standard no gap schedule.

Overview of computing when to schedule maintenance for all scheduling policies

An overview of how the 3 different maintenance policies use RUL predictions is presented in Figure 15. In Figure 15, the respective policy is shown on the left with the inputs to that policy directly to the right. The

DDP and the ADDP make use of the data-driven RUL predictions, while the IBP uses the inspection based RUL estimations. The output of each policy is the target RUL for each engine. The ADDP has the locations of the created maintenance opportunities for all engines as additional outputs.

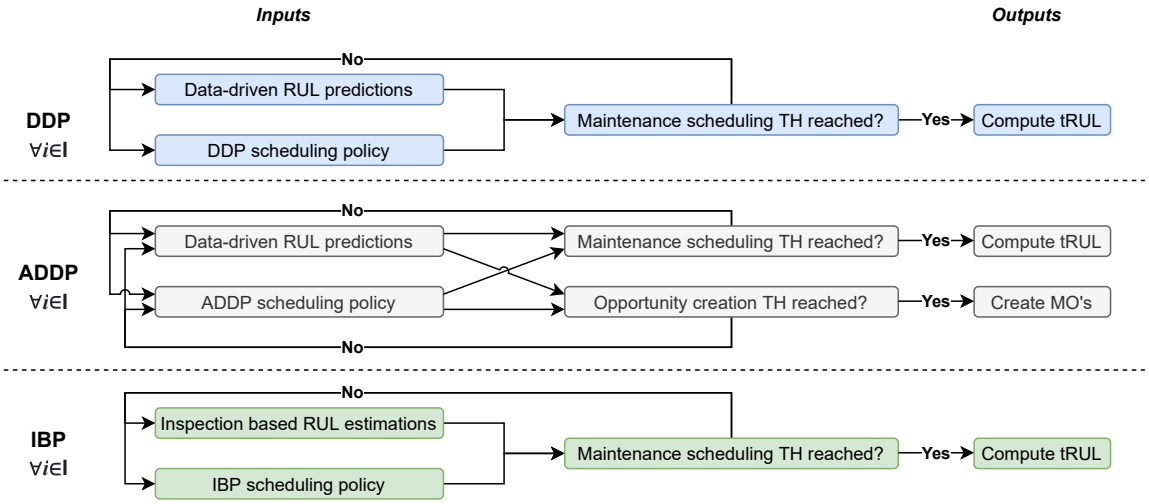


Figure 15: Overview of how the target RUL value is computed for all maintenance scheduling policies using RUL information as input. For the ADDP strategy, the created MO's are additional outputs

4.1.2 Integer Linear Programming model to select the optimal maintenance opportunity

This section will elaborate on how the optimal maintenance opportunity is selected when the target RUL has been computed for an engine. The optimal maintenance opportunity is picked using an Integer Linear Programming (ILP) model that makes use of the target RUL obtained via one of the 3 maintenance policies described in the previous section. Note that the ILP model is only used for maintenance strategies that make use of maintenance opportunities during which maintenance could be scheduled. The maintenance strategies NGDDP and NGIBP operate using a schedule that has no standard opportunities and both strategies solely rely on flight pair cancellation to create an opportunity during which maintenance can be carried out. Therefore, it is not needed to use this ILP model to pick the optimal opportunity as for those strategies there will only be a single slot that is created using cancellation. The nomenclature of the ILP model can be found in Table 7. Note that the terms maintenance opportunity, slot and gap are used interchangeably throughout this work but all refer to a moment in time during which an aircraft is located at the base airport for sufficient time such that maintenance can be carried out if required.

Table 7: Nomenclature of the Integer Linear Programming model used for maintenance scheduling

Sets	
K	Set of all aircraft
I_k	Set of all engines that are part of aicraft k
T_k	Set of all maintenance slots that are available to aicraft k in the current optimization period
Decision variable	
$x_{i,k,t}$	Equals 1 if engine i of aircraft k in assigned to maintenance slot t, 0 otherwise
Parameters	
$C_{k,i,t}$	Penalty cost of assigning engine i of aircraft k to maintenance slot t
H	Number of hangars available

The objective function of the ILP model is mathematically formulated in Equation 21a. The constraints that the model is subject to are denoted in Equation 21b and Equation 21c.

$$\min \quad \mathcal{C} = \sum_{k \in K} \sum_{i \in I_k} \sum_{t \in T_k} x_{k,i,t} \cdot c_{k,i,t} \tag{21a}$$

s.t.

$$\sum_{t \in T_k} x_{k,i,t} = 1, \qquad \forall i \in \mathbf{I_k}, \forall k \in \mathbf{K}, \tag{21b}$$

$$\sum_{k \in K} \sum_{I \in I_k} x_{k,i,t} \leq H, \quad \forall t \in \mathbf{T_k} \tag{21c}$$

Objective function of the Integer Linear Programming model

The objective function only consists of a single term that computes the cost of assigning engine i of aircraft k to maintenance slot t. In order to compute this cost, the cost parameter $c_{k,i,t}$ is introduced and defined in Equation 22. In this equation, $tRUL_{k,i}$ indicates the target RUL of engine i of aircraft k and $FC^*_{k,t}$ indicates the flight cycle at maintenance slot t of aircraft k. The cost parameter takes a value of 10^9 if $FC^*_{k,t} \geq tRUL_{k,i}$ as that would mean that a slot is picked that is located after the target RUL. If a slot is picked that is located before the target RUL, so $FC^*_{k,t} < tRUL_{k,i}$, the cost parameter takes a value equal to the difference in flight cycles between $FC^*_{k,t}$ and $tRUL_{k,i}$. This causes the cost parameter to be minimal when the picked slot is as close as possible to the computed target RUL. The term $\frac{t}{100}$ is added to ensure that if more than one slot is available in between two flight cycles, the first one is always picked. Using this objective function the model will always attempt to schedule maintenance as close as possible to the target RUL that is computed using the maintenance policy, which is desired as this maximizes engine utilization and minimizes wasted life.

$$c_{k,i,t} = \begin{cases} tRUL_{k,i} - FC^*_{k,t} + \frac{t}{100} & \text{if} \quad FC^*_{k,t} < tRUL_{k,i} \\ 10^9 & \text{if} \quad FC^*_{k,t} \geq tRUL_{k,i} \end{cases} \tag{22}$$

Constraints of the Integer Linear Programming model

The first constraint that is defined in Equation 21b simply states that each engine that requires maintenance is allocated to one slot and one slot only. The second constraint defined in Equation 21c states that for any moment in time the hangar capacity H can not be exceeded. The hangar capacity H is defined as the number of engines i that can be replaced in maintenance slot t. The sum of all engines i of all aircraft k at during all maintenance slots t should be smaller than H.

Overview of selecting a maintenance opportunity

An overview showing the inputs and outputs of the ILP model is shown in Figure 16. Note that the ILP model is only used for the Gapped (G) flight schedules, as then the ILP model can be used to pick to optimal slot out of all possible ones. For the No Gapped (NG) flight schedules, the ILP model is not used as for these schedules opportunities are created by cancelling a flight pair of 2 FC right before the computed target RUL. This created slot is then used to perform the maintenance action.

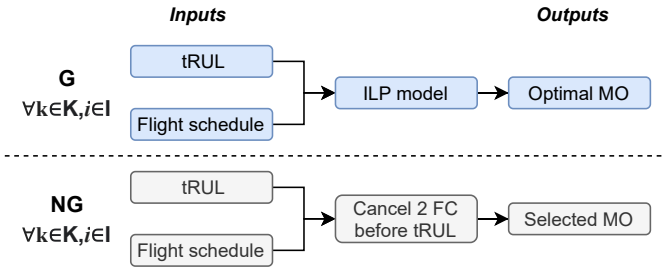


Figure 16: Overview of selecting a MO for both a Gapped (G) and a No Gapped (NG) flight schedule. Note that the ADDP strategy falls under the Gapped schedule in this figure

4.1.3 Rolling horizon method to simulate maintenance scheduling

In order to simulate maintenance scheduling for the long run using the 5 maintenance strategies that are introduced in subsubsection 4.1.1, a rolling horizon method is used. This method is able to progress in time in by continuously optimizing and realizing a period of time. First, the maintenance schedule is optimized for a time period ψ, called the optimization time. After that, the optimized schedule is realized for a time period τ, called the realization time. A visualization of how the rolling horizon method continuously optimizes and realizes the schedule can be observed in Figure 17.

Optimization period

At the start of the optimization period, the sensor recordings are obtained for all engines on all aircraft. These sensor recordings will be given as input to the CNN model that predicts the RUL distribution time series for the previous week. These RUL predictions are then fed into the maintenance policy that is applied. If the scheduling threshold is reached, the engine will be given a target RUL as explained in subsubsection 4.1.1. The

optimization is performed if there is at least one engine i on all aircraft k that has its target RUL computed to be within the next optimization period, meaning that the target RUL has to fall in the range $[S_{\text{start},n}, S_{\text{start},n} + \psi]$. If this is the case, the ILP model is used to find the optimal allocation of maintenance slots for all engines that require maintenance. The outputs of the optimization period are thus the chosen maintenance slots for all engines for which a target RUL is computed. Note that engines that do not have a target RUL have not crossed the scheduling threshold yet according to the maintenance policy that is used. If none of the engines on all aircraft have a target RUL, the optimization is not required and realization of the realization period will occur.

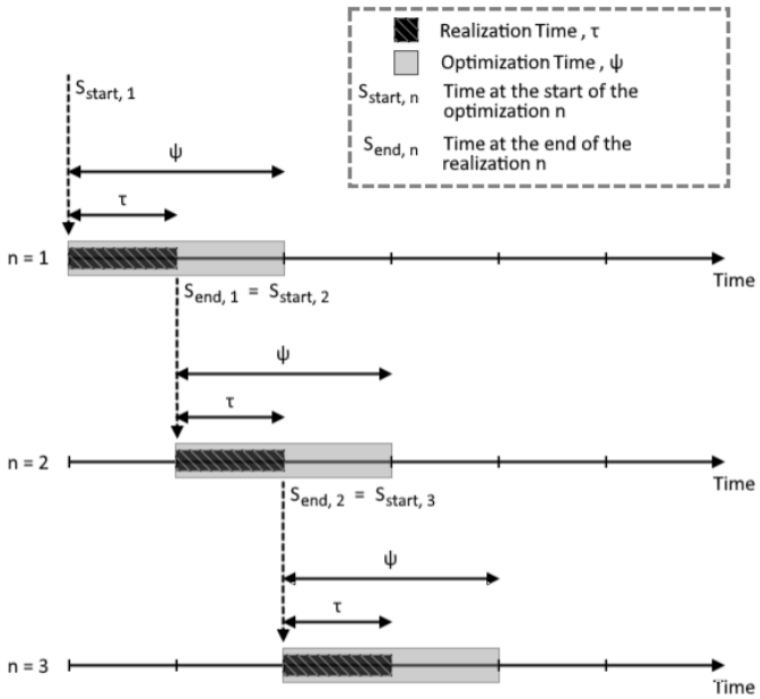


Figure 17: Rolling horizon method used to perform maintenance scheduling including important parameters [31]

Realization period

After the optimization is complete, the realization occurs. This simply means that the optimized schedule and maintenance allocations obtained from the optimization model are realized for a period τ. During realization, all performance indicators are collected, such as the flight cycles flown and all maintenance operations carried out. If an aircraft undergoes maintenance, this is realized by collecting the data about the engine that is replaced, such as the wasted life, flight cycles used and moment of replacement. A new engine is then put on the aircraft, which will then continue to fly until one of its engines requires maintenance again. At the end of the realization period, the sensor data for all flights that occured in that realization period is saved and prepared for RUL prediction. These RUL predictions will then be used in the next optimization period to allocate aircraft to maintenance slots using the target RUL of its engines.

Overview of the long-term maintenance scheduling simulation logic

This paragraph will present an overview of the maintenance scheduling simulation process. The logic of the GDDP maintenance will summarized using the flow chart found in Figure 18. After that, it will be stated how the other four maintenance strategies, those being NGDDP, ADDP, GIBP and NGIBP, differ from the logic presented in Figure 18. Figure 18 can be seen as the combination of Figure 15, Figure 16 and Figure 17 as it shows the inputs and outputs of all parts of the simulation model by making use of all the previously defined parts.

The simulation starts by initializing the aircraft and engine objects. Each engine is given 2 engines with no initial degradation or usage. Once this initialization is completed, the actual simulation can start. First, a realization period happens. During this period, all aircraft fly the schedule with a weekly gap. The simulation increments the number of flight cycles flown by 1 for each aircraft until the total number of flight cycles in that realization period is reached. After this realization period, all sensor data is collected for all engines of all aircraft. If an engine does not have a target RUL yet, RUL predictions are made using the obtained sensor data.

Using the data-driven maintenance policy, it is checked if the newly obtained RUL predictions of the previous realization period are below the threshold for maintenance scheduling. If this is the case, the parameters of the DDP are used to compute the target RUL for the engine. All engines now either have a target RUL or have not yet shown sufficient signs of degradation in order for them to be included in the maintenance scheduling optimization.

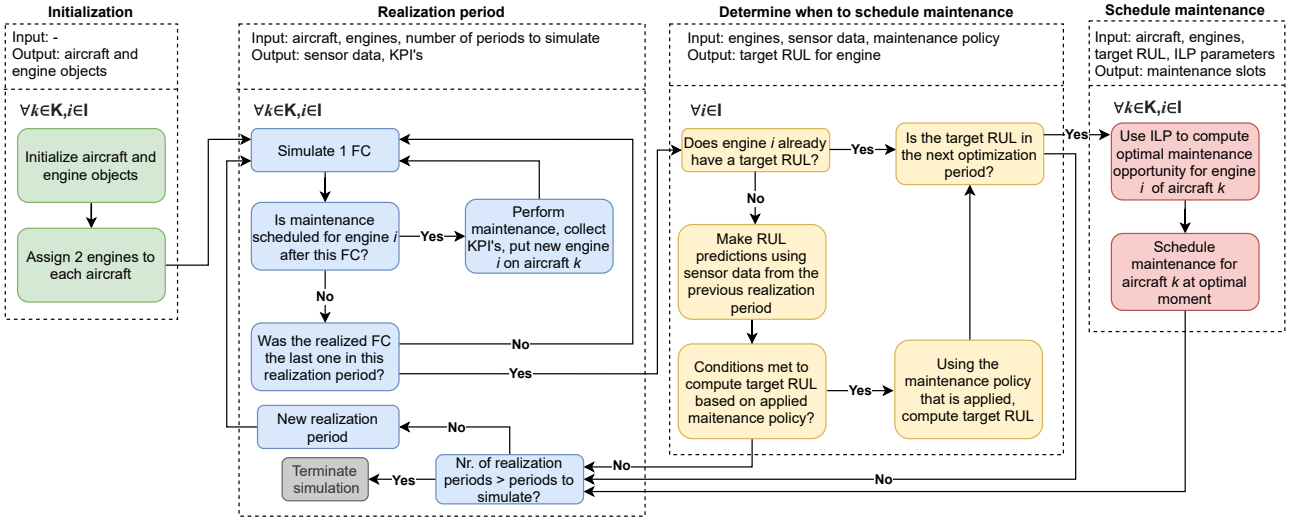


Figure 18: Logic of the maintenance scheduling simulation for the GDDP strategy

All engines that have a target RUL within the next optimization period are fed into the ILP model. Using the flight schedule and the corresponding locations of maintenance opportunities in the schedule, the ILP model selects a maintenance opportunity for each aircraft that has an engine with a target RUL in the optimization period. The outputs of the ILP model are the optimal maintenance allocations for all aircraft that have an engine that requires maintenance in the optimization period. The model then progresses to the next realization period, which continues until the maximum number of prespecified realization periods is reached. If during a realization period an aircraft has maintenance scheduled after a flight cycle, this maintenance is performed. A new engine is then put on the aircraft and all the relevant data regarding the maintenance operation is collected.

Even though Figure 18 is specific to the GDDP strategy, most blocks can be found in the other 4 strategies. The NGDDP uses the green, blue and yellow blocks to realize the flight schedule and compute the target RUL for each engine. The difference is that the NGDDP does not use the ILP model to pick the optimal opportunity, but just cancels the flight pair right before the target RUL and uses that opportunity to carry out the maintenance operation. The ADDP uses all blocks shown in Figure 18. It furthermore includes blocks that check if the opportunity creation threshold is reached and create maintenance opportunities if this threshold is indeed reached. The process of scheduling maintenance is equivalent to that of the GDDP.

The inspection based policy does not make use of realization and optimization periods. When using this policy, the number of flight cycles flown is incremented by 1 in a loop for each aircraft. An inspection occurs according to the inspection rate of that policy. If the obtained RUL estimation at an inspection is lower than the threshold value of the policy, maintenance happens directly after the inspection. For the NGIBP, an opportunity is created by cancelling the flight pair directly after the inspection, while for the GIBP the first possible opportunity in the schedule with weekly opportunities is picked.

4.1.4 Monte Carlo simulation

The maintenance scheduling simulations include variables and actions that include randomness. For example, if a new engine is placed on an aircraft that just underwent maintenance, a random engine is selected from all engines for which run-to-failure sensor data is available. In order to minimize the effects of randomness on the simulation outcome, 2 measures should be taken: (i) let the simulation run for a long period of time, and (ii) run the simulation multiple times using the same initial settings. This second point is referred to as Monte Carlo simulation as the model is ran with the same initial conditions for MC runs in order to average out the effects of randomness on the model output. This method ensures the robustness of the outcomes and allows for comparison with other maintenance scheduling strategies. Furthermore, confidence intervals on the model results can be obtained using the Monte Carlo simulation method.

4.1.5 Maintenance scheduling policy optimization

In subsubsection 4.1.1 a total of 3 different maintenance policies were introduced: a data-driven policy (DDP), an advanced data-driven policy (ADDP) and an inspection based policy (IBP). It was furthermore explained how each of those policies can be represented as a vector containing the parameters of the policy. This section will explain how the optimal values for each of the parameters in those policy vectors can be found. First, policy optimization for the DDP and the ADDP is described. After that, the policy optimization of the IBP is elaborated on.

Genetic algorithm for optimization of data-driven policy and advanced data-driven policy

For the DDP and the ADDP, policy optimization is performed using a genetic algorithm (GA). As an example, the DDP is taken to explain how this GA is used to find the optimal values of the parameters that make up this policy. The DDP can be represented as a vector containing the policy parameters as follows: DDP= $[TH_s, nA_s, p_s, f_s, sm_s]$. The values of these 5 parameters determine when a target RUL should be computed for an engine and determine the value of the computed target RUL. The following paragraphs explain the most important aspects of the GA.

Agents and fitness function

In the GA, a population of A agents is created in which each agent represents a DDP policy vector with random initial values. The full maintenance scheduling simulation logic as shown in Figure 18 is then applied for each of the agents. This simulation process is repeated for MC Monte Carlo runs for each agent in order to minimize the effects of randomness on the simulation outcome. After all simulations for all agents are performed, the results for each agent are collected. Using the performance indicators and input cost values, the performance of the model is expressed in a single monetary value, namely the average profit per aircraft per week, also called the profit rate. How the profit rate value is computed exactly for the maintenance scheduling simulation model will be explained in subsection 4.3. Using this profit rate value, the fitness of each agent is then computed using $f = (\frac{\text{profit rate}}{200{,}000})^3$, where f indicates the fitness of the agent. This non-linear fitness function is designed to give agents a fitness that increases with their profit value to the power of 3.

Filling the genepool

After the fitness is computed for each agent, a genepool is filled using copies of the agents of the previous generation. In this work, the number of copies of each agent that are placed in the genepool is c_{ag} times the agent's fitness. The agent with the highest fitness during a single generation receives additional copies in the genepool, namely a total number of copies equal to $c_{ag}*$ times the highest fitness of that generation. Now that the genepool is filled, 2 random parents are selected from all the copies present in the genepool. This process of parent selection happens $(A - P_{new} \cdot A - A*)$ times, where A is the number of agents in the previous generation. The term $P_{new} \cdot A$ means that at each new generation, P_{new} percent of all agents in that new generation are generated with a random policy vector. This is done to ensure that policy value diversity is kept high and to ensure that the optimal policy does not get stuck in a local maximum. Finally, the $A*$ agents with the highest fitness in the previous generation go again in the next generation, ensuring the the best policy values found will never get lost. By filling the genepool in this way is is ensured that the number of agents in the next generation is equal to the number of agents in the previous generation.

Crossover and mutation

After 2 parents are selected, crossover is applied to generate a child. Crossover is applied by taking the average of the values of the policy parameters at each index of the parents. For example, if parent 1 has a DDP policy vector equal to [40,3,0.8,1.0,10] and parent 2 has a DDP policy vector equal to [20,1,1.0,0.9,14], the resulting policy vector for the child would be [30,2,0.9,0.95,12]. Note that some parameters in this policy are required to be integers, such as the number of alerts required in order to perform maintenance scheduling, nA_S, for example. In this case, the average policy value obtained using the values of both parents is rounded down to the nearest integer. After crossover, mutation is applied to each of the policy values of all child agents with a probability of p_{mut}. Mutation is done by adding or subtracting a small random value drawn from a uniform distribution. After mutation it is ensured that all values that are required to be integer are converted to integers. After this, a new generation of A agents is created and the process of simulating maintenance scheduling repeats. This process ends if the maximum number of specified generations is reached, which is denoted as N_{gen}. An overview of the complete GA data-driven policy optimization loop is shown in Figure 19.

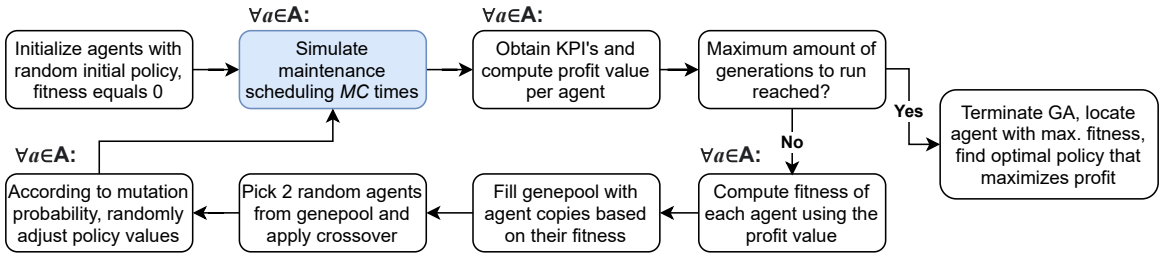


Figure 19: Genetic algorithm (GA) used to optimize the data-driven and advanced data-driven policy vectors

For the ADDP, this GA policy optimization works equivalently. The only difference is that the ADDP is made up of 9 parameters in stead of the 5 parameters the DDP is made up of. This however does not influence the way the GA performs the policy vector optimization.

Optimization of inspection based policy

The IBP can be represented as a simple vector that is made up of only 2 parameters: the inspection interval Int_I and the scheduling threshold TH_I. Because this IBP is a simple policy, optimization will be done by simulation maintenance scheduling using all combinations of Int_I and TH_I when varying both parameters along a specified range. For each combination of Int_I and TH_I, the policy profit value is computed. After all combinations are tested, it can simply be observed which combination led to the highest profit. This combination of Int_I and TH_I will then be used as the optimal IBP vector. This procedure is mathematically shown in Equation 23, where IBP* indicates the optimal policy vector of the IBP and where profit indicates the function that computes the profit of the IBP that makes use of parameters Int_I and TH_I.

$$\text{IBP}^* = \underset{Int_I, TH_I}{\text{argmax}} \left(\text{profit}(Int_I, TH_I)\right) \tag{23}$$

4.2 Experimental set-up

This section will describe the experimental set-up of the maintenance scheduling simulation test case. First, the input data used in this experiment is described. After that, the actual flight schedules used in this paper are highlighted. Lastly, the model input values for both the ILP + rolling horizon model and the GA are shown. In this work, all experiments that are conducted are ran on the HP ZBook Studio G3 with an Intel Core i7-6700HQ CPU. All the coding work is done in Python 4.0 where use is made of Gurobi 9.1 for the ILP optimization.

4.2.1 Run-to-failure engine sampling

The data that is used to simulate maintenance scheduling is obtained from the C-MAPSS dataset described in subsubsection 3.2.1 [3, 6]. The data from this dataset is used by the CNN model developed in subsubsection 3.1.1 to make probabilistic RUL predictions. These probabilistic RUL predictions are then used as inputs to the scheduling model. An important consideration is that in order to use a time series of RUL predictions as input to the scheduling model, sensor data of an engine needs to be available until the moment of failure. If no run-to-failure sensor data is available, RUL predictions can not be made for that engine until the moment of failure, meaning that the use of this engine in a maintenance scheduling simulation can not be represented. As described in subsubsection 3.2.1, the engines in the C-MAPSS testing set are not run-to-failure. The sensor recordings for these engines are terminated at some unknown point prior to failure, and the goal is then to predict the RUL value at the moment the sensor recordings have ended. For this reason, the engines from the C-MAPSS testing set can not be used when simulating maintenance scheduling.

On the other hand, the training engines in the C-MAPSS dataset are run-to-failure. This means that for the engines in the training set, sensor recordings are included from the moment the engine is fully healthy up until the moment the engine fails. For this reason, the training engines are suitable to use when simulating maintenance scheduling. From each of the 4 C-MAPSS instances, a fixed percentage of the training engines is randomly drawn. For FD001, FD002, FD003 and FD004 this means that 14, 37, 14 and 34 engines are sampled respectively. The CNN models for all 4 instances are then trained on the remaining training engines that are not sampled. The outcome is that there are now 4 CNN models that are able to make run-to-failure predictions on a total of 99 sampled engines that the CNN models have not seen during training. The sampled engines are given an ID that includes the instance they are sampled from and the index of the engine in that instance. For example, engine E1.06 means that this engine is the sixth engine from the FD001 training set. An example of a full run-to-failure RUL prediction time series can be observed in Figure 20. Note that this figure is the same

as Figure 10 and Figure 13, but without the annotation and zoomed out. It can be seen that the CNN model predicts the RUL distributions very accurately for this engine, especially near the engine's end of life.

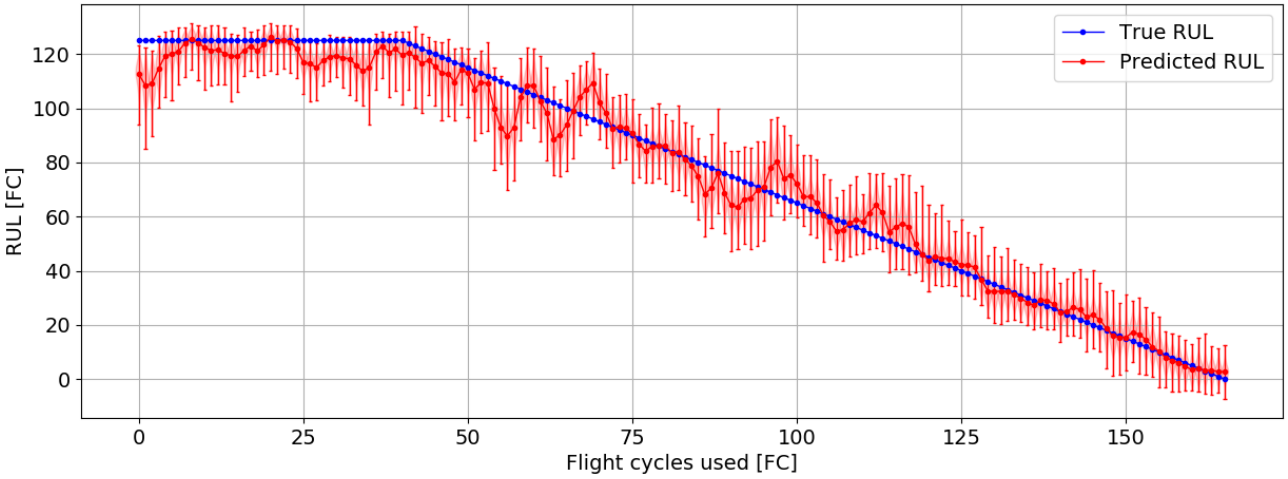


Figure 20: Full run-to-failure RUL distribution predictions for engine 34 from the FD001 training set that the model is not trained. This engine can now be used for simulations in which run-to-failure prediction data is needed

Now that a full time series of run-to-failure RUL predictions is obtained for 99 engines, all metrics introduced in subsection 3.3 can be used to evaluate the prediction performance of the developed CNN models, including the metrics that are not included in Table 4. The results for the first 5 sampled training engines can be found in Table 8. For all metrics that are included in both this table and Table 4, the input values to these metrics are the same. For the SRR metric, the sampling frequencies that are compared are evaluation after every flight cycle and evaluation after every 5 flight cycles, meaning that $f1 = 1$ and $f2 = 5$. Regarding the PH metric, the same input values as for the $\alpha - \lambda$ metric are used, namely an accuracy bound of 30% and a requirement of at least a probability mass of 0.75 between the accuracy bounds in order for a prediction to be counted as acceptable for the probabilistic version of this metric. For the WCRA metric, linear weighting is applied, meaning that the RUL prediction at the highest true RUL value has a weight of 1 and the RUL prediction at the lowest true RUL value has a weight equal to the length of the prediction series. For the convergence metric, the convergence of the CRPS is metric is tracked. The full results for all the 99 sampled training engines can be found in Appendix G. Note that all of the tables that show the results include the mean and standard deviation of the respective metric across all engines in the top two rows.

Table 8: The first 5 sampled training engines that are used for run-to-failure testing evaluated on the most important metrics. The mean and standard deviation of the metrics across all 99 sampled training engines are also shown

	Prognostic metric											
Engine ID	**MAE**	**MAPE**	**RMSE**	**ESTD**	**SRR**	**FP**	**FN**	**TP**	**TN**	**Score**	**W. score**	**Acc.**
Mean	10.22	21.22	13.57	8.84	0.24	0.03	0.16	0.42	0.32	4.43	2.9	0.79
St. dev.	4.67	11.38	5.4	3.02	0.18	0.04	0.17	0.23	0.15	6.88	4.5	0.18
E1.06	9.32	18.31	12.79	8.76	0.92	0.09	0.12	0.6	0.13	2.92	1.93	0.78
E1.16	13.19	18.61	17.66	11.75	0.27	0.09	0.06	0.58	0.19	3.71	1.5	0.84
E1.19	10.17	18.59	14.1	9.77	0.26	0.05	0.17	0.44	0.21	3.96	1.89	0.75
E1.26	10.72	23.97	16.1	12.01	0.26	0.05	0.12	0.44	0.37	4.33	1.7	0.83
E1.34	4.34	8.14	5.86	3.93	0.08	0.01	0.02	0.56	0.33	0.53	0.34	0.97

Engine ID	**Prec.**	**Recall**	**F1**	**PH(d)**	**PH(p)**	**$\alpha - \lambda$(d)**	**$\alpha - \lambda$(p)**	**CRA**	**WCRA**	**CRPS**	**W. CRPS**	**Conv.**
Mean	0.53	0.7	0.75	17.96	8.26	0.8	0.7	0.79	0.7	11.14	4.04	106.16
St. dev.	0.26	0.3	0.27	26.87	0.44	0.17	0.18	0.11	0.16	5.11	6.37	44.03
E1.06	0.82	0.83	0.85	9.0	9.0	0.79	0.7	0.82	0.75	9.16	2.52	77.86
E1.16	0.75	0.91	0.89	33.0	9.0	0.86	0.72	0.81	0.79	12.8	1.72	74.78
E1.19	0.68	0.72	0.8	9.0	9.0	0.78	0.64	0.81	0.79	11.1	2.55	53.9
E1.26	0.54	0.79	0.84	8.0	8.0	0.84	0.76	0.76	0.65	11.42	2.16	72.94
E1.34	0.63	0.97	0.98	166.0	9.0	0.98	0.89	0.92	0.89	5.03	0.57	81.14

4.2.2 Aircraft flight schedules

This work will make use of two different types of flight schedules: one flight schedule that is full and thus has no standard maintenance opportunity of at least 24 hours every week, and one flight schedule that does have

a 24 hour maintenance opportunity every week. An aircraft that flies the full schedule flies 10 flight cycles per week, while an aircraft that flies the schedule with weekly opportunities flies only 8 flight cycles per week. Note that if maintenance is desired to be scheduled in the full schedule, a flight pair of 2 flight cycles is required to be cancelled in order to create a maintenance opportunity. The schedules are based on real life flight schedules for a single week from the KLM A330 fleet. The flight schedule for the long term planning is simply obtained by repeating the same weekly flight schedule for each aircraft. Visualizations of both types of flights schedules for 5 aircraft can be found in Appendix H.

4.2.3 Experimental input values

This section presents the input values that will be used during the maintenance scheduling simulations. First of all, Table 9 shows the model parameters of the ILP model and the rolling horizon method. In this research, a fleet size of 5 aircraft is considered and each aircraft has 2 engines. The hangar capacity is limited to only a single hangar that all 5 aircraft compete for. The optimization period is set to 4 weeks and the realization period is set to 1 week. This means that after every week, the maintenance scheduling is optimized for the next 4 weeks, after which 1 week is realized. The total simulation period is equal to 260 weeks, or 5 years. This simulation period is then repeated for 50 Monte Carlo simulations for all of the maintenance strategies that are considered in this work.

Table 9: Scheduling model parameters

Model parameters	**Symbol**	**Value**	**Unit**
Nr. of aircraft	N_K	5	[aircraft]
Nr. of engines per a/c	N_E	2	[engines]
Nr. of hangars	H	1	[hangar]
Optimization period	ψ	4	[weeks]
Realization period	τ	1	[week]
Simulation period	P	260	[weeks]
Monte Carlo simulations	MC	50	[simulations]

Next, Table 10 displays the parameters of the GA that is used to optimize the data-driven policies. The GA uses 25 agents and runs for 30 generations. The probability of mutation for each of the values in the policy vector for all child agents is set to 35%. The genepool is filled with $10 \cdot f$ copies of each agent, and $50 \cdot f$ copies of the agent with the highest fitness are added. The single best agent of the previous generation is used again in the next generation and per generation 33% of all agents are newly added agents with a random policy vector.

Table 10: Genetic algorithm optimization parameters

GA parameters	**Symbol**	**Value**	**Unit**
Nr. of agents	A	25	[agents]
Nr. of generations	N_{gen}	30	[generations]
Probability of mutation	p_{mut}	35	[-]
Copies of agent in genepool	c_{ag}	$10 \cdot f$	[agents]
Copies of agent with highest fitness in genepool	$c_{ag}*$	$50 \cdot f$	[agents]
Nr. of best agents that go again	$A*$	1	[agent]
Percentage new random agents	P_{new}	33	[%]

4.3 Maintenance scheduling simulation model performance evaluation

This section will discuss the performance analysis of the maintenance scheduling simulation model. All the performance indicators that are collected during the simulations are described and their use when analyzing the model's performance is discussed. First, the operational performance indicators are mentioned, after which the monetary performance indicators are elaborated on. This second part also includes the cost structure that is used in this paper.

4.3.1 Operational performance indicators

During the maintenance scheduling simulation, all relevant data that indicates the performance of the strategy that used is collected. An operational performance indicator (OPI) is an indicator that can be measured as the result of an operational action and is thus non-monetary. The first OPI that is collected is the number of replacements denoted by R. In this work, if an engine is scheduled for maintenance, the engine is not repaired directly. The maintenance operation consists of removing the old engine and replacing it with a healthy engine. It is then assumed that the old engine is overhauled to perfect state and ready to put on a new aircraft afterwards. Therefore, the number of replacements counts the number of maintenance operations carried out. A maintenance operation can be scheduled using the ILP model, but can also be unscheduled if for example no feasible maintenance opportunity can be found in a given optimization period. In both cases, the number of replacements is incremented by one. The fact that an unscheduled replacement leads to a higher degree of inconvenience will be accounted for when collecting data on flight cancellations.

The next OPI that is collected is the number of flight cancellations. It is of course unwanted to cancel flights that passengers might have already booked tickets for, and therefore it is desired to apply a maintenance scheduling strategy that has the minimal amount of flight cancellations. Some strategies, such as the NGDDP and the NGIBP, rely on flight cancellation to create opportunities to carry out maintenance as in these strategies flight schedules are used with no standard maintenance slots. Flight cancellations are further divided into 3 types, depending on the moment of when the flight is cancelled. A short-term cancellation is defined as a flight that is cancelled less than 2 weeks before departure and denoted by Can_S. If a flight is cancelled 2 to 6 weeks before departure, a mid-term cancellation denoted by Can_M is counted. Lastly, if a flight is cancelled more than 6 weeks before departure, a long-term cancellation denoted by Can_L is counted. The more short term the cancellation occurs, the higher the inconvenience for passengers that have already booked the flight. Also, a more short term cancellation leads to more inconvenience for the airline as some aspects need to be arranged in a shorter time window, such as finding a suitable hanger to carry out the replacement or putting together a team of engineers. For those reasons, it is desired to have the minimum amount of short-term cancellations.

Other OPI's are the number of safety incidents denoted by SI and the number of inspections denoted by I. A safety incident is counted when engine failure happens during a flight and must thus be kept at 0 at all times. The number of inspections only applies to the inspection based policies and simply counts how many inspections occur to the engines of an aircraft in a given period of time. As inspections cost time and money, it is desired to minimize the number of inspections needed. Furthermore, the number of flight cycles flown after an engine's target RUL is counted as OPI and is denoted by $FC_{>tRUL}$. This indicator is related to the risk of operation, as flying many flight cycles after the target RUL leads to a higher probability of engine failure during a flight. Using the ILP model, the number of flight cycles flown after the target RUL is minimized, but due to the busy schedule and the limited hangar capacity it can never be reduced to 0. Note that when the NGDDP, GIBP or NGIBP is used, an aircraft can not fly after the target RUL as either a replacement always occurs right before the target RUL (for the NGDDP) or the concept of a target RUL is not applicable (for the GIBP and NGIBP). A OPI that is used to compute the generated revenue is the total number of flight cycles flown by an aircraft denoted by FC_{flown}. It is of course desired to maximize this value. Lastly, the mean remaining life of an engine is collected by computing the difference between the moment of replacement and the moment of true failure of the engine and is denoted by $\overline{WL}$. This OPI is also referred to as the wasted life of an engine at replacement and is desired to be minimized. All OPI's introduced in this section are normalized with respect to the number of aircraft and the number of weeks the simulations lasts. This means that the units of all OPI's are all per aircraft per week, except for the mean wasted life at a replacement.

4.3.2 Monetary performance indicators

This subsection will explain the monetary performance indicators (MPI) that are used when the maintenance scheduling simulation model is evaluated. This work includes 4 monetary performance indicators: cost, revenue, profit and the revenue to cost ratio. In this paper, the cost, revenue and profit are all normalized with respect to the number of aircraft and the total simulation time. This means that the unit of these values will be per aircraft per week. For this reason, the indicators are referred to as a rate. For example, the weekly profit per aircraft is equivalent to the profit rate. In order to explain how each of those indicators can be computed, the monetary structure needs to be explained first.

All the parameters that are considered in the monetary structure can be found in Table 11. The first MPI considered in this work is the cost rate. The cost rate is made up of many different components, each of which is explained separately fist. The fist cost component is the cost of replacement, which is defined in Equation 24. The cost of a replacement is made up of the hourly rate of the employees, the usage of tools required, the cost of towing the aircraft to the hangar and constant hangar visit cost.

Table 11: Cost parameters and other parameters that are used in the monetary evaluation structure of the maintenance scheduling simulation model

Cost parameter	**Symbol**	**Unit**
Ticket price	T	\$/Pax.
Man hour rate	C_{MHR}	\$/Empl./h
Cost of tools required	C_{TO}	\$
Cost towing per hour	C_{TW}	\$/h
Cost of hangar visit	C_H	\$
Cost of safety incident	C_{SI}	\$
Cost flying 1 FC after tRUL	C_T	\$
Cost engine overhaul	C_O	\$
Other parameters	**Symbol**	**Unit**
Duration replacement	D_R	h
Employees for replacement	E_R	Empl.
Total life of tools used	L_{TO}	Repl.
Duration towing to hangar	D_{TW}	h
Duration inspection	D_I	h
Employees for inspection	E_I	Empl.
Max. passengers per flight	M_{pax}	Pax.
Short-term load factor	LF_s	-
Mid-term load factor	LF_m	-
Long-term load factor	LF_l	-
Engine mean life	$\overline{L}$	FC

$$C_R = C_{MHR} \cdot D_R \cdot E_R + \frac{C_{TO}}{L_{TO}} + 2 \cdot D_{TW} \cdot C_{TW} + C_H \tag{24}$$

The next cost component is the cost of inspection and is defined in Equation 25. The cost of an inspection is made up of the hourly man rate, the duration of an inspection, the number of employees required for an inspection, the cost and life of the tools required and the cost and duration of towing the aircraft to a location where the inspection can be carried out. This cost only applies to the inspection based polices.

$$C_I = C_{MHR} \cdot D_I \cdot E_I + \frac{C_{TO}}{2 \cdot L_{TO}} + 2 \cdot D_{TW} \cdot C_{TW} \tag{25}$$

The cost of cancellation depends on the moment a flight is cancelled. As explained in the previous section, three types of cancellations are considered: short-term, mid-term and long-term. The cost of a short-term cancellation is defined in Equation 26. It consists of a refund to all passengers that is equal to 3 time the ticket price to compensate for passenger inconvenience and two times the cost of a replacement to cover the additional logistical and operational inconvenience to the airline. For the mid-term cancellation cost shown in Equation 27, it is assumed that only a ticket price refund i given to the passengers and that the flight is also only partially booked yet indicated by the mid-term load factor LF_m. To account for airline operational inconvenience, the cost of a mid-term cancellation also consists of the cost of a replacement once.

$$C_{SC} = 3 \cdot LF_s \cdot M_{\text{pax}} \cdot T + 2 \cdot C_R \tag{26}$$

$$C_{MC} = 1 \cdot LF_m \cdot M_{\text{pax}} \cdot T + C_R \tag{27}$$

Regarding the cost of a long-term cancellation, passengers are only compensated for an additional 20% of the ticket price, besides the refund of their original ticket payment. This is denoted in Equation 28. The total cancellation cost can then be computed the summing the number of cancellations times the costs of cancellation for all cancellations types as shown in Equation 29. The superscript p, k is used to indicate a flight cancellation of aircraft k in realization period p.

$$C_{LC} = 0.2 \cdot LF_l \cdot M_{\text{pax}} \cdot T \tag{28}$$

$$C^{p,k}_{C,\text{tot}} = C_{SC} \cdot Can^{p,k}_S + C_{MC} \cdot Can^{p,k}_M + C_{LC} \cdot Can^{p,k}_L \tag{29}$$

The final cost rate can now be computed by making use of all the previously defined cost components as is shown in Equation 30. The total cost is equal to the sum over all realization periods p for all aircraft k for all engines i of the cost of replacement, the cost of inspection, the cost of safety incidents, the cost of flying after the target RUL and the cost of wasting engine life. The cost of wasting engine life is computed by computed what fraction of the average engine life is wasted and multiplying that by the cost of overhaul. The cost rate is obtained by normalizing the cost by the number of aircraft N_k and the number of realization periods P.

$$\text{Cost rate} = \frac{\sum_p \sum_k \sum_i (C_R \cdot R^{p,k,i} + C_I \cdot I^{p,k,i} + C^{p,k}_{C,\text{tot}} + C_{SI} \cdot SI^{p,k} + C_T \cdot FC^{p,k,i}_{>tRUL,\text{tot}} + C_O \cdot \frac{WL^{p,k,i}_{\text{tot}}}{\bar{L}})}{N_k \cdot P} \tag{30}$$

The revenue can be computed by summing over all the flight cycles that are flown by aircraft k in realization period p and multiplying that by the short-term load factor, the maximum number of passengers on a flight and the ticket price each passenger pays. The short term load factor LF_s is thus also used as the default average load factor for all flights that are not cancelled. The revenue rate is obtained by normalizing the revenue by the number of aircraft N_k and the number of realization periods P as is shown in Equation 31.

$$\text{Revenue rate} = \frac{\sum_p \sum_k FC^{p,k}_{\text{flown}} \cdot LF_s \cdot M_{\text{pax}} \cdot T}{N_k \cdot P} \tag{31}$$

The obtained profit rate can simply be computed by subtracting the cost rate from the revenue rate as is shown in Equation 32. Note that the profit per aircraft per week is the value used to compute the fitness of an agent when using the GA for policy optimization. The profit rate is thus the most important indicator to evaluate the performance of a maintenance scheduling strategy. Lastly, the revenue to cost ratio is defined in Equation 33.

$$\text{Profit rate} = \text{Revenue rate} - \text{Cost rate} \tag{32}$$

$$\text{Ratio revenue to cost} = \frac{\text{Revenue rate}}{\text{Cost rate}} \tag{33}$$

4.4 Experimental results and analysis

This section will show the results that are obtained when simulating maintenance scheduling using data-driven RUL prognostics for the 5 strategies that are considered in this work. First, the results of the optimization of all the maintenance policies in the 5 maintenance scheduling strategies are presented. After that, the performance

of the 5 maintenance scheduling strategies that use the optimal policy values are compared with each other using the both the operational and monetary performance indicators.

