Development of a Machine Learning Tool as a Predictive Maintenance Solution for the Flight Control System of the Embraer E190

Nirina Sofia Leão Raharitahiana
nirina.leao@tecnico.ulisboa.pt

Instituto Superior Técnico, Universidade de Lisboa, Portugal

January 2021

Abstract

In a reality where the demand for air transport has abruptly dropped due to a situation without precedent, in a market which had already adapted to high demand, and which was already extremely competitive, it is necessary to reduce operational costs to survive in a sector which has been mutilated by these exterior circumstances. One of the areas in which this reduction has most potential is maintenance. Technological advancements which facilitate the acquiring of flight data and new emerging machine learning solutions enable the introduction of new methods to address problems deriving from faults in aircraft components, which not only lead to delays, but many times the impossibility of flying, causing a condition known as Aircraft-on-Ground (AOG), inevitably leading to major financial consequences. This project had as objective the creation of a machine learning solution for the predictive maintenance of the Flight Control System of Portugália Airlines' fleet, comprised of 13 Embraer aeroplanes. For this, flight data from the various aircraft sensors were used, as well as alert messages generated by the aircraft's systems, and the reports provided by the maintenance teams. Several variables were created to model the degradation level of the system, so as to provide the necessary information to the created model, so it could estimate how many flights the system has until it is likely to experience a fault, based on previous flights. The results show the potential of the solution, and that the model succeeds in identifying degradation patterns in the system

Keywords: Flight data, Predictive maintenance, Machine learning, Flight control system

1. Introduction

In an age of generally ever growing demand for air transportation, yet along with a general decrease in cost of airfares, and especially now with the crisis of the COVID-19 pandemic which took a blow at the aeronautical sector worldwide, it becomes increasingly important to decrease operational costs.

Maintenance is a major factor in that regard, with the global Maintenance, Repair and Overhaul (MRO) spend in 2018 being valued at $69 billion, representing 9% of airlines operational costs [1]. In fact, maintenance improvements have been stated as being one of the top three savings for airlines, these savings listed as [1]:

- Health monitoring and predictive maintenance driven by improved dispatch reliability, labour productivity,
- Fuel cost savings, and
- Delay reduction through improved turnaround process.

The maintenance actions of interest to this work may be classified as one of the four main types: corrective, preventive, condition monitoring, and predictive maintenance.

Corrective maintenance, also possibly referred to as run-to-failure maintenance, is the act of repairing a certain component only after a failure on it has occurred.

Preventive maintenance, on the other hand, involves a scheduled, regularly performed activity on a component with the objective of bettering its chances of not failing. The scheduling of this maintenance type may be done under either of two principles: time-based (i.e. at a fixed period) or usage-based (e.g. at a fixed mileage on a vehicle). Preventive maintenance is recommended when a component has an increasing probability of failure over time; if its failures are random, preventive maintenance is futile.

Condition monitoring involves the assessing of a component's reliability based on available operation data, attempting to identify wear or degradation in

order to act accordingly.

Finally, predictive maintenance is a step up from condition monitoring and regards acting to prevent a failure based on knowledge of when it is going to occur. In other words, data pertaining to the component at hand may be used to apply machine learning and analytics to assess its reliability, thus performing maintenance actions accordingly and as needed. As more and more data becomes available, there is a gradual trend in adopting this technique to avoid the shortcomings of the others.

The aviation industry is no exception to this trend. In fact, many condition monitoring solutions exist (specifically referred to as Aircraft Health Monitoring systems), and more recently an emerging market for predictive maintenance solutions grows with the increasing desire to reduce maintenance costs and improve the provided services. Fig. 1 and Fig. 2 present the results of a survey to study the adoption of these maintenance solutions by the surveyed airlines, showing that over half of these airlines had already Health Monitoring Systems in use, and, concerning Preventive Maintenance, while it had not been as widely adopted as the Health Monitoring Systems, already a very high portion of the airlines relied on it.

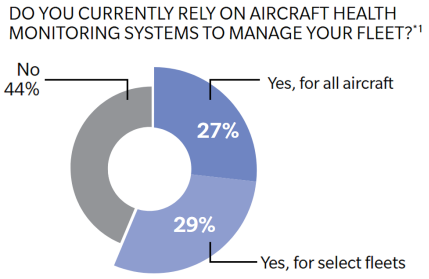

Figure 1: Adoption of Aircraft Health Monitoring systems, according to a survey by Oliver Wyman [2]

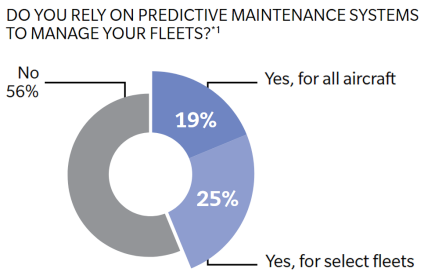

Figure 2: Adoption of Preventive Maintenance Systems, according to a survey by Oliver Wyman [2]

Machine learning has become an ever so present concept in the world we live in and so many technological advancements we rely on today.