Before the results can be discussed, it is first important to show the monetary and operational input parameter values that are used in this paper. These values are required to compute the monetary performance indicators. The most important performance indicator, the profit rate, is also required to perform the policy optimization using the GA as the fitness of an agent is proportional to the obtained profit rate. The monetary and operational input parameters used in this paper are shown in Table 12.

Table 12: Monetary and operational input values to the scheduling model that are used to compute the monetary performance indicators

Input parameter	**Symbol**	**Value**	**Input parameter**	**Symbol**	**Value**
Ticket price	T	200 [$]	Duration replacement	D_R	24 [h]
Man hour rate	C_{MHR}	38 [$]	Employees for repl.	E_R	15 [empl.]
Cost tools	C_{TO}	10^6 [$]	Duration of towing per repl.	D_{TW}	360 [s]
Cost towing per hour	C_{TW}	3,600 [$]	Duration inspection	D_I	8 [h]
Cost hangar visit	C_H	1,000 [$]	Employees for inspection	E_I	8 [empl.]
Cost safety incident	C_{SI}	10^9 [$]	Max passengers per flight	M_{pax}	400 [pax.]
Cost flying FC after $tRUL$	C_T	50,000 [$]	Short-term load factor a/c	LF_s	0.8 [-]
Cost engine overhaul	C_O	$3{\cdot}10^6$ [$]	Mid-term load factor a/c	LF_m	0.64 [-]
Tools mean life	L_{TO}	1000 [Repl.]	Long-term load factor a/c	LF_l	0.48 [-]
Engine mean life	$\overline{L}$	196.2 [FC]			

4.4.1 Results of policy optimization for all maintenance strategies

The results of the maintenance policy optimization for all 5 maintenance strategies are shown in Table 13. The optimized policy vectors for the ADDP, GDDP and NGDDP are obtained using the GA that is explained in subsubsection 4.1.5. The maintenance strategies that make use of the inspection based policies, the GIBP and NGIBP, are optimized using a brute-force search in which all combinations of values of both policy parameters along a specified range are evaluated.

Table 13: Values of the optimized policy vectors for each of the maintenance scheduling strategies

Strategy	**Parameters of the policy vector**	**Optimal values**
ADDP	[TH_{mo}, nA_{mo}, p_{mo}, nMO, TH_s, nA_s, p_s, f_s, sm_s]	[95, 2, 0.96, 1, 28, 2, 0.66, 0.81, 13.5]
GDDP	[TH_s, nA_s, p_s, f_s, sm_s]	[26, 1, 0.692, 0.890, 12.83]
GIBP	[Int_I, TH_I]	[11, 26]
NGDDP	[TH_s, nA_s, p_s, f_s, sm_s]	[47, 1, 0.681, 0.912, 13.38]
NGIBP	[Int_I, TH_I]	[9, 16]

It can be observed in Table 13 that for the ADDP, the threshold for opportunity creation is relatively high with a value of 95 FC. This means that this strategy is able to create maintenance opportunities in the flight schedule in the far future, ensuring that flight cancellation mostly happens long-term. In this way, the high cost of short-term cancellation can be avoided, which is the exactly why this policy was designed. It is furthermore interesting to see that for the ADDP, only 1 opportunity is created for an engine if the opportunity creation threshold is reached. This means that the early RUL prediction is accurate enough to create the opportunity close the ideal moment of maintenance. This also means that if the single opportunity that is created is not close to the actual moment of engine failure, a flight pair is required to be cancelled short-term in order to create an opportunity. Still, using the GA it was found that for the ADDP it is optimal to only create a single opportunity.

For both the ADDP and the GDDP, the threshold for maintenance scheduling TH_s has a similar value of 28 FC and 26 FC respectively. Around these values it is thus optimal to schedule maintenance, as the RUL predictions are then accurate enough for all engines to schedule maintenance while there is still time to find an opportunity before the target RUL. The GDDP is able to have a low value for TH_s as the flight schedule has weekly opportunities, meaning that it does not have to consider short-term flight cancellations often. On the other hand, the value of TH_s for the NGDDP is much higher compared to the ADDP and the GDDP. This is because in the NGDDP, no standard opportunities are present in the schedule, meaning that this strategy relies on flight cancellation in order to create maintenance slots. The high value of TH_s ensures that maintenance is scheduled much earlier compared to the ADDP and the GDDP, which ensures that most flight cancellations are mid-term in stead of short-term. This severely decreases the cost of the NGDDP, even though the target RUL is likely computed at a less optimal moment due to the more inaccurate RUL predictions when the engines are

still healthier. Using the GA, the NGDDP thus learned that by scheduling maintenance earlier the profit can be maximized as most flight cancellations then occur mid-term instead of short-term. The values of f_s and sm_s that are used to compute the target RUL are both similar in value for the ADDP, GDDP and NGDDP.

For the strategies that make use of an inspection based policy, it can be seen that the inspection interval is lower than the maintenance scheduling threshold in both cases. This makes sense as otherwise it often occurs that a after an inspection a RUL estimation is obtained that is just higher than the threshold, and due to the relatively long inspection interval the engine has already failed before a new inspection can take place. Furthermore, the inspection interval and the threshold value are quite different when comparing the GIBP and the NGIBP. This can be explained by the fact that for the GIBP, maintenance can not always occur directly after an inspection during which a RUL value lower than TH_I is obtained. For example, if an inspection has occured and the obtained RUL estimation is lower than TH_I, the aircraft has to wait until the next available opportunity in the standard schedule. In the mean time, flights are still scheduled that the aircraft has to be used for. The optimized policy for the GIBP should thus be more on the cautious side with a higher value for TH_I, as maintenance can not always happen as soon a the scheduling threshold is reached. For the NGIBP, this threshold value can be lower because maintenance can be carried out directly after an inspection during which it was found that the estimated RUL is lower than the threshold value.

4.4.2 Results of maintenance scheduling

This section will analyze the results that are obtained when simulating maintenance scheduling using the 5 maintenance strategies considered in this work. The evaluation of the results will be done using the operational and monetary performance indicators that were introduced in subsection 4.3. Table 14 presents the obtained results when simulation maintenance scheduling using all 5 strategies. The results on all key performance indicators (KPI) are obtained by simulating each strategy for 50 Monte Carlo runs using the optimized policy vectors that were presented in Table 13. The total simulation time for 50 Monte Carlo runs is also added as KPI, as well as the computational time per Monte Carlo simulation run. Table 14 shows the mean value for all KPI's of all Monte Carlo simulations, as well as the 95% confidence interval. For each KPI, the strategy that achieved the best value is highlighted in green. Note that all values in this table are normalized such that the unit is per aircraft per week, except for the remaining life at replacement. The remaining life at replacement metric indicates the average number of FC that is wasted for an engine once it is replaced. For example, when the ADDP is used, each aircraft undergoes 0.110 engine replacements per week on average.

Table 14: Results for all maintenance scheduling strategies measured using multiple KPI's (50 MC simulations)

	Maintenance scheduling strategy				
Operational KPI's	**ADDP**	**GDDP**	**GIBP**	**NGDDP**	**NGIBP**
Replacements	0.110 [0.109-0.111]	0.086 [0.085-0.087]	0.090 [0.089-0.091]	0.115 [0.114-0.116]	0.106 [0.105-0.107]
Short-term cancel.	0.084 [0.082-0.086]	0.001 [0.000-0.001]	0.013 [0.012-0.014]	0.008 [0.007-0.009]	0.228 [0.225-0.230]
Mid-term cancel.	0.011 [0.010-0.013]	0.0 [0.0-0.0]	0.0 [0.0-0.0]	0.228 [0.226-0.230]	0.0 [0.0-0.0]
Long-term cancel.	0.209 [0.206-0.210]	0.0 [0.0-0.0]	0.0 [0.0-0.0]	0.0 [0.0-0.0]	0.0 [0.0-0.0]
Safety incidents	0.0 [0.0-0.0]	0.0 [0.0-0.0]	0.0 [0.0-0.0]	0.0 [0.0-0.0]	0.0 [0.0-0.0]
Inspections	0.0 [0.0-0.0]	0.0 [0.0-0.0]	0.731 [0.721-0.741]	0.0 [0.0-0.0]	1.182 [1.172-1.191]
FC flown after tRUL	0.177 [0.171-0.182]	0.049 [0.047-0.052]	0.0 [0.0-0.0]	0.0 [0.0-0.0]	0.0 [0.0-0.0]
Total FC flown	9.834 [9.831-9.837	7.999 [7.999-8.000]	7.986 [7.984-7.987]	9.764 [9.761-9.766]	9.772 [9.770-9.775]
Remaining life at repl.	20.18 [20.02-20.33]	17.63 [17.50-17.77]	20.87 [20.51-21.23]	22.08 [21.80-22.35]	12.75 [12.60-12.91]
Monetary KPI's [$]					
Total cost (10^4)	3.534 [3.480-3.587]	0.639 [0.625-0.654]	1.218 [1.191-1.246]	2.317 [2.286-2.347]	6.347 [6.293-6.401]
Revenue (10^5)	6.294 [6.292-6.296]	5.120 [5.120-5.120]	5.111 [5.110-5.112]	6.249 [6.247-6.250]	6.254 [6.253-6.255]
Profit (10^5)	5.940 [5.934-5.947]	5.056 [5.054-5.058]	4.989 [4.986-4.993]	6.017 [6.013-6.021]	5.620 [5.613-5.627]
Ratio revenue/cost	17.86 [17.59-18.14]	80.58 [78.77-82.39]	42.20 [41.28-43.11]	27.03 [26.67-27.39]	9.86 [9.78-9.95]
Other KPI's [s]					
Total comp. time	231.23	139.93	58.21	151.87	62.67
Comp. time / MC run	4.059 [3.928-4.189]	2.216 [2.203-2.228]	0.701 [0.695-0.725]	2.431 [2.161-2.700]	0.853 [0.840-0.865]

Regarding the number of replacements, all policies show quite similar results. The finding that the NGDDP has the most replacements comes from the fact that the NGDDP's optimal policy is such that maintenance is scheduled at a relatively early moment in time. This leads to a very low number of short-term cancellations, but also leads to an increase in the number of replacements because maintenance is scheduled a times where the RUL predictions are not as accurate as in later stage when engines are closer to failure. This can also be observed when looking at the remaining life at replacement for the NGDDP, which is high due to the early moment of maintenance scheduling.

When looking at the results for flight cancellation, it can be seen that the GDDP and the GIBP have very low numbers of flight cancellations due to the fact that these strategies already have fixed opportunities in their

weekly flight schedules. For the NGDDP, it is indeed observed that the number of short-term cancellations is very low compared to the number of mid-term cancellations. This shows that the GA has indeed optimized the NGDDP's policy vector to make do maintenance scheduling early on, thereby eliminating most costly short-term cancellations. For the ADDP, the number of long-term cancellations is relatively high, which is the desired result for this strategy. This value is high because of the opportunities that are created at a very early moment in time using the RUL predictions at that moment. Furthermore, the number of short-term cancellations is also quite high for the ADDP. This is due to the fact that in the optimal ADDP policy vector, the number of opportunities created in the far future is only 1, meaning that it also happens quite often that this single opportunity is not located near the engine's true moment of failure. In this case, a flight pair has to be cancelled short-term in order to create an opportunity. For the NGIBP, all cancellations are short-term because a maintenance opportunity is simply created by cancelling a flight pair directly after the estimated RUL at an inspection is lower than the threshold value.

Only the GIBP and the NGIBP make use of inspections and therefore the number of inspections is equal to 0 for the other strategies. The number of inspections for the NGIBP is higher compared to the GIBP because of the lower inspection interval that is found to be optimal for this strategy.

Only the ADDP and the GDDP have nonzero values for the number of FC flown after the target RUL. The inspection based strategies do not make use of a target RUL, while the NGDDP always schedules maintenance right before the computed target RUL by cancelling the flight pair before it. The ADDP has a relatively high number of FC flown after the target RUL due to the fact that in each optimization period each engine that requires maintenance only has one opportunity that is created for that engine. It might occur that both engines of the same aircraft compete for the same opportunity if the opportunity that is created for one of the engines is located at a incorrect moment in time. This reason, in combination with the fact that the hangar may be occupied during the only opportunity that an engine has, leads to the possibility that an engine is replaced after the target RUL in the ADDP relatively often.

When looking at the number of flight cycles flown, it can be clearly seen that all strategies that make use of the no gap flight schedule fly more cycles than the strategies that have a standard maintenance slot in their flight cycles. Furthermore, the ADDP has the highest number of flight cycles flown as it creates very few opportunities once the opportunity creation threshold is reached, and redundant opportunities are used to reschedule flight pairs in order to maximize revenue. When looking at the remaining life at replacement, the NGIBP is found to perform best, followed by the GDDP.

When looking at the monetary performance indicators, it can be seen that no single strategy outperforms the others in term of both cost and revenue. The GDDP was found to have the lowest cost due to the low number of replacements, the very low number of flight cancellations and the low wasted life at replacement. Even though the GDDP also has a relatively low revenue due to the low number of FC flown, the ratio of revenue to cost is still highest for this policy. The revenue is highest for the ADDP due to the fact that only few opportunities are created and that redundant opportunities are used to reschedule flights. Due to the fact that the cost of the ADDP is relatively high due to the high number of both long-term and short-term cancellations, its profit value is not the highest of all.

The NGDDP was found to have the highest profit value, even though the cost, revenue and revenue to cost ratio are all not the best. The NGDDP has the highest profit due to the very low number of short-term cancellations, the fact that no FC are flown after the target RUL, no inspections are carried out and the number of flight cycles flown is very high. This policy thus combines a low cost value with a very high revenue, which leads to the highest profit value across all strategies that are considered. As the profit is considered the most important evaluation metric for an airline, it can be concluded that the NGDDP is the optimal maintenance strategy to use. In Figure 21, the profit values and revenue to cost ratios for all strategies are visualized. It can be seen that for most strategies, the boxes are thin, meaning that the variance in the monetary performance indicators is low across all Monte Carlo simulations.

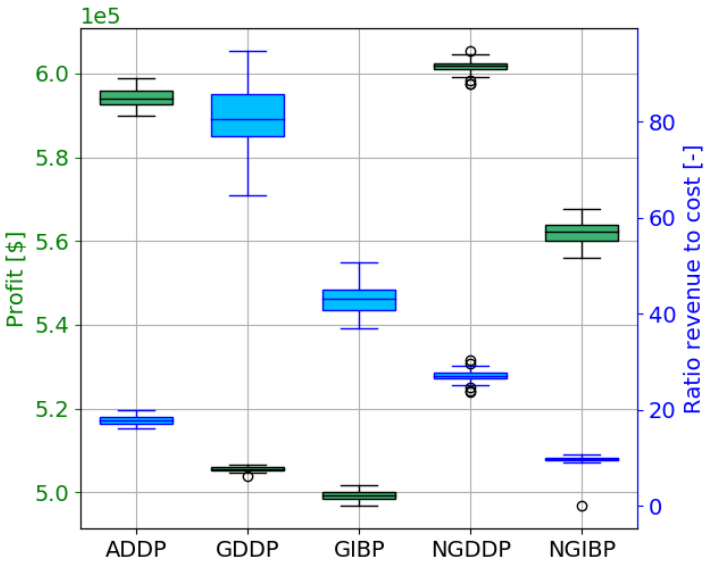


Figure 21: Box plot showing the profit and revenue to cost ratio for all maintenance scheduling strategies (50 MC simulations)

4.5 Sensitivity analysis

This section will provide a sensitivity analysis on the optimal policy vectors and most important inputs that affect the model performance. In order to keep this analysis concise, the sensitivity analysis will only be performed in detail for the maintenance scheduling strategy that was found to have the highest profit, being the NGDDP. All plots that present the sensitivity of the results to when the parameters in the optimal policy vectors for the other 4 strategies are varied can be found in Appendix I.

4.5.1 Sensitivity analysis on the optimal policy vector of best strategy

This section will investigate the sensitivity of the NGDDP optimal policy vector. For the NGDDP, the optimal policy vector consists of 5 parameters. In this analysis, each of those 5 parameters is varied along a specified range and the results on the monetary performance indicators are collected. In this section, only the 2 parameters that are most interesting to analyze are discussed in detail with associating plot. For the remainder of the policy vector sensitivity plots of the NGDDP, the reader is referred to Appendix I.

The first parameter of the NGDDP optimal policy vector that is analyzed is the multiplication factor used to determine the target RUL, f_s, as was introduced in Equation 19. The results of varying this parameter along the range [0.77-1.07] with step size 0.03 can be found in Figure 22. Note that the other values of the policy vector are kept at the optimal values shown in Table 13.

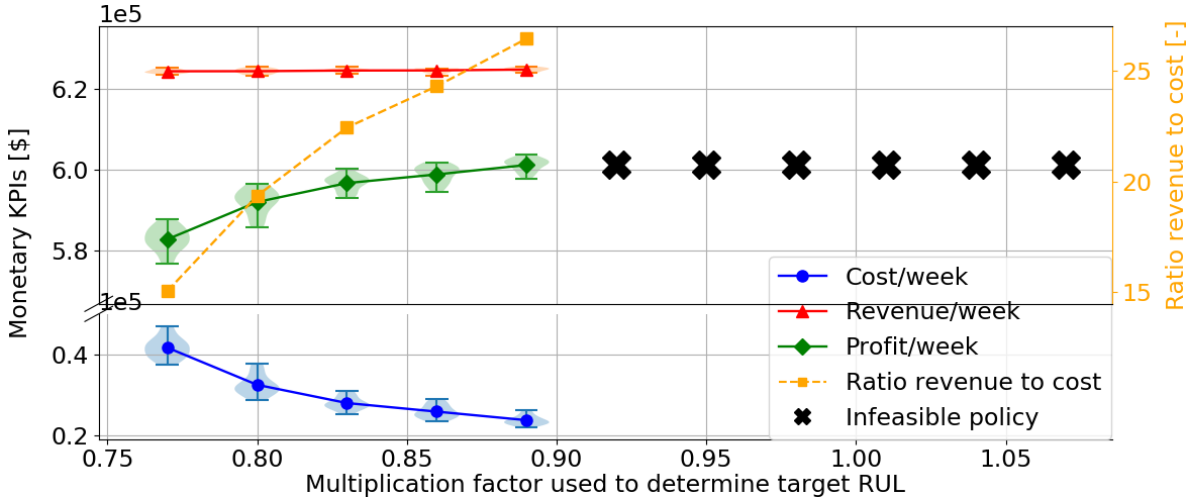


Figure 22: Sensitivity results on the multiplication factor used to compute the target RUL parameter of the optimal policy vector of the NGDDP (50 MC simulations)

In Figure 22, the cost, revenue and profit values are shown using a violin plot that shows the distribution of the outcome for 50 Monte Carlo simulations. The orange dashed line shows the mean revenue to cost ratio, which has a secondary y-axis on the right hand side. A black cross indicates an infeasible policy. This occurs if the multiplication factor that determines the target RUL becomes too big, as then the target RUL can be placed after the true moment of failure of one or more engines. This leads to engine failure during flight, which is leads to an infeasible policy. The same phenomenon would happen with very low value for the multiplication factor. If this value is very low, the target RUL can be computed to be before the current time, meaning that the time window to schedule maintenance is in the past, making the policy infeasible.

It can furthermore be seen that the profit obtained using the NGDDP steadily increases with increasing multiplication factor until the factor reaches a value of 0.89. Up to this point, the cost steadily decreases while the revenue stays constant. This leads to an increase in profit and revenue to cost ratio. If the value of the multiplication factor reaches 0.92 or higher, the NGDDP policy vector becomes infeasible. This indicates that the GA was indeed able to find the optimal value for the multiplication factor, even though this value balances right on the edge of making the policy infeasible. It is therefore concluded that the policy is quite sensitive to the multiplication factor value, as both policy infeasibility and a sharp drop in obtained profit can occur when changing the value in either direction. If an airline uses the NGDDP in its daily maintenance scheduling operations, it is advised to not be located right on the edge of policy infeasibility as this would be accompanied by severe safety risks. A solution for this is to include a safety factor on these very sensitive policy parameters such that the distance to policy infeasibility is increased.

The second parameter of the NGDDP that is analyzed is the threshold for maintenance scheduling, TH_s. This parameter is varied along the range [39-55] in with a step size of 2. The sensitivity results of varying this parameter is shown in Figure 23.

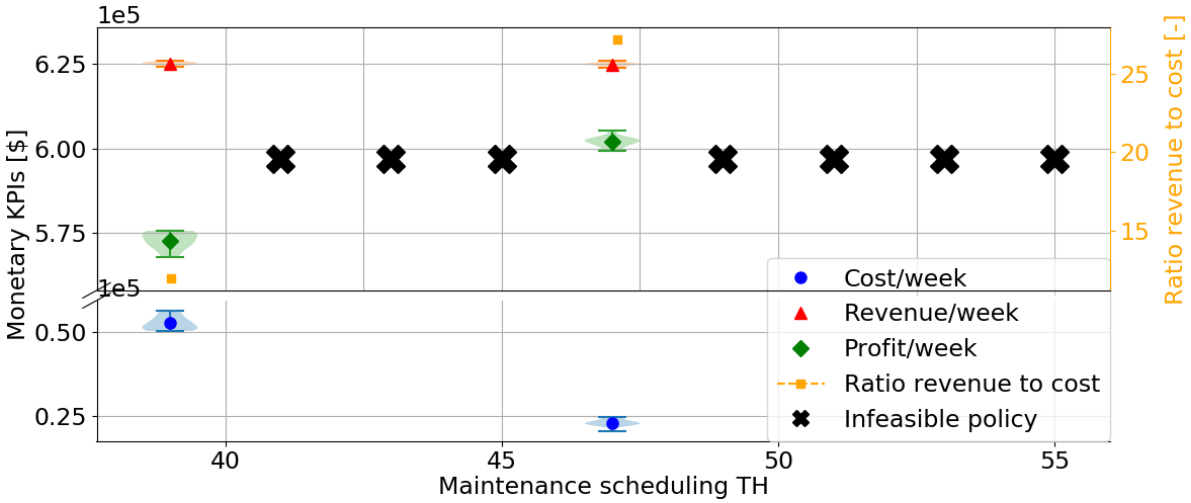


Figure 23: Sensitivity results on the threshold parameter of the optimal policy vector of the NGDDP (50 MC simulations)

In this figure it can be clearly observed that almost all maintenance scheduling threshold values surrounding the optimal value of 47 lead to an infeasible policy. This shows that the outcomes of the NGDDP policy vector are very sensitive to the scheduling threshold parameter value. Using the precise sampled engines used in this simulation model, it turns out that if a value of 47 is used for the scheduling threshold in combination with the other optimal policy vector values, the policy is never infeasible. This means that the target RUL is never computed before the current time and that the target RUL is never computed to be located after the moment of engine true failure. This shows that the policy optimization is a highly non-linear and unstable problem in which the optimal solution is surrounded by very low quality solutions. As this policy is so unstable and sensitive to the maintenance scheduling threshold value, it is not recommended to use the optimal value of 47 as this would mean that for any engine added to the simulation with slightly different degradation characteristics the model could be infeasible. On the other hand, if the value of 40 is used as shown in Figure 23, the NGDDP profit drops from $6.017 \cdot 10^5$ to only $5.687 \cdot 10^5$. This would mean that the ADDP strategy then obtains a higher profit than the NGDDP as the ADDP obtains a profit value of $5.940 \cdot 10^5$. This shows that the NGDDP is thus likely not the best policy to use in practice and only obtained the highest profit value because of a single unstable and sensitive solution. Regarding the ADDP, it was found that this strategy has a much more stable and a-sensitive optimal policy vector, meaning that in practice the ADDP is likely better to use than the NGDDP.

4.5.2 Sensitivity analysis on cost input values

In this section the performance of all the maintenance scheduling strategies will be evaluated when varying the most important cost input values along a specified range. When looking at the results obtained for the strategies with optimized policies, it was found that the cancellation cost and the inspection cost have the highest impact on the profit value of the maintenance strategy. Therefore, a sensitivity analysis on these two cost input values will be shown in this section. For the cost sensitivity analysis, the policies will not be re-optimized. If the cost input values change, it might be the case that the policy that is optimized using the GA is no longer optimal. However, it is assumed in this section that an airline that wants to make use of these maintenance scheduling strategies optimizes the scheduling policy using the best possible cost input estimations. This sensitivity analysis can then be used to analyze how the obtained profit changes if the input cost values differ from the estimations that the policies are optimized with.

Table 15 shows the profit values per aircraft per week for all the maintenance scheduling strategies when the cancellation cost is multiplied by the factor shown in the top row. The cancellation cost includes the short-term cancellation costs, the mid-term cancellation costs and the long-term cancellation costs as is defined in Equation 29. The column on the right that has a factor of $f_C = 2$ thus indicates that all other cost values are kept the same while all costs related to cancellation are twice as high. It can be seen in Table 15 that the NGDDP remains the best maintenance strategy, even if the cost of cancellation becomes twice as high. It can also be concluded that the NGIBP is very sensitive to the cancellation cost, as the profit value decreases by

14.8% when changing the multiplication factor f_C from 0.25 to 2. This high sensitivity is due to the fact that in the NGIBP, many short-term cancellations occur. If the cancellation cost value then changes by a lot, the effect on the profit value of this policy is also high. The GDDP is almost completely insensitive to the cancellation cost input value, as in the GDDP almost no cancellations occur.

Table 15: Results of the sensitivity analysis of the cancellation cost value for all maintenance scheduling strategies. The values shown in this table represent the profit values [10^5/aircraft/week] for the maintenance strategies shown in the first column

	Factor the cancellation cost is multiplied with f_C							
	0.25	*0.5*	*0.75*	*1.0*	*1.25*	*1.5*	*1.75*	*2.0*
ADDP	6.098	6.046	5.993	5.940	5.888	5.835	5.783	5.730
GDDP	5.057	5.056	5.056	5.056	5.055	5.055	5.055	5.054
GIBP	5.013	5.005	5.000	4.989	4.983	4.976	4.968	4.961
NGDDP	6.147	6.104	6.062	6.017	5.976	5.933	5.890	5.847
NGIBP	6.001	5.876	5.748	5.620	5.493	5.366	5.238	5.111

In Table 16, the cost sensitivity results for the inspection cost value can be seen when it is multiplied by a factor f_I. Again, this table shows the profit values for all strategies considered in this work. Note that the profit value for the ADDP, GDDP and NGDDP does not change with varying input inspection cost value as none of those strategies make use of inspections. Furthermore, it can be concluded that the GIBP and the NGIBP are both not very sensitive to the input inspection cost value. For the GIBP and the NGIBP, the profit values both only decrease by 2.6% when changing the multiplication factor f_I from 0.25 to 2. For all inspection cost input values, the NGDDP strategy remains the best strategy in terms of profit per week per aircraft, even when the inspection cost value is multiplied by a factor of 0.25.

Table 16: Results of the sensitivity analysis of the inspection cost value for all maintenance scheduling strategies. The values shown in this table represent the profit values [10^5/aircraft/week] for the maintenance strategies shown in the first column

	Factor the inspection cost is multiplied with f_I							
	0.25	*0.5*	*0.75*	*1.0*	*1.25*	*1.5*	*1.75*	*2.0*
ADDP	5.940	5.940	5.940	5.940	5.940	5.940	5.940	5.940
GDDP	5.056	5.056	5.056	5.056	5.056	5.056	5.056	5.056
GIBP	5.048	5.029	5.009	4.989	4.971	4.952	4.933	4.914
NGDDP	6.017	6.017	6.017	6.017	6.017	6.017	6.017	6.017
NGIBP	5.684	5.663	5.642	5.620	5.600	5.579	5.557	5.536

5 Conclusion and Discussion

This section will present the conclusions of this paper. First, the academic novelty of this work is highlighted. After that, the main conclusion are presented, as well as a discussion on the results that are obtained. Lastly, recommendations for further work are presented.

5.1 Academic novelty

This paper has researched a variety of topics: RUL prediction using a machine learning algorithm, presentation of the performance evaluation results for a prognostic algorithm and maintenance scheduling using data-driven RUL prognostics. For each of those three sub-topics, the academic novelty will be highlighted.

Regarding the RUL prediction using the CNN model, not much academic novelty is introduced. Most methods that are applied, such as the CNN architecture, data normalization, Monte Carlo dropout and sensor selection, are methods have been used before in literature. The novelty of this sub-topic thus does not lie in the specific method used in this work, but rather in the combination of those methods to obtain RUL prediction results. For example, no other works are found that use Monte Carlo dropout to obtain a probabilistic RUL predictions. Also, no other works that use the C-MAPSS dataset are found that show the prediction results for several combinations of selected input sensors. The novelty of this sub-topic is thus marginal, but sufficient to obtain superior prediction results compared to other papers.

Since Saxena et al. introduced their framework of prognostic algorithm performance evaluation, no recent literature can be found that actually makes use of this framework the evaluate the results of its prognostic model

[13]. This paper sets a standard when it comes to prognostic algorithm performance evaluation by applying a large variety of metrics to the obtained model output. No other works are found that spend this much effort on evaluating the performance of their prognostic algorithms. The metrics used in this work are both developed prior or introduced in this paper.

Lastly, no other works are found that use data-driven RUL predictions directly to schedule maintenance. This is the first work that develops several scheduling methods that can be used to schedule maintenance while making use of raw engine time series recordings. This makes the methods designed in this sub-topic novel and highly relevant for airlines that want to incorporate data-driven techniques into their daily scheduling operations.

5.2 Conclusions

In this paper it was first investigated how sensor recordings from an aircraft engine can be used to predict the engine's RUL. A convolutional neural network (CNN) was developed that uses 1D-filters to slide over the 2D-input arrays along the temporal dimension. In this way, the filter kernels of the CNN act as input smoothing filters that are able to extract useful features from the sensor data. The CNN consists of 5 convolutional layers followed by a fully connected layer in which Monte Carlo dropout is applied. This method allows for obtaining probabilistic RUL distributions that include model uncertainty. The proposed prognostic model was then tested on the C-MAPSS dataset. A large framework of prognostic model performance evaluation metrics was set-up, taking into account the most important requirements for prognostic model performance. By evaluating the obtained results it can be concluded that the CNN obtains low RMSE values on all instances of the C-MAPSS dataset. Furthermore, the use of all performance metrics is emphasized as the metrics allow for a thorough comparison of this prognostic model to other models. The obtained error values are also significantly lower than the error values obtained in recent literature. This shows the potential of the CNN method. Further reasons as to why this model outperforms other recent prognostic models are the normalization of the data with respect to the current operating condition, the extensive experimentation on the features to include as inputs and the use of Monte Carlo dropout. It can thus be concluded that the prognostic model developed in this work outperforms models introduced in recent literature, which was one of the goals of this research.

Secondly, it was investigated how probabilistic data-driven RUL predictions can be used in the daily scheduling routine of an airline. First, 3 different maintenance policies were developed that take as input the probabilistic data-driven RUL predictions obtained using the CNN and return a target RUL as output. The target RUL indicates the latest possible moment in time maintenance can be scheduled for an engine. The 3 maintenance polices investigated in this work are a data-driven policy (DDP), an advanced data-driven policy (ADDP) and an inspection based policy (IBP). Furthermore, the ADDP is able to use the probabilistic data-driven RUL predictions to create maintenance opportunities in the flight schedule at a very early stage. The DDP and IBP are then combined with both a flight schedule with a standard weekly maintenance slot and a flight schedule that has no standard maintenance slots. This creates a total of 5 different maintenance strategies that are investigated in this paper. Picking the optimal maintenance opportunities for the strategies with a standard maintenance opportunity is done using an Integer Linear Programming (ILP) model that minimizes the time between the moment of maintenance and the target RUL of an engine while respecting the hangar capacity. The data-driven maintenance policies are optimized using a genetic algorithm.

The input data for the maintenance scheduling simulation model is obtained by sampling run-to-failure training engines from the C-MAPSS training datasets that the CNN model is not trained on. These run-to-failure RUL predictions were then used by the data-driven maintenance strategies to simulate maintenance scheduling. The evaluation of the 5 maintenance strategies is done using both operational and monetary performance indicators, where the airline profit rate is considered to be the most important metric. It was found that the data-driven policy that uses a flight schedule with no standard maintenance slots achieves the highest profit rate due to the fact that the optimized policy vector has a relatively high value for the maintenance scheduling threshold. This ensures that flight cancellations that are needed to create a maintenance opportunity are mostly mid-term in stead of short-term, resulting in lower cancellation costs. Furthermore, it can be concluded that all inspection based strategies underperform the data-driven strategies with the same type of flight schedule. Also, strategies that make use of the gapped flight schedule severely underperform the strategies that make use of flight cancellation in order to create maintenance slots. This is due to the fact that in the gapped flight schedule strategies, aircraft utilization is much lower. Lastly, using a sensitivity analysis on the optimal policy vector of the NGDDP strategy that obtained the highest profit, it was found that the policy is very sensitive to a change in the maintenance scheduling threshold. It can be concluded that the optimal maintenance scheduling strategy, the NGDDP, only achieved a high profit value due to a very specific combination of policy vector parameter values. It can therefore not be concluded that the NGDDP maintenance strategy actually is the optimal strategy to implement in practice. The belief is created that the advanced data-driven policy strategy is the optimal maintenance strategy as this strategy is far less sensitive to a change in policy vector parameter

values. Also, if more accurate RUL predictions at an early stage could be obtained by improving the prognostic model, this policy will be able to create maintenance opportunities at even better moments in time, thereby increasing the performance of this strategy even more. Overall, it was shown how data-driven RUL prognostics can be incorporated into the daily scheduling routine using data-driven maintenance scheduling strategies. The results are hopeful, as the data-driven strategies outperform current strategies that rely on physical inspections for a given type of flight schedule.

5.3 Recommendations for further work

This section will enumerate the recommendations for further work on the topic of machine learning for RUL predictions and on using RUL prognostics to schedule maintenance. These recommendations mainly focus on extending the developed models in order to improve the results and make them more realistic.

1. In stead of manually tuning the hyperparameters of the CNN model, an automated algorithm that is able to find the optimal combination of hyperparameters that results in the best prediction performance could be used.

2. In stead of using the MAE as loss function when training the CNN, a more complex function such as the CRPS or weighted scored CRPS that can take into account specific requirements about the prognostic algorithm could be used.

3. When both engines of the same aircraft have a target RUL close to each other, replacing both engines during the same maintenance opportunity to save money on towing and hangar visits, and to increase aircraft utilization could be considered.

4. In this research, the time required for an inspection is not taken into account when considering the schedule for the inspection based strategies. This assumption can be eliminated, resulting in even worse performance of the inspection based strategies.

5. To obtain an even better picture of how all maintenance strategies perform, more scenarios can be tested, such as scenarios with more than 5 aircraft or where aircraft have more than 2 engines.

6. In this research, spare part and stock management is not taken into account. It is just assumed that if maintenance is performed there always is an overhauled engine available that is assumed to be as good as new. These assumptions could be relaxed.

Taking these recommendations into consideration will improve how realistic the maintenance scheduling model is in comparison to a real-world scenario. Lastly, this maintenance simulation model for aircraft engines can also be combined with other aircraft parts that can be monitored using sensor data. In this way, this model could schedule maintenance for all aircraft in a fleet for multiple components. In this case, the model can be extended to be able to group maintenance actions of several components during a single hangar visit.

References

[1] KLM Board of Managing Directors. Annual report 2019 royal dutch airlines. Retreived from `https://www.klm.com/travel/nl_nl/images/KLM-Jaarverslag-2019_tcm541-1063986.pdf`, 2019.

[2] V. Nguyen, M. Kefalas, K. Yang, A. Apostolidis, M. Olhofer, S. Limmer, and T. Bäck. A review: Prognostics and health management in automotive and aerospace. *International Journal of Prognostics and Health Management*, 10(2), 11 2019.

[3] A. Saxena and K. Goebel. Turbofan engine degradation simulation data set. `https://ti.arc.nasa.gov/c/13/`, 2008. NASA Ames, Moffet Field, CA.

[4] G. S. Babu, P. Zhao, and X. Li. Deep convolutional neural network based regression approach for estimation of remaining useful life. *Database Systems for Advanced Applications*, 9642:214–228, 2016.

[5] X. Li, Q. Ding, and J. Sun. Remaining useful life estimation in prognostics using deep convolution neural networks. *Reliability Engineering and System Safety*, 172:1–11, 2018. `doi:10.1016/j.ress.2017.11.021`.

[6] A. Saxena, K. Goebel, D. Simon, and N. Eklund. Damage propagation modeling for aircraft engine run-to-failure simulation. In *2008 International Conference on Prognostics and Health Management*, pages 1–9, 2008. `doi:10.1109/PHM.2008.4711414`.

[7] F. O. Heimes. Recurrent neural networks for remaining useful life estimation. In *2008 International Conference on Prognostics and Health Management*, pages 1–6, 2008. doi:10.1109/PHM.2008.4711422.

[8] S. Zheng, K. Ristovski, A. Farahat, and C. Gupta. Long short-term memory network for remaining useful life estimation. In *2017 IEEE International Conference on Prognostics and Health Management (ICPHM)*, pages 88–95, 2017. doi:10.1109/ICPHM.2017.7998311.

[9] A. L. 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 183*, 183:240–251, 2019. doi:10.1016/j.ress.2018.11.027.

[10] O. O. Aremu, D. Hyland-Wood, and P. R. McAree. A machine learning approach to circumventing the curse of dimensionality in discontinuous time series machine data. *Reliability Engineering and System Safety*, 195, 2020. doi:10.1016/j.ress.2019.106706.

[11] W. Yu, Y. Kim, and C. Mechefske. An improved similarity-based prognostic algorithm for rul estimation using an rnn autoencoder scheme. *Reliability Engineering and System Safety*, 199, 2020. doi:10.1016/j.ress.2020.106926.

[12] V. TV, Diksha, P. Malhotra, L. Vig, and G. Shroff. Data-driven prognostics with predictive uncertainty estimation using ensemble of deep ordinal regression models, 2021. arXiv:1903.09795.

[13] A. Saxena, J. Celaya, E. Balaban, K. Goebel, B. Saha, S. Saha, and M. Schwabacher. Metrics for evaluating performance of prognostic techniques. In *2008 International Conference on Prognostics and Health Management*, pages 1–17, 2008. doi:10.1109/PHM.2008.4711436.

[14] A. Saxena, J. Celaya, B. Saha, S. Saha, and K. Goebel. Metrics for offline evaluation of prognostic performance. *International Journal of Prognostics and Health Management*, 1(1):2153–2648, 2010.

[15] M. Baur, P Albertelli, and M Monno. A review of prognostics and health management of machine tools. *International Journal Advanced Manufacturing Technologies*, 107:2843—-2863, 2020.

[16] T. Gneiting and A. E Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007. doi:10.1198/016214506000001437.

[17] K. Schneider and R. Cassady. Evaluation and comparison of alternative fleet-level selective maintenance models. *Reliability Engineering and System Safety*, 134, 02 2015. doi:10.1016/j.ress.2014.10.017.

[18] J. Y. J. Lam and D. Banjevic. A myopic policy for optimal inspection scheduling for condition based maintenance. *Reliability Engineering and System Safety*, 144:1–11, 2015.

[19] H. Vu, P. Do, A. Barros, and C. Bérenguer. Maintenance grouping strategy for multi-component systems with dynamic contexts. *Reliability Engineering and System Safety*, 132:233–249, 2014.

[20] S. Wu and I. Castro. Maintenance policy for a system with a weighted linear combination of degradation processes. *European Journal of Operational Research*, 280:124–133, 2020.

[21] L. Zhang and J. Zhang. A data-driven maintenance framework under imperfect inspections for deteriorating systems using multitask learning-based status prognostics. *IEEE Access*, PP:1–1, 12 2020. doi:10.1109/ACCESS.2020.3047928.

[22] I. de Pater and M. Mitici. Predictive maintenance for multi-component systems of repairables with remaining-useful-life prognostics and a limited stock of spare components. *Reliability Engineering and System Safety*, 214, 10 2021. doi:10.1016/j.ress.2021.107761.

[23] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989. doi:10.1162/neco.1989.1.4.541.

[24] L. V. Jospin, W. Buntine, F. Boussaid, H. Laga, and M. Bennamoun. Hands-on bayesian neural networks - a tutorial for deep learning users. *ACM Computing Surveys*, 1(1), 2020.

[25] Y. Gal and Z. Ghahramani. Bayesian convolutional neural networks with bernoulli approximate variational inference. In *International Conference on Learning Representations 2016*, 2016. arXiv:1506.02158.

[26] B. Zhang, K. Zheng, Q. Huang, S. Feng, S. Zhou, and Y. Zhang. Aircraft engine prognostics based on informative sensor selection and adaptive degradation modeling with functional principal component analysis. *Sensors*, 20(3):920, 2020. doi:10.3390/s20030920.

[27] T. Wang, J. Yu, D. Siegel, and J. Lee. A similarity-based prognostics approach for remaining useful life estimation of engineered systems. In *2008 International Conference on Prognostics and Health Management*, pages 1–6, 2008. doi:10.1109/PHM.2008.4711421.

[28] B. Zhang, K. Zheng, Q. Huang, S. Feng, S. Zhou, and Y. Zhang. Aircraft engine prognostics based on informative sensor selection and adaptive degradation modeling with functional principal component analysis. *Sensors*, 20(3):920, 2020. doi:10.3390/s20030920.

[29] P. Malhotra, T. Vishnu, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff. Multi-sensor prognostics using an unsupervised health index based on lstm encoder-decoder, 2016. arXiv:1608.06154.

[30] C. Zhang, L. Pin, A. K. Qin, and K. C. Tan. Multiobjective deep belief networks ensemble for remaining useful life estimation in prognostics. *IEEE Transactions on Neural Networks and Learning Systems*, PP:1–13, 07 2016. doi:10.1109/TNNLS.2016.2582798.

[31] S. A. Boekweit. Fleet level multi-unit maintenance optimization subject to degradation. Msc thesis, Delft University of Technology, 02 2021.

[32] R. J. Hyndman and A. B. Koehler. Another look at measures of forecast accuracy. *International Journal of Forecasting*, 22:679–688, 2006.

[33] S. Makridakis, A. Andersen, R. Carbone, R. Fildes, M. Hibon, R. Lewandowski, J. Newton, E. Parzen, and R. Winkler. The accuracy of extrapolation (time series) methods: Results of a forecasting competition. *Journal of Forecasting*, 1(2):111–153, 1982.

[34] K. Goebel and P. Bonissone. Prognostic information fusion for constant load systems. In *2005 7th International Conference on Information Fusion*, volume 2, pages 9 pp.–, 2005. doi:10.1109/ICIF.2005.1592000.

[35] G. Vachtsevanos, F. Lewis, M. Roemer, A. Hess, and B. Wu. *Intelligent Fault Diagnosis and Prognosis for Engineering Systems*. John Wiley & Sons, Inc., Hoboken, NJ, USA, 2006.

[36] D. C. Hoaglin, F. Mosteller, and J. W. Tukey. *Understanding Robust and Exploratory Data Analysis*. John Wiley & Sons, Inc., Hoboken, NJ, USA, 1983.

Appendices

A Appendix 1 - General error metrics

Table 17: General metrics used for the assessment of accuracy and precision as stated in [13]

Metric name	Definition	Range	References
	accuracy based metrics		
Error	$\epsilon(i) = r_{\text{true}}(i) - r_{\text{pred}}(i)$	(-∞, ∞)	-
Mean Absolute Error	$MAE(i) = \frac{1}{L}\sum_{l=1}^{L}\lvert\epsilon^l(i)\rvert$	[0, ∞)	[32]
Mean Absolute Percentage Error	$MAPE(i) = \frac{1}{L}\sum_{l=1}^{L}\left\lvert\frac{100\cdot\epsilon^l(i)}{r_{\text{true}}}\right\rvert$	[0, ∞)	[32] [33]
Mean Squared Error	$MSE(i) = \frac{1}{L}\sum_{l=1}^{L}\epsilon^l(i)^2$	[0, ∞)	[32]
Root Mean Squared Error	$RMSE(i) = \sqrt{\frac{1}{L}\sum_{l=1}^{L}\epsilon^l(i)^2}$	[0, ∞)	[32]
Root Mean Squared Percentage Error	$RMSPE(i) = \sqrt{\frac{1}{L}\sum_{l=1}^{L}\left\lvert\frac{100\cdot\epsilon^l(i)}{r_{\text{true}}}\right\rvert}$	[0, ∞)	[32]
False Positive	$FP(r^l_{\text{true}}(i)) = \begin{cases} 1 & \text{if } \epsilon^l(i) > t_{FP} \\ 0 & \text{otherwise} \end{cases}$	[0, 1]	[34]
False Negative	$FN(r^l_{\text{true}}(i)) = \begin{cases} 1 & \text{if } -\epsilon^l(i) > t_{FN} \\ 0 & \text{otherwise} \end{cases}$	[0, 1]	[34]
	precision based metrics		
Error Standard Deviation	$ESTD(i) = \sqrt{\frac{\sum_{l=1}^{n}(\epsilon^l(i) - M)^2}{n-1}}$	[0, ∞)	[35] [36]
Mean Absolute Deviation	$MAD(i) = \frac{1}{n}\sum_{l=1}^{n}\lvert\epsilon^l(i) - Md\rvert$	[0, ∞)	[36]
Median Absolute Deviation	$MdAD(i) = \text{median}(\lvert\epsilon^l(i) - Md\rvert)$	[0, ∞)	[36]

B Appendix 2 - Additional information on the C-MAPPS dataset

The C-MAPSS data set simulates engines with a 90,000 lb thrust class, for several operating conditions, altitudes, Mach numbers and temperatures. In Figure 24, the general layout of a turbofan engine can be seen. Figure 25 shows the different modules within the engine, and shows how those modules are connected. C-MAPPS takes 14 inputs to simulate the various degradation scenarios of the simulated engine [3, 6].

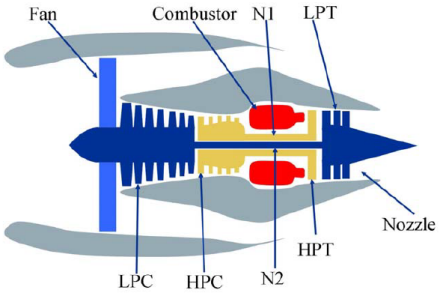


Figure 24: Overview of the simulated turbofan layout [3, 6]

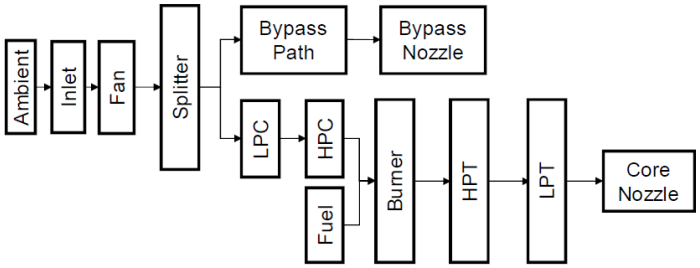


Figure 25: Modules and connections of the turbofan engine [3, 6]

The output of the C-MAPSS software is a run-to-failure data set of 21 time series signals that can be considered to be the sensor measurements of the engine. The sensors that are present in the C-MAPSS data set that can be used to predict degradation of the engine can be found in Table 18.

Table 18: Overview of the 21 sensors that are included in the C-MAPSS data set [3, 6]

Symbol	Description	Units
Parameters available as sensor data		
T2	Total temperature at fan inlet	°R
T24	Total temperature at LPC outlet	°R
T30	Total temperature at HPC outlet	°R
T50	Total temperature at LPT outlet	°R
P2	Pressure at fan inlet	psia
P15	Total pressure in bypass-duct	psia
P30	Total pressure at HPC outlet	psia
Nf	Physical fan speed	rpm
Nc	Physical core speed	rpm
epr	Engine pressure ratio (P50/P2)	–
Ps30	Static pressure at HPC outlet	psia

Symbol	Description	Units
Parameters available as sensor data		
phi	Ratio of fuel flow to Ps30	pps/psi
NRf	Corrected fan speed	rpm
NRc	Corrected core speed	rpm
BPR	Bypass Ratio	–
farB	Burner fuel-air ratio	–
htBleed	Bleed Enthalpy	–
Nf_dmd	Demanded fan speed	rpm
PCNfR_dmd	Demanded corrected fan speed	rpm
W31	HPT coolant bleed	lbm/s
W32	LPT coolant bleed	lbm/s

C Appendix 3 - Raw non-normalized sensor data for FD001

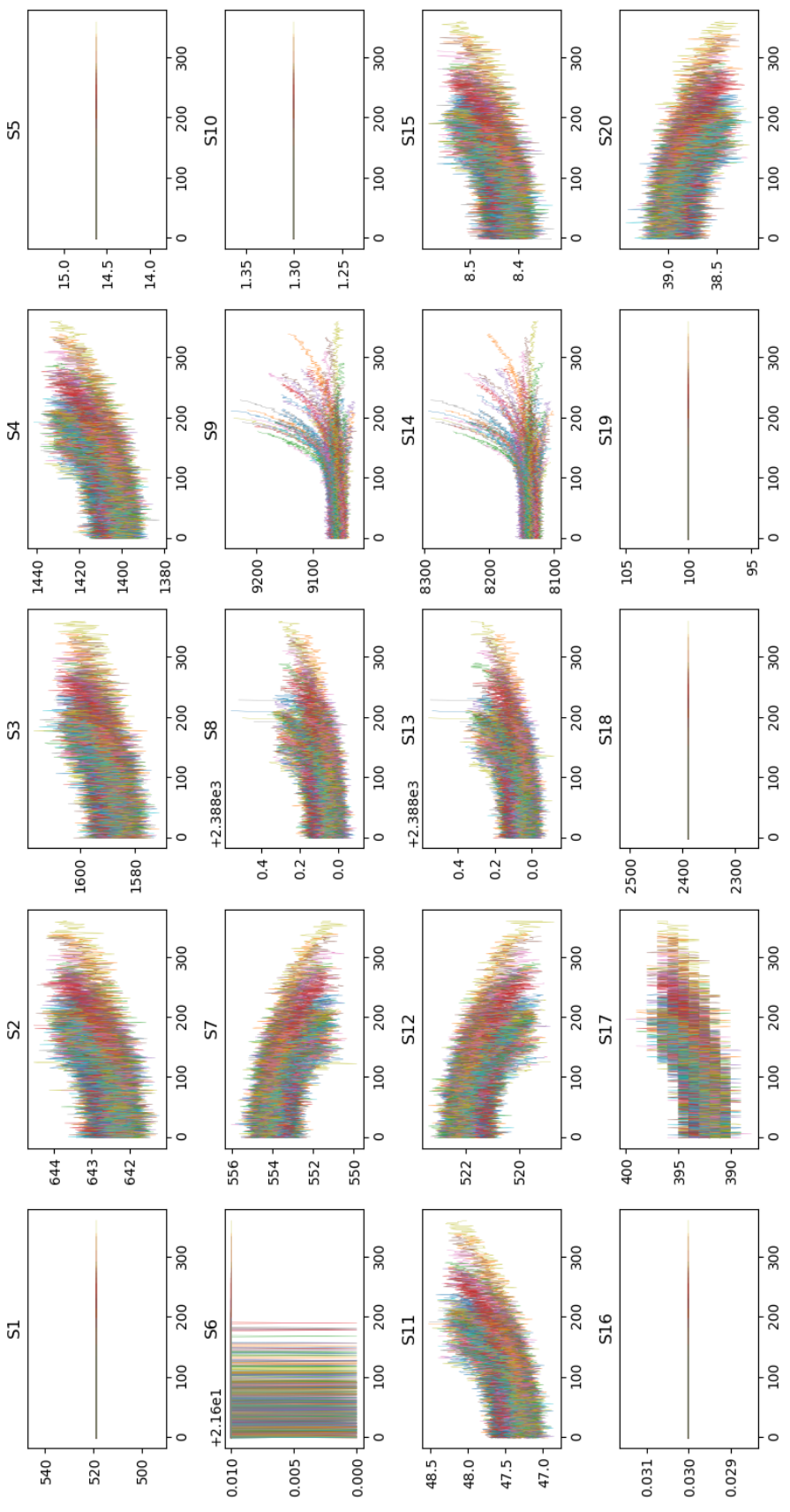


Figure 26: Raw non-normalized sensor signals for all 21 sensors in instance FD001. The values on the x-axis indicate the number of flight cycles the sensor has recorded

D Appendix 4 - Normalized sensor data for FD004 used as input

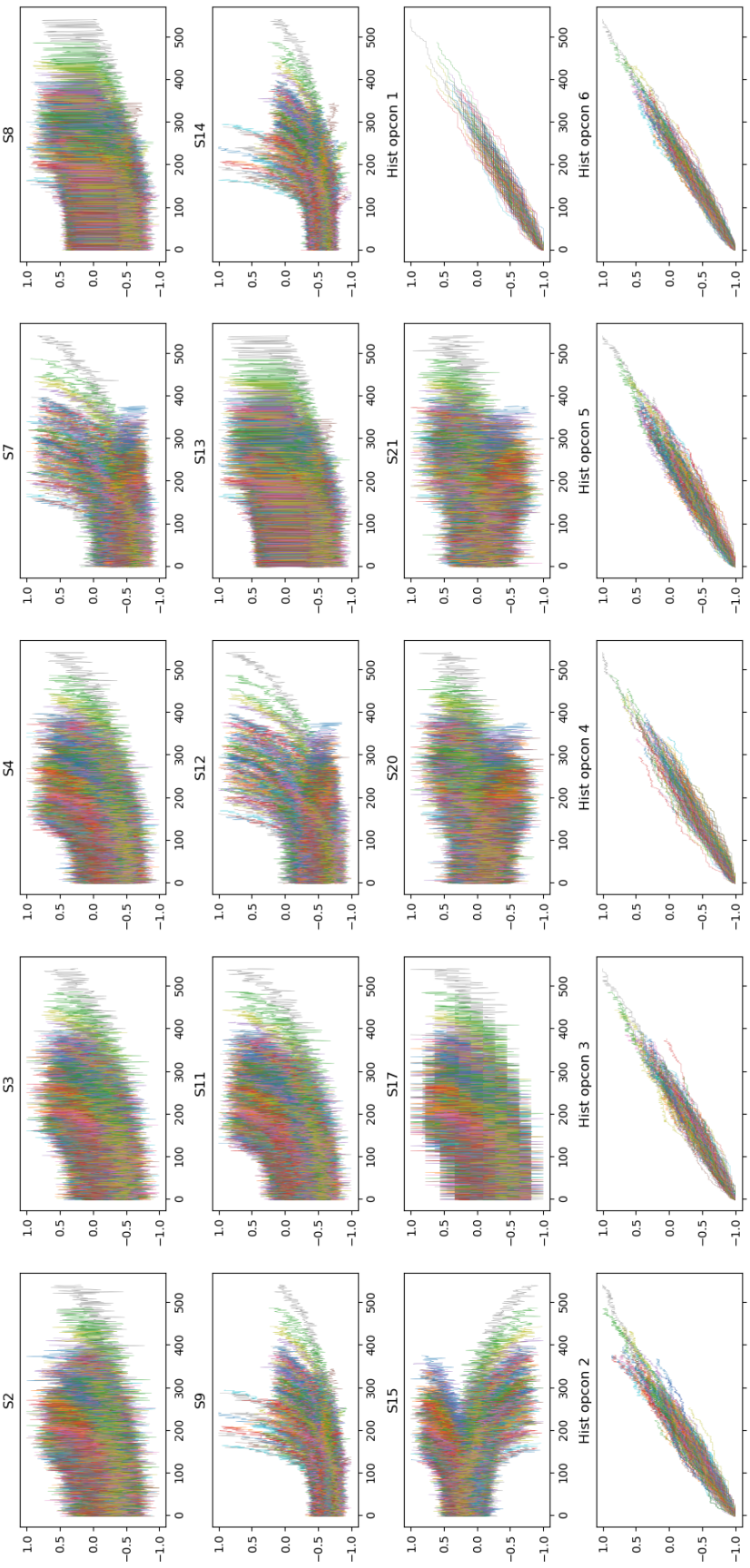


Figure 27: Normalized sensor signals for the 14 selected sensors in instance FD004 as well as the normalized operating condition history for the 6 operating conditions in FD004. The values on the x-axis indicate the number of flight cycles the sensor has recorded

E Appendix 5 - Additional RUL prediction results for FD002-FD004

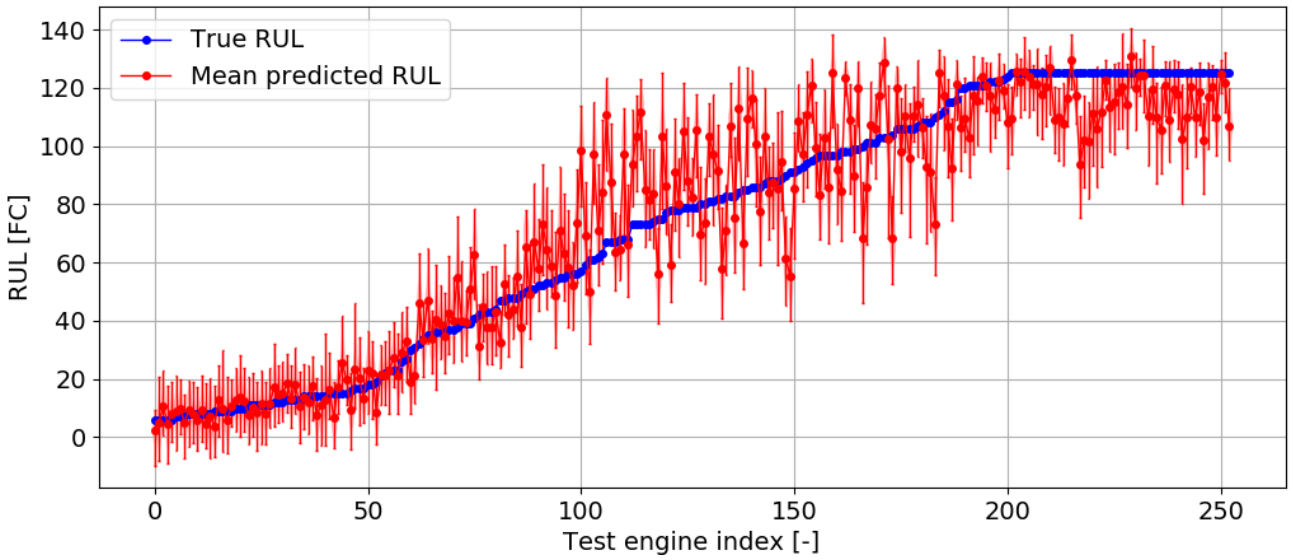


Figure 28: True RUL and predicted RUL distributions for all 259 engines in FD002 sorted by true RUL

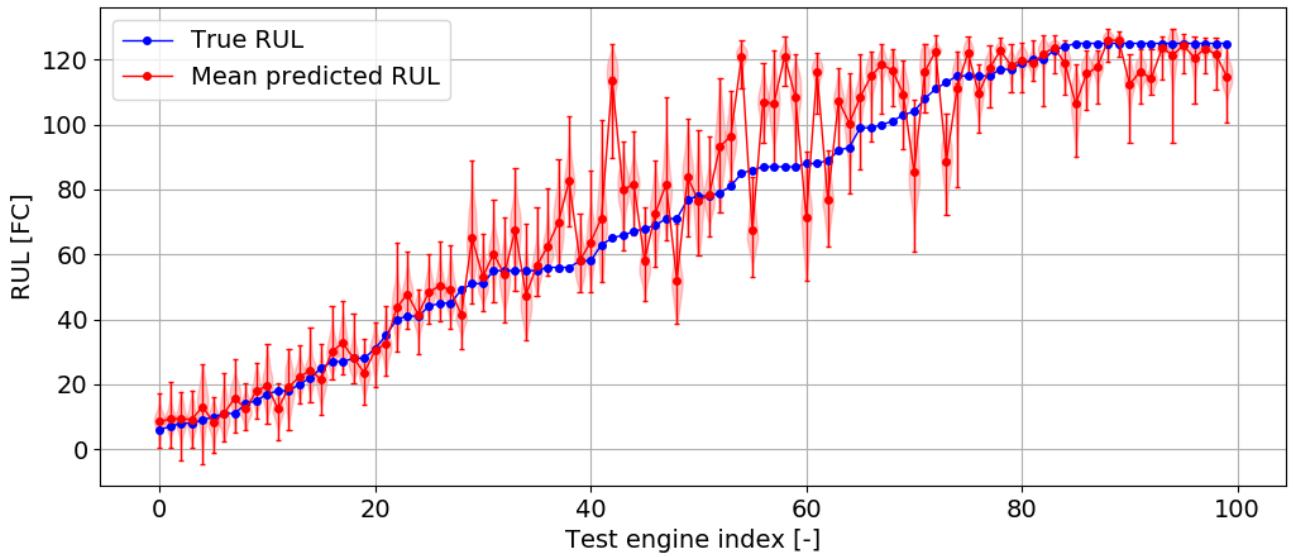


Figure 29: True RUL and predicted RUL distributions for all 100 engines in FD003 sorted by true RUL

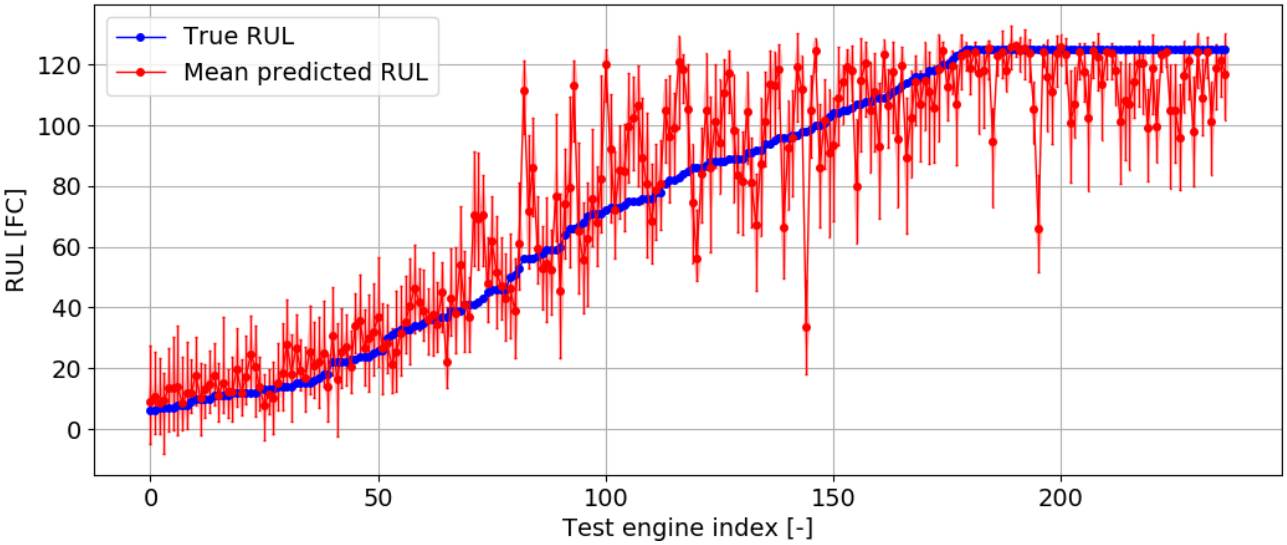
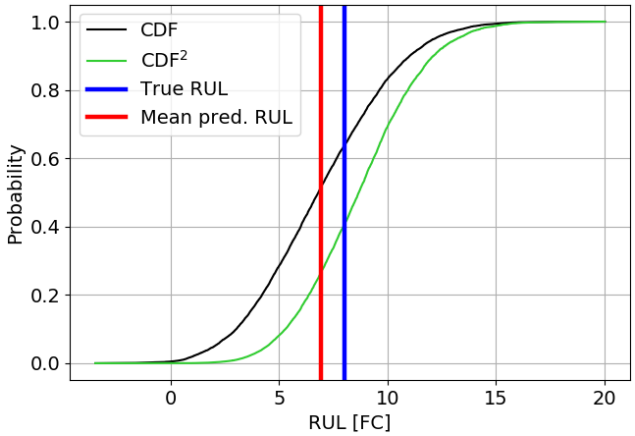
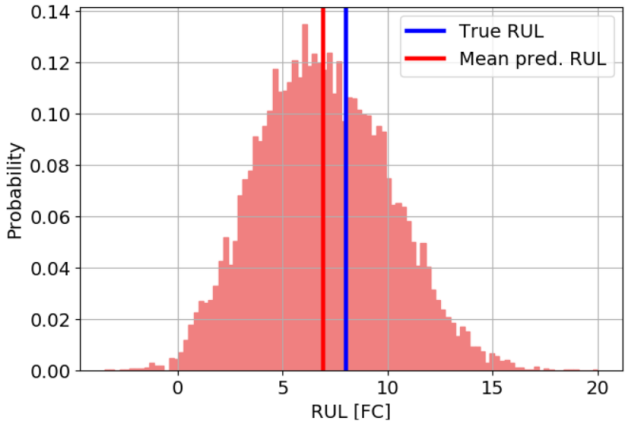


Figure 30: True RUL and predicted RUL distributions for all 248 engines in FD004 sorted by true RUL

F Appendix 6 - Additional RUL prediction results for individual engines at with different true RUL values

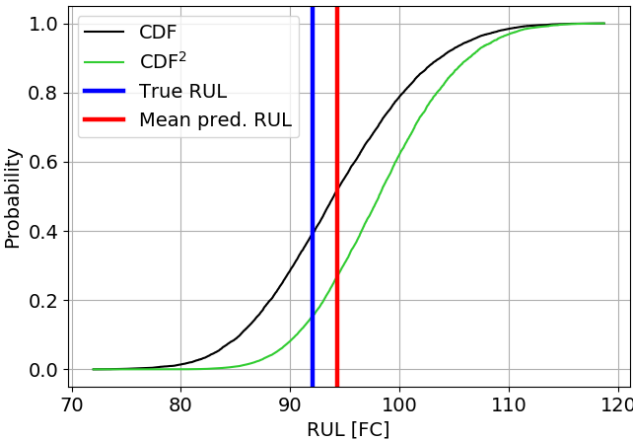


(a) CDF and CDF^2 of 10,000 MC predictions

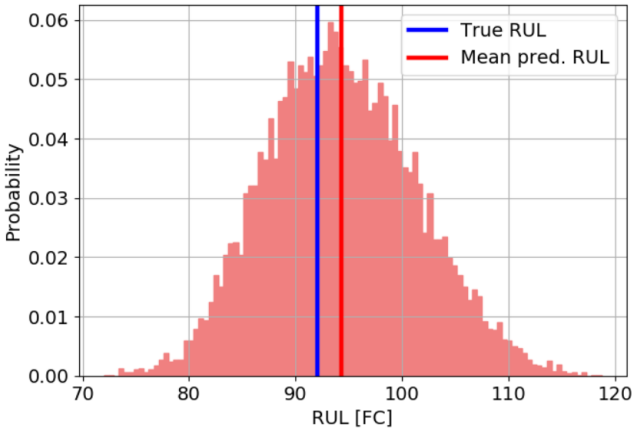


(b) Histogram of 10,000 MC predictions

Figure 31: MC prediction results on test engine 1 from FD001

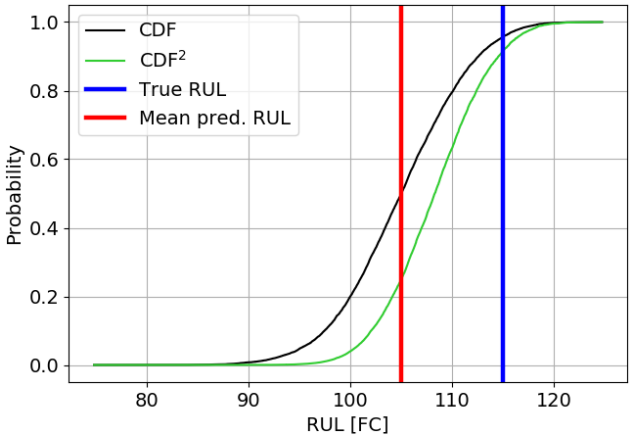


(a) CDF and CDF^2 of 10,000 MC predictions

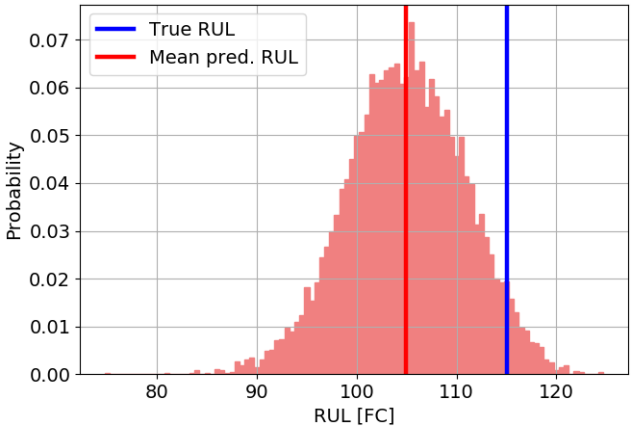


(b) Histogram of 10,000 MC predictions

Figure 32: MC prediction results on test engine 56 from FD001



(a) CDF and CDF^2 of 10,000 MC predictions



(b) Histogram of 10,000 MC predictions

Figure 33: MC prediction results on test engine 80 from FD001

G Appendix 7 - Performance evaluation of all sampled training engines using all metrics

Table 19: Sampled training engines E1.6-E2.248 that are used for run-to-failure testing evaluated on the first half of the most important metrics. The mean and standard deviation of the metrics across all 99 sampled training engines are also shown

	Prognostic metric											
Engine ID	**MAE**	**MAPE**	**RMSE**	**ESTD**	**SRR**	**FP**	**FN**	**TP**	**TN**	**Score**	**W. score**	**Acc.**
Mean	10.22	21.22	13.57	8.84	0.24	0.03	0.16	0.42	0.32	4.43	2.9	0.79
St. dev.	4.67	11.38	5.4	3.02	0.18	0.04	0.17	0.23	0.15	6.88	4.5	0.18
E1.6	9.32	18.31	12.79	8.76	0.92	0.09	0.12	0.6	0.13	2.92	1.93	0.78
E1.16	13.19	18.61	17.66	11.75	0.27	0.09	0.06	0.58	0.19	3.71	1.5	0.84
E1.19	10.17	18.59	14.1	9.77	0.26	0.05	0.17	0.44	0.21	3.96	1.89	0.75
E1.26	10.72	23.97	16.1	12.01	0.26	0.05	0.12	0.44	0.37	4.33	1.7	0.83
E1.34	4.34	8.14	5.86	3.93	0.08	0.01	0.02	0.56	0.33	0.53	0.34	0.97
E1.38	5.6	12.61	7.75	5.36	0.17	0.0	0.04	0.46	0.44	0.96	0.56	0.96
E1.45	13.52	25.36	17.75	11.5	0.16	0.02	0.43	0.09	0.33	7.15	4.26	0.49
E1.51	6.33	9.96	7.71	4.41	0.49	0.0	0.03	0.54	0.38	0.85	0.47	0.97
E1.66	12.37	19.13	16.43	10.82	0.2	0.05	0.08	0.57	0.23	3.34	1.25	0.86
E1.67	11.17	16.22	14.47	9.19	0.21	0.03	0.05	0.84	0.04	2.21	1.15	0.92
E1.68	6.34	10.91	9.24	6.72	0.24	0.0	0.06	0.62	0.27	1.03	0.41	0.94
E1.75	13.53	30.96	16.53	9.5	0.09	0.15	0.13	0.59	0.1	3.95	3.12	0.71
E1.77	18.09	38.67	21.34	11.32	0.56	0.04	0.47	0.09	0.3	8.73	5.48	0.43
E1.87	10.98	21.81	13.84	8.42	0.34	0.07	0.14	0.5	0.25	3.05	1.89	0.78
E2.6	6.43	18.97	9.16	6.52	0.02	0.0	0.16	0.21	0.51	1.46	1.28	0.82
E2.19	9.19	23.39	10.97	5.99	0.21	0.19	0.05	0.1	0.64	1.93	1.37	0.75
E2.23	7.54	19.29	9.87	6.38	0.5	0.0	0.13	0.49	0.32	1.63	1.34	0.86
E2.28	7.77	16.98	10.07	6.41	0.07	0.11	0.04	0.48	0.29	1.3	1.04	0.85
E2.30	14.0	34.69	17.71	10.84	0.03	0.0	0.39	0.14	0.33	6.2	4.57	0.54
E2.38	6.65	22.45	10.05	7.53	0.64	0.0	0.19	0.25	0.53	1.96	1.94	0.81
E2.52	12.09	21.46	15.89	10.31	0.74	0.0	0.3	0.19	0.43	4.84	3.23	0.68
E2.62	6.56	22.68	8.06	4.69	0.12	0.0	0.13	0.41	0.4	1.06	0.94	0.86
E2.67	18.47	40.28	22.09	12.12	0.26	0.0	0.59	0.01	0.35	11.79	6.1	0.38
E2.69	28.1	66.8	32.98	17.27	0.48	0.0	0.69	0.0	0.14	46.66	32.37	0.17
E2.75	12.09	18.28	17.05	12.02	0.29	0.06	0.11	0.54	0.22	4.33	1.95	0.81
E2.80	11.42	16.5	14.29	8.59	0.44	0.02	0.09	0.64	0.17	2.28	1.45	0.88
E2.84	10.68	15.04	14.8	10.24	0.02	0.04	0.02	0.7	0.18	2.43	0.93	0.93
E2.89	6.67	17.3	8.04	4.49	0.13	0.0	0.08	0.56	0.31	0.97	0.72	0.92
E2.100	9.48	18.84	13.05	8.97	0.13	0.04	0.07	0.63	0.2	1.95	1.11	0.88
E2.103	11.15	15.15	14.2	8.79	0.21	0.03	0.09	0.67	0.14	2.77	2.54	0.88
E2.106	10.87	15.52	14.92	10.22	0.14	0.01	0.04	0.63	0.26	2.49	1.12	0.95
E2.117	7.77	14.54	10.6	7.21	0.18	0.0	0.14	0.41	0.32	1.99	1.73	0.84
E2.136	14.42	33.9	18.34	11.34	0.55	0.04	0.44	0.12	0.35	7.14	5.51	0.5
E2.139	6.17	15.05	8.66	6.07	0.52	0.01	0.05	0.81	0.07	0.87	0.53	0.94
E2.146	8.34	20.29	12.15	8.83	0.38	0.0	0.17	0.4	0.38	2.95	3.22	0.82
E2.150	5.39	13.34	6.92	4.35	0.27	0.02	0.03	0.5	0.38	0.75	0.63	0.95
E2.153	18.13	24.01	22.49	13.31	0.5	0.13	0.03	0.73	0.06	6.35	2.27	0.83
E2.159	10.98	13.57	13.8	8.36	0.21	0.03	0.01	0.81	0.08	1.91	0.88	0.95
E2.188	19.13	37.71	23.89	14.3	0.16	0.03	0.58	0.02	0.29	18.28	9.77	0.34
E2.198	9.41	29.26	14.44	10.96	0.14	0.07	0.12	0.53	0.27	2.87	1.47	0.81
E2.202	5.31	13.91	6.76	4.18	0.45	0.07	0.02	0.67	0.18	0.61	0.53	0.9
E2.206	7.54	14.24	10.52	7.33	0.19	0.04	0.16	0.25	0.47	1.91	1.7	0.79
E2.214	18.11	34.18	22.64	13.58	0.22	0.04	0.53	0.06	0.26	12.53	7.98	0.36
E2.222	7.17	17.36	9.0	5.45	0.01	0.02	0.11	0.6	0.24	1.06	0.99	0.86
E2.226	10.64	34.87	13.98	9.06	0.07	0.0	0.27	0.2	0.5	3.76	2.54	0.72
E2.230	6.35	15.72	7.88	4.67	0.32	0.0	0.15	0.38	0.38	1.04	1.02	0.84
E2.232	9.63	12.52	13.32	9.2	0.1	0.07	0.0	0.77	0.1	1.86	0.83	0.93
E2.234	16.51	32.59	21.59	13.92	0.02	0.0	0.43	0.08	0.35	13.36	9.16	0.5
E2.245	14.73	20.82	21.35	15.46	0.39	0.18	0.05	0.55	0.16	7.03	2.2	0.75
E2.248	10.29	15.34	13.95	9.43	0.62	0.04	0.02	0.72	0.16	2.18	0.93	0.93

Table 20: Sampled training engines E2.249-E4.233 that are used for run-to-failure testing evaluated on the first half of the most important metrics. The mean and standard deviation of the metrics across all 99 sampled training engines are also shown

	Prognostic metric											
Engine ID	**MAE**	**MAPE**	**RMSE**	**ESTD**	**SRR**	**FP**	**FN**	**TP**	**TN**	**Score**	**W. score**	**Acc.**
Mean	10.22	21.22	13.57	8.84	0.24	0.03	0.16	0.42	0.32	4.43	2.9	0.79
St. dev.	4.67	11.38	5.4	3.02	0.18	0.04	0.17	0.23	0.15	6.88	4.5	0.18
E2.249	10.9	21.48	13.89	8.62	0.22	0.03	0.22	0.46	0.24	3.54	4.17	0.74
E3.6	12.95	19.61	16.05	9.47	0.19	0.08	0.04	0.73	0.08	2.73	1.41	0.87
E3.16	4.24	11.72	6.3	4.66	0.04	0.0	0.03	0.56	0.37	0.55	0.61	0.96
E3.19	6.74	10.68	9.54	6.76	0.16	0.0	0.05	0.26	0.62	1.56	1.08	0.95
E3.26	7.1	19.59	10.84	8.19	0.11	0.0	0.13	0.59	0.25	3.01	3.28	0.86
E3.34	4.26	6.05	7.52	6.19	0.34	0.0	0.02	0.62	0.32	0.69	0.5	0.98
E3.38	9.22	14.39	14.81	11.59	0.08	0.01	0.16	0.26	0.49	5.97	4.0	0.82
E3.45	8.73	17.09	13.69	10.54	0.32	0.0	0.16	0.21	0.56	4.55	3.01	0.82
E3.51	9.63	19.89	12.17	7.45	0.08	0.0	0.17	0.24	0.56	2.72	1.79	0.83
E3.66	14.99	35.56	19.68	12.75	0.41	0.0	0.42	0.09	0.46	9.45	5.49	0.56
E3.67	12.75	27.32	15.76	9.27	0.24	0.07	0.05	0.7	0.12	2.71	1.65	0.88
E3.68	7.37	13.3	9.84	6.52	0.18	0.0	0.05	0.16	0.72	1.6	1.1	0.94
E3.75	4.33	7.49	6.33	4.61	0.0	0.0	0.02	0.46	0.43	0.73	0.62	0.98
E3.77	4.58	7.45	7.01	5.31	0.27	0.0	0.04	0.3	0.58	0.86	0.79	0.96
E3.87	10.19	26.22	13.96	9.54	0.05	0.0	0.28	0.37	0.32	4.16	2.74	0.71
E4.6	5.38	13.8	7.84	5.7	0.06	0.0	0.07	0.53	0.35	0.94	1.12	0.92
E4.12	8.55	32.85	11.57	7.8	0.07	0.0	0.18	0.34	0.48	2.29	2.86	0.82
E4.19	9.25	21.17	11.54	6.89	0.32	0.13	0.04	0.66	0.1	1.46	1.18	0.82
E4.28	8.69	19.75	12.36	8.79	0.28	0.01	0.16	0.21	0.51	2.72	2.1	0.81
E4.30	6.98	14.16	9.53	6.48	0.15	0.08	0.01	0.59	0.25	1.09	0.86	0.9
E4.31	10.63	14.62	13.97	9.07	0.13	0.1	0.02	0.62	0.23	2.27	1.1	0.88
E4.38	7.22	23.08	8.86	5.14	0.48	0.08	0.08	0.38	0.34	1.08	0.92	0.82
E4.52	4.05	8.02	6.06	4.5	0.17	0.02	0.03	0.47	0.45	0.53	0.54	0.96
E4.67	8.31	20.44	11.5	7.95	0.44	0.0	0.17	0.28	0.45	2.4	1.45	0.81
E4.69	9.14	19.8	13.02	9.27	0.15	0.02	0.21	0.29	0.43	3.21	2.88	0.76
E4.75	12.47	20.91	18.2	13.26	0.67	0.14	0.04	0.69	0.02	6.11	2.34	0.79
E4.80	4.24	9.51	6.95	5.51	0.02	0.01	0.06	0.54	0.33	0.99	0.91	0.92
E4.84	15.71	26.56	20.49	13.15	0.26	0.0	0.39	0.05	0.42	12.35	6.64	0.54
E4.89	10.49	20.44	13.33	8.22	0.1	0.02	0.26	0.15	0.47	3.14	2.04	0.69
E4.101	18.01	25.27	23.07	14.42	0.43	0.16	0.09	0.57	0.13	7.55	3.05	0.74
E4.106	5.49	10.02	7.82	5.57	0.02	0.01	0.05	0.39	0.46	0.89	0.84	0.93
E4.117	6.76	13.63	10.25	7.7	0.01	0.02	0.05	0.44	0.44	1.4	1.23	0.93
E4.131	5.14	10.12	7.62	5.63	0.08	0.01	0.05	0.61	0.29	0.83	0.98	0.94
E4.134	8.35	23.77	13.54	10.66	0.53	0.0	0.23	0.38	0.33	4.54	4.71	0.75
E4.136	11.49	32.17	15.49	10.39	0.17	0.0	0.35	0.11	0.49	4.69	3.8	0.63
E4.139	6.65	15.97	9.4	6.65	0.42	0.0	0.09	0.53	0.31	1.22	1.02	0.9
E4.146	7.99	13.49	10.69	7.1	0.15	0.04	0.02	0.62	0.27	1.46	1.07	0.94
E4.150	17.69	42.87	20.26	9.87	0.15	0.0	0.5	0.01	0.39	7.79	5.92	0.44
E4.153	15.33	29.53	20.1	13.0	0.05	0.0	0.35	0.18	0.37	11.17	7.96	0.62
E4.159	7.56	16.98	11.68	8.9	0.28	0.0	0.13	0.28	0.57	2.53	2.15	0.87
E4.174	4.36	6.26	6.76	5.17	0.08	0.0	0.01	0.57	0.34	0.75	0.63	0.98
E4.198	20.33	45.21	23.74	12.25	0.27	0.05	0.64	0.0	0.29	12.78	9.61	0.29
E4.201	13.25	22.55	17.59	11.57	0.07	0.11	0.04	0.63	0.16	3.86	2.31	0.84
E4.214	27.24	66.39	31.8	16.41	0.23	0.0	0.84	0.0	0.09	45.83	28.7	0.1
E4.219	13.46	60.72	17.45	11.11	0.26	0.0	0.39	0.12	0.42	6.28	5.71	0.58
E4.229	5.76	14.49	7.95	5.49	0.36	0.03	0.04	0.56	0.32	0.83	0.82	0.92
E4.230	4.9	10.24	8.01	6.33	0.16	0.02	0.06	0.56	0.3	1.12	0.81	0.91
E4.232	9.81	13.33	14.09	10.11	0.26	0.06	0.05	0.58	0.23	2.95	1.32	0.88
E4.233	11.78	14.66	16.36	11.35	0.23	0.04	0.04	0.71	0.14	3.24	1.2	0.91

Table 21: Sampled training engines E1.6-E2.248 that are used for run-to-failure testing evaluated on the second half of the most important metrics. The mean and standard deviation of the metrics across all 99 sampled training engines are also shown

	Prognostic metric											
Engine ID	**Prec.**	**Recall**	**F1**	**PH(d)**	**PH(p)**	$\alpha - \lambda$**(d)**	$\alpha - \lambda$**(p)**	**CRA**	**WCRA**	**CRPS**	**W. CRPS**	**Conv.**
Mean	0.53	0.7	0.75	17.96	8.26	0.8	0.7	0.79	0.7	11.14	4.04	106.16
St. dev.	0.26	0.3	0.27	26.87	0.44	0.17	0.18	0.11	0.16	5.11	6.37	44.03
E1.6	0.82	0.83	0.85	9.0	9.0	0.79	0.7	0.82	0.75	9.16	2.52	77.86
E1.16	0.75	0.91	0.89	33.0	9.0	0.86	0.72	0.81	0.79	12.8	1.72	74.78
E1.19	0.68	0.72	0.8	9.0	9.0	0.78	0.64	0.81	0.79	11.1	2.55	53.9
E1.26	0.54	0.79	0.84	8.0	8.0	0.84	0.76	0.76	0.65	11.42	2.16	72.94
E1.34	0.63	0.97	0.98	166.0	9.0	0.98	0.89	0.92	0.89	5.03	0.57	81.14
E1.38	0.51	0.93	0.96	10.0	8.0	0.96	0.88	0.87	0.82	6.55	0.81	76.46
E1.45	0.2	0.17	0.28	20.0	9.0	0.57	0.36	0.75	0.7	15.86	5.73	63.59
E1.51	0.59	0.94	0.97	184.0	8.0	0.97	0.92	0.9	0.87	6.89	0.75	87.63
E1.66	0.72	0.88	0.9	9.0	8.0	0.87	0.72	0.81	0.77	12.26	1.45	69.35
E1.67	0.95	0.95	0.96	8.0	8.0	0.92	0.83	0.84	0.78	9.83	1.11	142.81
E1.68	0.7	0.91	0.95	9.0	8.0	0.94	0.91	0.89	0.86	6.82	0.6	71.2
E1.75	0.86	0.82	0.81	8.0	8.0	0.73	0.61	0.69	0.53	12.65	3.84	106.36
E1.77	0.22	0.16	0.26	18.0	8.0	0.49	0.33	0.61	0.52	20.19	7.64	55.12
E1.87	0.67	0.78	0.83	9.0	9.0	0.79	0.74	0.78	0.71	11.41	2.49	67.35
E2.6	0.29	0.56	0.72	40.0	8.0	0.84	0.67	0.81	0.71	8.78	2.09	80.35
E2.19	0.13	0.65	0.44	9.0	9.0	0.75	0.68	0.77	0.67	11.24	1.87	62.96
E2.23	0.61	0.79	0.88	9.0	8.0	0.87	0.8	0.81	0.7	8.83	2.13	100.01
E2.28	0.63	0.93	0.87	17.0	9.0	0.85	0.7	0.83	0.75	8.21	1.25	91.0
E2.30	0.3	0.26	0.41	8.0	8.0	0.61	0.45	0.65	0.51	16.59	6.79	70.85
E2.38	0.32	0.57	0.73	8.0	8.0	0.81	0.7	0.78	0.64	8.99	2.94	84.63
E2.52	0.3	0.39	0.56	39.0	8.0	0.71	0.54	0.79	0.74	14.48	4.74	74.31
E2.62	0.5	0.76	0.86	8.0	8.0	0.86	0.77	0.77	0.63	7.92	1.54	85.7
E2.67	0.02	0.01	0.03	8.0	8.0	0.4	0.24	0.6	0.51	21.18	8.41	49.07
E2.69	0.0	0.0	0.0	8.0	8.0	0.3	0.17	0.33	0.19	30.77	44.01	49.28
E2.75	0.71	0.83	0.86	8.0	8.0	0.82	0.65	0.82	0.76	12.1	2.27	101.54
E2.80	0.79	0.87	0.92	9.0	8.0	0.89	0.75	0.83	0.8	11.04	1.85	96.43
E2.84	0.79	0.97	0.96	11.0	8.0	0.93	0.88	0.85	0.81	10.11	0.98	106.95
E2.89	0.64	0.88	0.94	8.0	8.0	0.92	0.84	0.83	0.74	7.2	1.05	80.2
E2.100	0.76	0.9	0.92	9.0	9.0	0.9	0.77	0.81	0.73	9.42	1.41	91.72
E2.103	0.82	0.88	0.92	22.0	9.0	0.88	0.74	0.85	0.8	10.99	3.66	141.78
E2.106	0.71	0.94	0.96	22.0	9.0	0.96	0.81	0.84	0.8	10.92	1.45	115.35
E2.117	0.56	0.75	0.86	18.0	9.0	0.86	0.7	0.85	0.82	9.28	2.58	76.12
E2.136	0.26	0.22	0.34	8.0	8.0	0.52	0.44	0.66	0.54	16.56	7.8	65.5
E2.139	0.92	0.94	0.97	9.0	8.0	0.95	0.85	0.85	0.78	5.9	0.64	81.84
E2.146	0.51	0.7	0.83	8.0	8.0	0.83	0.74	0.8	0.67	9.95	4.87	149.7
E2.150	0.57	0.94	0.96	16.0	8.0	0.96	0.83	0.87	0.79	6.76	1.07	107.59
E2.153	0.92	0.96	0.9	28.0	8.0	0.84	0.67	0.76	0.7	16.53	2.04	118.36
E2.159	0.91	0.99	0.97	47.0	8.0	0.96	0.83	0.86	0.84	9.62	0.78	99.02
E2.188	0.05	0.03	0.05	9.0	8.0	0.39	0.29	0.62	0.57	22.26	13.73	52.59
E2.198	0.66	0.81	0.85	8.0	8.0	0.81	0.74	0.71	0.52	9.94	1.78	85.53
E2.202	0.78	0.97	0.93	12.0	9.0	0.9	0.83	0.86	0.79	5.4	0.58	86.19
E2.206	0.35	0.62	0.72	9.0	9.0	0.8	0.69	0.86	0.8	9.74	2.69	112.16
E2.214	0.18	0.1	0.17	8.0	8.0	0.44	0.3	0.66	0.63	20.66	11.14	51.12
E2.222	0.71	0.84	0.9	8.0	8.0	0.86	0.77	0.83	0.74	7.45	1.37	88.79
E2.226	0.29	0.43	0.6	8.0	8.0	0.74	0.64	0.65	0.45	12.83	3.69	76.63
E2.230	0.5	0.72	0.84	14.0	8.0	0.84	0.75	0.84	0.77	8.01	1.63	86.43
E2.232	0.88	0.99	0.96	16.0	8.0	0.93	0.82	0.87	0.85	8.85	0.78	104.53
E2.234	0.18	0.15	0.26	8.0	8.0	0.57	0.46	0.67	0.6	19.36	13.35	62.49
E2.245	0.77	0.91	0.82	9.0	8.0	0.78	0.66	0.79	0.73	13.85	2.05	103.11
E2.248	0.82	0.97	0.96	19.0	9.0	0.93	0.82	0.85	0.8	9.72	1.0	94.0