And this is so as machine learning is such a versatile concept, expandable to a plethora of subjects, because instead of relying on programmers hard-coding ways in which an algorithm should behave, the machine is expected to learn by itself how to deal with a certain input. This represents an obvious advantage to traditional programming: it would be impossible to input every possible face into an algorithm so as to have it recognise faces in photographs, whereas with machine learning, the program is required to learn the general features of a face, and apply that knowledge into some other example, being then able to tell, with some degree of certainty, whether or not there is a face in the picture.

With such potential, it would only be a waste to not try to apply machine learning to engineering problems. And indeed, "machine learning is becoming a driving force in the field of industry-grade predictions, delivering significantly more reliable forecasts than traditional statistical methods, particularly where there is access to vast quantities of 'unstructured' data" [3].

The objective of this work is hence to provide Portugália Airlines with a machine learning solution to aid in applying predictive maintenance on the Flight Control System of the aircraft of their fleet. The solution should be able to output an estimated time until the next failure on this system, i.e the Remaining Useful Life (RUL), taking advantage of the flight data generated by each aircraft, so that maintenance may rely less on preventive and especially corrective methods, hence becoming more efficient and less costly.

2. Fundamentals

Following is an introduction to the aircraft in analysis, along with an overview of its Flight Controls System which should acquaint the reader with the basic workings of each subsystem, along with their main components, and some terminology. After this, the available data are presented, and a brief discussion on the relevant machine learning concepts is had in order to introduce the terms later used in the Methodology chapter.

2.1. Embraer E190 and its Flight Controls System

The Embraer E190 is part of the Embraer E-Jet family, which is comprised of four different aircraft models — E170, E175, E190, and E195 — all of which narrow-body twin engine aircraft capable of short to medium range flights, and with a seating capacity of 70 to 130 seats, being marketed by Embraer as regional aircraft with 'the big jet feel'. [4]

Portugália Airlines' fleet is currently composed

of nine Embraer E190 and 4 Embraer E195 [5]. These two variants differ from each other in fuselage length (thus seating capacity) and not much else. In fact, in terms of the Flight Control System, and this work, both models are the same, hence treated the same.

According to Airlines for America, formerly known as the Air Transport Association of America (ATA), the Flight Control System belongs to the ATA chapter 27 [6] (henceforth ATA 27, for short), which is a numbering system standard for all commercial aircraft documentation [7].

The flight control system is the system responsible for allowing directional control of the aircraft, being comprised of surfaces on wings and tail, and being made up of primary and secondary flight control systems [8]. The primary controls allow control of the aircraft about the lateral, longitudinal, and vertical axes and include the following subsystems: Ailerons, Elevators, Multifunction Spoilers, and Rudder.

The secondary controls serve as aid in lift generation and handling of the aircraft. These include the subsystems concerning the Flaps, Slats, Spoilers/Speed Brakes, and Horizontal Stabiliser subsystems. All these surfaces are represented in Fig. 3.

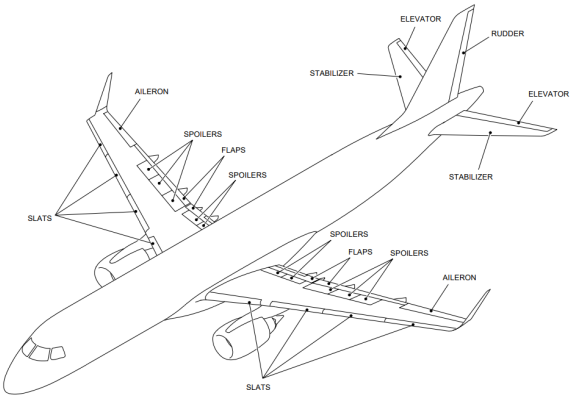

Figure 3: Flight controls system surfaces on the aircraft [8]

Additionally, an Electrical System is responsible for operating the electronically controlled fly-by-wire system.

All controls have position sensors providing readings of where each control for each surface — pilot's or copilot's — is, along with the applied force when applicable. All control surfaces have surface position sensors which provide information on their deflection. Additionally, the flaps and slats also have skew sensors which are used to prevent a skew condition from occurring. A skew condition pertains to when either the inboard or outboard edge of the flap moves further than the other [9]; should this happen, the skew sensors detect this, and the flap is disallowed from moving further.

Within the subsystems corresponding to each of the surfaces of the Flight Controls System, the most critical components in terms of system reliability were found to be the Power Control Units (PCUs), which provide movement to the surfaces of the Ailerons, Elevators, Rudder and Spoilers; the Aileron cables, which provide a mechanical control over the Aileron surfaces; the Flap and Slat skew sensors; and the various sensors for the positions of the aforementioned control surfaces from Fig. 3.

2.2. Available Data

The available data used in this work pertains to three different sources: flight data consisting of sensor data from the aircraft with readings of the sensors throughout each flight, maintenance and crew alerting messages generated by the aircraft's Central Maintenance Computer (CMC), and maintenance reports.

The sensor data was obtained via the company Sagem's Analysis Ground Station (AGS) software, which is the flight analysis software tool used by Portugália Airlines. After exporting, the data consisted of files — a file per flight — with sensor readings of the Flight Control System throughout the flight.