Table 22: Sampled training engines E2.249-E4.233 that are used for run-to-failure testing evaluated on the second half of the most important metrics. The mean and standard deviation of the metrics across all 99 sampled training engines are also shown

	Prognostic metric											
Engine ID	**Prec.**	**Recall**	**F1**	**PH(d)**	**PH(p)**	$\alpha - \lambda$**(d)**	$\alpha - \lambda$**(p)**	**CRA**	**WCRA**	**CRPS**	**W. CRPS**	**Conv.**
Mean	0.53	0.7	0.75	17.96	8.26	0.8	0.7	0.79	0.7	11.14	4.04	106.16
St. dev.	0.26	0.3	0.27	26.87	0.44	0.17	0.18	0.11	0.16	5.11	6.37	44.03
E2.249	0.66	0.68	0.79	8.0	8.0	0.76	0.66	0.79	0.7	11.95	5.87	99.21
E3.6	0.9	0.95	0.92	11.0	8.0	0.88	0.79	0.8	0.73	11.68	1.32	122.38
E3.16	0.6	0.94	0.97	8.0	8.0	0.97	0.93	0.88	0.8	4.48	0.8	197.56
E3.19	0.29	0.84	0.91	30.0	8.0	0.96	0.84	0.89	0.86	8.12	1.62	115.93
E3.26	0.7	0.82	0.9	8.0	8.0	0.86	0.84	0.8	0.68	7.53	4.95	131.82
E3.34	0.66	0.97	0.99	117.0	8.0	0.98	0.95	0.94	0.91	4.29	0.61	280.71
E3.38	0.35	0.62	0.76	31.0	9.0	0.83	0.74	0.86	0.82	10.65	5.19	94.14
E3.45	0.27	0.56	0.72	8.0	8.0	0.81	0.72	0.83	0.76	10.28	3.89	100.94
E3.51	0.3	0.59	0.74	9.0	8.0	0.83	0.71	0.8	0.72	11.2	2.6	85.11
E3.66	0.16	0.17	0.3	8.0	8.0	0.58	0.49	0.64	0.51	17.3	7.52	65.52
E3.67	0.86	0.94	0.92	8.0	8.0	0.88	0.76	0.73	0.58	11.74	1.77	113.71
E3.68	0.19	0.76	0.86	24.0	9.0	0.95	0.76	0.87	0.82	9.38	1.9	91.61
E3.75	0.51	0.95	0.98	22.0	8.0	0.98	0.88	0.93	0.89	5.28	0.97	140.48
E3.77	0.34	0.88	0.94	20.0	8.0	0.96	0.84	0.93	0.89	5.79	1.24	140.12
E3.87	0.54	0.57	0.73	8.0	8.0	0.71	0.62	0.74	0.62	11.61	3.98	72.76
E4.6	0.6	0.88	0.94	8.0	8.0	0.92	0.86	0.86	0.77	6.23	1.83	191.04
E4.12	0.41	0.65	0.79	8.0	8.0	0.81	0.76	0.67	0.42	9.89	4.42	159.47
E4.19	0.86	0.94	0.88	8.0	8.0	0.84	0.69	0.79	0.68	8.21	1.22	100.5
E4.28	0.29	0.57	0.72	13.0	9.0	0.84	0.65	0.8	0.71	10.99	3.45	99.12
E4.30	0.71	0.98	0.93	9.0	9.0	0.91	0.84	0.86	0.78	6.68	0.94	158.63
E4.31	0.73	0.97	0.91	9.0	9.0	0.88	0.83	0.85	0.81	10.22	1.28	116.21
E4.38	0.52	0.82	0.82	21.0	8.0	0.84	0.68	0.77	0.64	8.67	1.36	76.83
E4.52	0.51	0.94	0.96	8.0	8.0	0.95	0.92	0.92	0.87	4.76	0.83	202.09
E4.67	0.39	0.63	0.77	8.0	8.0	0.84	0.65	0.8	0.71	10.86	2.37	67.36
E4.69	0.41	0.58	0.72	10.0	8.0	0.77	0.67	0.8	0.7	11.02	4.34	123.99
E4.75	0.97	0.94	0.88	8.0	8.0	0.81	0.7	0.79	0.71	11.23	1.97	99.27
E4.80	0.62	0.9	0.94	9.0	9.0	0.93	0.86	0.9	0.84	5.22	1.35	175.78
E4.84	0.11	0.12	0.22	25.0	8.0	0.6	0.41	0.73	0.71	18.29	8.96	59.66
E4.89	0.24	0.37	0.52	10.0	9.0	0.72	0.6	0.8	0.74	13.03	3.26	67.4
E4.101	0.81	0.86	0.82	8.0	8.0	0.74	0.58	0.75	0.67	16.87	3.22	112.63
E4.106	0.46	0.88	0.93	9.0	9.0	0.94	0.82	0.9	0.85	6.76	1.46	144.27
E4.117	0.5	0.89	0.93	8.0	8.0	0.93	0.85	0.86	0.78	7.3	1.74	196.84
E4.131	0.68	0.92	0.96	11.0	9.0	0.94	0.87	0.9	0.84	5.46	1.49	218.4
E4.134	0.54	0.62	0.76	8.0	8.0	0.76	0.72	0.76	0.6	9.86	7.45	144.57
E4.136	0.18	0.24	0.38	8.0	8.0	0.65	0.51	0.68	0.49	14.01	5.91	105.09
E4.139	0.63	0.85	0.92	18.0	8.0	0.9	0.81	0.84	0.74	7.34	1.53	129.27
E4.146	0.7	0.97	0.96	9.0	9.0	0.95	0.86	0.87	0.8	7.71	1.39	164.45
E4.150	0.02	0.02	0.03	25.0	8.0	0.5	0.32	0.57	0.44	20.98	9.4	56.0
E4.153	0.32	0.34	0.51	30.0	8.0	0.66	0.45	0.7	0.63	17.87	12.24	70.05
E4.159	0.33	0.69	0.82	8.0	8.0	0.88	0.75	0.83	0.72	9.34	3.36	168.52
E4.174	0.63	0.98	0.98	77.0	9.0	0.98	0.9	0.94	0.91	5.18	0.95	206.6
E4.198	0.0	0.0	0.0	9.0	8.0	0.31	0.28	0.55	0.4	22.87	13.42	72.48
E4.201	0.8	0.94	0.89	8.0	8.0	0.85	0.74	0.77	0.65	12.06	2.41	209.57
E4.214	0.0	0.0	0.0	8.0	8.0	0.12	0.06	0.34	0.23	30.45	41.74	43.96
E4.219	0.22	0.23	0.38	8.0	8.0	0.61	0.51	0.39	0.05	16.07	8.85	95.97
E4.229	0.64	0.93	0.94	13.0	8.0	0.93	0.88	0.86	0.76	6.05	1.09	150.81
E4.230	0.65	0.9	0.93	9.0	8.0	0.92	0.87	0.9	0.84	5.85	1.24	130.98
E4.232	0.72	0.92	0.91	13.0	8.0	0.89	0.78	0.87	0.83	9.72	1.57	128.2
E4.233	0.84	0.95	0.95	12.0	8.0	0.92	0.84	0.85	0.82	10.97	1.29	127.83

H Appendix 8 - Flight schedule visualizations

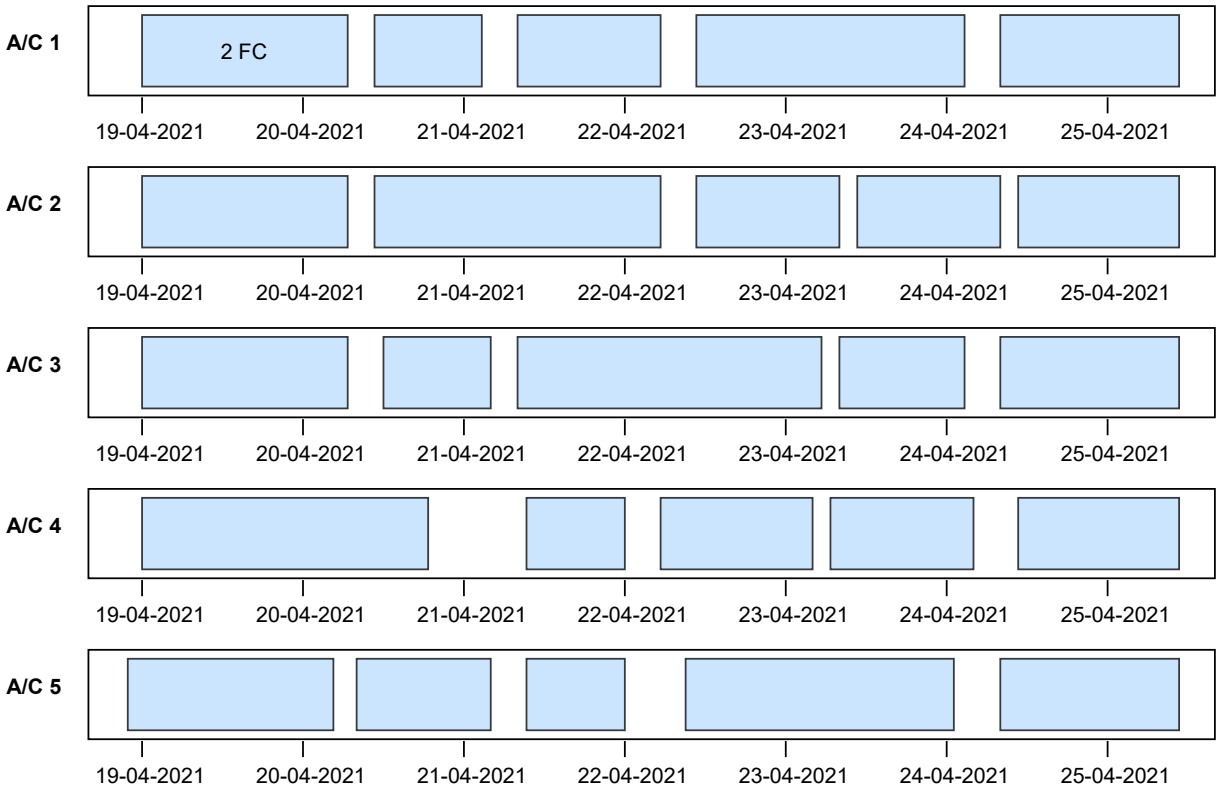


Figure 34: Full flight schedule in which each aircraft has 10 flight cycles per week

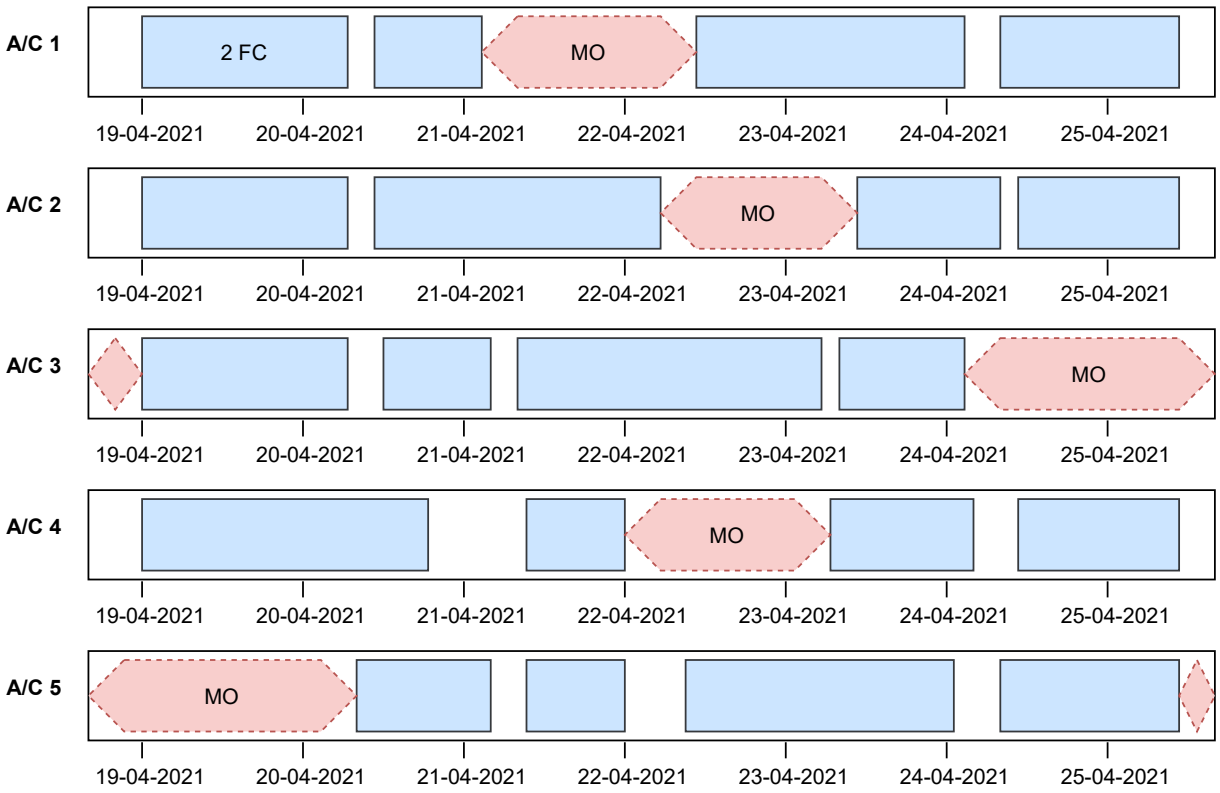


Figure 35: Flight schedule with weekly gaps in which each aircraft has 8 flight cycles per week

I Appendix 9 - Policy sensitivity plots for all strategies

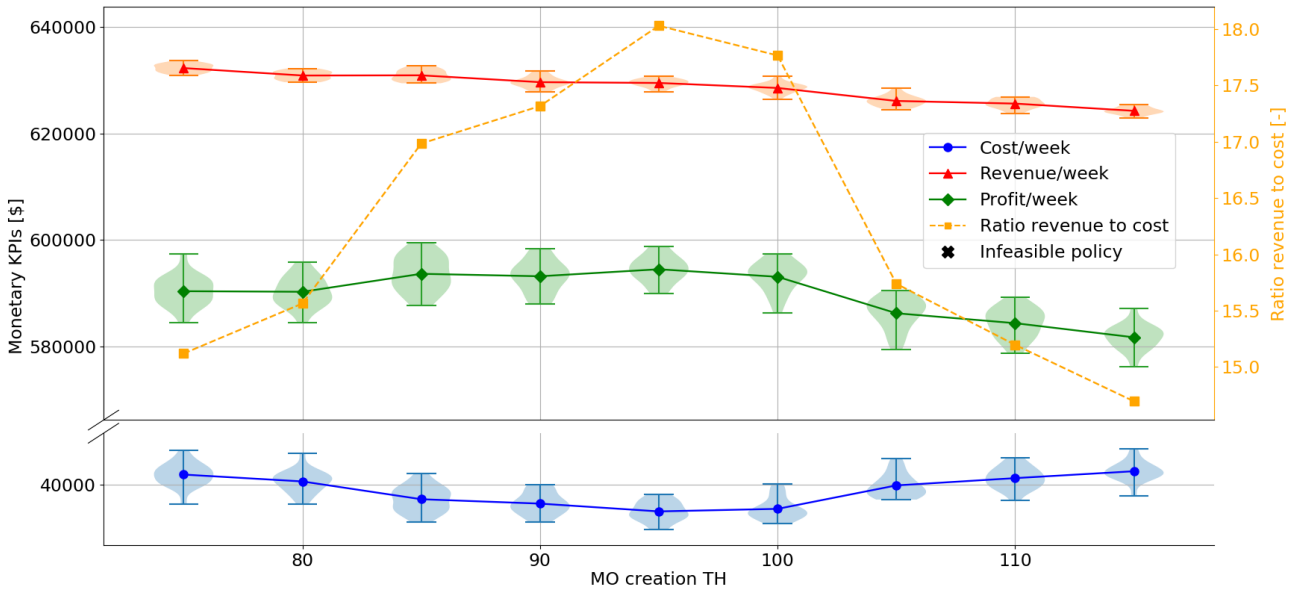


Figure 36: Sensitivity analysis on the MO creation TH parameter for the ADDP strategy

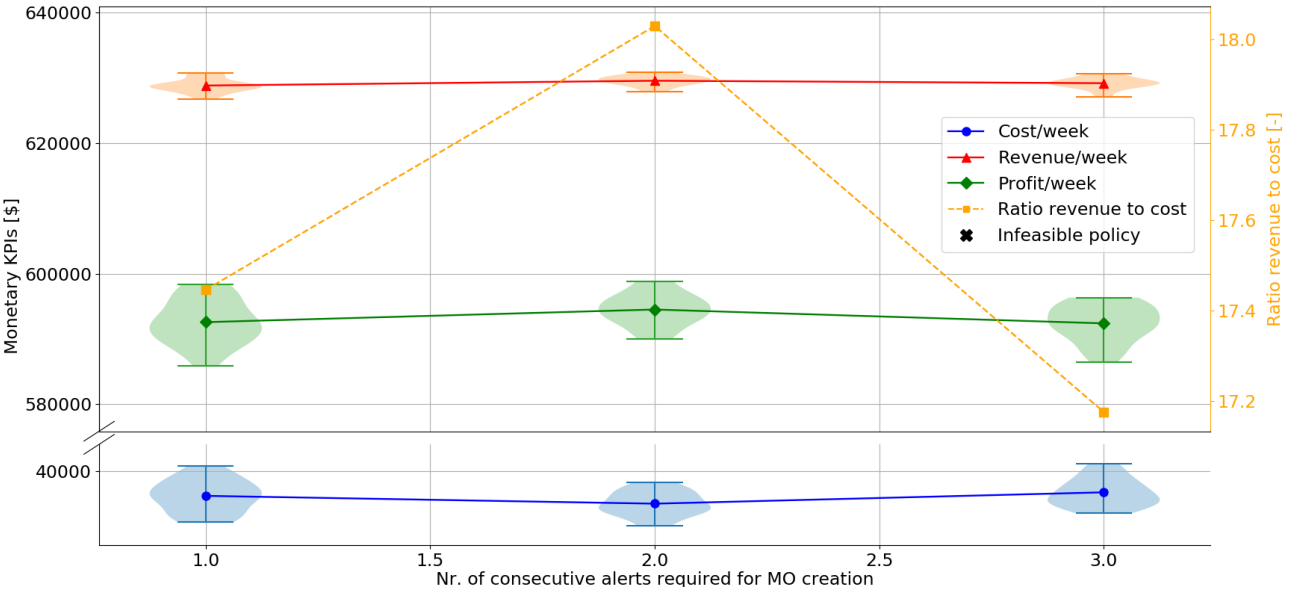


Figure 37: Sensitivity analysis on the number of consecutive alerts required for MO creation parameter for the ADDP strategy

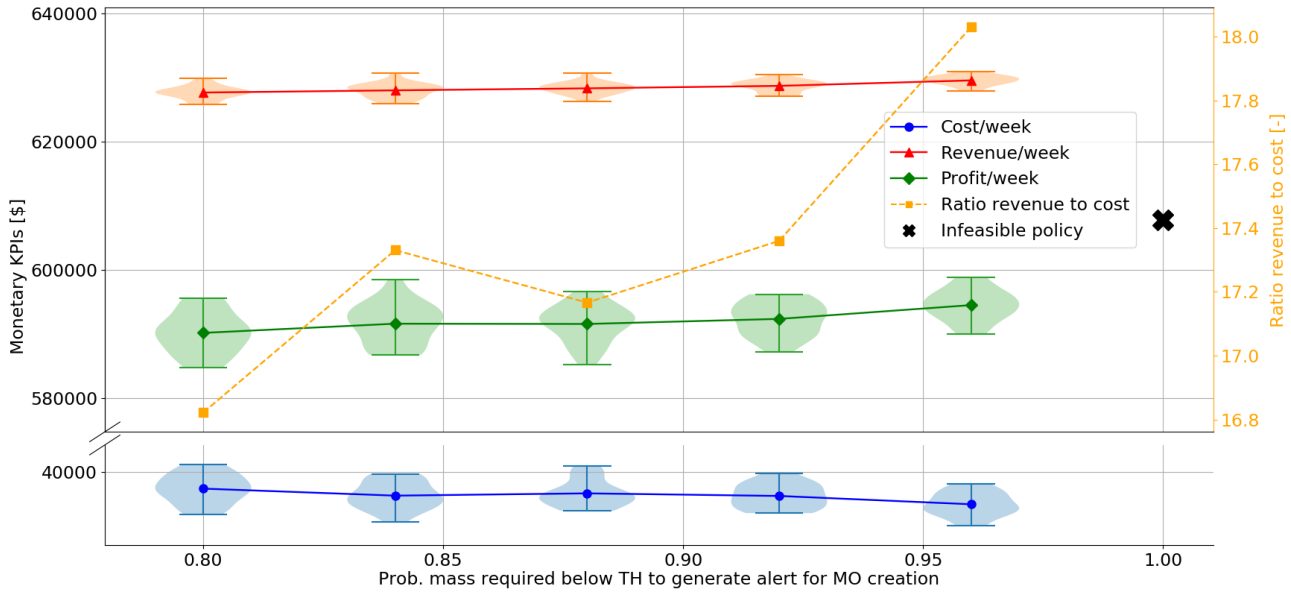


Figure 38: Sensitivity analysis on the probability mass required below TH parameter for the ADDP strategy

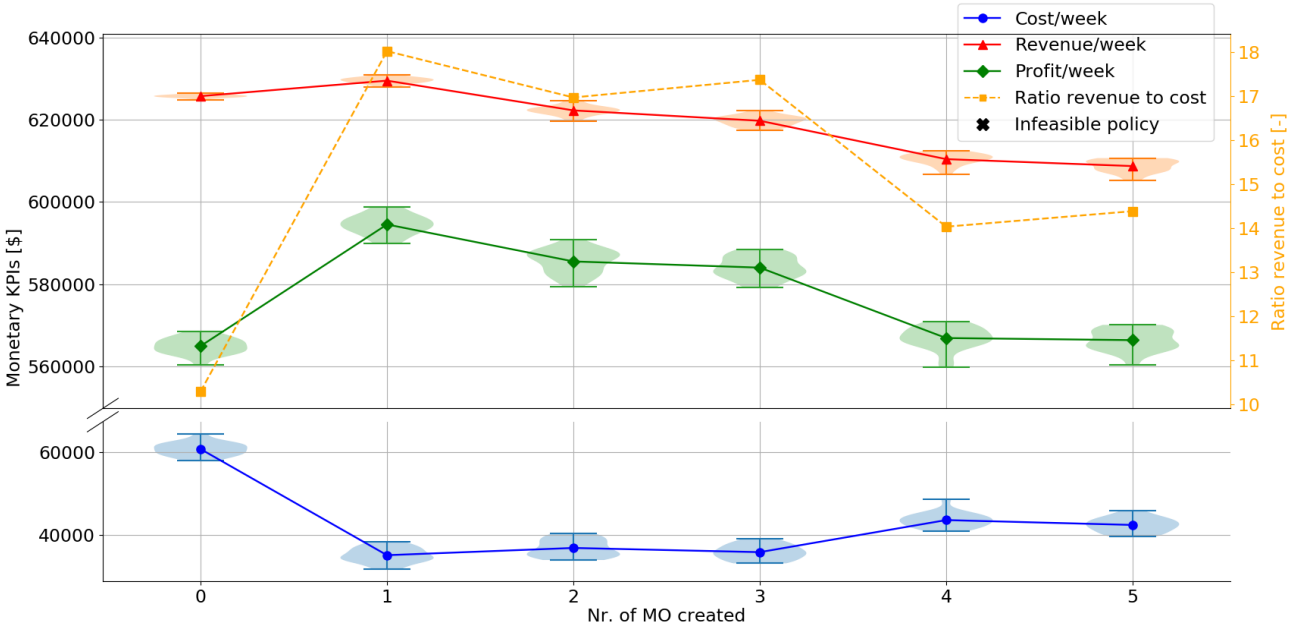


Figure 39: Sensitivity analysis on the number of MO created parameter for the ADDP strategy

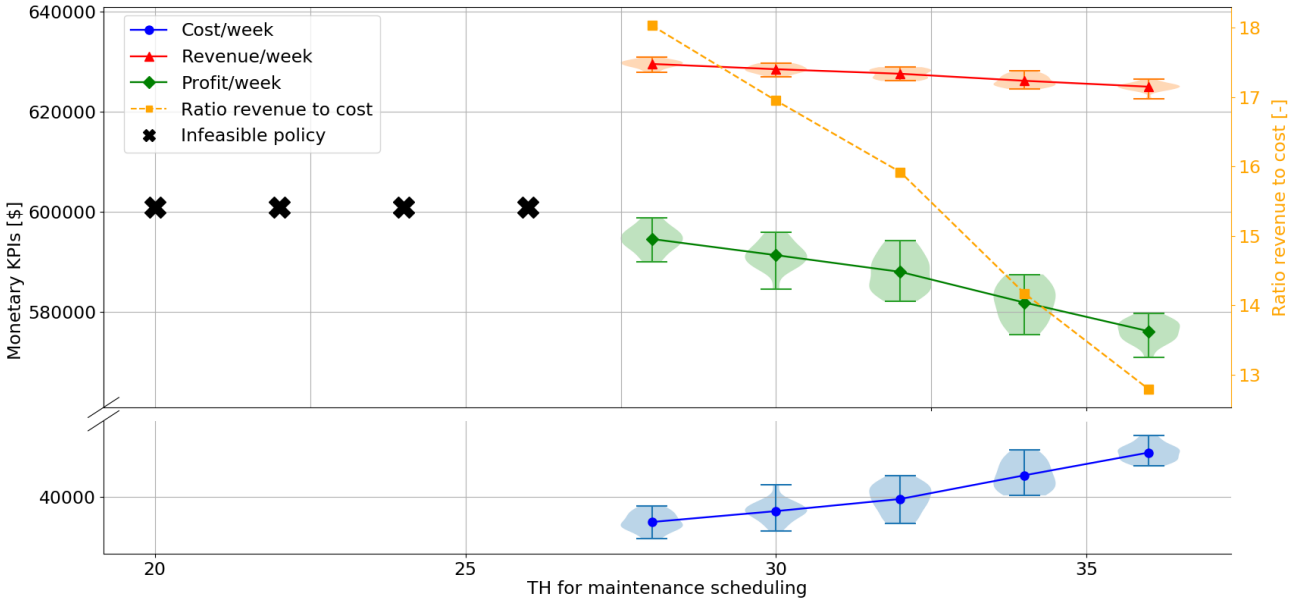


Figure 40: Sensitivity analysis on the maintenance scheduling TH parameter for the ADDP strategy

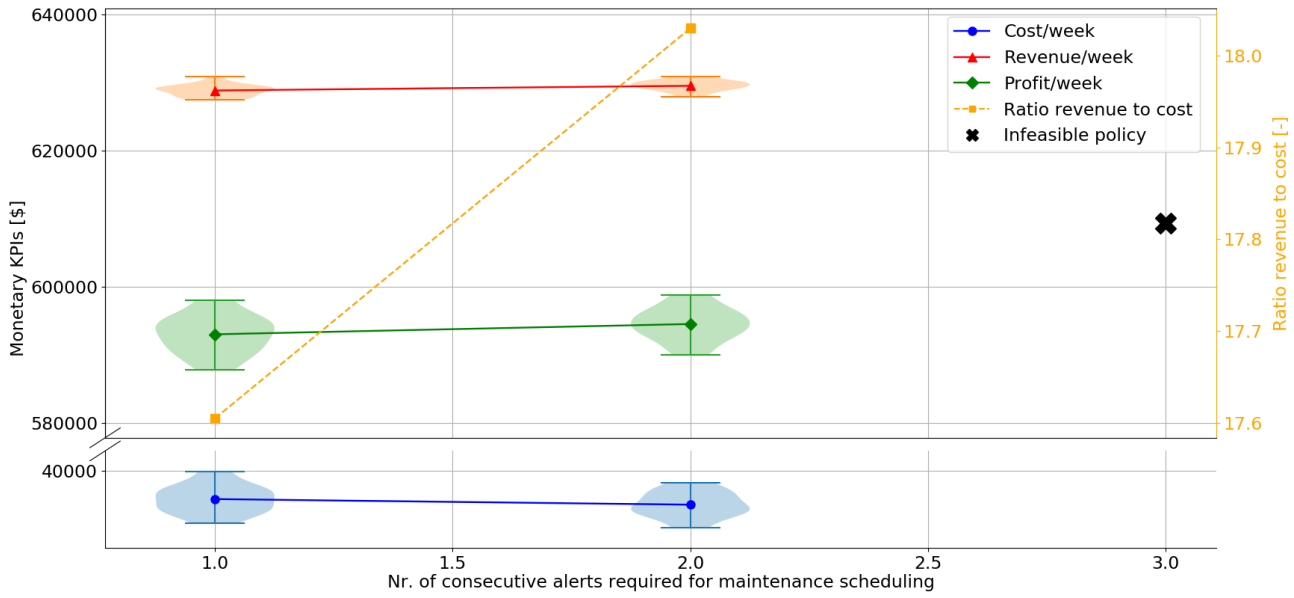


Figure 41: Sensitivity analysis on the number of consecutive alerts required for maintenance scheduling parameter for the ADDP strategy

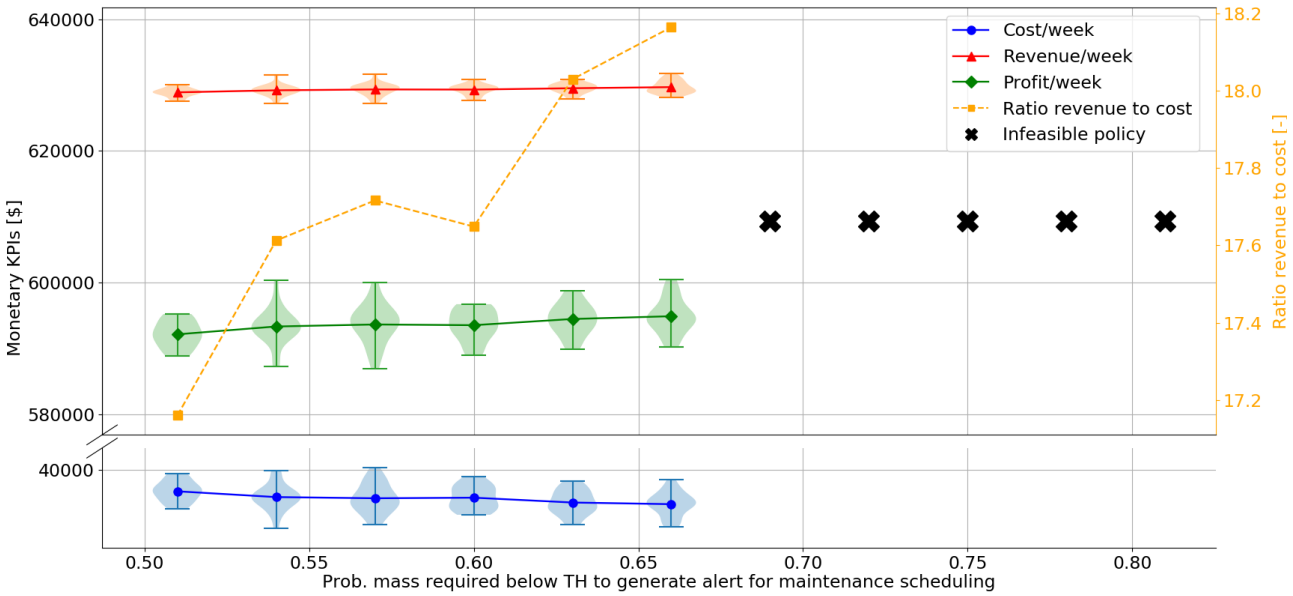


Figure 42: Sensitivity analysis on the probability mass required below TH for maintenance scheduling parameter for the ADDP strategy

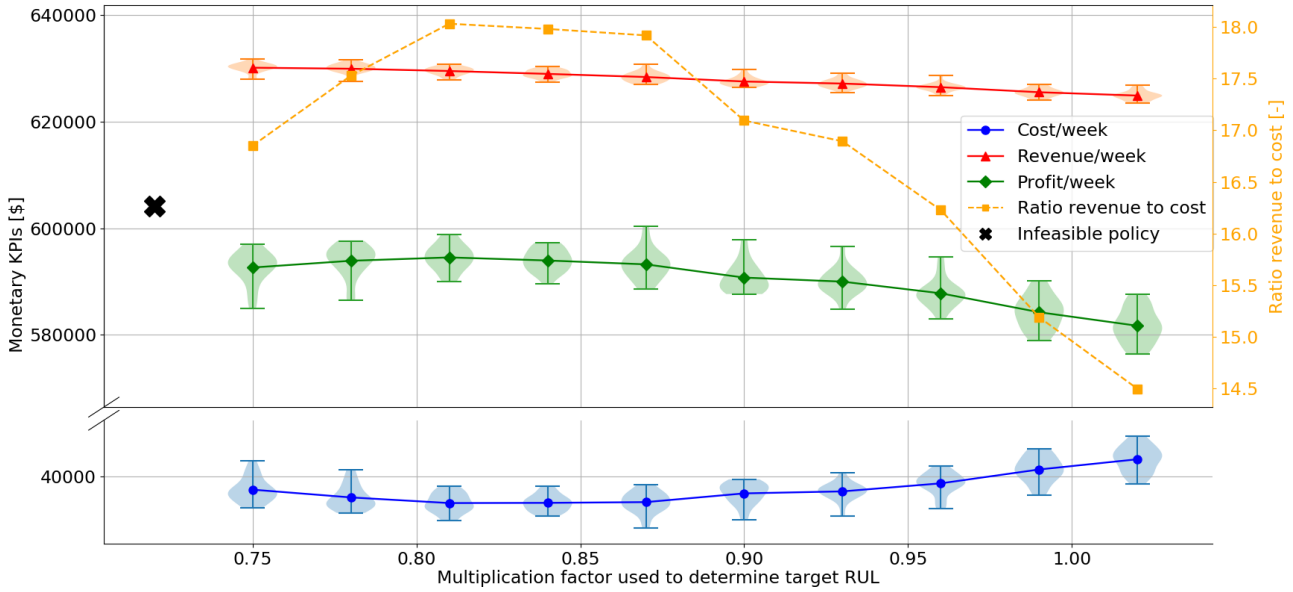


Figure 43: Sensitivity analysis on the multiplication factor used to determine target RUL parameter for the ADDP strategy

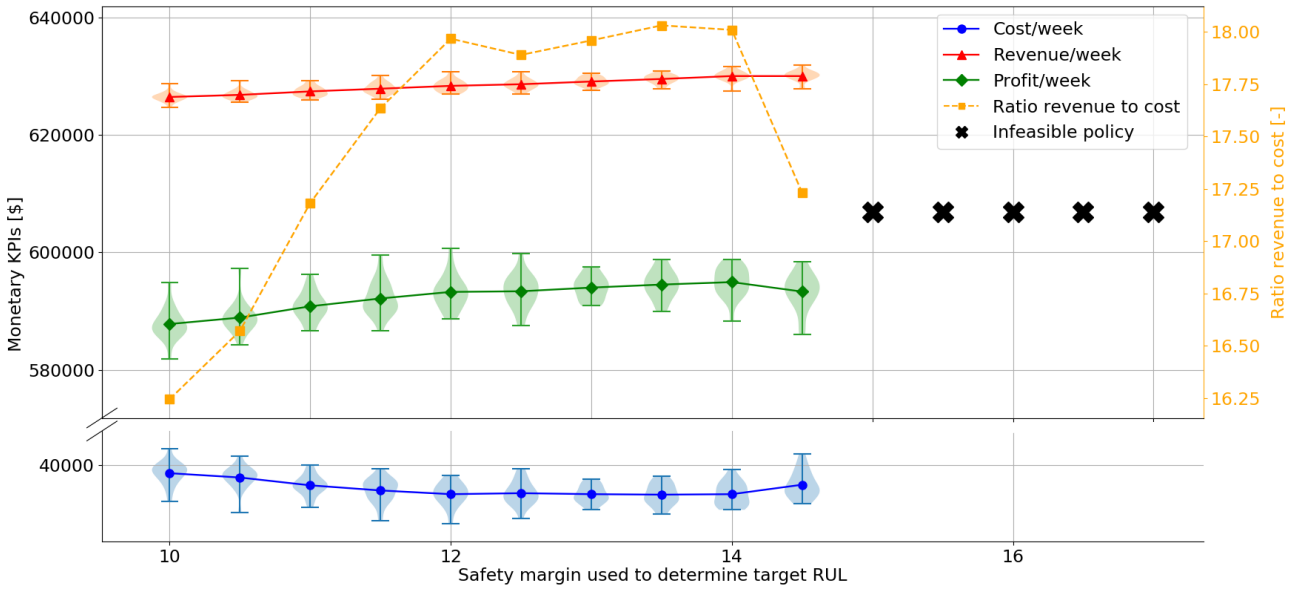


Figure 44: Sensitivity analysis on the safety margin used to determine target RUL parameter for the ADDP strategy

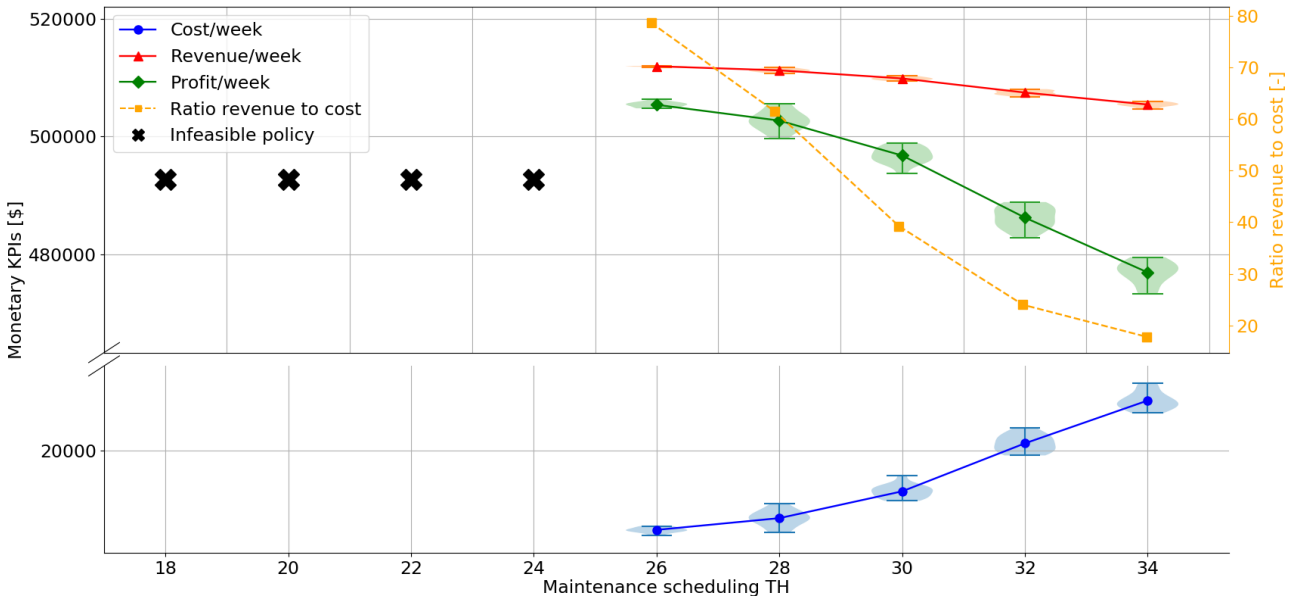


Figure 45: Sensitivity analysis on the maintenance scheduling TH parameter for the GDDP strategy

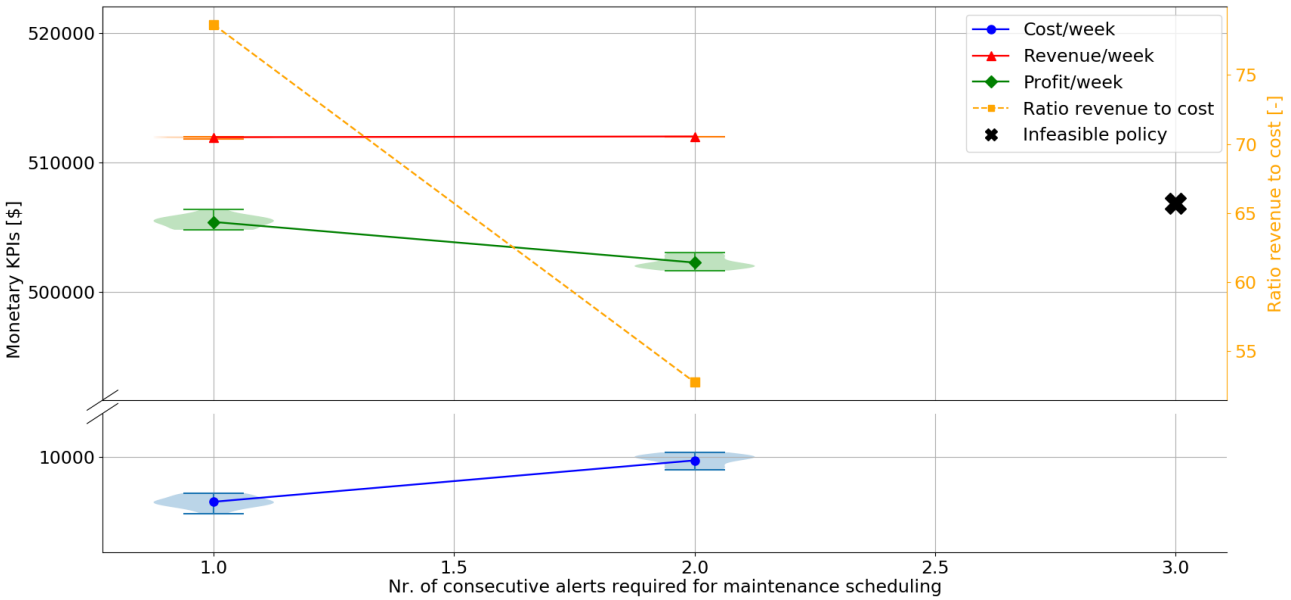


Figure 46: Sensitivity analysis on the number of consecutive alerts required below TH parameter for the GDDP strategy

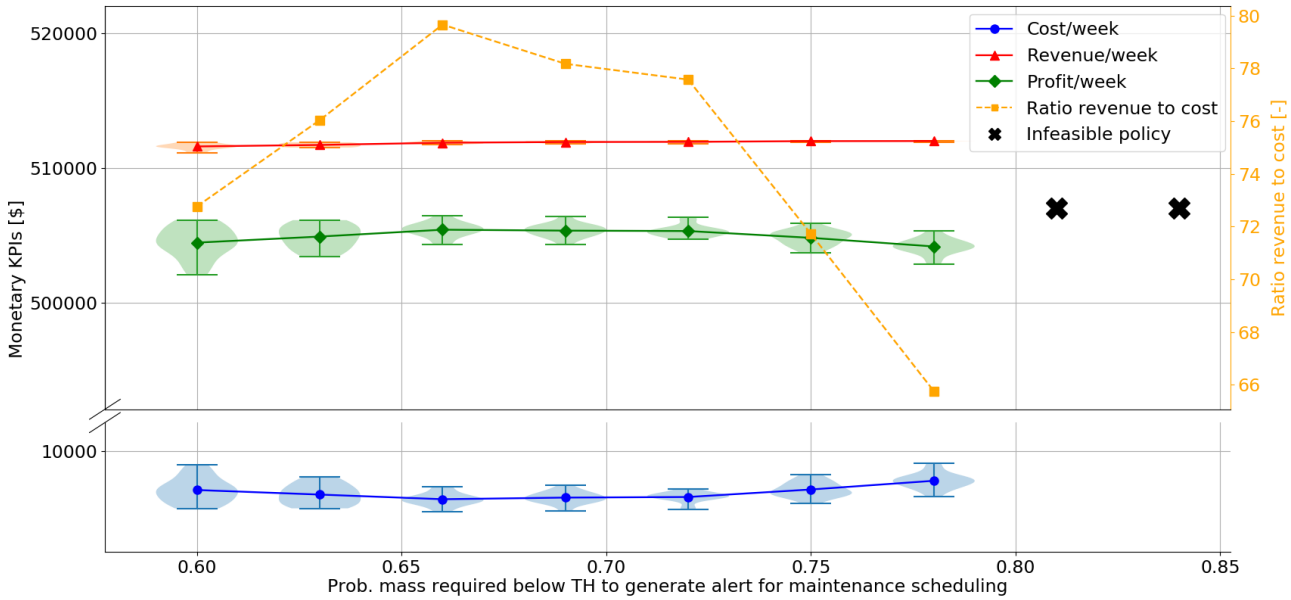


Figure 47: Sensitivity analysis on the probability mass required below TH parameter for the GDDP strategy

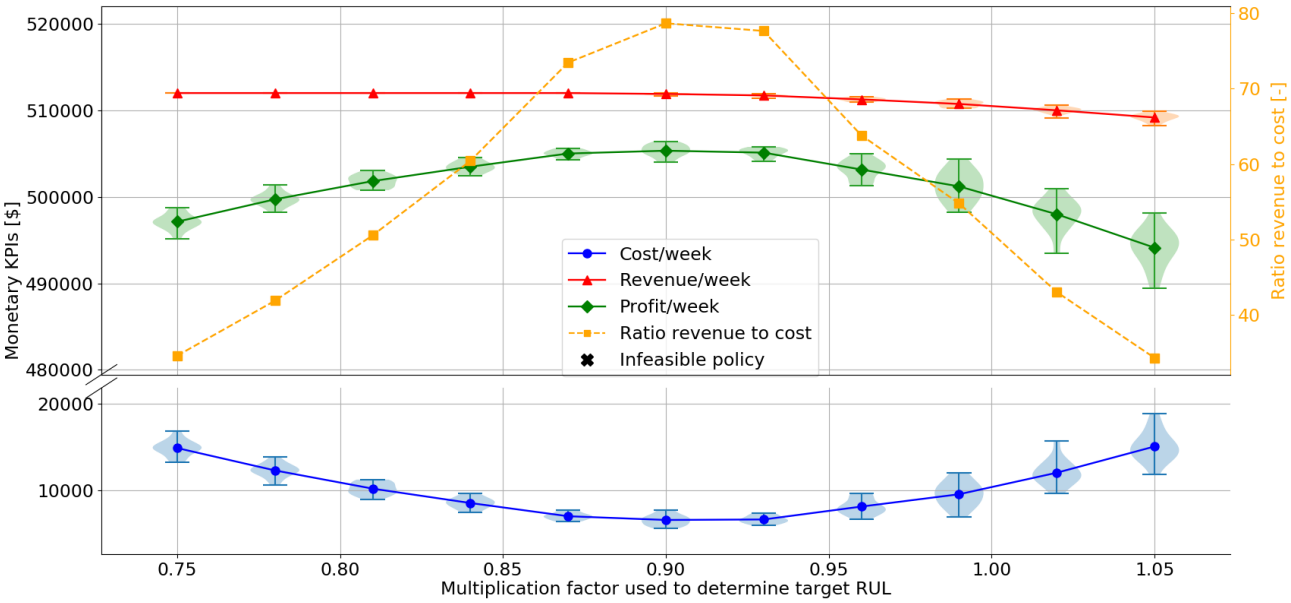


Figure 48: Sensitivity analysis on the multiplication factor used to determine target RUL parameter for the GDDP strategy

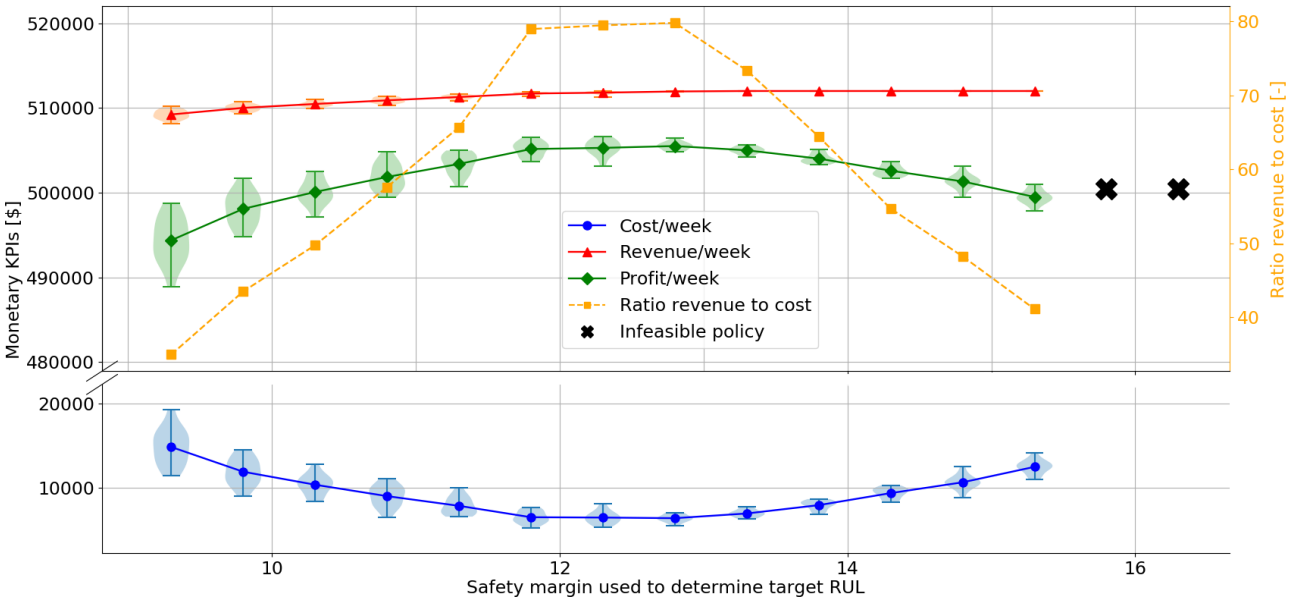


Figure 49: Sensitivity analysis on the safety margin used to determine target RUL parameter for the GDDP strategy

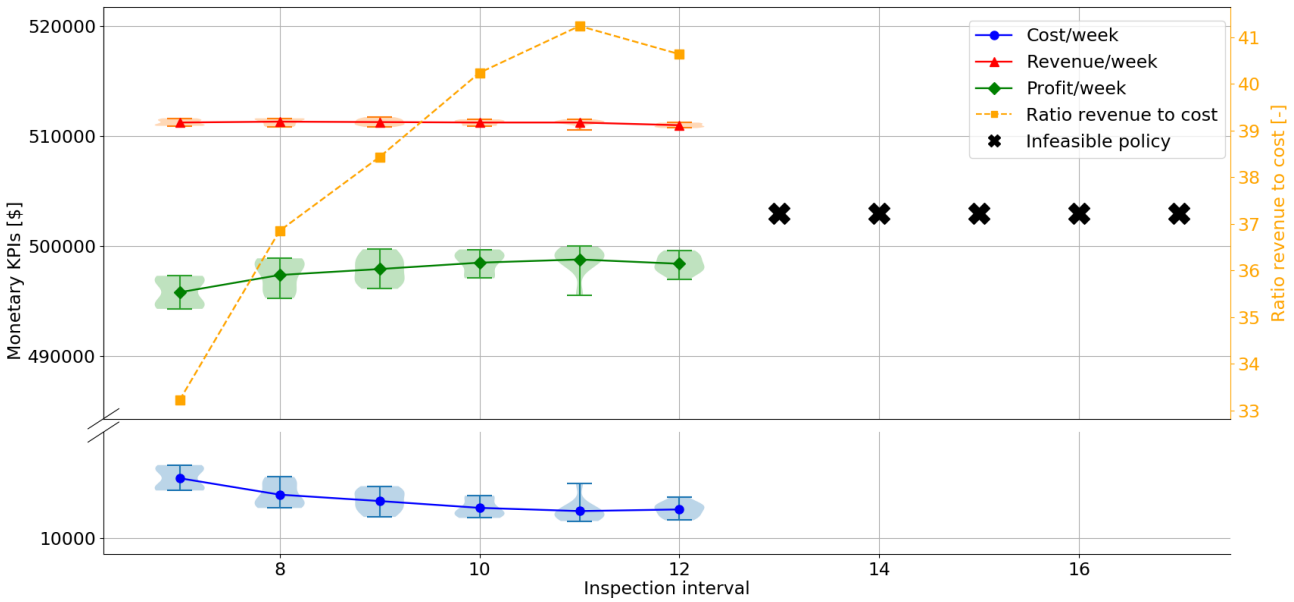


Figure 50: Sensitivity analysis on the inspection interval parameter for the GIBP strategy

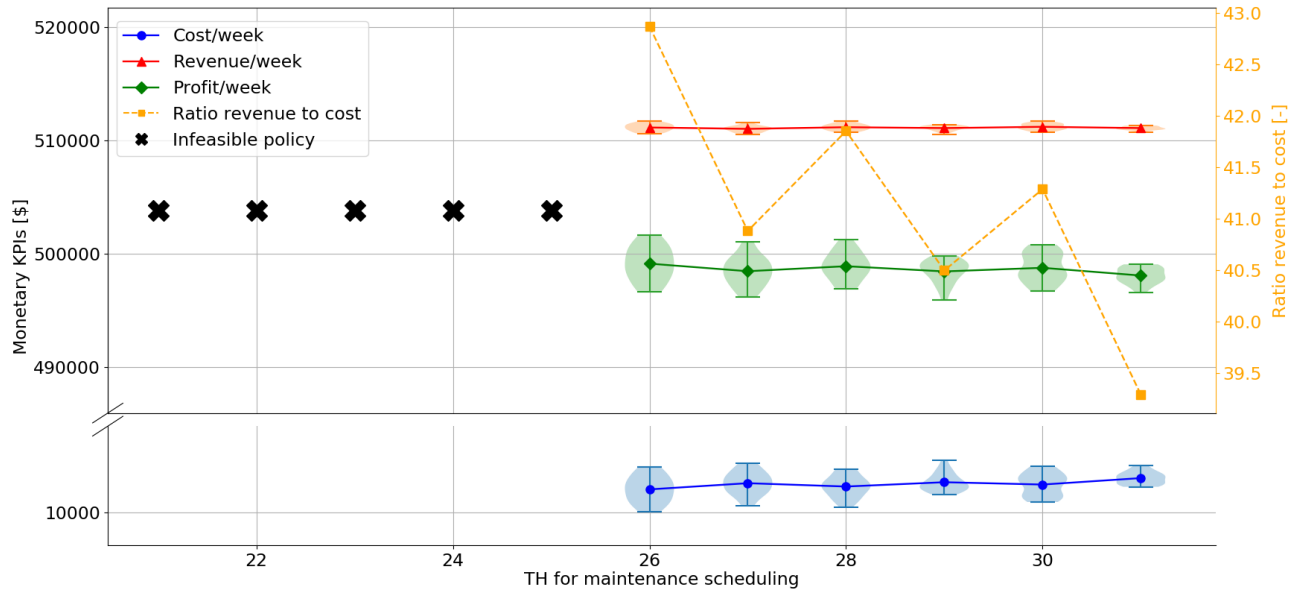


Figure 51: Sensitivity analysis on the TH for maintenance scheduling parameter for the GIBP strategy

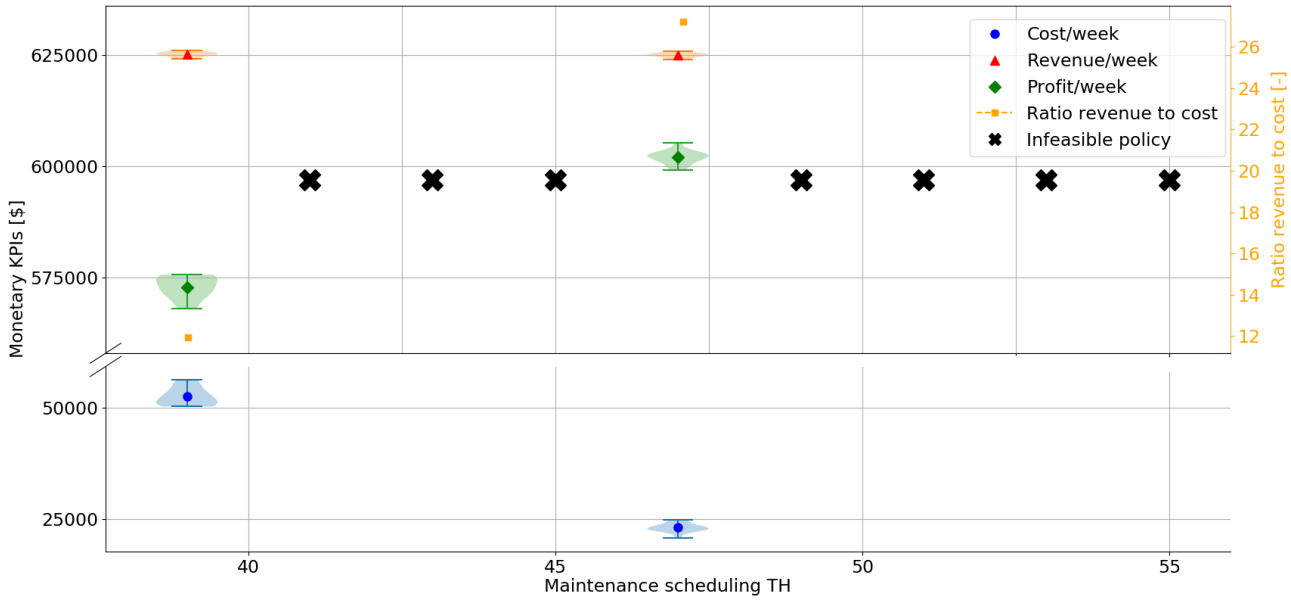


Figure 52: Sensitivity analysis on the maintenance scheduling TH parameter for the NGDDP strategy

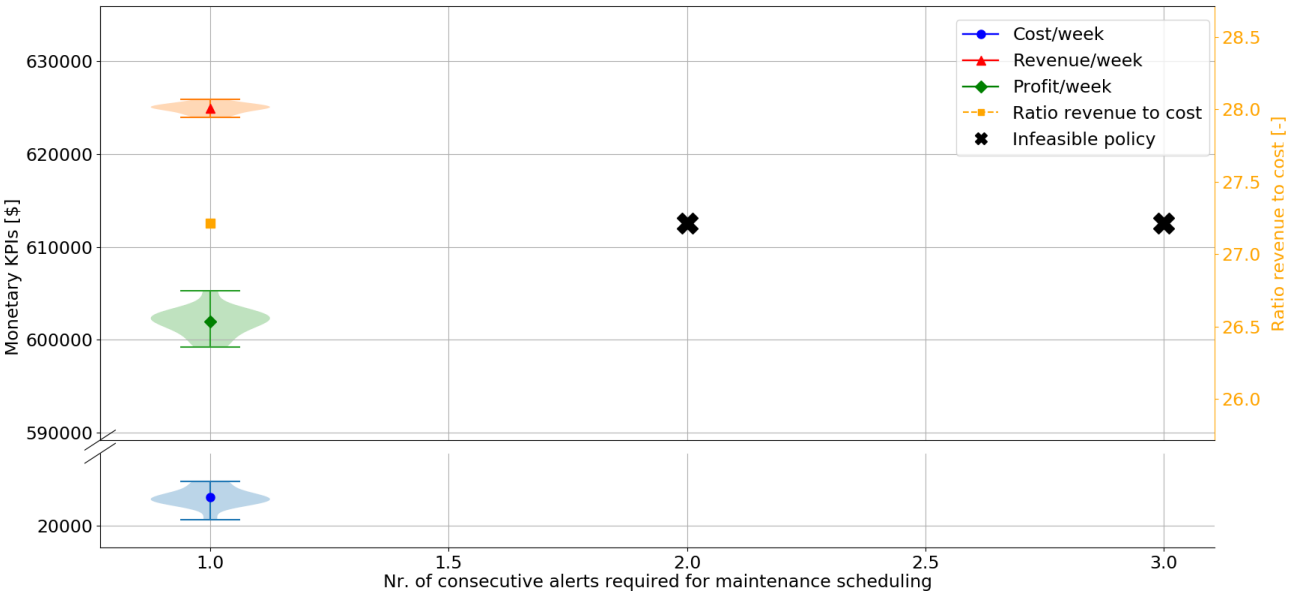


Figure 53: Sensitivity analysis on the number of consecutive alerts required below TH parameter for the NGDDP strategy

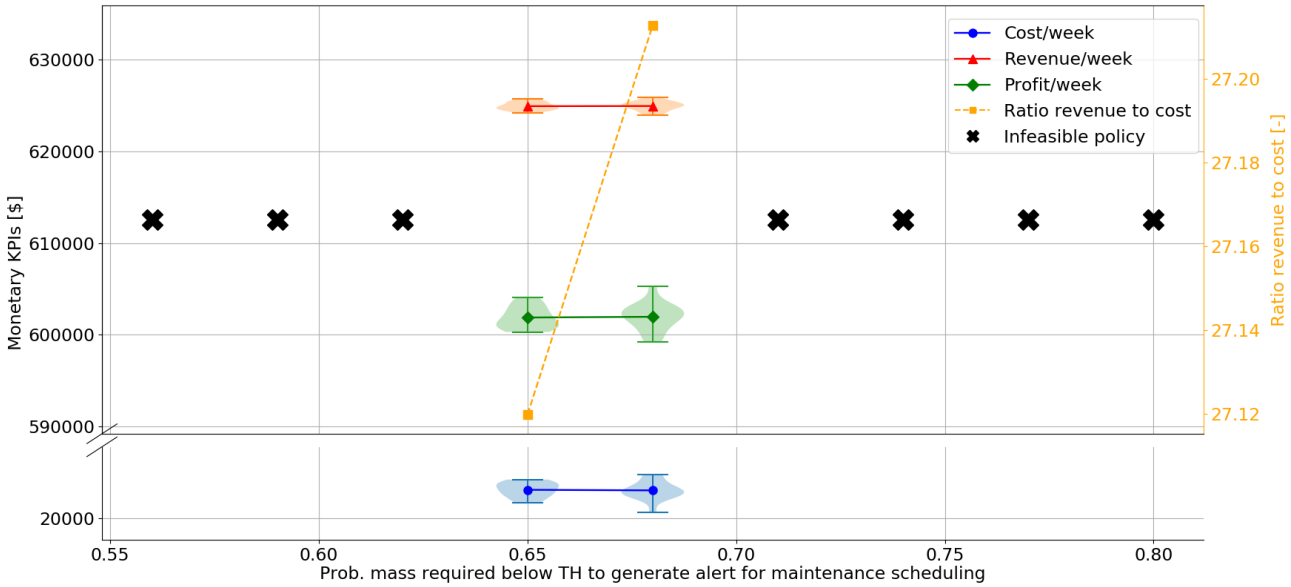


Figure 54: Sensitivity analysis on the probability mass required below TH parameter for the NGDDP strategy

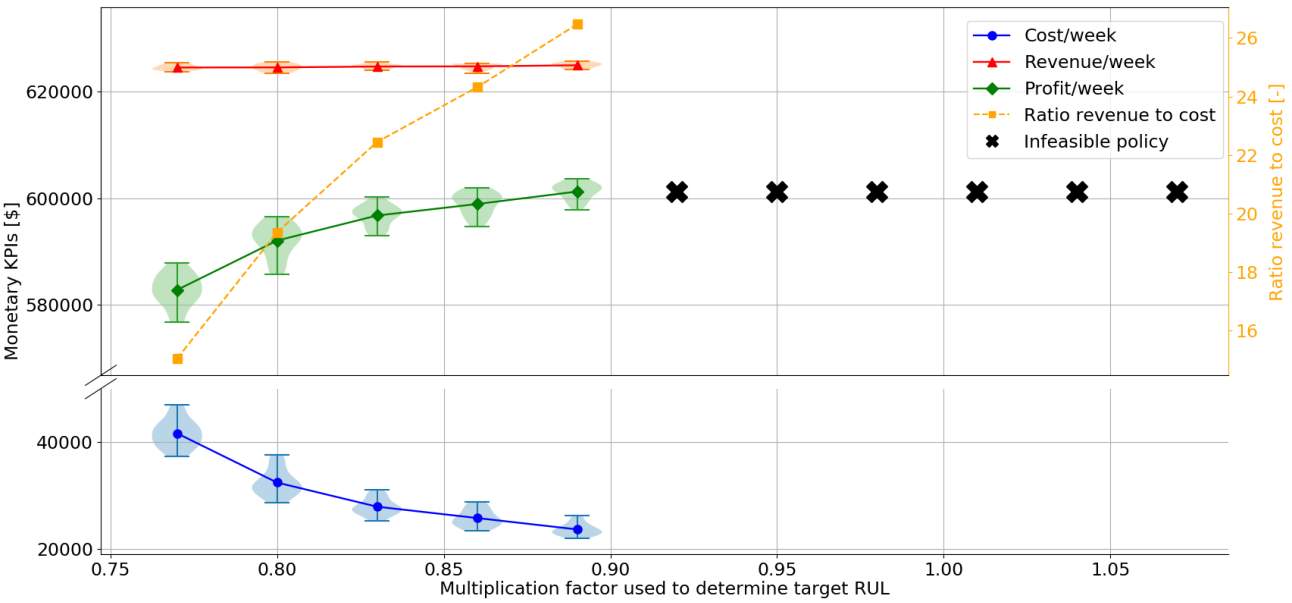


Figure 55: Sensitivity analysis on the multiplication factor used to determine target RUL parameter for the NGDDP strategy

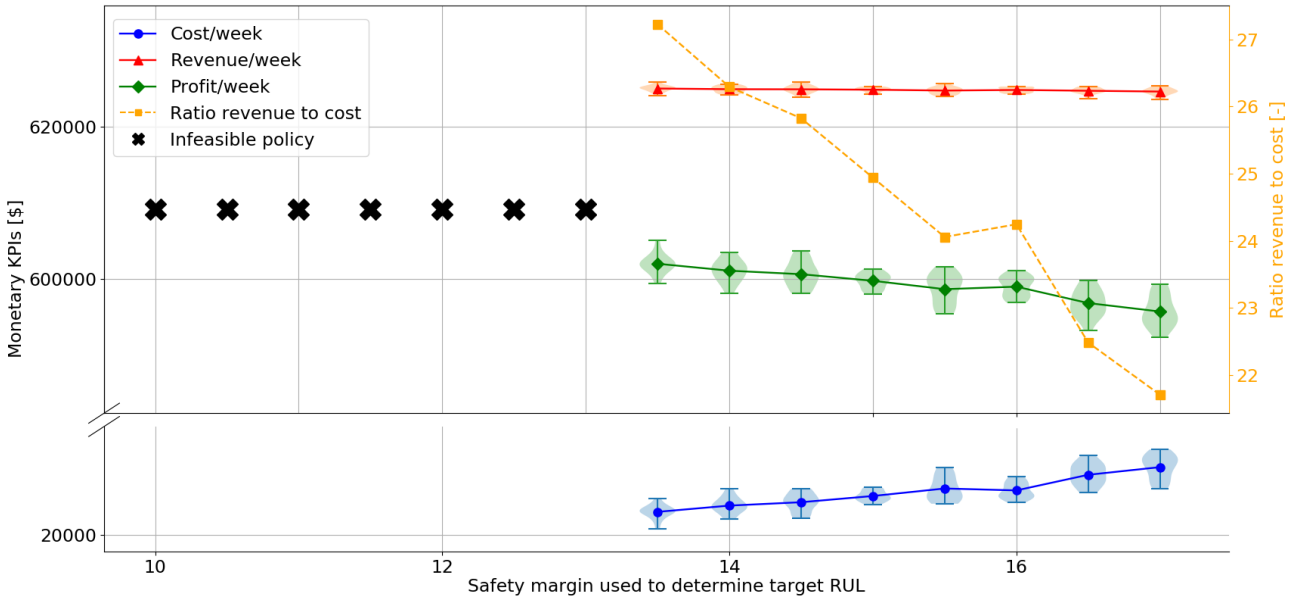


Figure 56: Sensitivity analysis on the safety margin used to determine target RUL parameter for the NGDDP strategy

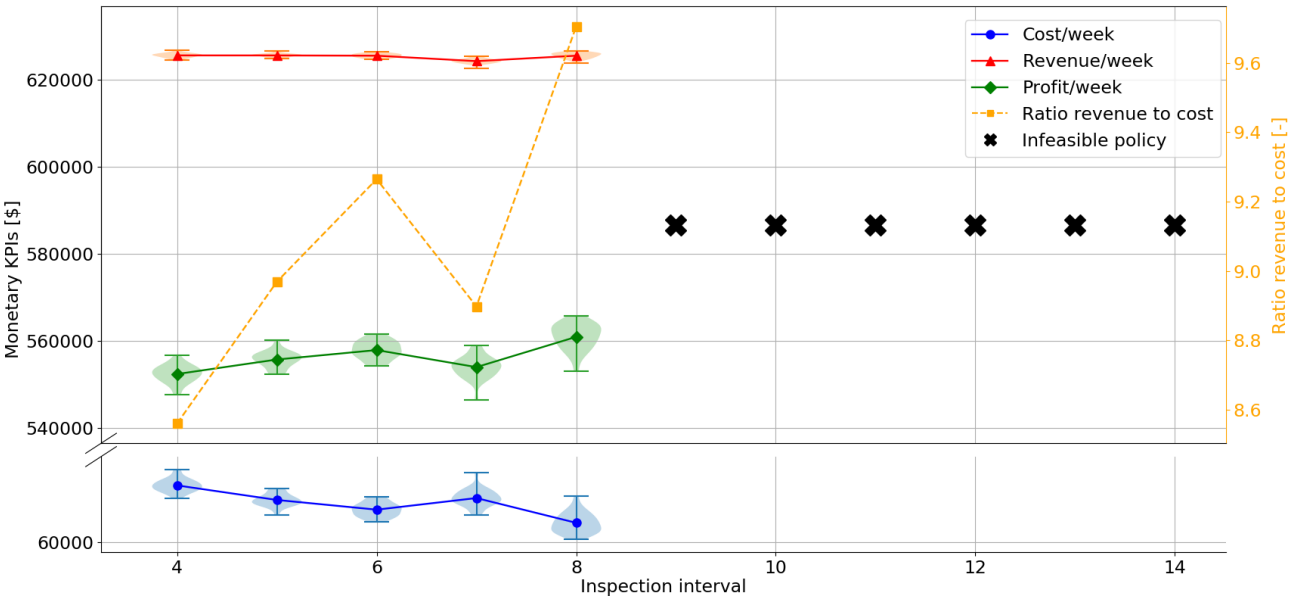


Figure 57: Sensitivity analysis on the inspection interval parameter for the NGIBP strategy

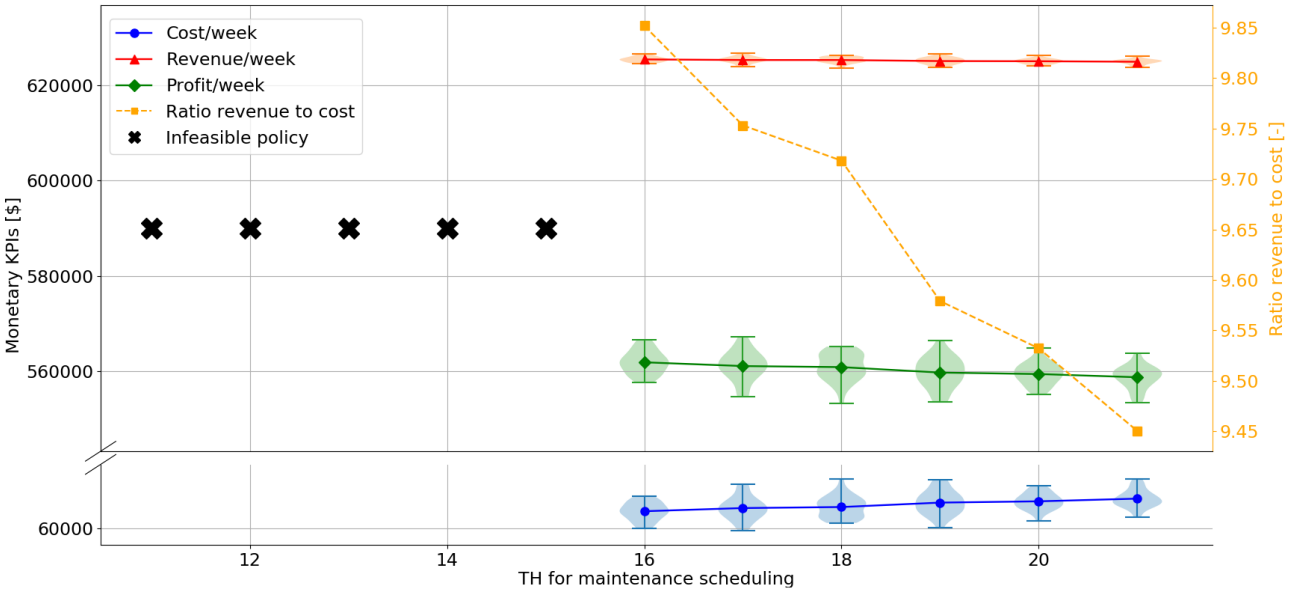


Figure 58: Sensitivity analysis on the TH for maintenance scheduling parameter for the NGIBP strategy

II

Literature Study
previously graded under AE4020

1 Literature study

1.1. Introduction

Repair and maintenance costs are among the top expenses of big aviation companies. According to the annual reports of the Royal Dutch Airlines KLM, the aircraft maintenance costs were around €882 million in 2019, equivalent to ~ 14.5% of their total yearly expenses [1]. Furthermore, not being able to fly an aircraft is also very costly. DHL has estimated that if an aircraft has to remain on ground due to technical issues, this can cost an airline up to €925.000 per day [2]. This shows the importance of being able to monitor the health of the aircraft and its components, as well as predicting when specific parts will fail. Over the last few decades, Prognostics and Health Management (PHM) has received increasing attention both from a practical and scientific perspective. It is considered as one of the key solutions to improve the reliability, safety, maintainability and economic affordability for many industrial assets, including aircraft [3]. PHM combines real-time and historical information of the system to improve the decision-making in terms of maintenance operations. It is considered to be an engineering disciple that has as a goal to minimize maintenance costs while ensuring adequate levels of safety and operationally. Is does so by assessing the health state of a system using available information from for example sensors, and tries to make predictions on the remaining useful life (RUL) accordingly. Accurate predictions for a components RUL can ensure that an aircraft does not undergo unnecessary maintenance when its components are still in healthy state, and can prevent high down times which might occur if a component were to fail unexpectedly [2].

Failure prognostics, including RUL estimation, is one of the core concepts of PHM. Predicting a components RUL refers to estimating the future condition of this component based on the current condition. Maintenance for this component or aircraft can then be scheduled accordingly [4]. The RUL of a component is usually modeled as a random variable that is to be computed using available information, such as sensor degradation measurements of similar components. Based on degradation trends in the sensor signals of historical data, the RUL can be predicted using the current state of a component. [5] describes the 4 steps of implementation of PHM for RUL prediction as follows: (i) define critical failure components, (ii) select appropriate sensors for degradation measurements, (iii) apply data analysis for feature evaluation, and (iv) prognostic methodology and tool evaluation metrics. RUL prediction comes down to using historical information in combination with current component state to compute how many more time cycles the component can safely be used. Over the past years, much research has been done on RUL prediction for aircraft components. The availability of a publicly available turbofan engine sensor measurements data set [6] has lead to numerous scientific attempts to find the best possible RUL estimation. A variety of those attempts will be discussed in more detail in this report, with a focus on literature that uses machine learning techniques. Machine learning has seen a huge increase in application in a variety of research areas over the past years and shows tremendous potential when it comes to making accurate predictions on large data sets. For this reason, machine learning methods will be the core focus of this study.

In order for prognostic algorithms that predict RUL to be viable in practice, metrics are developed that assess the performance of such algorithms [7]. Those metrics can be used to directly compare several algorithms against each other to see which one performs best, but can also be used to see how well an algorithm performs compared to the true RUL values. For aviation company decision makers, it is of vital importance

to have a standardized framework that consists of a set of indicative and informative metrics that can be used to check whether or not a certain prognostic algorithm can be used in practice. Current prognostic algorithm performance metrics suffice for comparing algorithms with each other, but fail to deliver inclusive outputs on algorithm reliability and safety for implementation in the real world. Furthermore, almost all current RUL prediction methods only output deterministic RUL estimates [8–12]. Company decision-makers cannot make reliable choices if no measure of algorithm uncertainty is included in the model output. For this reason, including a degree of confidence in the model RUL predictions is of great value for algorithm implementation in practice.

This research will consist of two main parts: (i) developing a machine learning RUL prediction algorithm, and (ii) developing a framework that can be used to analyze the performance of prognostic algorithms and assess readiness for use in practice. These accompanying research questions for both parts are stated below including explanation:

How can machine learning be used on sensor measurements of an aircraft turbofan engine for remaining useful life prediction?

The novelty of this study lies not so much within this first part. Much research has already been done on RUL prediction using machine learning techniques. This first part primarily focuses on continuing on previous work, by taking the aspects of other papers that worked best. Authors in [8] and [9] found that Convolutional Neural Networks (CNN) hold great potential for RUL estimation for turbofan engines. Despite this, no other RUL estimation attempts using CNN can be found in literature. The main focus of this first part will therefore be on developing a CNN for RUL prediction by optimizing the architecture used in [9] and by including physics based degradation information as input to the network by pre-computing component health indicators (HI). This method of combining a CNN with HI information has not been seen before in literature. The goal of this part is to achieve superior RUL prediction results on the commonly used C-MAPSS data set compared to recent literature [6] [13]. Also, uncertainty estimation will be included in the model output. This report discusses both scientific literature that attempts turbofan RUL prediction, and literature that covers neural network uncertainty estimation.

How can the performance and readiness of an aircraft component prognostic algorithm be assessed such that the algorithm can be used in real-world applications?

The second part of this research focuses on developing a framework that can directly be used by aviation companies to see if a prognostic algorithm fits their requirements and if it can be used in their daily operations. As stated, many metrics that quantify algorithm performance currently exist [7], but no standardized approach on algorithm evaluation can yet be found in academic literature. Such an approach is of great value for algorithm implementation in practice, and therefore this research attempts to develop such a novel method. This method might consist of a great number of both current and novel metrics that test the performance of an algorithm. If this algorithm then achieves satisfactory threshold levels on all metrics, this algorithm can be used for airline maintenance planning. Scientific literature that previously looked into this topic is discussed in this report, including all currently used performance metrics.

This report is set up as follows: first, section 1.2 describes methods of data-driven health indicator extraction, as well as general methods of the remaining useful life prediction of components. Next, section 1.3 describes a commonly used turbofan engine degradation data set and a variety of machine learning techniques used on this data set for remaining useful life prediction. Then, section 1.5 describes methods of including uncertainty estimates in neural network outputs. Currently used metrics that asses the performance of prognostic algorithms and algorithm implementation in practice are discussed in section 1.6. In section 1.7, the research questions and method are discussed. Lastly, a conclusion is presented in section 1.8.

1.2. General Health Monitoring and RUL Prediction Methods

Over the past years, much research has been done in the field of health monitoring and RUL prediction. This chapter aims to shed light on the most important findings of the scientific literature that was published on those topics. First, the area of health monitoring of an aircraft component will be discussed. In this report, health monitoring is solely defined as being able to use the components sensor data to extract a health indicator that reveals information on the degradation state of the component. After this, scientific papers that

aim to predict the RUL of a component or system are discussed. These papers are subdivided into three categories: papers that predict RUL using a data-driven approach, papers that predict RUL using a model based approach, and papers that predict RUL using a hybrid approach. Several different RUL prediction attempts in literature will be presented to give a holistic overview of the methods that have recently been used.

1.2.1. Data driven health indicator extraction methods

Measurements on the degradation state of a component or system are valuable as they provide information about the health state of a component. Often it is not possible to measure the degradation level from a component or system directly. However, multiple other system features can usually be measured directly using sensors. For an aircraft turbofan engine, some examples of those sensor measurements are: the temperature at fan inlet, the pressure at fan inlet, the physical fan speed, the bypass ratio etc. Please refer to subsection 1.3.1 for a full description of the turbofan engine and the corresponding sensor measurements data set. The goal of health monitoring is then to use this available sensor data to construct a single variable that indicates the degradation state of the system, the health indicator. The following paragraphs discuss several attempts to fuse available sensor data into a useful health indicator.

In [14], the authors predict a health indicator (HI) for the aircraft air conditioning system (ACS) using available sensor data. First, they provide a detailed description of the ACS, including a list of all built-in sensors. They then plot the values of the most important sensors against the number of fight cycles, to see if any clear visual degradation trends are present. It is concluded at this point that the sensor data is affected by several factors, such as the operating mode, environmental conditions and the system health state. Advanced analytic methods are required to obtain useful prognostic information from the raw sensor data. For computation of the HI, the non-parametric modeling technique Multivariate State Estimation Technique (MSET) is used. This method stores past data samples in a matrix, and when a new state estimation is required, those parameter's values are computed using the weighted averages of the stored historical data [15].

The state of the system is described by M variables (sensors), collected in a $M \times 1$ vector. The state of the system at time t_i is described as:

$$\mathbf{X}(t_i) = \begin{bmatrix} x_1(t_i) & x_2(t_i) & x_3(t_i) & \dots & x_M(t_i) \end{bmatrix}^T \tag{1.1}$$

Collecting K of those observations, the training matrix can be constructed as follows:

$$\mathbf{D} = \begin{bmatrix} x_1(t_1) & x_1(t_2) & \dots & x_1(t_K) \\ x_2(t_1) & x_2(t_2) & \dots & x_2(t_K) \\ \vdots & \vdots & \ddots & \vdots \\ x_M(t_1) & x_M(t_2) & \dots & x_M(t_K) \end{bmatrix} \tag{1.2}$$

An estimate of the observed state $\mathbf{X}_{obs}$ can then be computed using the training matrix $\mathbf{D}$ and weight matrix $\mathbf{W}$, where $\mathbf{W}$ is computed using $\mathbf{D}$ and $\mathbf{X}_{obs}$ and where the $\otimes$ sign denotes the Gaussian kernel similarity operator $\mathbf{K}_h(x, x_i)$ [16]. All equations are presented below:

$$\mathbf{X}_{est} = \mathbf{D} \cdot \mathbf{W} \tag{1.3}$$

$$\mathbf{W} = (\mathbf{D}^T \otimes \mathbf{D})^{-1} \cdot (\mathbf{D}^T \otimes \mathbf{X}_{obs}) \tag{1.4}$$

$$\mathbf{K}_h(x, x_i) = \frac{1}{\sqrt{2\pi}h} e^{\frac{(x-x_i)^2}{2h^2}} \tag{1.5}$$

The health indicator is then computed as the parameter residuals of a indicative sensor, meaning a sensor that shows a clear visual degradation pattern over time. For the ACS, the HI was computed by taking the difference between the measured value and the estimated value of RAMT (bleed air temperature) sensor as follows:

$$HI = T_{\text{RAMT,measured}} - T_{\text{RAMT,estimated}} \tag{1.6}$$

Sensor selection for the training matrix was done by computing the correlation coefficients for all sensor signals. Only sensors with a strong correlation ($>\sim 0.7$) were selected to be part of the state vector $\mathbf{X}$. The training matrix was then constructed using data in normal condition, meaning no anomalies were observed and no maintenance was performed during that time. To reduce the computational burden, min-max and vector ordering was used to minimize the size of the training data while still including the optimal subset of historical observations [17], leading to a training matrix of size 405×5. Using the MSET method, the authors were able to extract a health indicator that was highly related to the ACS degradation state as the HI value grows higher in magnitude over time, and falls back to around zero when maintenance is performed [14].

In [18], the authors aim to predict the RUL of a turbofan engine using health index similarity. In this paragraph, the method used to construct the HI will be discussed. The procedure starts with partitioning the data from 21 sensors inside the turbofan engine into 6 different operating regimes. If this would not be done, no clear degradation trend was observed over time. Using a simple 3D plot, it was found that 6 distinct clusters in the data are present, avoiding the need for sophisticated clustering algorithms. The next step is to select useful sensors that might reveal information about the engine degradation. Some of the 21 sensors only attain single or multiple constant values that show no clear degradation pattern. Also, some sensors shown inconsistent trends near the unit's end of life. For example, for some units the sensor value might increase over time, while for some units the value remains approximately constant. These sensors are also not suitable for HI construction. All sensors that are selected, a total of 9 out of 21, are continuously-valued with a clear degradation trend. The question remains whether all 9 sensors actually contribute to improving degradation assessment quality of the HI. Therefore, the model was trained using only the 7 sensors with lowest noise variance and clearest degradation trends in all operating regimes.

The model that is trained in [18] is a linear regression model shown in Equation 1.7. The sample training set $\Omega = \{(\mathbf{x}, \text{HI})\}$ consists of sensor measurement vectors $\mathbf{x}$ that are either from cycles very close to engine failure, in which case the value of HI is set to 0, or from cycles in the very early life of the engine, in which case the value of HI is set to 1. In [18], if an engine has 5 or less cycles remaining, the HI is considered to be 0, and the HI is set to 1 if an unit has more than 300 cycles until failure remaining. Training 6 different models leads to the possibility of predicting the HI time series given the sensor measurements time series of that unit.

$$HI = \alpha + \boldsymbol{\beta}^T \cdot \mathbf{x} + \epsilon = \alpha + \sum_{i=1}^{N} \beta_i x_i + \epsilon \tag{1.7}$$

The final data-driven HI extraction method is based on the correlation between the unit under consideration and a healthy unit. In [19], the authors compute the HI of a degraded component by performing accelerated degradation experiments in which they collected multiple vibration signals from both a nominal healthy bearing and a degraded bearing. The correlation coefficient between the vibration sensor signals is directly used as a HI for the degraded component. This HI is computed as follows:

$$HI = \frac{\sum_{i=1}^{N}(x_i - \bar{x}) \cdot \sum_{i=1}^{N}(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{N}(x_i - \bar{x})^2} \cdot \sqrt{\sum_{i=1}^{N}(y_i - \bar{y})^2}} \tag{1.8}$$

In this equation, x_i and y_i are vectors of length N describing the sensor vibration signals, and $\bar{x}$ and $\bar{y}$ the corresponding means. Using this approach, a HI can be constructed for any time point by comparing a degraded unit to a set of healthy units. Due to the high frequency output of the HI over time, the authors in [19] use sampling and smoothing methods to obtain a more useful output. For sampling, the authors decide to only keep 1 point over 10. For smoothing, a simple moving average was used. Lastly, a exponential curve of the form $HI(t) = a \cdot e^{-bt}$ was fitted through the obtained smoothed values of the HI.

1.2.2. General RUL prediction methods

Methods used for RUL prediction can generally be grouped into 3 categories: (i) model-based approaches, (ii) data-driven approaches and (iii) hybrid approaches [9]. This section will explain the characteristics of each category of approaches and will present relevant examples in literature for each. Note that as this research is focused primarily on machine learning (ML) methods, data-driven and ML approaches are more relevant to this study and are extensively described in section 1.3. However, it was found that if system degradation can be modeled well, model-based approaches tend to yield more accurate results [20]. Therefore, a review of model-based approaches is included as parts of those methods can be used to create hybrid approaches that perform superior to both pure data-driven approaches and pure model-based approaches.

Model-based approaches rely on using damage models that match with the degradation characteristics of a component. Using such a model, degradation of a component is modeled step by step in an iterative manner in which the new state of the component is computed taking into account the previous state and the accumulated damage over time. In practice, applying model-based approaches is usually challenging as extensive knowledge on the exact physics of the working of the component is required in order to make accurate degradation predictions. For complex physical systems with many parts operating in different environments, such as an airplane turbofan engine, such damage accumulation models are not usually available [9]. The

most used model-based approaches for RUL prediction in literature are particle filter, Weibull distribution and Eyring modelling. The main concepts of those methods will be briefly described.