The event of failure was defined as the appearance of the message FLT CTRL NO DISPATCH in the Crew-Alerting System (CAS), which is a cautionary message which dictates immediate maintenance intervention, without which the aircraft is not allowed to fly.

Messages generated by the aircraft's CMC were accessed via the Fault History Database (FHDB). These pertain to maintenance messages generated by the aircraft when it is in operation, indicating alerts on its various systems.

Finally, the maintenance reports were accessible via the AMOS software, the MRO solution in use by Portugália Airlines.

2.3. Relevant Machine Learning Concepts

The dataset pertaining to the problem at hand consists of sensor data which characterises the functioning of the control system of the aircraft throughout time, and this therefore consists of time series data. Not all machine learning solutions are suited for this type of data. Recurrent Neural Networks (RNNs) have the ability to remember the past and are therefore capable of capturing temporal relationships between the current input in the series and what happened in the past. However, despite being able to remember the past, an RNN suffers from the issue of short-term memory. Given a long input series, the RNN might be able to relate the current input with its neighbours but cannot relate it to elements

which are further away in the series. This leads to an RNN potentially forgetting useful information simply due to the series being too long. The solution to this is using instead models which have longer memory, such as Long Short-Term Memory (LSTM) or Gated Recurrent Units (GRU).

Of specific interest to this work is the concept of Autoencoder. It revolves around an algorithm learning to encode a signal into a latent space, in practicality compressing the signal, and its applicability is surprisingly large, especially with respect to anomaly detection, for example in both [10] and [11].

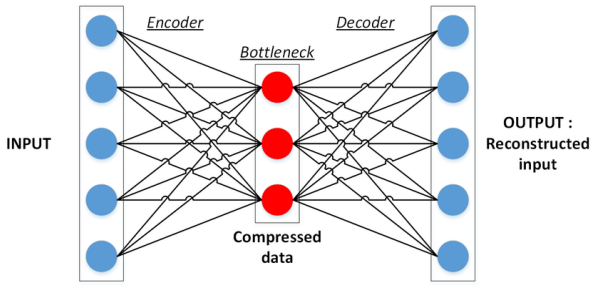

Figure 4: Basic architecture of an autoencoder [12]

To have an Autoencoder learn to encode a signal, it is necessary to have both an Encoder and a Decoder (see Fig. 4). A signal is input in the Encoder, which compresses it and feeds it to the Decoder, which in turn tries to reconstruct the signal back to its original state. In the end of this cycle, the algorithm measures how close it was to the original signal and the reconstruction error is used to iterate again, repeating the cycle until the desired accuracy is achieved. At this point, the Encoder can look at the most relevant characteristics of the input signal and construct a good encoding of said signal.

The most common way of making use of the concept of Autoencoder in anomaly detection is via its reconstruction error. The main idea behind this is that if the Autoencoder is trained to be able to encode and decode back (read reconstruct) normal data, then, when tasked with reconstructing anomalous data, the reconstruction error will be greater. This error can therefore be used as a measure of how anomalous the data is.

2.4. Health Indicator

Many methods to estimate the Remaining Useful Life of a component revolve around the calculation of a Health Index (HI). As the nomenclature suggests, this is a numerical value representative of the system's health. There are three types of health index, depending on how or on what basis it is calculated [13]:

- The *Physical Health Index* is defined from physical characteristics or parameters of the current state of the component and its operation, such as cracks;
- the *Probabilistic Health Index* is defined by the probability of the component being in a healthy state, or its reliability; its value ranges from 0 to 1, 1 being the best possible state;
- the *Mathematical Health Index* has only mathematical meaning and it can present any value. This is one of the most prevalent Health Index types in Remaining Useful Life estimation via machine learning methods, where HI is defined by the algorithm from a set of data.

3. Methodology

The purpose of this work is to obtain an estimate of how long a certain aircraft has until it is likely to experience an event of failure of the Flight Control System, based on the sensor data available. The machine learning solution chosen as the basis for this was proposed in [10], where an LSTM-Autoencoder was employed to try to determine the remaining useful lives of several instances in two different publicly available datasets: the C-MAPSS Turbofan Engine Dataset [14] and the Milling Machine Dataset [14].

Furthermore, the methodology used in the present work to apply this algorithm to the specific problem can be divided into four different stages:

- Data gathering — feature creation (which is to say the creation of the possible input variables for the model, of which a selection would be made according to their usefulness in describing the system's degradation);
- Data pre-processing and feature selection;
- Autoencoder, and
- Data post-processing.

3.1. Algorithm Overview

Given the system of interest's run-to-failure sequences, an Autoencoder is made use of to try to predict the Remaining Useful Life of a given test instance which has not yet failed. A run-to-failure sequence should be understood, in this context, as the evolution, along time, of the set of features chosen to represent the system's health: each sequence begins at a healthy state and ends at the instant where there is a failure.

For this purpose, first, the Autoencoder is trained to encode and decode again (i.e. reconstruct) healthy data only, which is to say, as an approximation, data where it is known a failure will not occur in a while. To achieve this, it is assumed that

the initial cycles of each sequence pertain to healthy data, so the Autoencoder is trained with those only.