Particle filter models are extensively described in [21]. Particle filtering relies on Monte Carlo sampling by using a set of particles representing the state of a system. The state model is initiated with a prior probability function, that reduces to a single point prior estimation when the initial state is fully known. Using a Markov Process, the state of a system can be updated over time using the initial distribution and a transition equation. Using the Chapman-Kolmogorov equation, the prior distribution at any time can be computed. Updating this state is performed when observations become known using Bayesian methods that compute the posterior distribution. The overall state distribution can be computed by taking random particles from the posterior distribution. When the numbers of particles goes to infinity, any arbitrary probability density function of the systems state can be approximated. For a complete description of all current research presented on the topic of particle filters used in prognostics, the reader is referred to [21].

In [22], Weibull distribution fitting is used to model the deterioration of bearings. This paper proposes a model that follows 3 steps: (i) feature extraction, (ii) classification, and (iii) RUL prediction. The authors use a Weibull failure rate function to model the state of bearings. The obtained results from this Weibull functions are used for training a machine learning model that performs classification and RUL prediction. The features of the input signals that are used to fit a Weibull distribution are found using careful analysis of the input data. The most indicative signal features found in the input data are the Root Mean Square, the kurtosis and the Root Mean Square Entropy Estimator. The type of distribution that was fitted is a special type of Weibull distribution, namely the Universal failure rate function. This distribution is especially used as a failure model to predict system reliability and maintainability. This work indicates the applicability of model-based approaches for component RUL prediction by fitting a failure function using the input data characteristics [22].

In [13] the authors introduce the C-MAPSS data set that will be described in subsection 1.3.1. Furthermore, this paper discusses various damage models that can be used to compute the degradation level of a system based on real physical quantities. First, the Arrhenius model is introduced that is able to compute the time to failure for certain mechanical models. This model is represented by an exponential function, and failure time depends on the temperature at failure time and activation energy. Secondly, the Coffin-Mason crack growth model is mentioned. This model has been shown to be able to predict crack growth in several methods based on the temperature range, the maximum temperature and cycling frequencies. The Eyring model is similar to the Arrhenius model, but also models how the time to failure varies with stress. This model is more accurate, but more physical parameters need to be determined before it can be applied. Lastly, the authors propose a health indicator extraction model specific for the C-MAPPS data challenge. This model takes into account the exponential degradation pattern and uses efficiency and flow of several components to model degradation based on physical processes [13]. In [20], the authors use a different model-based approach to model the RUL of bearings. The state of a bearing is computed using the Modified Paris crack growth model that estimates the fatigue life of a component based on the crack length. RUL estimation is performed accordingly using particle filter methods.

Data-driven models do not rely on specific knowledge or expertise of the system that is modelled. Data-driven models rely on historical sensor data of the system and try to find relations and causality between multiple signals. Based on specific and complex patterns in the sensor data, such approaches are able to predict the remaining useful life of a system or component. The downside of data-driven approaches is that a lot of training data is required in order for the model to truly learn the correct relations and causalities between sensor signals. If not enough training data can be provided, the model will not be able to learn the correct complex data structure and will then not output accurate predictions. Some examples of data-driven approaches used for RUL estimation are Neural Networks (NN), Hidden Markov models, Gaussian processes regression and Relevance/Support Vector Machines (RVM/SVM). An extensive analysis on data-driven/machine learning methods used on RUL prediction will be presented in subsection 1.3.2.

Lastly, hybrid approaches aim to combine the benefits of both physics based model-approaches and pure data-driven approaches. For example, a model-based approach can first use expertise on the system physics to extract important characteristics and features. A data-driven or machine learning algorithm can then be used as a classifier that uses the extracted features to make RUL predictions. For example, [22] can be classified as hybrid model as first a Weibull fitting method is used for feature extraction, and a machine learning approach is used after this for classification and RUL estimation. [23] is a hybrid model used for determining Li-ion battery state of charge. In this work, a variety of sensor measurements such as voltage, current and temperatures are used to train a standard NN that predicts the battery state of charge. After this, an Extended Kalman Filter approach is used on the NN output estimations to compute the final state of charge predictions. Only the NN output estimations gave highly fluctuating results that were not able to follow the correct degradation trend well, but when the KF was applied the output was filtered and followed the correct state of charge degradation trend. These papers show the applicability of combining data-driven or machine learning methods with physics based models.

Lastly, Figure 1.1 shows a brief but clear overview of how RUL estimation methods can be classified. In this work, the main focus will be on machine learning RUL prediction methods, but data-driven health indicator extraction methods with a physical interpretation will be combined into the network architecture with the goal of achieving superior RUL prediction performance.

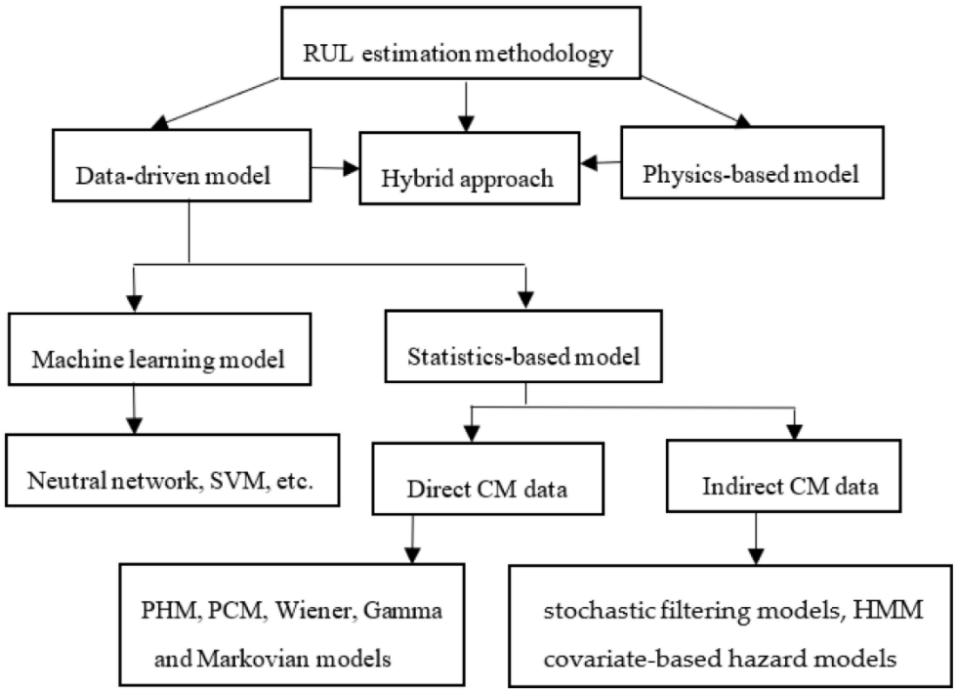


Figure 1.1: Overview of RUL estimation method categories [24]

1.3. Machine Learning for RUL Prediction on Turbofan Degradation Sensor Data

The main focus of this research is on the use of machine learning for RUL prediction. Therefore, this chapter will no longer look at general RUL prediction methods, but will discuss specific approaches to estimate turbofan engine RUL prediction using a variety of different ML algorithm structures. In order to have a solid understanding of the system we want to predict, it is of great importance to first have a closer look at how the system works and what sensors are actually present in a turbofan engine. The general turbofan engine layout will be examined on a high level, and the sensors that provide data are presented. Also, the popular C-MAPSS turbofan engine degradation data set is described. This data set developed by NASA is the most popular data set used for RUL prediction for turbofan engines in scientific literature. Next, recent machine learning RUL prediction attempts on the C-MAPSS data set are discussed. This includes a variety of different ML algorithms structures, such as artificial neural networks (ANN), recurrent neural networks (RNN), long short term

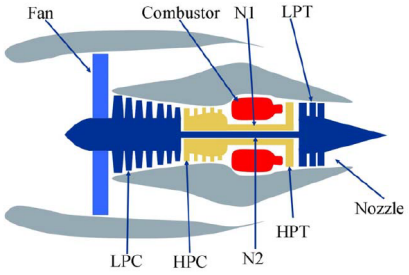


Figure 1.2: Overview of the simulated turbofan layout [13]

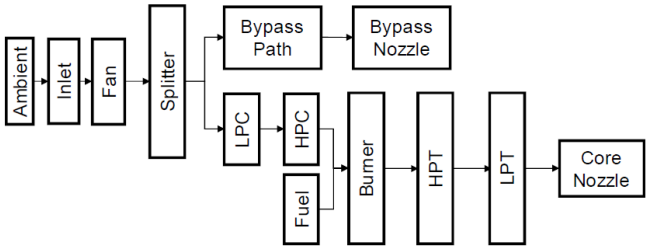


Figure 1.3: Modules and connections of the turbofan engine [13]

memory networks (LSTM), deep neural networks (DNN) and multi-objective deep belief network ensemble (MODBNE). Also, a genetic algorithm (GA) approach for network hyperparameter tuning is included in the discussion. RUL estimation using convolutional neural networks (CNN) is discussed separately in more detail, as this research will primarily focus on the use of CNN. Lastly, all relevant ML attempts on RUL prediction are summarized in a table for clarity.

1.3.1. The turbofan engine model and C-MAPSS data set

On of the core challenges in data-driven prognostics is the availability of run-to-failure data sets. Usually, aviation companies that perform maintenance on their fleet gather data that they keep private and use to improve their own maintenance solutions. Publicly available sensor data is therefore scarce. Furthermore, run-to-failure data of turbofan engines is hard to collect, as due to maintenance operations the engines are almost never used until complete failure. For those reasons, the publicly available NASA turbofan engine degradation data set is a popular benchmarking problem for comparing the performance of prognostic algorithms [6] [13]. This data set is developed using a model-based simulation program called C-MAPSS (Commercial Modular Aero-Propulsion System Simulation), which is a tool made by NASA that is able to simulate degradation in turbofan engines in a Matlab and Simulink environment.

As this research focuses on the RUL prediction of a turbofan engine, it might aid in the process of understanding to have a better image of what such an engine looks like. The C-MAPSS data set simulates engines with a 90,000 lb thrust class, for several operating conditions, altitudes, Mach numbers and temperatures. In Figure 1.2, the general layout of a turbofan engine can be seen. Figure 1.3 shows the different modules within the engine, and shows how those modules are connected. C-MAPPS takes 14 inputs to simulate the various degradation scenarios of the simulated engine [13].

The output of the C-MAPSS software is a run-to-failure data set of 21 time series signals that can be considered to be the sensor measurements of the engine. The sensors that are present in the C-MAPSS data set that can be used to predict degradation of the engine can be found in Table 1.1.

Table 1.1: Overview of the 21 sensors that are included in the C-MAPSS data set [13]

Symbol	Description	Units
Parameters available as sensor data		
T2	Total temperature at fan inlet	°R
T24	Total temperature at LPC outlet	°R
T30	Total temperature at HPC outlet	°R
T50	Total temperature at LPT outlet	°R
P2	Pressure at fan inlet	psia
P15	Total pressure in bypass-duct	psia
P30	Total pressure at HPC outlet	psia
Nf	Physical fan speed	rpm
Nc	Physical core speed	rpm
epr	Engine pressure ratio (P50/P2)	–
Ps30	Static pressure at HPC outlet	psia

Symbol	Description	Units
Parameters available as sensor data		
phi	Ratio of fuel flow to Ps30	pps/psi
NRf	Corrected fan speed	rpm
NRc	Corrected core speed	rpm
BPR	Bypass Ratio	–
farB	Burner fuel-air ratio	–
htBleed	Bleed Enthalpy	–
Nf_dmd	Demanded fan speed	rpm
PCNfR_dmd	Demanded corrected fan speed	rpm
W31	HPT coolant bleed	lbm/s
W32	LPT coolant bleed	lbm/s

The complete C-MAPSS data set consists of 4 sub-data sets, named FD001, FD002, FD003 and FD004, where the difference between them is the number of operating conditions and fault modes. Each data set is made up of a training data set and a testing data set. The run-to-failure data is simulated for multiple engines that start with varying degrees of wear and is stored as time series data for all 21 sensors. After some

time, a fault develops in an engine and grows until the engine can no longer fulfil its intended function, after which the engine is declared unhealthy. The final recorded time step for each engine is the time at which the engine has failed. For the testing data set, the time series is terminated at some unknown point before failure. The goal is to predict the remaining cycles until failure would occur given the time series sensor data of that engine up to that point. The correct RUL values of the test data are also provided. The data set is structured as a n-by-26 matrix, where n is the number of time cycles until failure for each engine. For the 26 columns, the first column is the engine number, the second column is the operational cycle number, the third to fifth columns are the operational settings, and the remaining 21 columns represent the sensor measurement values. Table 1.2 gives a comprehensive overview of the number of engines in the C-MAPSS sub-data sets.

Besides the C-MAPSS data set, another popular turbofan degradation data set is the "PHM08 Challenge Data Set". This data set is also provided by NASA, as part of a competition held at the first international conference on Prognostics and Health Management in 2008. This data set similar to the C-MAPSS data set, except that for the test data set the correct RUL values are not presented. Participants of the competition were able to upload their predictions on the test set, after which the organization would compute their score [6]. For the coming sections, if any literature uses the PHM08 Challenge data, keep in mind that the structure of this data set is equivalent to that of the C-MAPSS data set.

Table 1.2: Overview of the C-MAPSS data set [6]

C-MAPSS data set	FD001	FD002	FD003	FD004
Nr. of engines for training	100	260	100	249
Nr. of engines for testing	100	259	100	248
Operating conditions	1	6	1	6
Fault modes	1	1	2	2

1.3.2. Machine learning RUL prediction efforts on the C-MAPSS data set

The C-MAPSS data set is the most widely used benchmarking problem in turbofan engine RUL estimation. Over the past years, many attempts to predict the RUL values of the engines in the test data set can be found. More recently, machine learning has risen to be one of the most promising methods to predict RUL using the C-MAPSS data set. This section aims to review the most relevant attempts at RUL prediction on this data set in literature using a variety of machine learning techniques.

[25] is one of the first attempts on RUL prediction using a publicly available turbofan degradation data set. As part of the PHM08 Challenge, the author attempts to predict the RUL values of the test data set using several machine learning methods. The novelty of this paper is that it solely relies on machine learning software, and that no attempts were done to analyze the data or find specific patterns that could indicate degradation based on any physics-based principles. As an initial try, the author investigates whether a simple Multi-Layer Perceptron (MLP) is able to differentiate between an engine in healthy state, and a engine in near-failure state. For each engine, the first 30 cycles are labeled as healthy, and the final 30 cycles before failure are labeled as unhealthy. A simple 3-layer network with 24 inputs (3 operating condition settings and 21 sensors values) was able to correctly identify the health state of a new test engine 99.1% of the time. Network training time was found to only be around 5 seconds. The next step is for the network to output a RUL estimate in stead of only a 'healthy' or 'unhealthy' label. This was done by training the model using the remaining operational cycles in the training data. For example, if an engine in the training data has a life of 200 cycles and the RUL has to be predicted after 50 cycles, the target RUL value for training was set at 150 cycles. Figure 1.4 shows the RUL prediction output for one specific engine. It can be seen that even though the prediction (red) is very far away from the target RUL values (blue), the MLP captures the trend of decreasing RUL over time and ends at 0 at approximately the same time as the true RUL.

Further observations of the MLP network output lead to the change in a very important aspect of RUL prediction: the target RUL during training. During the initial healthy cycles of an engine, no fault has occured and therefore no degradation propagates through the sensors over time. Therefore, degradation is only expected to be noticeable after a certain amount cycles used. For this reason, the target RUL value when a component is still very healthy is set to a constant value, as an healthy engine's sensor values provide no way

for the model to differentiate between for example a RUL value of 200 or 300 cycles. Therefore, the authors set the constant RUL target value for training at 130 cycles, as this is more than the minimum number of life cycles for all engines in the data set and the MLP output shows a steady state output near cycle number 140. To improve the prediction accuracy of the model, the author then reverts to a recurrent neural network (RNN), as this model is able to incorporate the time dependence of the data and use information of the past states. RNN uses internal memory of past information and can learn the dependency on past hidden states. Finally, Differential Evolution (DE) is used to find to optimal structure of the RNN network. Using a population of solutions, the RNN structure, the number of hidden nodes and some other model hyperparameters were tuned. By evaluating the fitness of each individual, the best solutions were crossed over. Using also mutation, the final RNN architecture was determined. Hundreds of different model structures were trained and tested in this manner, which saved tremendous time as human trial and error methods were deemed unnecessary. The final RNN RUL prediction output can be seen in Figure 1.5 for the same singe engine unit. A large improvement compared to the MLP can be observed. Using a combination of the 3 best performing RNN models, a second place was obtained in the PHM08 Challenge [25].

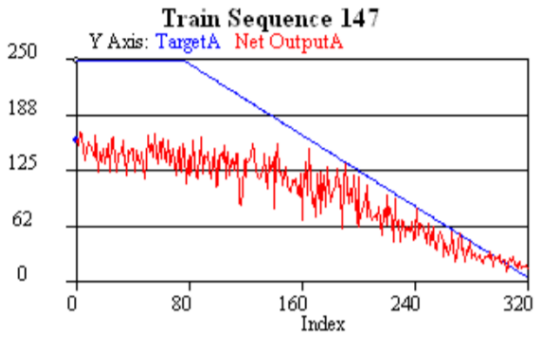


Figure 1.4: MLP output versus target output [25]

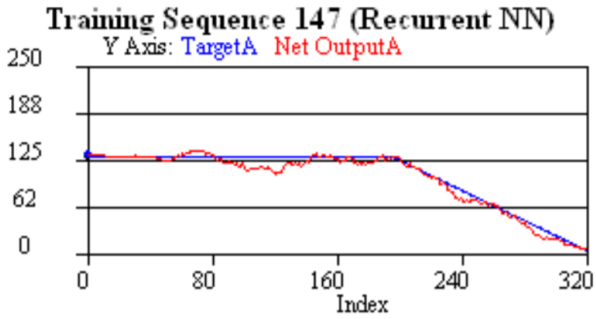


Figure 1.5: RNN output versus target output [25]

In [26], the author tries to predict RUL for the PHM08 Challenge using a Kalman filter ensemble of neural network models. The main contribution of this paper to this research are two aspects. First, the high dimensional feature data is mapped into a two dimensional image using neuroscale mapping. Using a Radial Basis Function neural network that minimizes the Sammon Stress Metric, the high dimensional data was mapped to 2D format such that Figure 1.6 could be created. From this image, it can be clearly observed that the engine data is clustered in 6 distinct operating regimes. Furthermore, it can be seen that points with lighter colour, which indicates a lower RUL, are located further away from their corresponding cluster. This means that the operating condition of the engine might aid in predicting the RUL of an engine unit. Figure 1.7 shows a 3D plot of all operational settings provided in the data set, and again all 6 operating regimes can be clearly seen. The challenge guidelines mention that the units operating regimes have a significant relationship with the engines performance. Therefore, the author decided to include the operating regime history of each engine as an additional feature to the model input, by adding six columns to the data set that indicate the number of cycles each unit has been in each of the 6 operating regimes.

Secondly, [26] contributes to this research by using data normalization. The author uses a standard method where the normalized value for a feature are computed by subtracting the mean and dividing by the standard deviation of all values of that feature. However, as this data set contains 6 clear data clusters, the data was normalized only by using the mean and variance of the values of a feature in the same operating regime in order to maximize variance in each mode. This method was proven to yield superior results compared to regular data normalization. The rest of this paper describes a Kalman Filter ensemble of neural network models, which turned out to win the PHM08 competition. As this approach is not directly related to the research in this study it is not further discussed.

[27] combines the approaches of computing a HI using linear regression and a standard neural network (NN). The authors use the exact same linear regression model approach as was explained in subsection 1.2.1, Equation 1.7 [18]. Labelling healthy engines with HI = 1, and near-failure engines with HI = 0, this linear model was trained to predict a HI at any time point. Note that again 6 different linear models were trained, one for each operating regime. In [27], the HI is computed for all training engines at any time point. The obtained HI are used as target output of a standard 2-layer NN, where the input is a concatenation of the sensor data and amount of time cycles spent in each operating condition at that time point. This simple network is now able

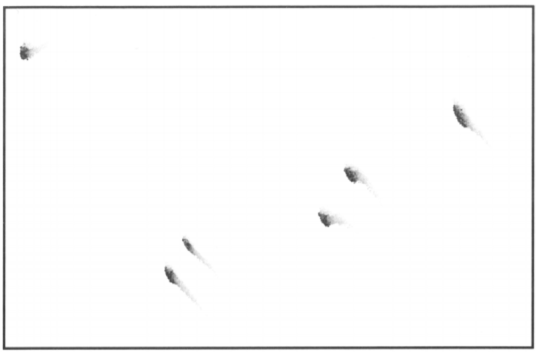


Figure 1.6: Neuroscale image of all points in the PHM08 data set. Lighter coloured dots indicate engines closer to failure. Six distinct clusters can be seen. Note that in neuroscale plots the axis are irrelevant [26]

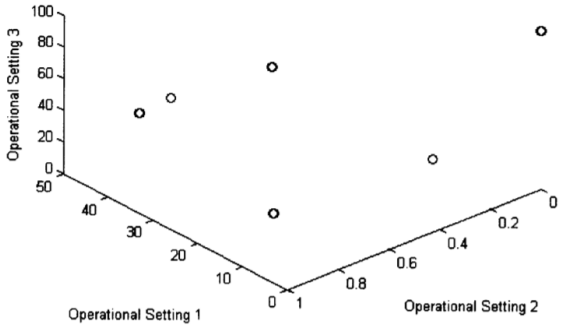


Figure 1.7: Three dimensional plot of the three operational settings clearly shows the six distinct operating regimes of all engines [26]

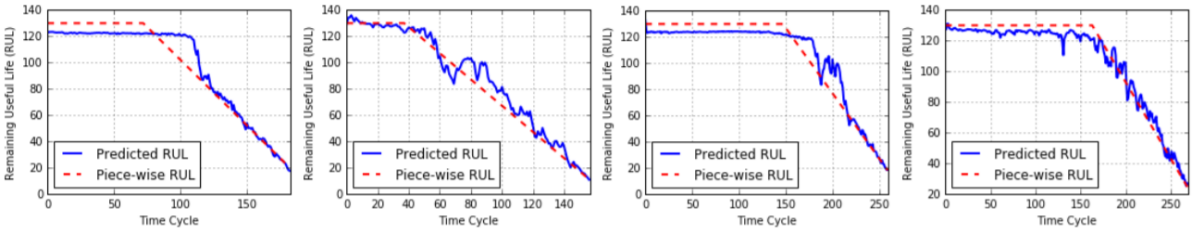
to predict a HI for a test engine at any time point. The RUL of a test engine is now predicted as follows: at all time cycles that sensor data is available, a HI is estimated using the NN. This time series of HI is smoothed using a moving average, and a higher order polynomial is fitted through these HI data points. The RUL is then computed by taking the difference between when the polynomial fit crosses HI = 0 and the last time cycle in the test engine data.

More recently, [28] uses an ensemble of Deep Belief Networks (DBN) for RUL prediction. Here, the authors take advantage of the proven fact that for an ensemble of multiple models, the outputs of each of the base models can be combined to improve generalization. Previous works have shown that for an optimal ensemble, each of the base models should generate small errors and have a good diversity among themselves in order to pursue superior generalization. [29] has shown that for a base learner model, accuracy and diversity are sufficient conditions for the ensemble to outperform any of its members. It was however found that those conditions often conflict, and therefore the authors revert to a Multi-objective Deep Belief Network Ensemble (MOBDNE) method. Many previous work on RUL prediction, such as [25], only focuses on a single objective such as minimization of RMSE or the PHM08 Challenge scoring function. This paper uses a multi-objective function to train the base learners of the ensemble. Each base model is DBN, which is composed of several stacked Restricted Boltzmann Machines (RBM). Using a Multi-objective Evolutionary Algorithm (MOEA), multiple DBN's are trained at the same time. The MOAE decision variables are the the DBN's structure parameters, such as the number of layers, the number of hidden neurons, the values of the weights and learning rates. An initial population of DBN's is evolved using crossover and mutation for a predetermined number of generations. The final generations of DBN's is used as the ensemble of base models. The DBN's in the final population are combined to an ensemble where the weights of each DBN are optimized using a single objective differential equation. Here, the only objective is to minimize the training prediction error of the ensemble network. Besides this method, the authors also use sensor selection and data normalization in a similar fashion as was mentioned before. This method achieved very good performance on instance 1 and 3 of the C-MAPSS data set, but significantly lower performance on instance 2 and 4 (where multiple fault modes are present). Still, this method outperformed traditional machine learning algorithms such as Support Vector Machines (SVM), and standard Multi-layer Perceptrons (MLP).

[10] is the first attempt of using Long Short-term Memory (LSTM) for RUL prediction on the C-MAPSS data set. LSTM is a branch of RNN's that is able to learn long-term dependencies in the data. It has shown to be effective for sequential data, such as speech and music recognition, text prediction and motion-capturing. Using forget, input and output gates for a certain states, a LSTM network has to ability remove or let information into the current cell state. This is different from a standard RNN, where the cell state is completely overwritten each time step. In this manner, LSTM is able to memorize long-term data patterns that aid in making more accurate predictions. This LSTM method was used to predict faults in engine and RUL from sensor data. RUL labeling was performed using training of a SVM that is used as an anomaly detector. This was done as component degradation will generally not be visible after the component has been used for some time and a fault has occured. Using a SVM, labeled RUL values were obtained for the training data such that initially the RUL was set at a constant value. The performance of the LSTM was compared to 3 different types of RNN's (simple RNN, GRU LSTM and simple LSTM), and an esemble LSTM method. It was found that the

simple LSTM method obtained the lowest error scores compared to other methods.

In [11], the authors also use a LSTM approach for RUL estimation. In addition to the previously discussed paper, in [11] the authors use 2 LSTM layers in combination with 2 feed forward fully connected layers. Again, the LSTM layers serve the purpose of learning long-term dependencies in the data, while the fully connected layers are now able to extract hidden patterns in the data structure. The addition of the fully connected layers improves the mapping of the LSTM output to the regression outcome. The objective that is optimized in this paper is the mean squared error between the true RUL and the predicted RUL. In this paper, 2 types of data normalization are used: Z-score normalization and min-max normalization. As used in similar papers on the same topic, a piece-wise linear target RUL function is used, where a constant target RUL value of 130 cycles is used in the engines early life. Using cross-validation, the architecture of the model was optimized by changing the number of layers and hidden nodes. The developed method was compared to a simple MLP, a SVR and a CNN. The LSTM method obtained the lowest scores, whit CNN performing just slightly worse (9% higher score).



(a) FD001 example (b) FD002 example (c) FD003 example (d) FD004 example

Figure 1.8: Example RUL prediction of the LSTM method in [11] for all C-MAPSS data instances

[12] is the most recent attempt in literature that predicts RUL using LSTM. In this paper, the authors note that unsupervised learning techniques are able to extract high-level features from raw input data. They find that the combination of unsupervised pre-training in combination with supervised learning has the possibility of achieving higher RUL prediction accuracy then just using only supervised learning techniques. The goal of this paper is thus to incorporate unsupervised learning methods to achieve superior prediction performance. The authors use a Restricted Boltzmann Machine (RBM) for unsupervised pre-training to learn abstract features from the raw input data. Using these features, the weights can be initialized near a local minimum before the supervised fine-tuning is performed. After this RBM layer, 2 LSTM layers are used. Lastly, a fully connected layer and an output layer are present. Additionally, the authors use a Genetic Algorithm (GA) for hyperparameter tuning. Using a population of networks with different hyperparameters, evolutionary methods such as crossover and mutation are applied to the individuals. In this manner, the population will evolve such that the networks architecture will be optimized and the hyperparameters are ideally tuned. Furthermore, this paper uses manual sensor selection, z-score input normalization, and again an RUL_{early} of 130 cycles. The average training time for each instance of the data set is around 60 hours. The algorithm developed in this paper obtained the best performance in terms of RMSE on all instances except FD002 up to that point in time (the CNN in [9] outperformed this LSTM method). In terms of the PHM08 Challenge score, this method won on all instances.

The authors of [30] attempt to predict RUL using degradation pattern learning. Their approach at RUL computation consists of two parts: a global part that is able to learn the distribution of all degradation patterns that reflects the structure in the data, and a local part that describes useful information on the state of a specific unit. The state of a unit is described by not only a HI, but also by the degradation speed and the corresponding pattern. A HI is adopted from previous work, computed using a very simply linear regression that just multiplies sensor weights with the corresponding sensor values. In this work, no sensor selection is performed as the authors state that sensors that do not reveal much useful information are automatically assigned a weight value close to 0. RUL is computed by an Adjacent Difference Neural Network (ADNN) that is similar to a standard MLP trained with back propagation, but an additional term is added to the loss function where to goal is to smooth weights corresponding to the same hidden node. This is beneficial as more smooth weights damp the effect of noise in the fixed-input length time series data. The weights of the trained ADNN are considered as the degradation pattern learned by the model based on the input HI information. Regarding the results, the sorted RUL predictions for all engines in the C-MAPSS test set are shown in Figure 1.9.

The obtained PHM08 Challenge score would have given them a 4th place in the competition, showing the potential of the method.

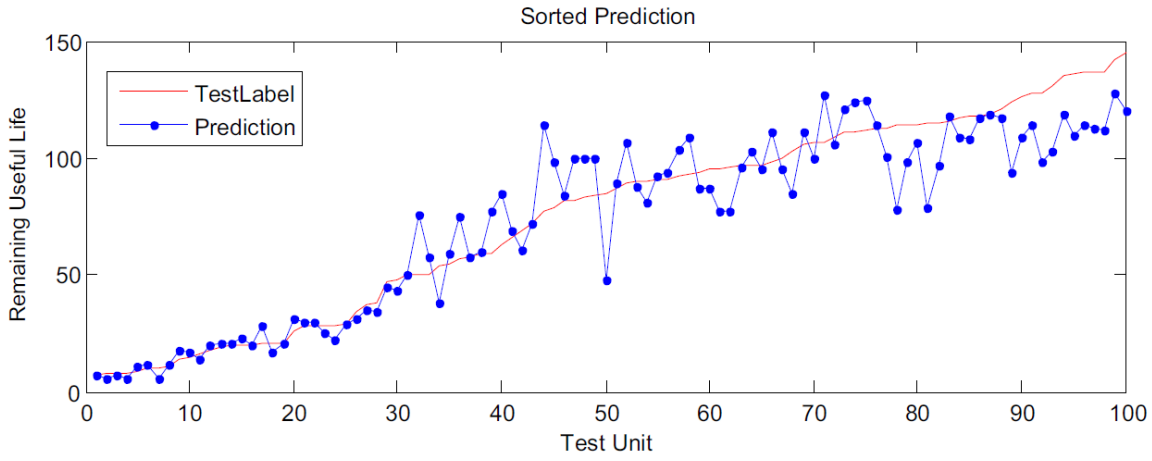


Figure 1.9: Sorted RUL prediction for all engines in the test set by [30]

Lastly, the most recent attempts on RUL prediction on the C-MAPSS data set are mentioned to show the current state of research. Note that some works are not highly related to this study, where the main goal is to use a CNN on raw sensor data, and therefore these recent papers will only be discussed briefly. In [31], the authors make use of a ML-based framework that clusters observations using a kernel density estimation. This is done in order to reduce the high dimensionality of the data, which is often the case for multiple time series sensor measurements. ML is used to learn the low-dimension representation of the identified clusters. Although regular DNN's are able to deal with high-dimensional data, they require many training samples which is often not available in the real world, especially not for run-to-failure applications. The newly develop method believes that "black box" DNN approaches are inferior to simpler ML methods that first reduce the data dimensionality, as there is an increasing need to understand the characteristics of ML algorithms applied in PHM.

[32] uses a RNN autoencoder for feature extraction and subsequent computation of a HI. The obtained values for the HI are used as training samples to train a linear regression model. Using these models, a library of HI degradation curves is made. For a test engine, a HI trajectory is computed and similarity matching is used to find the most similar degradation trend in the HI curve library. The final RUL computation is performed by computing the the weighted average of the RUL's of the most similar engines in the training set.

1.3.3. Convolutional neural networks for RUL prediction on the C-MAPSS data set

The previous section focused on a large variety of machine learning techniques applied to the turbofan engine RUL prediction problem. In this section, literature that focuses specifically on RUL prediction using convolutional neural networks (CNN) will be reviewed, as the CNN's are the predominant focus of this research. This study will continue by developing an algorithm for turbofan RUL prediction using a CNN method, and therefore it is of great importance to have a solid understanding of how CNN's can be applied to the C-MAPSS data set for RUL estimation. In literature only two attempts on RUL prediction using a CNN can be found. The full methods of those papers will be discussed, including the initial data feature extraction, the data preprocessing, the network architecture, network training, and network output results.

The strength of CNN's lies within its potential identify multiple hidden patterns of input sensor data. With increasing number of convolutional layers, more abstract features of the input data can be revealed. For example, in image processing, lower level features might be just simple lines or edges, while higher level features that are obtained by combining lower level features might be more sophisticated shapes such as eyes or ears. For RUL estimation using CNN, the authors in [8] define two main challenges: (i) processing input in the CNN should be performed along the temporal dimension and (ii) sharing and unifying the units of the CNN among a variety of sensors.

[8] is the first attempt in scientific literature on RUL prediction using a CNN. Similar to many papers discussed before, the authors first note that the data points are clustered into 6 different operating regimes

by plotting the 3 operating setting values. Using the same method mentioned in [26], the amount of cycles spent in each of the 6 operating regimes is added as additional model input features. Furthermore, Z-score data normalization is used for all input values (being the sensor measurements and the number of cycles spent in each operating regime). Again, just as in [26] this normalization is performed with respect to each operating regime. The final aspect the authors mention before explaining their method is that similar to many previous papers, a piece-wise linear degradation target RUL function is used due to the fact that degradation information in the early life of a component is almost absent. The popular value for the constant early RUL is set at 130 cycles.

An overview of the CNN architecture developed in [8] can be found in Figure 1.10. The input to the model is a $D \times S$ matrix, where D is the number of attributes (the number of sensors plus the 6 additional columns representing the operating condition history) and where S is the amount of time cycles history. In this paper, the input size is equal to 27×15. The value for 15 was chosen because in the test data set there is one engine that only has 15 time cycle data points. Increasing the value of S might lead to increased prediction performance as more time cycles history are included, but this comes at the cost of higher computational load. This input matrix is fed into a convolutional layer, after which a pooling layer is added. Then, another convolution and pooling operation is performed, after which the extracted high level features are fed into a fully connected MLP layer. The output layer is a single neuron that estimates the RUL.

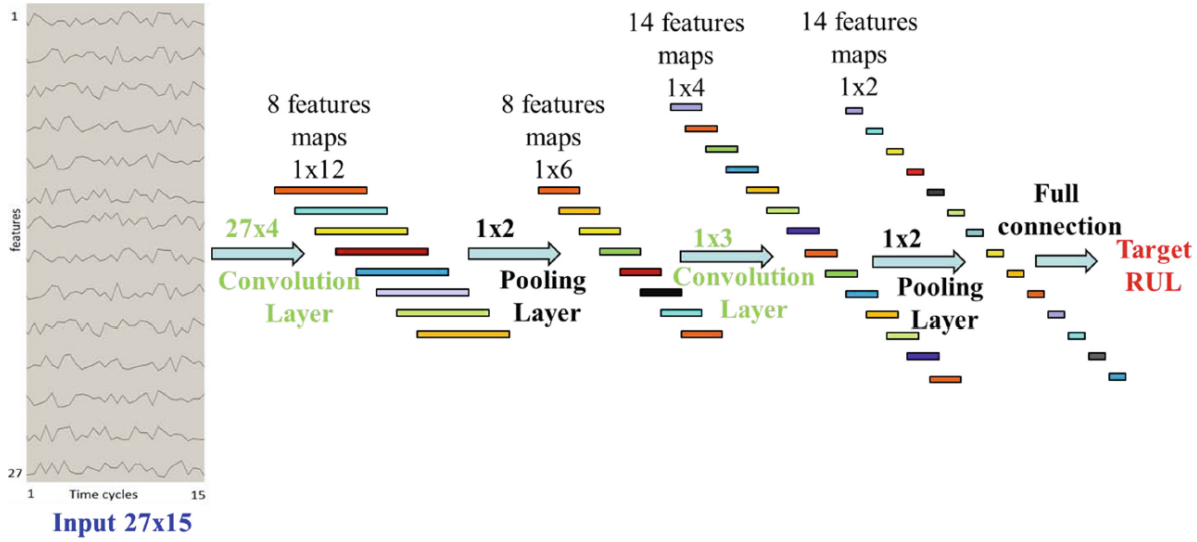


Figure 1.10: Overview of the CNN architecture used in [8]

A convolution operation is performed by sliding a window (kernel) over the input data and applying element-wise multiplications between the weights of the kernel and the location of the input where the kernel is located. This process of sliding a kernel over the 2D input matrix and applying convolution operations can be formally expressed as follows:

$$\mathbf{x}_j^l = Sig(\mathbf{z}_j^l) \tag{1.9}$$

$$\mathbf{z}_j^l = \sum_i \mathbf{x}_j^{l-1} * \mathbf{k}_{ij}^l + b_j^l \tag{1.10}$$

Here, $\mathbf{x}_j^{l-1}$ is the filter input, $\mathbf{k}_{ij}^l$ is the kernel, b_j^l is a bias that is to be learned, $\mathbf{z}_j^l$ is the input to the sigmoid activation function, and $\mathbf{x}_j^l$ is the output feature map of the convolution operation. Sig in Equation 1.9 stands for the sigmoid activation function, which was chosen due to its simplicity. In [8], the filter size in the first convolutional layer is $D \times 4$, meaning that the filter has the same height as the input data matrix and slides along the temporal dimension of the input data. Given the input matrix width, the filter can perform 12 convolution operations as no zero-padding of the input is used. If it were to be desired to retain the same size as the input, padding operations could have been used. In Figure 1.10, it can be seen that the in the first convolution operation, a total of 8 filters are used that each transform the 27×15 input matrix to a 1×12 feature map. For the second convolutional layer, the filter size is now 1×3. This reduces the size of the 14 feature maps to 1×4.

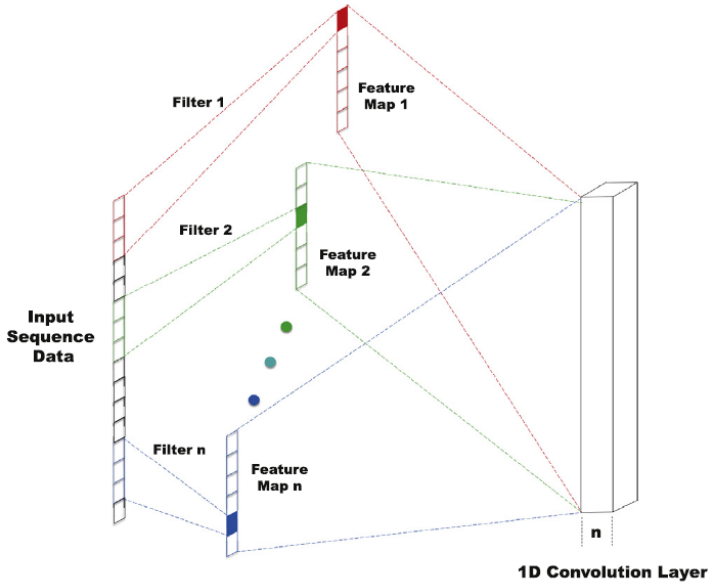


Figure 1.11: Convolution operation of n filters on 1D input data [9]

In between the convolutional layers, average pooling layers are placed. The goal of pooling layers is to sample the feature maps such that their size is reduced without the loss of the most indicative features. In [8], the size of the pooling filters is 1×2. Pooling then works as follows: on each of the first 8 feature maps of size 1×12, the average of the first and second index becomes the first index of the feature map after pooling. The average value of the third and the fourth index of the feature map before pooling becomes the second index of the new feature map, etc. In this manner, the size of each feature map is cut in half. After the final pooling layer, all 28 values that are present in the 14 feature maps are flattened and connected to a fully connected layer for RUL estimation. The network is trained using back-propagation and gradient descend. As almost all current programming languages have sophisticated packages that can train and optimize network weights automatically, the mathematical theory on network training is not discussed here.

The CNN developed in [8] was trained and tested on the C-MAPSS data set. The performance of the CNN was also compared to other previously used machine learning techniques, such as a Multi-layer Perceptron (MLP), a Support Vector Machine (SVM) and a Relevance vector machine (RVR). In terms of RMSE, the CNN outperformed all other methods for all sub-data sets. In terms of the PHM08 Challenge score, that punishes late RUL predictions harder than early predictions, only the RVR outperformed the developed CNN. Based on those results, in combination with the fact that this is only the first ever attempt at RUL prediction using CNN, the authors conclude that this method is promising.

[9] is the second, more recent attempt in literature that attempts to predict turbofan RUL using a CNN. This work is highly correlated to the work in [8], but the methods and network architecture are slightly altered. The input to the network is once again a 2D matrix of size $S \times D$, where S is the time sequence history and D the number of input features. Opposed to the method used in [8], authors of [9] note that there is no relationship between spatially neighbouring features in the 2D input data. That is, no relationship between different sensors at the same time is present, there is only a temporal relationship between a single sensors measurements over time. For this reason, the size of the filter that is used for the convolution operation on the input layer is $F_L \times 1$, where F_L is the filter size with default value of 10. This means that although convolution is applied on a 2D input matrix, the actual convolution operation is only 1-dimensional as the filter size is also 1-dimensional. This process of 1D convolution on an input sequence can be visualized as in Figure 1.11, where n 1D filters are used to create n feature maps that are then presented as a 3D output with depth n.

The remainder of the deep CNN architecture can be found in Figure 1.12. The full model consists of 5 convolution layers able to extract high level features. Note that opposed to the model in [8], no pooling layers added after each convolution layer. This is not done as the authors state that even though computational load is reduced, useful feature information might be lost. In addition, the input size is not very large compared to other popular CNN applications such as image processing. Furthermore, zero-padding of the input is

used to ensure that the output size of a convolution layer is the same as the input size of that layer. Each convolution layer uses 10 filters, such that the output of each convolution layer is 3D, where the height is the time history sequence, the width the number of sensors that are included in the input, and the depth the number of feature maps generated by the filters. At each layer, a hyperbolic tangent activation function is used. After the convolution layers, the data is flattened and a fully connected layer is added to map the features to a corresponding RUL estimate. During training, dropout is applied at the fully connected layer to prevent network overfitting on the training data. Dropout is incorporated by setting each of the weights in connected to the hidden neurons to 0 with a specified probability. As will be discussed in section 1.5, dropout can also be used to represent a ensemble model. The input data is normalized using min-max normalization. Sensor selection is performed such that only sensors that provide useful information on degradation and have continuous values are included in the input. This leads to an input matrix size of $S \times 14$, as 14 sensors are included. Compared to [8], this paper does not include a history of the engines operating conditions in the input. The time history sequence input for each sensor differs for each sub-data set, and reaches from 15 to 30 cycles. Finally, this paper also uses a piece-wise linear RUL target function, with a early constant RUL value of 125. Network training is done using Back-propagation (BP) using Adam optimization.

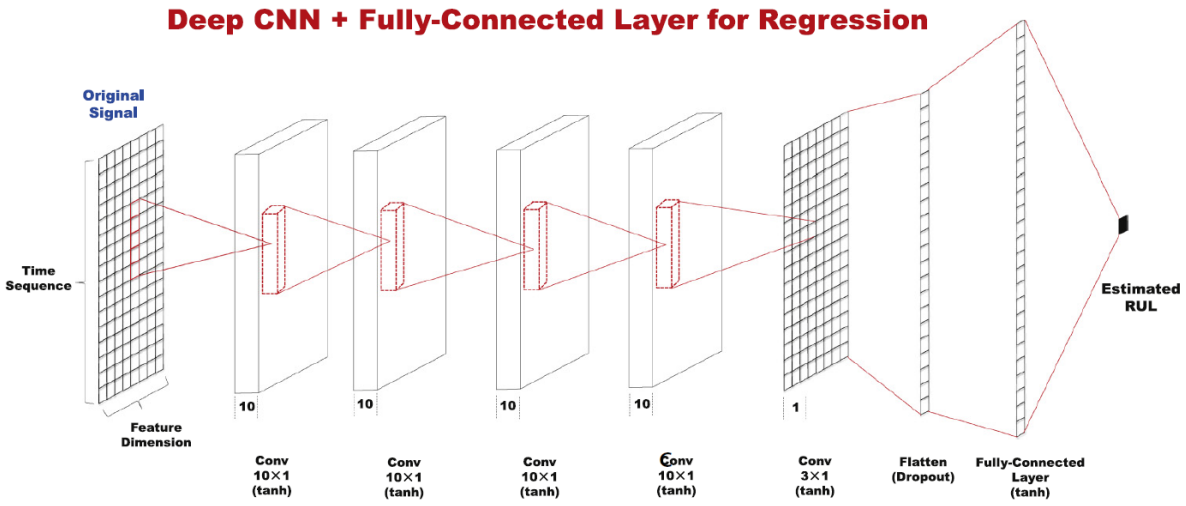


Figure 1.12: Overview of the CNN architecture used in [9]

The results of the RUL prediction of the method used in [9] can be found in Figure 1.13. In this image, the test engines are sorted by increasing RUL (blue line), and the model predictions are shown in orange. It can be seen that generally, the predictions are quite good, especially for engines with a small RUL value (RUL < 30). Furthermore, this method was compared to several other ML methods, such as a standard MLP, a DNN, a RNN and a LSTM network. The deep CNN outperformed all other techniques for RUL prediction, both in terms of RMSE and PHM08 Challenge score. Lastly, authors include a sensitivity analysis on the network architecture by changing the number of convolutional layers and the size of the time window that is included in the input. It was found that RUL prediction performance increased with both a higher number of convolutional layers and with a larger time window size. However, no significant improvement was found after adding more than 5 convolutional layers and making the time window size larger than 25, while computational training time increased linearly for both parameters.

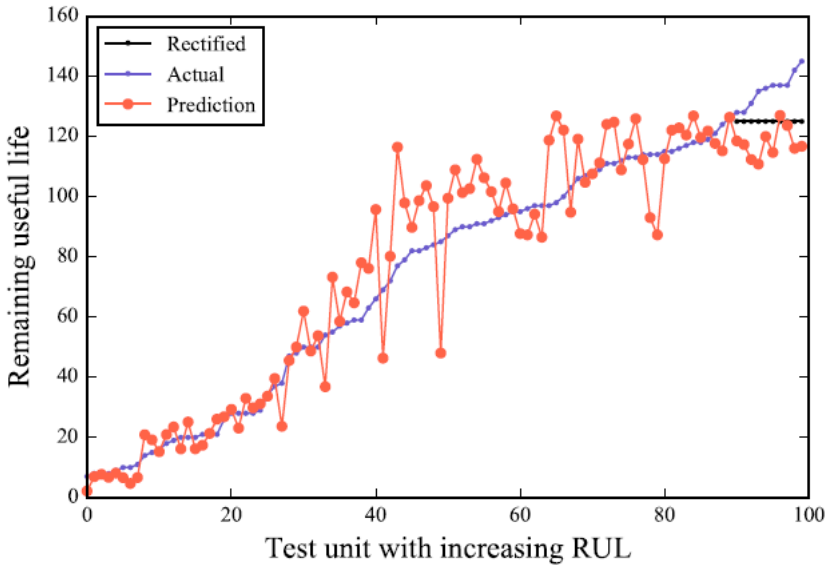


Figure 1.13: Sorted RUL prediction for all engines in the test set by [9]

Lastly, an overview of all tune-able parameters of a CNN used for RUL estimation can be found below. This includes both network hyperparameters, as well as input and output parameters based on the desired RUL prediction performance. When developing a CNN for RUL estimation, each of those parameters should be carefully tuned in order to achieve desired performance.

- Nr. of convolutional layers
- Nr. of pooling layers
- Nr. of fully connected layers
- Nr. of filters per layer
- Size of the filters
- Stride of the filters
- Use of input padding
- Type of activation function
- Type of weight initialization
- Type of pooling
- Size of input matrix
- Nr. of neurons in FCL
- Dropout rate
- Learning rate
- Batch size
- Nr. of training epochs
- Type of loss function
- Type of data normalization
- Type of output
- Early constant RUL value
- Nr. of relevant sensors

1.4. Overview of machine learning RUL prediction attempts

This section presents a comprehensive overview of all RUL prediction attempts discussed in this chapter. This overview can be found in Table 1.3. This table shows the initial RUL estimation attempts, the most relevant attempts and the most recent attempts. The following points should be noted:

- This table presents both scores on the C-MAPSS data set and the PHM08 Challenge data set, depending on what data set is used in the paper. The scoring function is a function used in the PHM08 Challenge that exponentially punishes late predictions harder than early prediction. This scoring function will be explained further in subsection 1.6.2. Similar to the RMSE, a lower value is better.
- The RMSE and score values shown for the C-MAPSS data set are all for instance FD001. Most papers also include results on the other 3 instances, but that would take up too much space in this table. Only showing the results for the first instance gives a good indication on the methods performance.
- Especially in the early papers there is no clear consensus on how to present the results. Some papers use MSE, some use RMSE and some use the scoring function. Also, some present their score values as average result per engine, while some papers sum all score values for the engines in the test set. This table is constructed to both show what methods are used and what results are obtained, but the results of the initials papers are therefore hard to compare directly.
- The best results in terms of RMSE and score on instance FD001 on the C-MAPSS data set are shown in bold.

- It can be seen that over time, the results in terms of score on the C-MAPSS data set have greatly improved. Even though the last attempt at RUL prediction using CNN is over 2 years ago, the methods results are still comparable to the best results in the table. This shows the importance of further researching CNN techniques.

- Besides the works shown in Table 1.3, many more attempts at RUL prediction on the C-MAPSS data set have been performed over the past decades, both in literature as on dedicated programming websites and blogs. This table is composed of the most relevant and recent scientific papers, but note that in order to obtain an even broader view of all previous work it might be valuable to also look at other sources, such as technical blogs and websites.

Table 1.3: Overview of initial, most relevant, and most recent RUL prediction attempts on the C-MAPSS (FD001) and PHM08 Challenge data sets (ordered by year of publication)

Authors	Paper	Year	Method	RMSE C-MAPSS	Score C-MAPPS	Score PHM08	Notes
Heimes	[25]	2008	RNN + DE	-	-	-	Total error on PHM08 = 519.8
Peel	[26]	2008	Ensemble of KF NN	-	-	-	MSE on PHM08 = 984
Lee et al.	[18]	2008	LR HI similarity	-	-	5606.06	Score computed on test set of 435 units
Riad et al.	[27]	2010	NN HI prediction	-	-	1540	Outperformed similar LR model
Babu et al.	[8]	2016	Deep CNN	18.448	1286.7	2056	First RUL prediction using CNN
Yuan et al.	[10]	2016	LSTM NN	-	-	-	% of units within error margin shown
Zhang et al.	[28]	2017	MODBNE, no TW	17.96	640.27	-	No time window processing applied
Zhang et al.	[28]	2017	MODBNE, TW	15.04	334.23	-	Time window processing applied
Zhao et al.	[30]	2017	pat. learning + ADNN	-	261	793.642	No RMSE values provided
Zheng et al.	[11]	2017	LSTM	16.14	338	-	Outperformed CNN, MLP, SVR and RVR
Li et al.	[9]	2018	Deep CNN	12.61	274	-	Second RUL prediction using CNN
Ellefsen et al.	[12]	2019	RBM + LSTM + GA	**12.56**	231	-	Best RMSE result on FD001 so far
Aremu et al.	[31]	2020	MLDR + classifier	-	-	-	RMSE only provided on FD002 and FD004
Yu et al.	[32]	2020	RNN + HI similarity	13.58	**228**	-	Most recent work on RUL prediction

1.5. Probabilistic Prediction and Uncertainty Estimation

One of the main drawbacks of almost all papers discussed in the previous chapter is that none of them include an estimation on the uncertainty of the obtained results. Only [9] obtains its results by averaging 10 forward passes of the input through the network and subsequently obtains 10 various output values. In this way, the authors were able to measure to some degree the variation in the network output by computing the standard deviation of the obtained outputs. Besides this one example, there seems to be a lack of uncertainty estimation in recent literature regarding RUL estimation. Inclusion of an estimate of network uncertainty is however of vital importance for adaptation of a prognostic algorithm in practice. Company decision makers are not only interested in the accuracy of a proposed RUL estimation method, but also in how certain and confident that method is in its predictions. Therefore, this chapter investigates methods of obtaining not only point estimates for RUL predictions, but also probability density functions as network outputs. Also, general neural network uncertainty quantification methods are touched upon. The combination of those two aspects related to this study will be briefly discussed.

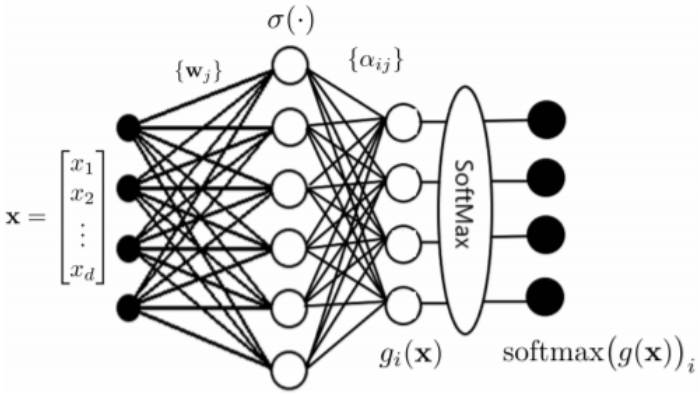


Figure 1.14: Example of a standard neural network using a softmax activation function on the output layer [33]

1.5.1. Obtaining a probabilistic prediction from a neural network

This section discusses how a neural network can output a probabilistic prediction in stead of only a deterministic value prediction. An example of such a deterministic value prediction can be seen in Figure 1.12, where all hidden neurons in the fully connected layer are connected to a single output neuron. This single output neurons value is directly used as the prediction value of the RUL of the engine. The first method of obtaining a probabilistic output is to let the output layer of the network be a vector representing all possible values the output can take instead of just a single neuron. In the case of RUL prediction, this output vector **O** might look as follows:

$$\mathbf{O} = \begin{bmatrix} p(RUL = 0) \\ p(RUL = 1) \\ p(RUL = 2) \\ \vdots \\ p(RUL \geq 130) \end{bmatrix} \tag{1.11}$$

Each index in this output vector now represents the probability that the model assigns to each possible RUL value. These probability values can be obtained using a softmax activation function on the output layer. An example of this is shown in Figure 1.14 [33], where a standard multi-layer neural network with input **x** provides probability estimates on the output using a softmax layer. The values in the output layer $g_i(\mathbf{x})$ are taken as input to the softmax function, that subsequently outputs the assigned probability values of each possible output. The softmax function that transforms the output value into a useful probability is defined as follows [33]:

$$\text{softmax}(g(\mathbf{x}))_i = \frac{e^{g_i(\mathbf{x})}}{\sum_{i=1}^{m} e^{g_i(\mathbf{x})}} \tag{1.12}$$

Such a network can be trained using a standard approach, where the loss function can be composed of the difference between the predicted probability vector and the true probability vector. The true probability vector is a null-vector with a value of 1 at the index corresponding to the correct RUL value. For example, if the true RUL of the vector shown in Equation 1.11 were to be 2, the vector that the network would use for training network would be $\begin{bmatrix} 0 & 0 & 1 & \dots & 0 \end{bmatrix}^T$. This approach is used in [34], where the authors use a neural network for classification of Alzheimer disease among patients. Here, the authors use the output of the softmax layer directly as an estimator of $P(Y = i|x)$, where Y is the network output and x the network input. Only 2 output neurons are used, one corresponding to "Alzheimer disease", and one corresponding to "no disease".

A different approach is to let the output neurons of the NN be the parameters of a statistical distribution. In the blog post of Shaked Zychlinski [35], he attempts to incorporate neural network uncertainty by letting the output neurons of the network be the parameters of a Normal distribution, μ and σ. This is a special type of NN, called a Mixed Density Network (MDN). First described in 1994 by C. M. Bishop [36], MDN's do not predict the expected value of the target, but also estimate the underlying probability distribution. A MDN can be interpreted as a mixture of m kernel functions, where a kernel function $\phi(\mathbf{t}|\mathbf{x})$ is the conditional density

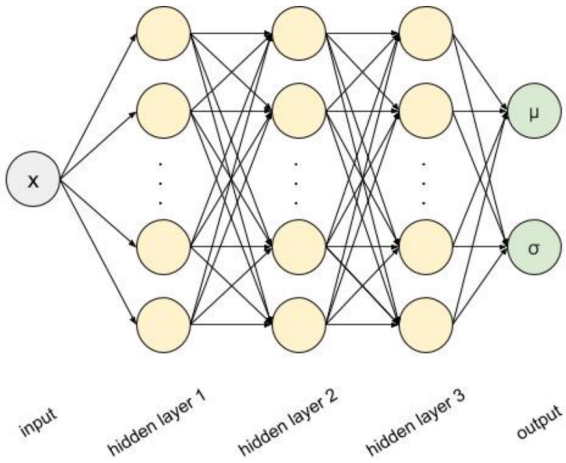


Figure 1.15: Architecture of a neural network where the output neurons represent the parameters of a Normal distribution [35]

of the target vector **t** given the input vector **x**. In [36], the discussed kernel function takes a Gaussian form according to the equation below:

$$\phi(\mathbf{t}|\mathbf{x}) = \frac{1}{(2\pi)^{c/2}\sigma(\mathbf{x})^c}\exp\left\{-\frac{(t-\mu(\mathbf{x}))^2}{2\sigma(\mathbf{x})^2}\right\} \tag{1.13}$$

A mixture of m of those kernel functions can be created by weighting those function with a coefficient $\alpha(\mathbf{x})$. The probability density of the target data can then be computed as follows:

$$p(\mathbf{t}|\mathbf{x}) = \sum_{i=1}^{m} \alpha_i(\mathbf{x})\phi_i(\mathbf{t}|\mathbf{x}) \tag{1.14}$$

Using this mixture model, any arbitrary conditional density function can be computed. By letting m be sufficiently large, and by making a NN with sufficient hidden neurons, any conditional density function can be approximated by an MDN [36].

The method in [35] uses the network architecture shown in Figure 1.15 to estimate the function $g(x) = f(x) + \epsilon(x) = x^2 - 6x + 9 + \epsilon(x)$, where $\epsilon(x)$ is a Normally distributed noise term that increases with increasing value of x. The network is trained using input values of x, and corresponding output targets sampled by $f(x)$. Using this network with output parameters μ and σ, it is tested whether the network indeed estimates higher uncertainty for increasing values of x (as the noise term is higher in the training data for higher input values). The model is trained using the negative log likelihood of the Normal distribution as loss function. In this way, by minimizing this function, the model is thought to predict the Normal distribution parameters μ and σ that given an input x, output variable y is the most likely outcome. This method is also proposed by [36].

The results in [35] are very satisfactory: not only does the network learn to almost perfectly predict the function $f(x)$, it also outputs higher uncertainty estimates for increasing vales of x due to the added noise term $\epsilon(x)$ that increases with x.

[37] aims to predict traffic flow over time. In this blog post, the author uses the parameters of negative binomial distribution as network output, n and p. The model itself is a LSTM sequence to sequence model that was previously developed for the same traffic flow prediction problem. In this article, the author only modifies the output layer to make the network return probabilistic distributions. Again, a custom negative log likelihood loss function was implemented specific to the negative binomial distribution. The obtained binomial distribution was used by the authors to construct confidence intervals around the model predictions, as can be seen in Figure 1.16, where several colors indicate a variety of confidence levels based on the properties of a binomial distribution. From this it becomes apparent that probability distribution outputs can be used as a degree of uncertainty in a neural network.

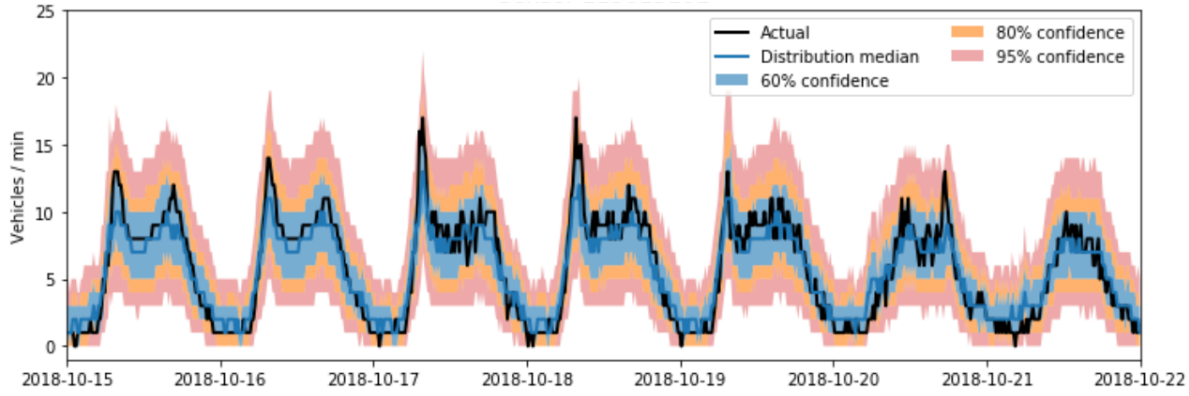


Figure 1.16: Output of the neural network created in [37]: traffic flow over time including confidence intervals based on a negative binomial distribution

1.5.2. Estimating neural network output uncertainty

Output probability distributions can be directly used as a degree of uncertainty in the network. For example, the output parameter σ in the network shown in Figure 1.15 contains information about how certain the network is on its mean prediction μ. More methods exist that are able to quantify the uncertainty of a NN prediction. The methods that will be discussed in this section are Bayesian NN's (BNN) and Monte Carlo Dropout (MCD).

A Bayesian NN is a stochastic NN, meaning that stochastic elements are introduced into the network. In a Bayesian NN, the network weights are not simple deterministic values but a probability distribution is learned over the weights. Suppose A is the prior distribution of the weights and biases in the network, and B is the training data in the form of input-output pairs. The goal of a Bayesian neural network is then to find the the posterior distribution $P(A|B)$ over the weights and biases that makes the output of the network most likely given the input. Given the stochastic nature of these networks, a Bayesian NN can be regarded as a special case of ensemble learning. The idea behind ensemble learning is to train several models and average the predictions made by all models. Averaging the ensemble of independent estimators is found to yield superior prediction compared to training only a single highly trained expert model [38]. Sampling from a Bayesian NN can be done by given a certain input multiple forward passes through the network. Due to the posterior distributions that are computed over the weights and biases, the network output will differ with each forward pass, resulting in a sampled set of outcomes. These outcomes can be used to improve prediction performance by averaging the output results and by computing the variance of between all outputs, giving an indication of model uncertainty. In [39], uncertainty in a BNN is further subdivided into aleatoric (statistical) uncertainty and epistemic (systematic) uncertainty, where both components are estimated using the properties of the BNN posterior distribution.

More recently, a different approach to estimate model uncertainty was proposed. In [40], the authors show using an extensive mathematical proof that the widely used technique of dropout can be used during testing of the model to approximate a Gaussian process. This leads to the conclusion that this way of applying dropout is another approach to obtain probabilistic outputs from the network. The authors mention that one of the main applications of this result is that using dropout can now be used during testing of the model to sample multiple outputs for a single input. The authors refer to this concept as Monte Carlo Dropout (MCD), which can be used to quantify model uncertainty. [41] follows this work by proving that MCD can also be used in CNN's by applying dropout after each convolutional layer as well as the inner-product layers. Using MCD in a CNN, it was shown that the model outperformed models that only use regular dropout during training.

[42] is a very recent work that describes how MCD can be used for uncertainty estimation. Using MCD, a the function $y = x^3 + \epsilon$ is approximated, where $\epsilon \sim \mathcal{N}(0,9)$. It is indeed found that within the range of the training input data $x \in [-4,4]$ the model outputs very accurate predictions with low uncertainty estimates. For the range of input value the model was not trained on ($x < -4, x > 4$), the model performs poorly and outputs very large uncertainty bounds. The paper furthermore experiments with several model inputs, such as dropout rate, activation function, model precision and number of training epochs to see what the effect on the predicted uncertainty is. Also, the network using MCD is compared to a standard dropout network where dropout is only using at training time. It was found that the MCD model did not improve the prediction

performance for several data sets. Therefore, this paper concludes that MCD is a great tool to use to include in a network for uncertainty analysis, but not so much for improving the network prediction accuracy.

In [43], the authors use MC sampling to fully model all sources of prediction uncertainty in their network, including sensor noise. Their approach is used to quantify two types of uncertainty: data uncertainty, e.g. sensor noise due to sensor imperfection, and model uncertainty, e.g. uncertainty introduced due to imbalances in the training data. In this paper, MCD is applied by keeping dropout on during test time and sampling the same input through the network several times. This approach, compared to equivalent methods of BNN's, is easy to implement and does only require the model to be trained once. Data uncertainty is estimated using Assumed Density Filtering (ADF), where the network activation, inputs and outputs are represented by a probability distribution. This modifies the network to not only output predictions μ, but also the respective data uncertainty v. The ADF propagates the initial sensor noise $v = v^{(0)}$ through the network to represent input uncertainty. Using MCD, an input sample if fed through the network T times. The model architecture can be seen in Figure 1.17. A single input image with noise characteristic $v = v^{(0)}$ is fed into the model that at each node computes the node value along with an uncertainty estimate. This procedure is done T times, such that T output pairs (μ_T, v_T) are sampled. The final prediction of the network given an input x is then given as the the mean value of all μ_T. The network uncertainty depends on the average noise propagated through the network and the distance of the prediction to the mean prediction, as can be seen in the bottom right of Figure 1.17. Lastly, an important finding is that the optimal dropout rate for MCD during testing is the same dropout rate that is applied during training [43].

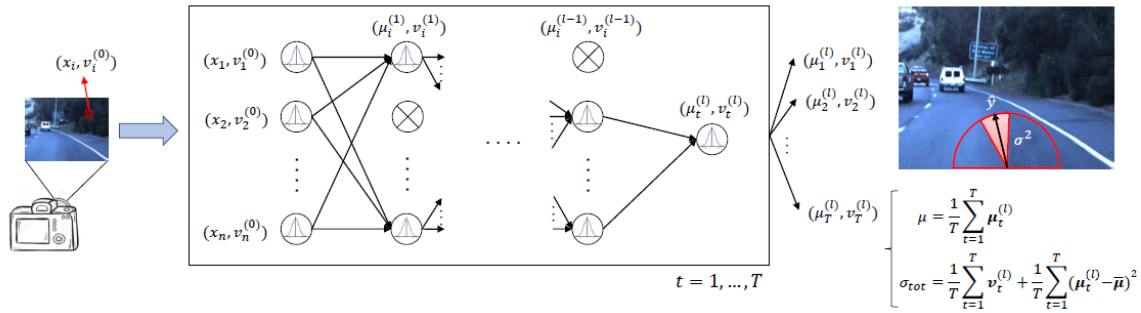


Figure 1.17: Overview of the model structure developed in [43]. This model predicts uncertainty using both a data noise characteristic and Monte Carlo Dropout sampling

[44] uses a very similar approach where they attempt to create an uncertainty map for a fully CNN for colorectal polyp segmentation using MCD. In this paper, the authors use T forward passes through the network, after which they apply a softmax function to the sampled output. The obtained values are averaged to obtain the posterior output distribution. Using this method, the authors find superior results compared to previous methods with very useful uncertainty maps that indicate at what locations the model is not fully sure whether or not a polyp is present in an image.

Lastly, [45] present a very easy to understand description for the implementation of MCD in Python using the Keras API. This post clearly describes how to pick the optimal dropout rates, and how MCD can subsequently be used to construct confident intervals around a prediction. Here, the input data is very similar to Figure 1.16. In as similar fashion as was used to create the output in this image, the author uses 20 forward passes through the network for each input with a dropout rate of 0.375. The results are similar to the output shown in Figure 1.16, where satisfactory confidence bounds that posses information about network uncertainty are plotted around the input data.

A simple way in which all of the above information can be used for engine RUL prediction and uncertainty estimation is as follows. A NN approach similar to the approaches shown in Table 1.3 could be used, but now the output neurons represent the parameters of a distribution, similar to the architecture shown in Figure 1.15. For example, for a normal distribution μ is the indication of the model RUL prediction, while σ reveals information on how certain the model is in its prediction. Then MCD can be used to sample T of those normal distribution outputs by keeping dropout on during test time and feeding a single input into the network T times. This is similar to the method used in [43]. These T normal distributions can then be mixed into a single output posterior using equal weighting coefficients. An example of this approach is shown in

Figure 1.18 and Figure 1.19. Figure 1.18 shows 4 possible output distributions of the network sampled using MCD, each with its own μ and σ. These can be mixed using equal weighting into the distribution shown in Figure 1.19. This posterior distribution shows the most accurate RUL prediction distribution from which uncertainty bounds and confidence intervals can be constructed. During the continuation of this study, the question on how those output distributions can be optimally combined for uncertainty analysis will be further examined.

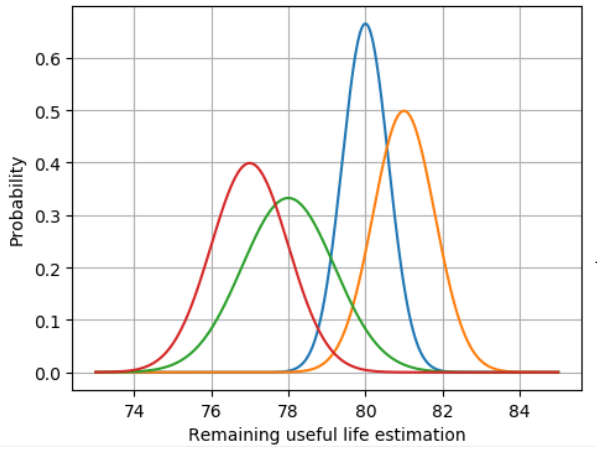


Figure 1.18: An example of 4 normal output distributions (probability density functions) sampled using Monte Carlo Dropout

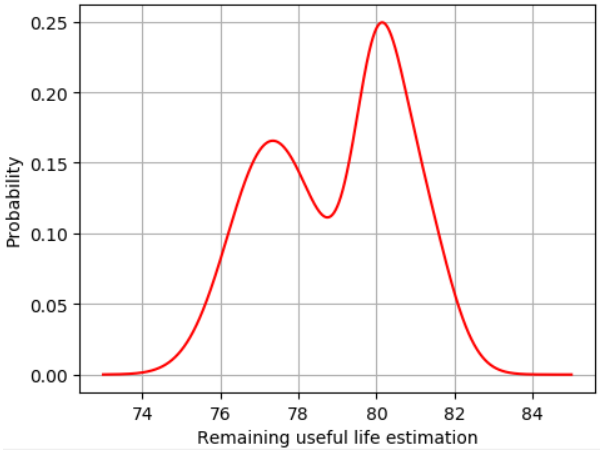


Figure 1.19: A single output distribution obtained by computing the weighted average of the 4 distributions shown in Figure 1.18

1.6. Prognostic Algorithm Performance Assessment

Prognostic algorithms (PA) aim to predict the health state and RUL of aircraft components. Faulty predictions may have catastrophic conditions, when for example it were to occur that a component would fail during flight while the PA estimated the component to be healthy. However, in RUL estimation various sources of uncertainty are present, such as future operating conditions, the component environment, errors in the PA and data noise [7]. These reasons impose strong requirements on the assessment of PA's before they can be used in practice. Therefore it is of vital importance to establish a wide source of metrics that asses a PA on all aspects. In this study, only metrics related to accuracy, reliability and safety are considered. Quantities that can be used to research economic feasibility of a PA are not taken into account in this research. This chapter first looks at general error metrics that are commonly used in combination with machine learning techniques. After, currently used metrics specific to the assessment of prognostic algorithms are introduced. As some PA's do not have deterministic outputs, it is also examined how uncertainty estimates can be included in the algorithms accuracy and reliability assessment.

1.6.1. General error metrics and machine learning performance indicators

This section covers a variety of general metrics that can be used to evaluate the performance of a series of predictions. This includes both metrics that directly relate an output value to a true value, and metrics that are commonly used with machine learning classification tasks.

A variety of standard metric used to assess the accuracy and precision of an algorithms output can be found in Table 1.4. This table presents the definition of the metric, as well as the range of values the metric can take. Also, references that provide further information on that specific metric are included. Over time, some metrics have become more popular than others. For example, in 1982 the RMSE was the most used metric with a use rate of 34%, while the MAPE was only used 15% of the time. More recently, in 2007 it was found that the RMSE was still used only 9% of the time, while the MAPE is now the most used metric with a use rate of 44% [46]. In Table 1.4, r_{true} is the correct true value, while r_{pred} is the predicted value. Furthermore, the symbols used for defining false positives and false negatives need to be explained. A false positive is defined as a prediction that is earlier than the true value minus some sort of early prediction threshold value. For example, if the true RUL is 70 and the value of t_{FP} is set at 10, than all predictions that estimate a RUL smaller than 60 are classified as false positives. On the contrary, false negatives deal with very late predictions. If the value of the threshold parameter t_{FN} is 10, then a RUL estimation larger than 80 is labeled as false negative.

Note that the values of t_{FP} and t_{FN} do not have to be constant. In RUL prediction, it might be beneficial to increase the required prediction accuracy over time, and the values of t_{FP} and t_{FN} can then for example be a function of the true RUL.

When testing an algorithms with many test units, the total number of false positives (FP) and false negatives (FN) can be computed. This information, in addition to the correctly identified true positives (TP) and true negatives (TN), can be visualized using a confusion matrix. Furthermore, several metrics can be defined using those four classes. [47] defines the following metrics:

$$\text{accuracy} = \frac{TP+TN}{N} \quad (1.15) \qquad \text{precision} = \frac{TP}{TP+TN} \quad (1.16) \qquad \text{recall} = \frac{TP}{TP+FN} \quad (1.17)$$

Table 1.4: General metrics used for the assessment of accuracy and precision as stated in [48]

Metric name	Definition	Range	References
accuracy based metrics			
Error	$\epsilon(i) = r_{\text{true}}(i) - r_{\text{pred}}(i)$	(-∞,∞)	-
Mean Absolute Error	$MAE(i) = \frac{1}{L}\sum_{l=1}^{L}\lvert\epsilon^l(i)\rvert$	[0, ∞)	[49]
Mean Absolute Percentage Error	$MAPE(i) = \frac{1}{L}\sum_{l=1}^{L}\left\lvert\frac{100\cdot\epsilon^l(i)}{r_{\text{true}}}\right\rvert$	[0, ∞)	[49] [50]
Mean Squared Error	$MSE(i) = \frac{1}{L}\sum_{l=1}^{L}\epsilon^l(i)^2$	[0, ∞)	[49]
Root Mean Squared Error	$RMSE(i) = \sqrt{\frac{1}{L}\sum_{l=1}^{L}\epsilon^l(i)^2}$	[0, ∞)	[49]
Root Mean Squared Percentage Error	$RMSPE(i) = \sqrt{\frac{1}{L}\sum_{l=1}^{L}\left\lvert\frac{100\cdot\epsilon^l(i)}{r_{\text{true}}}\right\rvert}$	[0, ∞)	[49]
False Positive	$FP(r^l_{\text{true}}(i)) = \begin{cases} 1 & \text{if } \epsilon^l(i) > t_{FP} \\ 0 & \text{otherwise} \end{cases}$	[0, 1]	[51]
False Negative	$FN(r^l_{\text{true}}(i)) = \begin{cases} 1 & \text{if } -\epsilon^l(i) > t_{FN} \\ 0 & \text{otherwise} \end{cases}$	[0, 1]	[51]
precision based metrics			
Error Standard Deviation	$S(i) = \sqrt{\frac{\sum_{l=1}^{n}(\epsilon^l(i)-M)^2}{n-1}}$	[0, ∞)	[52] [53]
Mean Absolute Deviation	$MAD(i) = \frac{1}{n}\sum_{l=1}^{n}\lvert\epsilon^l(i) - Md\rvert$	[0, ∞)	[53]
Median Absolute Deviation	$MdAD(i) = \text{median}(\lvert\epsilon^l(i) - Md\rvert)$	[0, ∞)	[53]

Another metric that can be used is the F1-score which is computed as the harmonic mean between the precision and the recall [47]. Tuning a prognostic algorithm might also be done usig a ROC (Receiver Operating Characteristic) curve. Using the FP and the FN, this plot is created by plotting the FP rate on the x-axis and 1-FN rate on the y-axis. A perfect algorithm would have 0 FP and 0 FN, and would therefore be located in the top left corner of this plot. This plot can be used to trade-off FP and FN to tune the algorithm to required performance[48].

The final method of algorithm reliability evaluation is the reliability diagram [48]. This diagram plots the observed frequency of an event against the computed probability of that event. The occurrence of the event is predicted T times and the range of probabilities is divided into k bins, for example like [0-5%, 5-15%, 15-25%...]. Suppose n_k out of N times the predicted probability of an event false within bin k that has a center value of p. Out of n_k predictions in that bin, the event actually occurs m_k times, such that the observed frequency of the event o_k is equal to m_k/n_k. The values of o_k can then be plotted against p to obtain the reliability diagram, where the diagonal represents perfect forecasting. Deviation from the diagonal can be computed using the Brier score that is defined mathematically below [48]:

$$BS = \frac{1}{K}\sum_{k=1}^{K}(p_k - o_k)^2 \tag{1.18}$$

If $p_k > o_k$, over-forecasting occurs, and when $p_k < o_k$, under-forecasting occurs.

1.6.2. Metrics for prognostic algorithm performance assessment

This section analyzes metrics that are developed specifically for assessing the performance of prognostic algorithms (PA). The requirement for specific metrics for PA is mainly due to two factors [7]:

- In RUL prediction, a late prediction is usually considered to be worse than an early prediction. Late predictions may cause high safety risks because a component is used in highly deteriorated state, while an early prediction only leads to unnecessary maintenance causing higher costs.
- The accuracy of a prediction should increase over time. If for example the true RUL of a component is 100, and a PA predicts a RUL of 140, this is a highly inaccurate prediction (+40%) but because the true RUL is still very high no maintenance has to be scheduled yet. Therefore, this inaccurate prediction does not influence airline decision making. If the true RUL were to be 15 and the predicted RUL 21 (again +40%, so same accuracy as before), it is likely that the airline already has to make a detailed flight and maintenance schedule for the aircraft based on the RUL estimate of 21 cycles. It will cost the airline a lot of money and leads to increased risks if the aircraft then suddenly fails after 15 cycles, 6 cycles earlier then expected.

This first point is incorporated in the score used to evaluate the RUL predictions in the PHM08 Challenge. This scoring function exponentially weights estimation errors, and also punishes late predictions harder than early predictions. The scoring function is defined as follows:

$$s = \begin{cases} e^{-\frac{d_i}{13}} - 1 & \text{for } d_i < 0 \\ e^{-\frac{d_i}{10}} - 1 & \text{for } d_i \geq 0 \end{cases} \tag{1.19}$$

Where d_i is equal to the difference between the predicted RUL and true RUL. This scoring function is a specific example with coefficients (13,10) of the general timeliness metric stated in [48].

[48] proposes a set of new metric that can be used for the assessment of PA. [7] further adopts those metrics and combines the most important ones into a framework that can be used to asses the performance of RUL predicting PA's. This framework, that consists of 4 metrics and is developed by taking into account to two points on top of this page, is explained below. Note that in this section, the output RUL estimation of a PA is only a single deterministic value.

The first metric defined in [7] is the Prognostic Horizon (PH) and is visualized in Figure 1.20. This metric is equal to the time difference between the time the RUL prediction enters a specified performance criteria and the component end of life (EoL). In Figure 1.20, the black line indicates the component true RUL. The grey areas around this line represent areas of constant error which are considered acceptable upper and lower bounds for the RUL prediction. As soon as a PA outputs a RUL estimation that is within the grey area, the prognostic horizon can be computed. Note that it may occur that a prediction at a certain time is within the grey zone, but a prediction made at a later time is no longer within the grey zone. The PH can then be defined as the time difference between the time the RUL prediction is within the grey area and no longer leaves this area at a later time, and the EoL. A high value for PH indicates that there is more time to act for an airline based on a prediction within a certain level of accuracy.

The second metric of importance is the $\alpha - \lambda$ metric, and is shown in Figure 1.21. This metric is based on the point that the performance of the PA should increase over time, as predictions near component EoL are vital for airline operational planning. A specific form of this metric is $\alpha - \lambda$ accuracy, which indicates that the RUL prediction should fall between specific α accuracy bounds that get tighter around the true RUL when time progresses. In Figure 1.21, the blue line indicates the true RUL and the blue shaded area shows the α bounds that get closer to the true RUL line over time. λ is a time-window modifier that can be chosen such that a specific accuracy α is achieved at a specific time instance. For example, a requirement that can be set is that the accuracy halfway of the true RUL should be 20%. If the prediction at a certain time falls within the α accuracy bounds, the $\alpha - \lambda$ accuracy takes a value of 1, and takes value 0 if the prediction is outside the

α bounds. This metric can be used to compute what percentage of all predictions meet the $\alpha - \lambda$ accuracy requirement, which reveals information on how useful and reliable the PA is [48] [7].

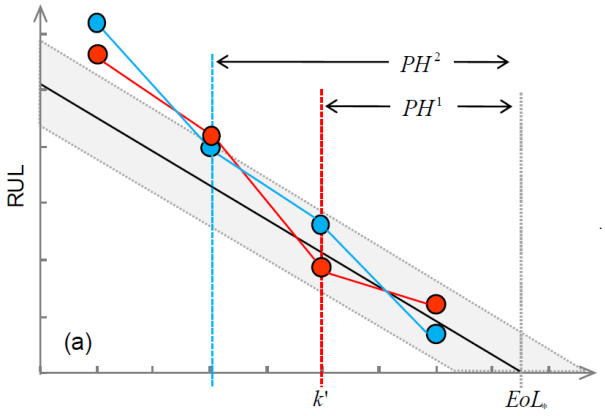


Figure 1.20: Prognostic horizon metric for point estimate [48]

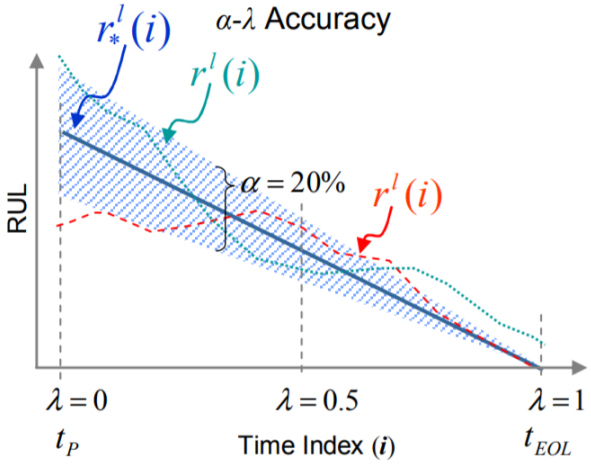


Figure 1.21: $\alpha - \lambda$ metric for point estimate [48]

The third metric in the framework is the Relative Accuracy (RA). This metric is defined as the measured error relative to the true RUL value at a specific moment in time i_λ. The RA can be computed as follows:

$$RA_\lambda = 1 - \frac{|r_{\text{true}}(i_\lambda) - r_{\text{pred}}(i_\lambda)|}{r_{\text{true}}(i_\lambda)} \tag{1.20}$$

This metric is similar to the $\alpha - \lambda$ metric, but the difference is that while the $\alpha - \lambda$ metric only tells whether a prediction in inside the α bounds or not, the RA gives a quantitative value that indicates how close a prediction is to the actual value. The RA reveals information on the accuracy at a specific time, but it might be useful to define a metric that evaluates the accuracy at multiple moments in time and aggregates those values into a single indicator. The Cumulative Relative Accuracy (CRA) is defined as the weighted sum of several RA values at different time points, normalized by the number of RA values. Mathematically, this comes down to the following:

$$CRA_\lambda = \frac{1}{|p_\lambda|} \sum_{i \in p_\lambda} w(r(i)) \cdot RA_\lambda \tag{1.21}$$

Where p_λ is the set of all time indexes at which a prediction is made, $|p_\lambda|$ is length of the set, and $w(r(i))$ are the weights based on the RUL at all time indexes [48] [7].

The final metric used in the framework developed in [7] is convergence. This metric is able to compute a single value that indicates how quickly any metric M improves with time. Figure 1.22 shows how 3 different arbitrary performance measures evolve over time. It can be seen that the blue metric does not decrease significantly, while the red and green metrics end up at a value near 0 at component EoL. The convergence metric determines how quickly each metric reached 0 by computing the center of mass of the area below the curve of each metric. In Figure 1.22, those centers of mass are the 3 colored circles. The value for the convergence metric is then the Euclidean distance between the performance metric center of mass and the bottom right point with coordinates (t_p,0). It can be clearly seen that for the red metric that converges faster than the green metric, the distance from the bottom right point to its center of mass is also smaller. A faster convergence is important to achieve high confidence predictions, and to keep the prognostic horizon high as possible.

In the very recent work of [54], the authors also look at the prognostic evaluation framework developed in [7]. The authors have found no new frameworks or other recently developed metrics for the assessment of PA's. The authors state: "..., the development of improved standardized metrics, suitable even for on-line applications, still represents a stimulating topic for the research community [54]". This indicates that the framework developed in [7] is still the most prevalent research performed on the development of new metrics for the evaluation of PA's.

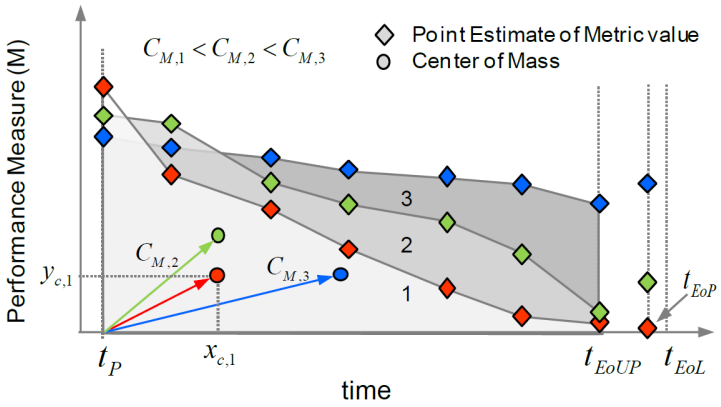


Figure 1.22: Convergence metric [48]

Besides this framework, [48] mentions two other metrics that are useful when assessing the performance of PA. The first one is called the Prediction Spread, and is defined as the variance of the predictions over time of any metric M. Secondly, the Sampling Rate Robustness (SRR) metric gives an estimate of how sensitive any metric M is to changing the data set sampling frequency. A certain sampling frequency is compared with a reference frequency that can for example be used as the recommended sampling frequency when using the PA.

1.6.3. Incorporating algorithm uncertainty estimates

The previous section only covered metrics that take as input single valued RUL predictions. However, as described in section 1.5, this research aims to predict a complete RUL output probability density function (PDF). As can be clearly seen in Figure 1.19, such an output PDF can be highly asymmetric, indicating that simply computing the mean and variance for uncertainty analysis does not suffice. For this reason, the metrics described in the previous section need to be altered such that they are able to deal with PDF estimations rather than deterministic values. An easy way of incorporating the PDF into the metrics is by defining error bounds based on the true RUL value. This method can be seen in Figure 1.23. Here, the True RUL value is marked with a white dot, and asymmetric error bounds α^+ and α^- are shown as well. These bounds are asymmetric due to the fact that late prediction is considered to be worse than early prediction. The α^+ bound is therefore located closer to the true RUL value than the α^- bound. Another example can be found in Figure 1.24, where the PDF of Figure 1.19 is shown with symmetrical bounds.

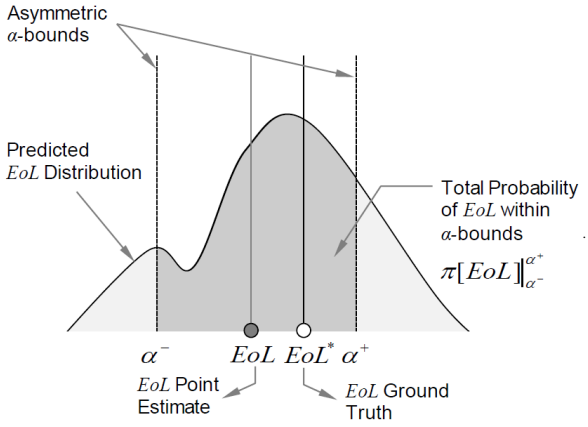


Figure 1.23: RUL PDF output with α-bounds [7]

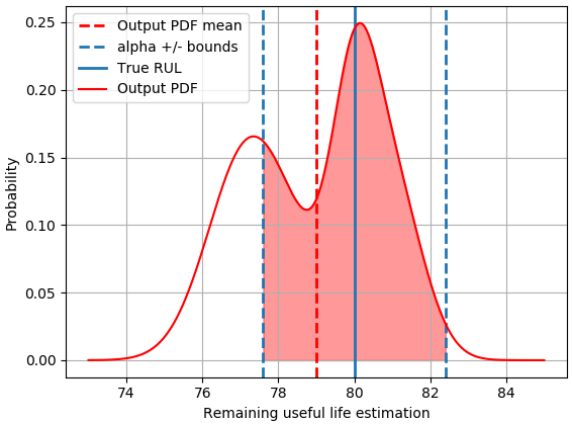


Figure 1.24: RUL PDF shown in Figure 1.19 with α-bounds

This concept can be used in combination with the previously defined metrics by computing the probability mass between the α^- and α^+ bounds. This probability mass then has to be higher than a specified level β in order for the prediction to be considered to be within the α-bounds. Mathematically, this comes down to the following:

$$\pi[r(i_\lambda)]_{\alpha-}^{\alpha+} \geq \beta \tag{1.22}$$

Here, $\pi[r(i_\lambda)]_{\alpha-}^{\alpha+}$ is the probability mass between the α bounds. Figure 1.25 and Figure 1.26 show how this method can be incorporated into the PH and the $\alpha-\lambda$ metrics. A prediction is considered to be within the grey area in both figures if the probability mass between the α bounds is larger than a specified value of β.