The Autoencoder is then tasked with reconstructing the full sequences of the train data (which is to say, the full run-to-failure sequences, with both healthy cycles and degraded cycles). Because it was trained on healthy data only, and these full sequences represent a system that is degrading over time, it is presumable that the Autoencoder will make an ever-increasing error throughout the reconstruction of each sequence. This error is then used as a measurement of the health of the system, and so a normalisation of it is used as a Health Index. So, for each cycle t of the sequence u, the error $e_t^{(u)}$ is normalised as [10]:

$$h_t^{(u)} = \frac{e_M^{(u)} - e_t^{(u)}}{e_M^{(u)} - e_m^{(u)}}, \tag{1}$$

where $e_M^{(u)}$ and $e_m^{(u)}$ are the maximum and minimum reconstruction errors obtained for sequence u, respectively.

The result of calculating this Health Index for a sequence u throughout its length $L^{(u)}$ is a Degradation Curve $H^{(u)}$ which represents the degrading health of the system, from healthy state to failure such that $H^{(u)} = [h_1^{(u)}\, h_2^{(u)} ... \, h_t^{(u)} ... \, h_{L^{(u)}}^{(u)}]$. The Degradation Curves obtained from the train data are then stored.

Following, the same curves are obtained for the test data. Because the test data represents instances which have not yet run into a failure, the resulting Degradation Curves are considered incomplete. It is by comparing — matching — the test Degradation Curves with the train Degradation Curves that an estimation of the RUL for a given test instance is made, as represented by Fig. 5, showing that that estimation is obtained by varying a time-lag (horizontal offset) between the two curves and calculating the RUL as the remaining time cycles on the train curve after the last cycle of the test curve.

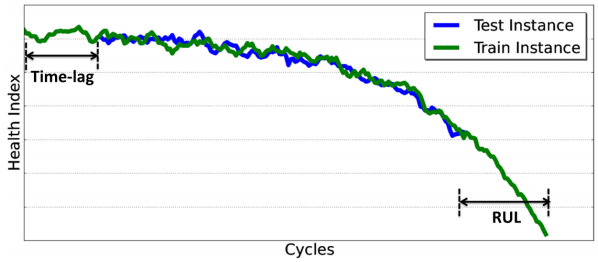

Figure 5: Example of RUL estimation using Degradation Curve matching [10]

There being various train curves and various possible values for the time-lag presented in Fig. 5, there are multiple estimates for the RUL of a single test instance, so a weighted mean of these estimations is taken as the final predicted RUL. The weights for each estimate are given by the similarity s between each test curve u^* and train curve u for given values of time-lag t, which is computed via the following expression [10]:

$$s(u^*, u, t) = \exp(-d^2(u^*, u, t)/\lambda), \tag{2}$$

where

$$d^2(u^*, u, t) = \frac{1}{L^{(u^*)}} \sum_{i=1}^{L^{(u^*)}} (h_i^{(u^*)} - h_{i+t}^{(u)})^2 \tag{3}$$

is the squared Euclidean distance between $H^{(u^*)}$ in its cycles 1 through $L^{(u^*)}$, and $H^{(u)}$ in its cycles t through $t + L^{(u^*)}$, and λ ($\lambda > 0$) is a parameter controlling the scale of the similarity, smaller values of λ implying larger difference in s, even when d is small.

Thus, each RUL estimate $\hat{R}^{(u^*)}(u, t)$ given by each train instance and each time lag is used to compute the final RUL estimate $\hat{R}_{final}^{(u^*)}$ by:

$$\hat{R}_{final}^{(u^*)} = \frac{\sum[s(u^*, u, t) \cdot \hat{R}^{(u^*)}(u, t)]}{\sum s(u^*, u, t)}. \tag{4}$$

Additionally, this summation is only over combinations of u and t such that

$$s(u^*, u, t) \geq \alpha \cdot s_{max} \tag{5}$$

with $0 \leq \alpha \leq 1$ and s_{max} as the maximum obtained similarity for a given test instance. This means that any RUL whose similarity was below this cutoff would not be used in the estimation.

The parameters λ, α, along with the maximum allowable time lag t_{max} between two curves would be parameters to be configured based on results of a validation set.

3.2. Data Gathering/Feature Creation

The data gathering stage of the work pertains to both obtaining the raw sensor data of the many flights of each aircraft via the AGS software, and the transformation of said data into usable information for the algorithm.

The raw data consists of sensor recordings of all flights from 13 different aircraft, concerning the period of around 4 years. For each flight, a file containing the evolution over time of over 70 sensor readings is stored. This results in over 72 000 files worth of data, each pertaining to the entirety of the corresponding flight, making them a considerable volume of data, and quite unfeasible to use 'as-is'.

As such, given the flight files, a summary of each flight containing its most important statistics stored in variables was created. For this, each subsystem of

the Flight Controls System was studied and the final line-up of features consisted of three main types of measurements for each given flight:

- Given pilot input how close the subsystem's response is to its theoretical response;
- The reading disparity between sensors which are supposed to read the same values;
- The applied force required by the pilots to move the controls.

Each subsystem was then studied based on these main criteria, when applicable (some controls such as the flap/slat lever do not have readings for force applied on them, for instance).

With respect to creating features to gauge the difference between theoretical and actual surface response to a pilot input, the method differed widely between each surface subsystem. For the Aileron subsystem, for instance, a table regarding control yoke rotation and expected left and right aileron deflection is given in the Embraer maintenance manuals, as presented in Table 1.

Table 1: Control yoke rotation *versus* left and right aileron deflection [8]

Control Yoke	Left Aileron	Right Aileron
40º (LEFT)	25º (UP)	15º (DOWN)
35º (LEFT)	20.64º (UP)	13.54º (DOWN)
30º (LEFT)	17.01º (UP)	11.84º (DOWN)
25º (LEFT)	13.59º (UP)	10.06º (DOWN)
20º (LEFT)	10.52º (UP)	8.20º (DOWN)
15º (LEFT)	7.62º (UP)	6.27º (DOWN)
10º (LEFT)	4.92º (UP)	4.27º (DOWN)
5º (LEFT)	2.36º (UP)	2.18º (DOWN)
0º	0º	0º
5º (RIGHT)	2.24º (DOWN)	2.29º (UP)
10º (RIGHT)	4.36º (DOWN)	4.74º (UP)
15º (RIGHT)	6.43º (DOWN)	7.33º (UP)
20º (RIGHT)	8.33º (DOWN)	10.15º (UP)
25º (RIGHT)	10.21º (DOWN)	13.22º (UP)
30º (RIGHT)	11.96º (DOWN)	16.63º (UP)
35º (RIGHT)	13.67º (DOWN)	20.51º (UP)
40º (RIGHT)	15º (DOWN)	25º (UP)

Table 2: Flap/Slat lever position *versus* flap deflection in degrees [8]

Flap/Slat Lever Position	Flap Position Reading (º)	Slat Position Reading (º)
0 (UP)	0	0
1	7	15
2	10	15
3	20	15
4	20	25
5	20	25
FULL (DOWN)	37	25

This allowed the creation of features such as the average, maximum and minimum recorded differences between actual and theoretical Aileron surface deflection, given pilot input, by interpolation of the above-mentioned table.

Other subsystems — such as flaps and slats — did not require interpolation due to the discrete nature of the readings: on one hand, the flap/slat lever only had 7 possible positions; on the other hand, despite there being a continuous transition between angles on the surfaces as they move, the AGS software rounds any value to its closest entry of Table 2.

For these two subsystems, it was possible to directly measure the delay of the surface response, after the lever is moved.

Finally, for the Elevator, Rudder and Spoiler subsystems, no table such as Table 1 or Table 2 is provided in the aircraft's maintenance manuals. In these cases, the workaround to still be able to find a theoretical input/output relation was to take files corresponding to flights which were deemed healthy (i.e., when it is known a failure is not going to occur in a while) and compute a linear regression model between the control's position and the corresponding surface's deflection. This can be illustrated by Fig. 6.

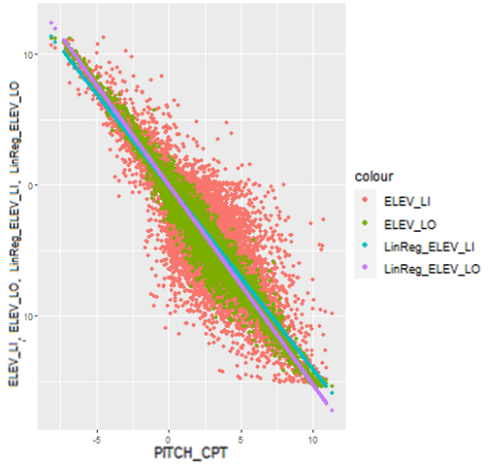

Figure 6: Elevator pilot input (PITCH_CPT) versus surface output; LI = left inboard surface sensor reading, LO = left outboard surface sensor reading, LinReg = linear regression

After this step, the same principle for feature creation as the Ailerons was used, where the difference between expected and actual surface deflection is computed throughout the flight, and an average, maximum and minimum differences are recorded for the flight summary.

The other source of data pertains to the system alert messages available in the FHDB. For these, a feature was created for each subsystem presented in Section 2.1, each consisting of a weighted sum of all the messages corresponding to said subsystem.

The weights for each message type were assigned based on message importance, which was assessed by the amount of times that message is present in maintenance reports from system failures.

3.3. Data Pre-Processing and Feature Selection

The data pre-processing stage pertains to the steps taken into preparing the gathered data to be input in the algorithm, such as dealing with abnormalities in the data, rearranging the flight summaries into run-to-failure sequences, selecting which features should be used in the model, normalising the data, and creating the train, test, and validation sets to train, tune, and test the model.

Of the 75 total features created across the Flight Controls System, only a selection was later used in the algorithm, since there were many which would only add to the run time and nothing more. This selection was made based on correlation to the faults. Such correlation is visible in the feature plots, and an example of this is shown in Fig. 7, where in the first quarter of 2018, an unusual amount of force seems to be needed to move the elevator controls; this period eventually ends after a fault in a Flight Control Module (FCM), an Electrical System component.