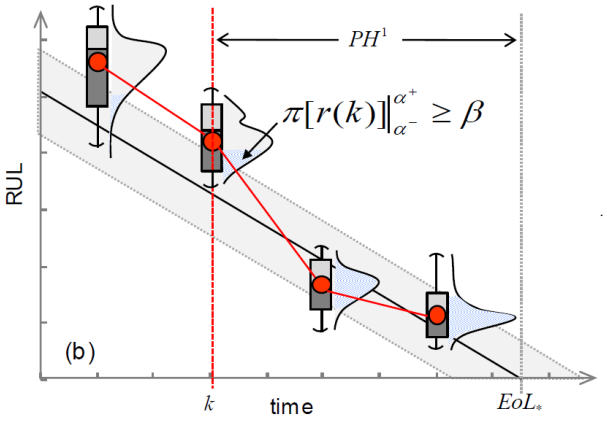


Figure 1.25: Prognostic horizon metric for PDF prediction [7]

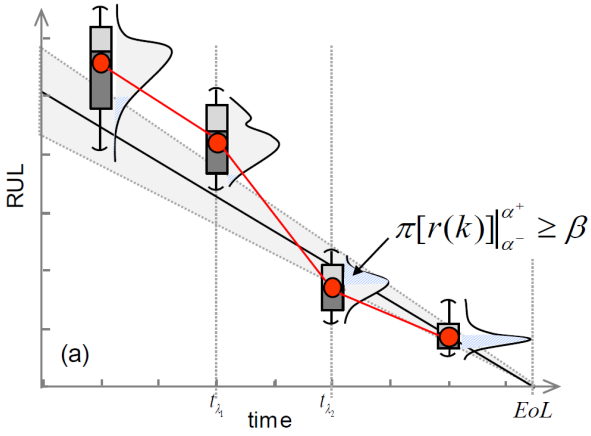


Figure 1.26: $\alpha-\lambda$ metric for PDF prediction [7]

The method of computing the probability mass between α-bounds is very useful, but has one main limitation: it reveals no information on how close the probability mass is to the true RUL prediction as it only tells how much mass it between the bounds. Two algorithms that for example both have a probability mass between α-bounds of 0.6 might perform very differently, as it might be the case that for one algorithm the mass of 0.6 is located very close to the true RUL value, but for the other algorithm it might be that the mass of 0.6 is located more towards the edge of the bounds. Also, this method reveals no information how close the remaining mass 0.4 is to the true RUL. A metric that does take these pitfalls into account is the Continuous Ranked Probability Score (CRPS) [55]. This metric computes the cumulative density function (CDF) of the RUL prediction PDF, squares this CDF and then computes the area between this squared CDF and a Heaviside step function located at the true RUL. This metric is visualized in Figure 1.27, where the CDF is computed using the PDF shown in Figure 1.19 and the true RUL value is equal to 80 cycles.

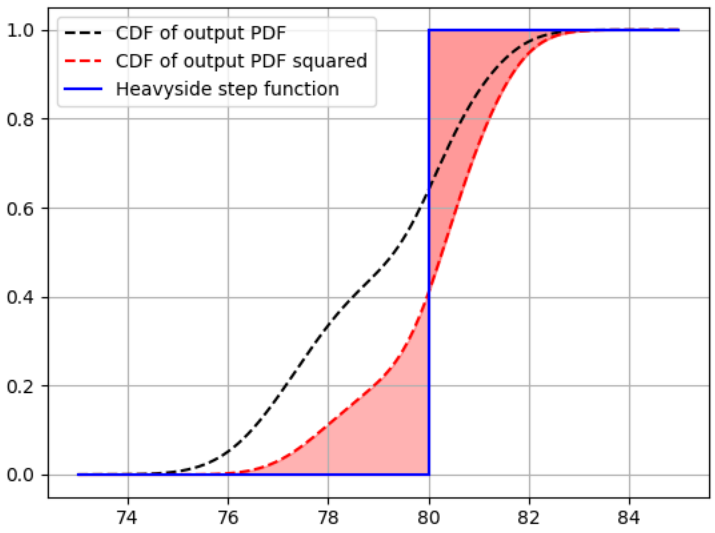


Figure 1.27: Application of CRPS metric to RUL PDF of Figure 1.19

The CRPS metric can thus be used to directly compare a probabilistic prediction to a deterministic true

value. Furthermore, the CRPS metric has the property that it is in the same unit as the observations. For a single point forecast, the CRPS reduces to simply the absolute error [55].

Lastly, [7] shares some insights on how the values of α, β and λ can be determined. The main goal of developing the performance metrics that were described in this chapter is help company management make decisions on whether or not an algorithm performs good enough on a variety of aspects to use in practice. The choice of the parameters α, β and λ plays a vital role in the outcome of those metrics, and determining the values of those inputs should thus be done with great attention. Some factors that determine how crucial it is to make a correct prediction, including the effect it has on the parameter value, are mentioned below [7]:

- Required time for problem mitigation: the time needed to mitigate an unforeseen problem, for example due to a late prediction made by the PA. This has an effect on the parameter λ.
- Cost of mitigation: cost of a corrective action. Related to the costs due to false positives (unnecessary early maintenance). Trade-off between FP and TP should be made by tuning the parameter α.
- Criticality of system/cost of failure: costs and reliability risks related to FN (late predictions) that lead to high down times, safety bottlenecks and unscheduled maintenance costs.
- Capability of managing uncertainty: deals with how well can a company deal with uncertainty and what are the costs of system failure. Determines the value of β.

1.7. Research Outline

Remaining useful life prediction for aircraft turbofan engines is not a new concept in the scientific literature. Many attempts on RUL prediction, both for turbofan engines and other components, have already been identified in this paper. Furthermore, the use of data-driven and machine learning approaches has seen a tremendous rise in use over the past decade. This research aims to continue upon the work that has been done in this field of work. Furthermore, many prognostic algorithm performance indicators have been developed over the years that indicate the accuracy and reliability of a prediction. However, no standardized framework has been developed that directly assesses the readiness of a prognostic algorithm for implementation in practice. This research also focuses on how an algorithm should be tested to ensure safe and reliable implementation in the real world. This chapter first clearly identifies the gap between the current literature and the problem at hand. Also, the research questions are mentioned, and the way those questions will be answered is discussed in the method section.

1.7.1. Research Gap

As stated, many attempts can be found in literature in which the authors try to make predictions on the RUL of a turbofan engine. Furthermore, the concept of machine learning techniques and the use of big data is also already applied to this problem many times. The novelty of this research is therefore not so much related to coming up with a completely new way of predicting turbofan RUL. This research focuses on combining methods of other authors that have seen great success with their approach, to reach even higher levels of RUL prediction performance. To be specific, the convolutional neural networks developed in [8] and [9] are examples of machine learning applied to the C-MAPSS data set for RUL prediction that have seen great success. Nevertheless, no further attempts on using a CNN for RUL prediction has been identified recently, even though the authors obtained very good results compared to similar approaches. Furthermore, besides the promising concept of only using raw sensor data as input to the CNN, methods of health indicator extraction based on the observed patterns in the physics of the engine could also leverage superior results [14] [18]. Therefore, this research aims to **combine both approaches by using a CNN that not only takes raw sensor data as input, but also uses pre-computed HI information for better RUL performance**.

Furthermore, all approaches to RUL prediction using the C-MAPSS data set build their model to output only a single deterministic RUL value prediction. For implementation of such an algorithm in practice, it is of vital importance for an algorithm to not only make accurate predictions, but also provide information about the uncertainty of the prediction that it made. This can for example be done by letting the output of your model be a probabilistic prediction instead of a single value. Also, sampling methods can be used to construct confidence intervals around the model predictions. Those methods are not used in any other turbofan RUL prediction papers, and therefore this research also attempts to **include uncertainty estimates in the form of probabilistic distributions for the RUL prediction of an engine**.

Lastly, although [7] defines a framework for prognostic algorithm performance evaluation, no standardized framework exists that directly assesses the readiness of a prognostic algorithm for use in practice. Therefore, this research will combine a set of currently available metrics and predefined threshold values, as well as newly developed metrics, to **create a general framework that assesses the reliability and application readiness of any prognostic algorithm with distribution output**.

1.7.2. Research Questions

The key research questions of this research are presented in this section in bold. Below each key question, a list of corresponding sub-questions is given.

How can machine learning be used on sensor measurements of an aircraft turbofan engine for remaining useful life prediction?

- What aircraft turbofan engine data can be used for health monitoring and remaining useful life prediction?
- What turbofan engine sensors are most useful when predicting remaining useful life?
- How can turbofan engine sensor data be used to construct a health indicator that shows the degradation level at any time?
- How can turbofan engine raw sensor data and pre-computed health indicators be fused into a convolutional neural network for remaining useful life predictions?
- How can a probabilistic output distribution be obtained from a neural network?
- How can neural network uncertainty estimation be included in the model output?

How can the performance and readiness of an aircraft component prognostic algorithm be assessed such that the algorithm can be used in real-world applications?

- What metrics and frameworks currently exist that evaluate the performance of prognostic algorithms?
- Based on airline operational performance, how can suitable threshold values for prognostic algorithm performance metrics be determined?
- What new metrics can be developed for prognostic algorithms that quantify the readiness for implementation in practice?
- How can both existing and newly developed metrics be fused into a general framework that airlines can use directly for making decisions on algorithm implementation?
- What are the remaining obstacles towards implementation of prognostic algorithms in practice?

1.7.3. Method

This research will focus on creating a convolutional neural network that takes as input raw sensor data. Furthermore, HI extraction models will be developed that will serve as additional inputs to the CNN. While developing the CNN architecture, close attention will be paid to previous related works in order to use the aspects that have been proven to work best, and to avoid the errors that already have been made. This will result in a network that is more advanced than other CNN's used for the problem of turbofan engine RUL prediction. Additional NN uncertainty estimation methods will be used to quantify the confidence of the model predictions. It will be attempted to make the networks output neurons the parameters of a statistical distribution such that confidence intervals can be computed. Also, dropout will be used during training to prevent model overfitting, and dropout will be used during testing to sample a variety of output distributions for a single input. These output distributions can then be mixed into a single PDF that reveals vital information on the confidence of a model prediction. Lastly, it will be attempted to construct a general framework that can be used to quantify the readiness of any prognostic algorithm for implementation in practice. This will be done by selecting and creating the most informative performance metrics that can be used to see how

well an algorithm is doing. Ideally, a single program is developed that takes as input the true RUL values and predicted output distributions of a prognostic algorithm, and returns how reliably this algorithm can be used in practice. Also, this model should return an indication on where in the system reliability bottlenecks might arise. Using this tool, aviation company decision makers should be able to make choices on whether or not a certain prognostic algorithm is suitable to use within their company. This program will be created by closely comparing prognostic algorithm output and true RUL values, and by implementing information on the company requirements. For example, some companies might have very strict requirements in term of how long in advance they plan their maintenance operations. For such a company, a much higher RUL prediction accuracy is required at higher RUL values. All of this information has to be fused into a single, ready-to-use decision making tool that aids in the implementation of prognostic algorithms in a reliable way.

1.8. Conclusion

In recent years, more and more attention has been paid to incorporate predictive maintenance actions into airline operations. Predictive maintenance has the potential to reduce operational costs and aircraft downtime, while increasing reliability and safety. For those reasons, prognostics and health management of aircraft components has been widely studied in literature. One of the key concepts of prognostics is predicting the remaining useful life of a component. This prediction is vital for determining when maintenance is to be scheduled for an aircraft, and thus also plays an important role in the entire scheduling problem of an airline. This study aims to use novel machine learning techniques to make high quality remaining useful life predictions on one of the aircrafts most interesting and complex components, the turbofan engine.

This report first mentions general methods that can extract health indicators and predict the remaining useful life of several components. With the main focus being data-driven degradation prediction, it was found that many methods currently exist that can predict component remaining useful life with satisfactory accuracy. Next, a widely used turbofan degradation data set is described to give insight in how component sensor data can be presented and how it can be used to make predictions. A variety of machine learning techniques applied to this data set was discussed, and their performances were analyzed. The application of Convolutional Neural Networks was analyzed in more detail, as it was found that this technique is not currently a topic that has received much attention even though multiple studies found this method to yield promising results.

Uncertainty in neural networks was analyzed by looking at how a model can output parameters that reveal information on the confidence on its predictions. It was found that a possible way of including uncertainty estimation is to let the network output neurons be the parameters of a statistical distributions. Also, dropout can be kept on during testing time such that a single input can be fed through the network multiple times and different outputs can be obtained. This method is called Monte Carlo Dropout and is equivalent to approximating a Gaussian process. Combining both methods can lead to a model output that not only attempts to make remaining useful life predictions, but also includes a measure of the prediction uncertainty.

At the moment, no standardized framework for the performance assessment of prognostic algorithms exists. A variety of prognostic algorithm performance metrics exist, but no method has been developed that combines that application of those metrics into a decision-making tool that can directly be used by aviation companies to check if a prognostic algorithm can be used in practice. Metrics specific to remaining useful life are included by introducing the fact that late prediction are usually worse than early predictions, and that the accuracy of an algorithm should increase over time. Those metrics suffice when simply comparing the performance of algorithms with each other, but can not yet be used to directly quantify the readiness of an algorithm for use in practice.

Taking the above results into account, it becomes apparent that although extensive research has been done on turbofan remaining useful life prediction, gaps still exist between literature and algorithm implementation. Therefore this study will focus on 3 main aspects: (i) improving current remaining useful life algorithm prediction performance by using a Convolutional Neural Network that includes physics based Health Indicator information, (ii) include measures of uncertainty in the model prediction, and (iii) develop a standardized framework that assesses the readiness of a prognostic algorithm for implementation in practice.

III

Supporting work

1

Verification and Validation

This chapter will elaborate on how verification and validation were applied throughout this research. Verification and validation are of great importance to ensure that the results of obtained during the research are reliable and can be used to set-up a data-driven maintenance decision framework in practice. Verification and validation for simulation models was extensively researched by Sargent, whose work will be the focus of this chapter when discussing how verification and validation was applied in this work [56].

This chapter is laid out as follows: first, section 1.1 covers the general verification and validation strategy applied in this research. Next, section 1.2 describes how model verification is performed. Lastly, section 1.3 discusses the model validation process.

1.1. Verification and validation strategy

Before the specific forms of verification and validation can be discussed, it is first important to introduce the definitions of verification and validation used in this model. These definitions are adopted from the work of Sargent, as this research is specific to simulation models. The definitions as used in this research are presented below:

- **Model verification**: 'ensuring that the computer program of the computerized model and its implementation are correct' [56]
- **Model validation**: 'substantiation that a model within its domain of applicability possesses a satisfactory range of accuracy consistent with the intended application of the model' [56]

From these definitions, it can be concluded that model verification focuses on the correctness of the implementation of the model, while model validation focuses on how suitable the model is with respect to its intended application. Verification is an internal process, as it looks at the correctness and condition of the model itself. Validation on the other hand is an external process, as it looks at how the model and its results relate to the needs of the customer and other identified stakeholders [57].

Sargent introduces three different decision-making approaches that can be used to determine if a model is valid. The first one is to let the model developer determine the model validity itself. This approach is not recommended, as the model developer might not always be fully aware of all the specific requirements posed by the customer. Secondly, the users of the simulation model can determine the model validity. This approach is thought of to be better as this approach allows for more credibility as it ensures that the users are actively engaged in the model development. Lastly, model validity can be monitored by a third, independent party. The Independent Verification and Validation (IV&V) team is a separate party that is required to have a solid understanding of the model that is being developed. Conducting the IV&V can both occur during the model development phase, as well as after the model has been created. The method of IV&V also leads to a higher level of credibility, as other parties are more likely to accept that the model is valid if an independent party has come to that conclusion [56].

In this research, model verification and validation is done by the model developer. This is done because the model that is developed at this stage is mainly a theoretical model with no specified user or customer.

Therefore, assessing the validity can not be done by an user. Furthermore, as the size of the model is relatively small and no other parties are involved, IV&V is not applicable when taking into account the scope of this research. For those reasons, verification and validation are performed by the model developer.

Assessing the overall model validity will be done by looking into the model verification and model validation separately. Model verification will focus on the input data is used in the model and on the specific implementation of the functions used in all parts of the model. Model validation will discuss the conceptual validation of the model, as well as the operational validation. Note that the model refers to both the prognostic model and the maintenance simulation model combined, meaning that the whole simulation process from raw engine sensor data up to maintenance scheduling results is being verified and validated.

1.2. Model verification

For this research, two types of verification are applied. The first one is model input verification, in which the correctness of the input data that is being used to construct and test the model is analyzed. Secondly, model function verification is applied by analyzing the computerized model specification and implementation of all parts of both the prognostic and the maintenance scheduling simulation model. A variety of validity tests are discussed that are used throughout the model design to minimize implementation errors and maximize internal correctness.

1.2.1. Input verification

This subsection will focus on the verification of the inputs used the model. First, C-MAPSS dataset that is used to generate the RUL prognostics is investigated [6, 13]. This dataset is of great importance to ensure model validity, as this dataset is used throughout the entire model and forms the basis of both the prognostic and the scheduling model. The C-MAPSS dataset is a widely used prognostics dataset used in many other works published in peer-reviewed journals. This creates a high credibility, as many other independent authors have approved the use of the dataset for high-end scientific research. Furthermore, the dataset is published by NASA. NASA is known as a reliable and trustworthy organization that produces high quality research. This adds confidence to the conclusion that the C-MAPSS dataset is a verified data source.

Secondly, the flight schedules used in this work can be taken a closer look at. The schedules the aircraft fly are based on real flight schedules taken from the KLM A330 fleet. This ensures that the flight schedules are very realistic and comparable to real world operations. In this research, the flight schedules of only 5 aircraft are used. This means that if a simulation is performed with more than 5 aircraft, the flight schedules will be repeated for certain aircraft, which would not be the case in practice. However, in this research, simulations are only performed with a fleet size of 5 aircraft.

Lastly, the cost input parameters are important inputs to the model as they determine the profit that is obtained using all 5 maintenance scheduling strategies. It was found to be very difficult to make an accurate estimation of all cost input values, such as for example the penalty cost of flying a FC after the target RUL. The cost input values were determined by consulting a large variety of online sources, thereby taking the most reasonable estimates as the input values. This may lead to inaccurate models inputs, as no fixed and peer-reviewed values could have been obtained. However, it is not believed that this will have a large effect on the model output. First of all, the model performance can also be assessed by only looking at the operational performance indicators. Secondly, the range of cost input value error is not believed to be large. The sensitivity analysis on the most important cost input values has furthermore shown that the profit obtained for all strategies is not sensitive to the cost input values. It can thus be concluded that the model input values contribute to high model validity.

1.2.2. Model function verification

This subsection will elaborate on how the specific functional blocks of the complete model are verified. The model is broken down into the following blocks in order to perform the functional verification: the prognostic model, the scheduling model, the optimization model and the rolling horizon model. Verification of each of those functions will be done by analyzing their implementation, the external solvers and packages that are used and their response to a variety of test cases.

Prognostic model verification

First, the prognostic model consists of all functions that transform the raw sensor data to RUL predictions. This model is first of all verified by carefully examining how the raw input data is processed and normalized. The normalization is done using default Skicit functions that are available in Python [1]. This machine learning library is commonly used in research and many other experienced programmers make use of these functions. The prognostic CNN model is implemented using the Keras deep learning API for Python [2]. This API is widely used and its application for deep learning models is considered to be verified amongst researchers. The results of the prognostic model can be verified by comparing them against similar models. The output error values are very comparable to those obtained by Li et al. who use a very similar prognostic model architecture [9]. This adds confidence that the model was implemented correctly. Furthermore, a variety of test cases were applied to the developed model. The first test is an extreme test, where the input to the model is set to all zeros. It was found that the CNN model was not able to be trained properly and just returned an output of zero as well, which is what is to be expected. Another test is the comparison to other models as stated by Sargent, in which the results and implementation of the model are compared with similar models [56]. It was observed that the results of the developed CNN model were in the same range as those of other comparable works found in literature. Lastly, smaller tests were performed that checked the overall correctness of model implementation, such as varying some input values along a specified range, tracking the impact on the output values of structural changes, implementing checkpoints that break the code if certain parameters attain values that do not make sense and by constantly monitoring the overall logic of the implemented code. This process is comparable to what Sargent refers to as a structural walk-through, but in this work the verification is done by the model developer himself. This entire process ensures the validity of the prognostic model implementation.

Scheduling model verification

The second part of the overall model is the scheduling model. This model includes all functions that transform the probabilistic RUL predictions obtained using the prognostic model to the target RUL values for each engine. The verification of the functions in this model was done similarly to the prognostic model. Throughout the code, multiple breakpoints were included that are able to break the code if an error occurs. Also, multiple print statements and output variables are used that are able to reveal information on the values of certain parameters at different stages of the computations. In this way, the state of the model at all stages was monitored in order to ensure that the model was implemented correctly. Extreme testing was performed to ensure that a zero input would result in a zero-valued output. The process of computing the target RUL for each engine was monitored by investigating the attributes of the aircraft and engine objects. After each week of simulating maintenance scheduling, the attributes were checked to see if the values of the target RUL, the scheduled moment of maintenance and the number of FC an engine is used are all updated in a logical and correct manner.

Optimization model verification

The optimization model contains all functions that select the optimal maintenance slot for an engine that required maintenance in the next optimization period. The first part checks for each engine if the target RUL is within the next optimization period. This process was checked manually by carefully monitoring the state of the engine's attributes and by printing the model outputs while running the simulation. The optimization model itself is written using Gurobi Optimization in Python [3]. Gurobi is an industry solver that is used for a large variety of optimization problems. This solver is considered to be accepted in academic research and is used by many industry leading companies and universities. It can therefore be concluded that the use of this solver contributes to the model verification.

Rolling horizon model verification

Lastly, the rolling horizon model is a combination of the the aforementioned models such that the simulation can be performed for a long period of time. The rolling horizon model consists of two alternating stages, namely the optimization period and the realization period. During the optimization period, the ILP optimization model is used to determine the optimal maintenance slots for all engines that might need maintenance.

[1] https://scikit-learn.org/stable/
[2] https://keras.io/
[3] https://www.gurobi.com/

Then, during the realization period the flight cycle is realized and the degradation information is obtained. Verification of this functional block is done by investigating the model input-output relation, meaning that the outputs of the model were carefully monitored and investigated when the model input is changed. Furthermore, the aircraft and engine object attributes were monitored to ensure that logical values were obtained after each rolling optimization and realization period. Higher code validity is obtained as the code for this part was partially obtained from Boekweit, meaning that more than one developer has looked at and has verified the implementation [58]. This adds confidence to the correctness of the model implementation.

1.3. Model validation

This section will cover the validation of the developed model. This will be done by both looking into the conceptual validation of the model and by looking into the operational validation. Again, the work by Sargent will be used as guideline to perform the validation of simulation models [56].

1.3.1. Model conceptual validation

Conceptual model validity determines that (i) the theories and assumptions underlying the conceptual model are correct and (ii) the model's representation of the problem entity and the model's structure, logic, and mathematical and causal relationships are 'reasonable' for the intended purpose of the model [56]. The main assumptions and theories used in this research are already introduced in part 1 of this thesis. Therefore, this section will mainly focus on whether or not the representation of the model is accurate enough for the intended purpose of this research. This will also include a discussion on how closely the model resembles reality under the assumptions included in this work.

The purpose of the developed model is to show how raw sensor recordings can be used in a daily scheduling routine. This purpose can be further broken down into smaller sub-goals. First, the a sub-goal is to transform sensor recordings into probabilistic RUL predictions at any stage of an engine's life. Secondly, the goal is to show how these data-driven RUL predictions can be adopted in the daily scheduling routine of an airline. It can thus be concluded that the overall goal is to contribute to the available knowledge on data-driven maintenance scheduling. When performing conceptual validation, it should thus be investigated whether or not the model that is developed has a structure and logic that is reasonable for this intended purpose.

The logic and structure of the model comply with the purpose of generating novel knowledge on how data-driven RUL prognostics can be incorporated into the daily scheduling routine. The model is structured to mimic a realistic scheduling scenario, in which aircraft fly real flight schedules. The model's logic can be summarized as follows: using realistic aircraft engine degradation data, predict using a widely used CNN model the RUL distribution of the engine. Then, use this RUL prediction to determine the moment in time maintenance is needed for that engine. Knowing the moment of required maintenance, schedule the aircraft containing the near-failure engine in a suitable opportunity. Repeat this logic for as many realization periods as desired using the rolling horizon model. This logic completely complies with the intented purpose of this research.

The main theories used in this model will be briefly discussed in order to prove their contribution to the overall model validity. First, the model makes use of a CNN to make RUL predictions. By historical comparison, it can be proven that this type of prediction model is widely used in academic literature, both for RUL prediction and for other regression problems [8, 9]. Furthermore, an ILP model is used for the optimization of the maintenance scheduling. This theory suits the model well, as this ILP model considers only 2 options for each decision: either one does schedule maintenance for engine i of aircraft k at opportunity t, or one does not. This binary decision making makes the ILP model very well suited for this type of scheduling model. It can thus be concluded that the theories used in this model contribute to high model validity.

Lastly, it is important to analyze the main assumptions made when developing the model. This paragraph will predominantly focus on how the assumptions affect the intended purpose and how different the model is when comparing it to a real world scenario. First, it can be noted that the assumptions made in this work are perfectly in line with the intended purpose of the model. Once again, the main purpose of the model is to show how data-driven RUL prognostics can be used to schedule engine maintenance for an airline. The assumptions made do not interfere with this goal. For example, in reality an aircraft undergoes maintenance on more than just its engines. Therefore, maintenance actions are usually grouped to minimize aircraft downtime. However, assuming that the aircraft only requires maintenance on the engine as is done in this research does not affect the purpose of finding out how data-driven RUL prognostics can be used to schedule engine

maintenance. Other assumptions such as that at least a period of 24 hours is required to perform maintenance also do not hinder the main purpose of the model. It can thus be concluded that even though multiple assumptions are included in the model, none of them hinder the original purpose. However, these assumptions do make the model less realistic when comparing to a real life situation. For example, other airlines will usually group maintenance actions, both for different components and for engines that are expected to fail close to each other. These actions are not considered in this work. The model thus overall resembles a very realistic maintenance scheduling scenario as it takes into account realistic flight schedules, an entire fleet of aircraft, hangar capacity, realistic engine degradation data and realistic input values, but still aspects are included that cause the model to be different than practice. However, as the main goal is to contribute to the body of knowledge on data-driven maintenance scheduling, these differences compared to practice are not important and beyond the scope of this research.

1.3.2. Model operational validation

Operational validation is determining whether the simulation model's output behaviour has the accuracy required for the model's intended purpose over the domain of the model's intended applicability [56]. For this research, this means that it should be determined if the obtained KPI's are in line with the purpose of the model when considering the use of all maintenance strategies in practice. This subsection will analyze the validity of the results using a variety of test cases that will be discussed individually.

Validity testing on the results

First, the validity of the obtained results will be tested using a variety of tests described by Sargent [56]. The first test is called a degenerate test and consists of evaluating the model's behaviour when picking the input values in a way that should find the limits of the model. For example, the number of aircraft in the simulation was increased to a high number such that it would be expected that the hangar capacity would be exceeded at every moment in time. It was then indeed found that when increasing the number of engines to a high value, many engines that are near-failure would not have a maintenance opportunity available in the next optimization period. Furthermore, it was checked that the model constraints were respected at all times. When looking into the results, no cases were found in which a single engine underwent maintenance more than once, and the hangar capacity was never violated for the default input parameters. Sargent furthermore introduces the internal validity test on the model results. In a simulation model, many MC simulations are performed. If there is a high variance between the results obtained during different MC runs, this might indicate questionable and uncertain results. For the developed model, the obtained results across all MC runs have a low variance as was shown in the table that presents the output results including the 95% confidence interval.

Sensibility of the results

This subsection will focus on whether or not the obtained output values make sense and if the output is within the expected range of output values. Also, the output is compared to real world values to see to what degree the model output is representative to a real world scenario.

First, it is investigated if the results that are obtained make sense. In general, all findings are very sensible and the reasons as to why certain output values are obtained are clearly explainable. For example, the RUL prediction RMSE values across all C-MAPSS instances are lower than those obtained in other paper, which makes sense as all aspects that worked well in literature so far have been combined into a novel model. Furthermore, regarding the results of the maintenance scheduling simulation model, it was clearly observed that for all maintenance strategies the results were easy to explain. For example, the no gap and ADDP strategies have much higher revenue due to the fact that aircraft utilization is maximized. On the other hand, the strategies that make use of a gapped flight schedule attain a much lower cost due to the lower number of flight cancellations required. This logic extends to all results obtained, meaning that the overall results are sensible. Also, the results vary in a logical manner when slightly changing the model input values, as was observed in the added sensitivity analysis.

Next, it is analyzed is the obtained results are as expected. Once again, this is clearly the case as for the ADDP strategy the goal is to develop a strategy that is able to maximize revenue and minimize cancellation costs. From the obtained results, it can be seen that this is indeed the outcome, meaning that the result is as expected. Furthermore, it was also expected that the inspection based policies would underperform the

data-driven policies for the same type of flight schedule. Overall, the results for all maintenance strategies are as expected when comparing the type of flight schedule and the type of maintenance policy.

When comparing the model results to real world values, it becomes clear to what degree the model resembles reality. It should of course be noted that due to the simplicity of the model that is developed, the obtained profit values will be vastly different than airlines obtain in practice. For example, in this work the only costs are the engine related maintenance cost and flight cancellation costs. In reality, airlines have much more costs, such as maintenance costs for other components, crew costs, landing costs etc. Therefore it would not make sense to compare the obtained profit values directly to the profit values of a real world aircraft. However, the profit values can be compared to see if the model is able to predict the profit values with the correct order of magnitude. Comparing the obtained profit values of $5-6\cdot10^5$\$ per aircraft per week to the values obtained by common airlines, it can be concluded that the obtained results have the correct order of magnitude, thereby adding confidence in the validity of the model output.

Applicability of the model

Even though it is attempted to make the maintenance scheduling model as realistic as possible, a large number of assumptions remain. For example, this research only considers the maintenance actions on engines, while in reality many other components also require maintenance. Furthermore, airlines usually group maintenance actions for components that are expected to fail around the same time. Neglecting these aspects limits the applicability of the model. This means that this model is not directly ready to be used to schedule maintenance for airlines using sensor recordings. However, the results of this model can be used to see in what way data-driven RUL prognostics can be incorporated best into the scheduling routine. The obtained profit values and the other KPI's present a clear picture on what maintenance strategy is optimal to implement when engine sensor data becomes available. The results of this research can thus be used by any airline to innovate their maintenance process into a data-driven scheduling routine. This work then forms the basis of how the engine sensor data can be used to find the optimal moments in time during which maintenance could be performed. As this work has no ties to real airlines, the actual validation of the model is a difficult task. True model validation should be done by incorporating the most prominent aspects of this research into the scheduling routine of a real airline to see how the profit of the airline then changes.

Bibliography

[1] KLM Board of Managing Directors. Annual report 2019 royal dutch airlines. Retreived from https://www.klm.com/travel/nl_nl/images/KLM-Jaarverslag-2019_tcm541-1063986.pdf, 2019.

[2] V. Nguyen, M. Kefalas, K. Yang, A. Apostolidis, M. Olhofer, S. Limmer, and T. Bäck. A review: Prognostics and health management in automotive and aerospace. International Journal of Prognostics and Health Management, 10(2), 11 2019.

[3] R. Li, W. J.C. Verhagen, and R. Curran. A systematic methodology for prognostic and health management system architecture definition. Reliability Engineering and System Safety, 193, 2020. doi:10.1016/j.ress.2019.106598.

[4] G. Vachtsevanos, F. Lewis, M. Roemer, A. Hess, and B. Wu. Intelligent Fault Diagnosis and Prognosis for Engineering Systems. John Wiley & Sons, Inc., Hoboken, NJ, USA, 2006.

[5] V. Atamuradov, K. Medjaher, P. Dersin, B. Lamoureux, and N. Zerhouni. Prognostics and health management for maintenance practitioners - review, implementation and tools evaluation. International Journal of Prognostics and Health Management, 8(3), 2017. doi:10.36001/ijphm.2017.v8i3.2667.

[6] A. Saxena and K. Goebel. Turbofan engine degradation simulation data set. https://ti.arc.nasa.gov/c/13/, 2008. NASA Ames, Moffet Field, CA.

[7] A. Saxena, J. Celaya, B. Saha, S. Saha, and K. Goebel. Metrics for offline evaluation of prognostic performance. International Journal of Prognostics and Health Management, 1(1):2153–2648, 2010.

[8] G. S. Babu, P. Zhao, and X. Li. Deep convolutional neural network based regression approach for estimation of remaining useful life. Database Systems for Advanced Applications, 9642:214–228, 2016.

[9] X. Li, Q. Ding, and J. Sun. Remaining useful life estimation in prognostics using deep convolution neural networks. Reliability Engineering and System Safety, 172:1–11, 2018. doi:10.1016/j.ress.2017.11.021.

[10] M. Yuan, Y. Wu, and L. Lin. Fault diagnosis and remaining useful life estimation of aero engine using lstm neural network. In 2016 IEEE International Conference on Aircraft Utility Systems (AUS), pages 135–140, 2016. doi:10.1109/AUS.2016.7748035.

[11] S. Zheng, K. Ristovski, A. Farahat, and C. Gupta. Long short-term memory network for remaining useful life estimation. In 2017 IEEE International Conference on Prognostics and Health Management (ICPHM), pages 88–95, 2017. doi:10.1109/ICPHM.2017.7998311.

[12] A. L. 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 183, 183:240–251, 2019. doi:10.1016/j.ress.2018.11.027.

[13] A. Saxena, K. Goebel, D. Simon, and N. Eklund. Damage propagation modeling for aircraft engine run-to-failure simulation. In 2008 International Conference on Prognostics and Health Management, pages 1–9, 2008. doi:10.1109/PHM.2008.4711414.

[14] J. Sun, C. Li, C. Liu, Z. Gong, and R. Wang. A data-driven health indicator extraction method for aircraft air conditioning system health monitoring. Chinese Journal of Aeronautics, 32(2):409–416, 2019. doi:10.1016/j.cja.2018.03.024.

[15] J.W. Hines, D. Garvey, R. Seibert, and A. Usynin. Technical review of on-line monitoring techniques for performance assessment, vol 2: Theoretical issues. 2007.

[16] C. L. Black, R. Uhrig, and J. Hines. System modeling and instrument calibration verification with a non-linear state estimation technique. In Maintenance and reliability conference, 1998.

[17] J. W. Hines and D. R. Garvey. Traditional and robust vector selection methods for use with similarity based models. In 2006 International Topical Meeting on Nuclear Plant Instrumentation Controls, and Human Machine Interface, 7 2006.

[18] Tianyi Wang, Jianbo Yu, D. Siegel, and J. Lee. A similarity-based prognostics approach for remaining useful life estimation of engineered systems. In 2008 International Conference on Prognostics and Health Management, pages 1–6, 2008. doi:10.1109/PHM.2008.4711421.

[19] K. Medjaher, N. Zerhouni, and J. Baklouti. Data-driven prognostics based on health indicator construction: Application to pronostia's data. In 2013 European Control Conference (ECC), pages 1451–1456, 2013. doi:10.23919/ECC.2013.6669223.

[20] Y. Qian, R. Yan, and R. X. Gao. A multi-time scale approach to remaining useful life prediction in rolling bearing. Mechanical Systems and Signal Processing, 83:549–567, 2016. doi:10.1016/j.ymssp.2016.06.031.

[21] M. Jouin, R. Gouriveau, D. Hissel, M.Péra, and N. Zerhouni. Particle filter-based prognostics: Review, discussion and perspectives. Mechanical Systems and Signal Processing, 72-73:2–31, 2016. doi:10.1016/j.ymssp.2015.11.008.

[22] J. B. Ali, B. Chebel-Morello, L. Saidi, S. Malinowski, and F. Fnaiech. Accurate bearing remaining useful life prediction based on weibull distribution and artificial neural network. Mechanical Systems and Signal Processing, 56-57:150–172, 2015. doi:10.1016/j.ymssp.2014.10.014.

[23] W. He, N. Williard, C. Chen, and M. Pecht. State of charge estimation for li-ion batteries using neural network modeling and unscented kalman filter-based error cancellation. Electrical Power and Energy Systems, 62:783–791, 2014. doi:10.1016/j.ijepes.2014.04.059.

[24] Z. Chen, S. Cao, and Z. Mao. Remaining useful life estimation of aircraft engines using a modified similarity and supporting vector machine (svm) approach. Energies, 11(1):28, 2018. doi:10.3390/en11010028.

[25] F. O. Heimes. Recurrent neural networks for remaining useful life estimation. In 2008 International Conference on Prognostics and Health Management, pages 1–6, 2008. doi:10.1109/PHM.2008.4711422.

[26] L. Peel. Data driven prognostics using a kalman filter ensemble of neural network models. In 2008 International Conference on Prognostics and Health Management, pages 1–6, 2008. doi:10.1109/PHM.2008.4711423.

[27] A.M. Riad, Hamdy K. Elminir, and Hatem M. Elattar. Evaluation of neural networks in the subject of prognostics as compared to linear regression model. International Journal of Engineering & Technology IJET-IJENS, 10, 12 2010.

[28] C. Zhang, L. Pin, A. K. Qin, and K. C. Tan. Multiobjective deep belief networks ensemble for remaining useful life estimation in prognostics. IEEE Transactions on Neural Networks and Learning Systems, PP:1–13, 07 2016. doi:10.1109/TNNLS.2016.2582798.

[29] D. Opitz and R. Maclin. Popular ensemble methods: An empirical study. Journal of Artificial Intelligence Research, 11:169—-198, 1999. doi:10.1613/jair.614.

[30] Z. Zhao, B. Liang, X. Wang, and W. Luc. Remaining useful life prediction of aircraft engine based on degradation pattern learning. Reliability Engineering and System Safety, 164:74–83, 2017. doi:10.1016/j.ress.2017.02.007.

[31] O. O. Aremu, D. Hyland-Wood, and P. R. McAree. A machine learning approach to circumventing the curse of dimensionality in discontinuous time series machine data. Reliability Engineering and System Safety, 195, 2020. doi:10.1016/j.ress.2019.106706.

[32] W. Yu, Y. Kim, and C. Mechefske. An improved similarity-based prognostic algorithm for rul estimation using an rnn autoencoder scheme. Reliability Engineering and System Safety, 199, 2020. doi:doi.org/10.1016/j.ress.2020.106926.

[33] B. Asadi and H. Jiang. On approximation capabilities of relu activation and softmax output layer in neural networks, 2020. arXiv:2002.04060.

[34] D. Jha and G. Kwon. Alzheimer's disease detection using sparse autoencoder, scale conjugate gradient and softmax output layer with fine tuning. International Journal of Machine Learning and Computing, 7(1):13–17, 2017. doi:10.18178/IJMLC.

[35] S. Zychlinksi. Predicting probability distributions using neural networks. https://towardsdatascience.com/predicting-probability-distributions-using-neural-networks\-abef7db10eac, 2018.

[36] C.M. Bishop. Mixture density networks. Workingpaper, Aston University, 1994.

[37] S. Blake. A guide to generating probability distributions with neural networks. https://medium.com/hal24k-techblog/a-guide-to-generating-probability-distributions-with-neural-networks-ffc4efacd6a4, 2019.

[38] L. V. Jospin, W. Buntine, F. Boussaid, H. Laga, and M. Bennamoun. Hands-on bayesian neural networks - a tutorial for deep learning users. ACM Computing Survices, 1(1), 2020.

[39] Y. Kwon, J. Won, B. J. Kim, and M. C. Paik. Uncertainty quantification using bayesian neural networks in classification: Application to biomedical image segmentation. Computational Statistics & Data Analysis, 142, 2020. doi:10.1016/j.csda.2019.106816.

[40] Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Insights and applications, 2015. arXiv:1506.02157.

[41] Y. Gal and Z. Ghahramani. Bayesian convolutional neural networks with bernoulli approximate variational inference, 2016. arXiv:1506.02158.

[42] R. Seoh. Qualitative analysis of monte carlo dropout, 2020. arXiv:2007.01720.

[43] A. Loquercio, M. Segu, and D. Scaramuzza. A general framework for uncertainty estimation in deep learning. IEEE Robotics and Automation Letters, 5(2):3153–3160, 2020. doi:10.1109/LRA.2020.2974682.

[44] K. Wickstrøm, M. Kampffmeyer, and R. Jenssen. Uncertainty and interpretability in convolutional neural networks for semantic segmentation of colorectal polyps. Medical Image Analysis, 60, 2020. doi:doi.org/10.1016/j.media.2019.101619.

[45] S. Blake. How to generate neural network confidence intervals with keras. https://medium.com/hal24k-techblog/how-to-generate-neural-network-confidence-intervals-with-keras-e4c0b78ebbdf, 2019.

[46] A. Botchkarev. A new typology design of performance metrics to measure errors in machine learning regression algorithms. Interdisciplinary Journal of Information, Knowledge, and Management, 14:45—-76, 2019. doi:10.28945/4184.

[47] P. A. Flach. The geometry of roc space: Understanding machine learning metrics through roc isometrics. In ICML03, volume 1, pages 194–201, 01 2003.

[48] A. Saxena, J. Celaya, E. Balaban, K. Goebel, B. Saha, S. Saha, and M. Schwabacher. Metrics for evaluating performance of prognostic techniques. In 2008 International Conference on Prognostics and Health Management, pages 1–17, 2008. doi:10.1109/PHM.2008.4711436.

[49] R. J. Hyndman and A. B. Koehler. Another look at measures of forecast accuracy. International Journal of Forecasting, 22(4):679–688, 2006. doi:10.1016/j.ijforecast.2006.03.001.

[50] S. Makridakis, A. Andersen, R. Carbone, R. Fildes, M. Hibon, R. Lewandowski, J. Newton, E. Parzen, and R. Winkler. The accuracy of extrapolation (time series) methods: Results of a forecasting competition. Journal of Forecasting, 1:111–153, 1982.

[51] K. Goebel and P. Bonissone. Prognostic information fusion for constant load systems. In 2005 7th International Conference on Information Fusion, volume 2, page 9, 2005. doi:10.1109/ICIF.2005.1592000.

[52] G. Vachtsevanos, F. Lewis, M. Roemer, A. Hess, and B. Wu. Intelligent Fault Diagnosis and Prognosis for Engineering Systems. John Wiley & Sons, Inc., Hoboken, NJ, USA, 2006.

[53] D. C. Hoaglin, F. Mosteller, and J. W. Tukey. Understanding Robust and Exploratory Data Analysis. John Wiley & Sons, Inc., Hoboken, NJ, USA, 2000.

[54] M. Baur, P Albertelli, and M Monno. A review of prognostics and health management of machine tools. International Journal Advanced Manufacturing Technologies, 107:2843—-2863, 2020.

[55] T. Gneiting and A. E Raftery. Strictly proper scoring rules, prediction, and estimation. Journal of the American Statistical Association, 102(477):359–378, 2007. doi:10.1198/016214506000001437.

[56] R.G. Sargent. Verification and validation of simulation models. Journal of Simulation, 7:12–24, 2013. doi:10.1057/jos.2012.20.

[57] Ieee draft guide: Adoption of the project management institute (pmi) standard: A guide to the project management body of knowledge (pmbok guide)-2008 (4th edition). IEEE P1490/D1, May 2011, pages 1–505, 2011. doi:10.1109/IEEESTD.2011.5937011.

[58] S. A. Boekweit. Fleet level multi-unit maintenance optimization subject to degradation. Msc thesis, Delft University of Technology, 02 2021.