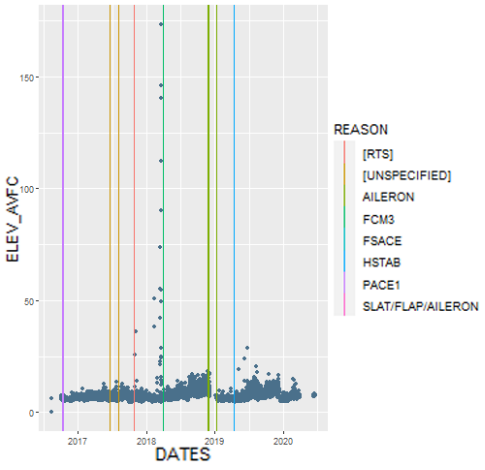

Figure 7: Plot of an elevator force feature throughout a given aircraft's lifecycle, measuring the average force needed on the elevator control per flight; vertical lines mark the dates where there was a fault in the system, each colour representing the reason for said fault

The flight summaries comprised of the features which were selected for the analysis were then organised in run-to-failure sequences, and the features were normalised by the following expression:

$$z = \frac{x - \min(x)}{\max(x) - \min(x)}, \tag{6}$$

where z is the re-scaled feature, x is the original feature, and min and max are the minimum and maximum values found for feature x, respectively.

The test and validation sets were sequences randomly taken from the original dataset, simply truncated at random so that the model had to estimate when failures were going to occur.

3.4. Autoencoder

The used autoencoder was built in the R programming language, making use of the Keras machine learning library. The encoder consisted of one LSTM layer with the rectified linear activation function and with a vector of 256 units as the output — 256 being the found number of units to be the best balance between computing time and reconstruction results. The decoder consisted of an LSTM layer with the ReLU activation function and with a 2D array with shape (*number of timesteps*, 256) as its output, followed by a time distributed layer so that the output of the autoencoder had the same shape as the original input. Moreover, between the encoder and decoder a repeat vector layer is required, so that the input of the decoder has the required shape for the decoder's LSTM layer (a 2D array rather than a vector).

With all this being set, the autoencoder model was then compiled and trained with the Mean Squared Error as the loss function, and the Adam optimiser, on 1000 epochs.

It was then tasked with reconstructing the sequences of the train, validation, and test sets.

3.5. Data Post-Processing

After obtaining the reconstructions of the train, validation, and test sets, the reconstruction errors throughout each sequence can then be computed. Considering a time series $Z = [z_1, z_2, ..., z_t, ..., z_L]$ for a sequence u with length L (where z_t is the vector of the features at time t), and considering its reconstruction $Z' = [z'_1, z'_2, ..., z'_t, ..., z'_L]$, this error $e_t^{(u)}$ is given by:

$$e_t^{(u)} = ||z_t^{(u)} - z_t'^{(u)}||. \tag{7}$$

Alternatively, a squared error can be considered so that larger reconstruction error later results in a much smaller health index:

$$e_t^{(u)} = ||z_t^{(u)} - z_t'^{(u)}||^2, \tag{8}$$

and both versions were tested, the best being chosen via a validation set.

Either of these error measurements would be normalised so as to obtain a health index $h_t^{(u)}$, as mentioned before in Eq. (1), to arrive at the degradation curves $H^{(u)}$.

Finally, these degradation curves were used as described in the Algorithm Overview to arrive at a final RUL estimation.

4. Results

Two sets of results were obtained, where the second was an attempt to obtain better estimations by filtering some outlier data points which were found to be due to either a recording, or software issue, concerning sensor readings on the various subsystems. These points were found to not relate to degradation of the Flight Control System, hence they were removed for testing via a cutoff in a feature which only presented higher values in this situation.

The performance metrics for assessing the model and configuring the hyper-parameters were the Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), number of False Positives and number of False Negatives. A False Positive in this context was defined as the model underestimating the RUL by under 15 days, while a False Negative meant that the model overestimated the RUL by over 15 days. The MAE and RMSE are presented in the following expressions:

$$MAE = \frac{1}{n}\sum_{u=1}^{n}|\hat{R}^{(u)} - R^{(u)}| \tag{9}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{u=1}^{n}(\hat{R}^{(u)} - R^{(u)})^2}, \tag{10}$$

4.1. Base Results

The best configuration found for the validation set was with $e_t^{(u)} = ||z_t^{(u)} - z_t'^{(u)}||^2$, $\lambda = 0.0005$, $\alpha = 0.74$, and $t_{max} = 50$. When applied this configuration to the test set, the following results were outputted by the model, as presented in Fig. 8. The performance metrics coming from these results follow in Table 3.

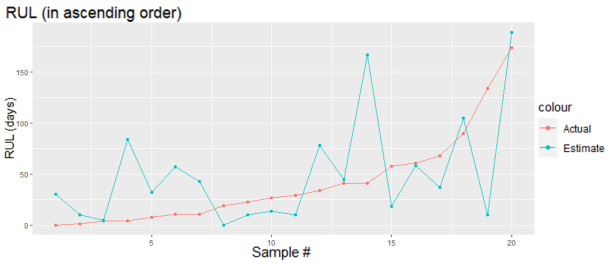

Figure 8: Actual and estimated RUL for the test set, in ascending order of the actual RUL

Table 3: Performance metrics on the RUL estimations from the base results

MAE	25.39
RMSE	31.19
Number of False Positives	5
Number of False Negatives	7

4.2. Data-Filtered Results

The best configuration for the validation set was found to be with $e_t^{(u)} = ||z_t^{(u)} - z_t'^{(u)}||^2$, $\lambda = 0.0005$, $\alpha = 0.95$, and $t_{max} = 50$. This corresponded to the estimates for the test set presented in Fig. 9, and performance metrics presented in Table 4.

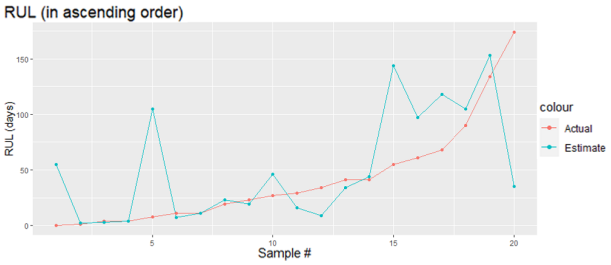

Figure 9: Actual and estimated RUL for the test set, in ascending order of the actual RUL

Table 4: Performance metrics on the RUL estimations from the system-wide-peak-filtered results

MAE	20.04
RMSE	27.97
Number of False Positives	2
Number of False Negatives	7

5. Result Discussion

Having obtained the predictions of the Remaining Useful Life of the test set, the interest lies in understanding the reasons for which the model outputted a certain RUL estimate and not another. When the prediction was accurate, the algorithm should understand why the system failed, thence showing that in the features corresponding to the failed subsystem; in the case of a False Positive, there should be some basis for the algorithm to predict a shorter RUL than its actual value; and finally, for a False Negative, it interests to know why the model did not see any anomalies and thus overestimated the RUL of a sample sequence.

5.1. Base Results

For the Base Results, it was found that for examples of True positives the model identified which subsystem was the reason for the failure of the sequence whose RUL it accurately estimated. This is showcased by the fact that upon inspection of the run-to-failure sequences in question, their features and respective reconstructions show a greater reconstruction error in features corresponding to the failed subsystem.

When looking at False Positive examples, however, no reason for the low RUL estimate compared to its true value was found. The fact is that the model identified anomalous behaviour where there were no true operational reasons for that behaviour to be taken as an anomaly: neither complaints from the crew were found, nor any sort of maintenance report pointing to any issue within the system.

As for False Negatives, where the estimated RUL is much greater than its real value, it was found that

indeed the model could not identify any anomalies in the behaviours of the features which could have tipped it into estimating a shorter RUL. This may stem from inadequate or insufficiently descriptive features, as the model was not able to associate the existing features' behaviours to anomalies. It is also possible that degradation only began much closer to the day of the fault, in which case it would simply not be possible for the model to accurately predict the RUL, with the length of sequence it was given.

5.2. Data-Filtered Results

The data-filtered results show a substantial improvement to the base results, confirming the assertion that the removed data points were not consequential to system degradation (at least not Flight Control System degradation, yet perhaps some other system not in study in this work).

For these results, it was found that the True Positives once again that the RUL estimate came from the model's ability to identify anomalous behaviour on the features corresponding to the failed subsystem.

As for False Positives, it was found that the behaviours considered anomalous by the model were indeed connected to the real operation of the aircraft. In this case, it translated into complaints from the pilots, or findings from the maintenance team resulting in small maintenance interventions. While these had not been considered faults *per se*, the algorithm was still able to identify them as a degraded state of the system.

Finally, with regards to False Negatives, it was found that the model was simply unable to identify degradation or an anomalous behaviour in the system.

5.3. Final Remarks on the Model

The filtered-data results show a promising ability to detect anomalies in the Flight Control System, where the True Positives stem from an understanding of which subsystem is degraded, and the False Positives still give valuable insight to a degradation which, while not necessarily leading to a major fault, still has implications in the operation of the aircraft.

False Negative estimations on both sets of results show the possible inadequacy of the features in representing the degradation of their respective subsystem, and thus the degradation of the overall Flight Control System. While it has been shown that certain behaviours in the feature plots seem to directly relate to some oncoming faults, in truth many other faults may not show in the gathered features, which later results in an inaccurate prediction of the RUL. As also discussed before, there is additionally the problem of feature behaviour relating to faults, yet not soon enough that this may be caught by the algorithm in time. This was indeed the case in many of the features, where anomalies were only detectable some days in advance, and even many times only on the actual day of the fault.

However, likely the major issue with this approach, was that too many subsystems which were mostly independent from each other were being modelled together. There were too many failure modes, so even if the algorithm might be able to predict a fault on the Electrical System for instance, that was no guarantee that it is capable of predicting a fault on the Aileron System. This would have been of interest to explore, by separating all systems and analysing them one by one, with only faults and features pertaining to the respective system, and perhaps the Electrical System being modelled together with each of the subsystems which depend on it. However, the lack of failure data, which had already been an issue with all the systems together, would now completely impede such study: there was no single system with enough failures to create an entire dataset of them, and the only system coming close to that possibility was the Electrical System, which would invariably require the analysis of features from more subsystems along with it anyway.

Still, the analysis done on the second set of results suggests that there is validity in the model's interpretation in what is an abnormal behaviour, so a RUL estimation corresponding to a small value, while not necessarily meaning there is an imminent fault, may be an indicative that the aircraft may benefit from a brief inspection.

6. Conclusions

The ultimate objective of this work was to provide a solution in failure prediction on the Flight Control System to allow for a better maintenance strategy beyond scheduled preventive maintenance, and run-to-failure maintenance, and in that front, being able to predict a few faults is still better than not predicting any, so a solution which was not prone to outputting False Positives was pursued, so that whenever a small RUL was predicted, the action should be to inspect the feature reconstructions and if possible identify the source of the problem, or to directly inspect the aircraft.

This work arrived at the objective of predicting the Remaining Useful Life of a set of test sequences of flights, with reasonable accuracy in predictions of up to 20 days, and granted that the number of flights to assess is great enough so as to give the model enough data points to make its prediction.

Additionally, many statistics to gauge the Flight Control System's health were created, for each of its many subsystems, and many of them proving to directly relate to faults or system degradation.

While some of the created features may have too much variability from many factors to conclude on the subsystem's degradation, they may still be useful to assess the performance of the controls, and to cross check crew complaints with. Other features were found to relate to some faults, and seeing as sometimes a maintenance intervention was found to worsen their respective behaviours, it may be practical to keep updated plots of each to catch any such changes in behaviour almost as soon as they happen. This is also valid for many other features, as many were seen correlating to some faults, even if only one or two in the entire dataset.

These features also allowed the identification of problems in either the recording system, or some software glitch, which, while not problematic in the aircraft's operations, may be a hindrance in further analysis of aircraft flight data.

Finally, implementation of the feature gathering routines in Portugália Airlines' environment could prove to be beneficial for the reasons already mentioned, and even if not for their predictive capabilities, then for their troubleshooting ones. As has already been shown, many features do relate to faults, and some of them only show signs of anomaly much too close to the day of the fault for their predictive capabilities to be of any worth. Many a time, troubleshooting the Flight Control System is not as simple as reading an error code and cross checking it with the maintenance manuals. Perhaps with the aid of these variables, troubleshooting becomes easier and hence more efficient, therefore avoiding, or otherwise minimising, flight delays. The program would then not only work as a Predictive Maintenance solution, but also a Health Monitoring one.

Acknowledgements

The author would like to thank this work's supervisors, Professor Sérgio Carvalho and Engineer Pedro Figueira, for allowing the opportunity to work alongside Portugália Airlines to deliver this project.

A further thankful note should be given to Engineer Nuno Lima from Portugália Airlines, for his patience and guidance, insightful advice whenever inquired, and generous availability to discuss the issues at hand. This thanks extends to everyone at Portugália Airlines, for always providing a welcome and warm environment, despite having the opportunities to be in the office been so scarce due to the COVID-19 pandemic.

References

[1] IATA, *Airline Maintenance Cost Executive Commentary*. International Air Transport Association, December 2019.

[2] T. Hoyland, C. Spafford, and A. Medland, *MRO Big Data – A Lion or a Lamb?* Innovation and Adoption in aviation MRO, Oliver Wyman, 2016.

[3] K. Bhattacharya, J.-P. Cresci, and P. Lortie, *Machine Learning: A Turning Point for Predictive Maintenance?* VELOCITY, Oliver Wyman, 2017.

[4] Embraer, "E-jets. the business warrior." `https://www.embraercommercialaviation.com/fleet/e-jets/`. [Online; Accessed 14 Dec. 2020].

[5] Embraer, "Portugália airlines fleet." `https://www.portugalia-airlines.pt/Pages/Fleet.aspx`. [Online; Accessed 17 Dec. 2020].

[6] A. Unlimited, "Ata 100 chapters complete list." `https://www.aerospaceunlimited.com/ata-chapters/`. [Online; Accessed 18 Dec. 2020].

[7] E. A. Solutions, "Ata chapters." `https://www.exsyn.com/wiki/ata-chapter`. [Online; Accessed 17 Dec. 2020].

[8] Embraer, *Aircraft Maintenance Manual*, ch. System Description Section. Embraer S.A., August 2008.

[9] E. White, "Flap slat accessory module and flap skew detection system," *Boeing Aero Magazine*, no. 6, 1999.

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

[11] N. Gugulothu, V. TV, P. Malhotra, L. Vig, P. Agarwal, and G. Shroff, "Predicting remaining useful life using time series embeddings based on recurrent neural networks." arXiv 1709.01073, 2017.

[12] J. Sublime and E. Kalinicheva, "Automatic post-disaster damage mapping using deep-learning techniques for change detection: Case study of the tohoku tsunami," *Remote Sensing*, vol. 11, no. 9, 2019.

[13] T. Wang, *Trajectory Similarity Based Prediction for Remaining Useful Life Estimation*. PhD thesis, University of Cincinnati, 2010.

[14] NASA, "Pcoe datasets." `https://ti.arc.nasa.gov/tech/dash/groups/pcoe/prognostic-data-repository/`. [Online; Accessed 2 May 2020].