

Marios Anastasopoulos

Optimization of satellite operations scheduling using an AI-based tool

School of Aerospace, Transport and Manufacturing

MSc in Astronautics and Space Engineering

MSc

Academic Year: 2022 - 2023

Supervisor: Dr. Nicola Garzaniti

September 2023



School of Aerospace, Transport and Manufacturing

MSc in Astronautics and Space Engineering

MSc

Academic Year 2022 - 2023

Marios Anastasopoulos

Optimization of satellite operations scheduling using an AI-based tool

Supervisor:  Dr. Nicola Garzaniti

September 2023

This thesis is submitted in partial fulfilment of the requirements for the degree of MSc in Astronautics and Space Engineering

© Cranfield University 2023. All rights reserved. No part of this publication may be reproduced without the written permission of the copyright owner.

Academic integrity declaration

I declare that:

* the thesis submitted has been written by me alone.
* the thesis submitted has not been previously submitted to this university or any other.
* that all content, including primary and/or secondary data, is true to the best of my knowledge.
* that all quotations and references have been duly acknowledged according to the requirements of academic research.

I understand that to knowingly submit work in violation of the above statement will be considered by examiners as academic misconduct.

Abstract

Click here to enter abstract text of not more than 300 words.

Keywords:

Click here to enter at least 6 keywords (not contained within the thesis title) – this makes the thesis searchable.

Acknowledgements

Click here to enter acknowledgement text.

Table of Contents

[Academic integrity declaration i](#_Toc143462420)

[Abstract ii](#_Toc143462421)

[Acknowledgements iii](#_Toc143462422)

[List of Figures vi](#_Toc143462423)

[List of Tables viii](#_Toc143462424)

[List of Equations ix](#_Toc143462425)

[List of Abbreviations x](#_Toc143462426)

[1 Introduction 1](#_Toc143462427)

[2 Literature Review 2](#_Toc143462428)

[2.1 Time Series Forecasting 2](#_Toc143462429)

[2.2 Recurrent Neural Networks 2](#_Toc143462430)

[2.2.1 GRU 3](#_Toc143462431)

[2.2.2 LSTM 4](#_Toc143462432)

[2.2.3 Transformers 5](#_Toc143462433)

[2.3 Model Building 7](#_Toc143462434)

[2.4 Hellas-Sat 3 10](#_Toc143462435)

[2.5 Tools 12](#_Toc143462436)

[3 Methodology 14](#_Toc143462437)

[3.1 Data Collection 14](#_Toc143462438)

[3.2 Data Pre-Processing 16](#_Toc143462439)

[3.3 Data Analysis and Preparation 18](#_Toc143462440)

[3.4 Hyperparameter Optimization 18](#_Toc143462441)

[3.5 Model Comparison 25](#_Toc143462442)

[3.6 Ticket Creation 31](#_Toc143462443)

[4 Results 35](#_Toc143462444)

[4.1 Model Selection 35](#_Toc143462445)

[4.1.1 Reaction Wheel 1 35](#_Toc143462446)

[4.1.2 Reaction Wheel 2 37](#_Toc143462447)

[4.1.3 Reaction Wheel 3 39](#_Toc143462448)

[4.1.4 Reaction Wheel 4 41](#_Toc143462449)

[4.2 Predictions 43](#_Toc143462450)

[4.3 Ticket 47](#_Toc143462451)

[4.4 Current Solution 48](#_Toc143462452)

[5 Conclusions 49](#_Toc143462453)

[5.1 Summary 49](#_Toc143462454)

[5.2 Limitations 49](#_Toc143462455)

[5.3 Future work 49](#_Toc143462456)

[References 50](#_Toc143462457)

[Appendices 51](#_Toc143462458)

# List of Figures

[Figure 2‑1 GRU cell structure 3](#_Toc143462548)

[Figure 2‑2 Vanilla LSTM cell structure 4](#_Toc143462549)

[Figure 2‑3 Architecture of Transformer forecasting model 6](#_Toc143462550)

[Figure 2‑4 LSTM cell architecture 8](#_Toc143462551)

[Figure 2‑5 LSTM memory network 8](#_Toc143462552)

[Figure 3‑1 Process diagram 14](#_Toc143462553)

[Figure 3‑2 Reaction Wheel rates Jan 2022 - Dec 2022, a) Reaction Wheel 1, b) Reaction Wheel 2, c) Reaction Wheel 3, d) Reaction Wheel 4 15](#_Toc143462554)

[Figure 3‑3 Non-normalized predictions for example LSTM model 17](#_Toc143462555)

[Figure 3‑4 Normalized predictions for example LSTM model 17](#_Toc143462556)

[Figure 3‑5 Multi-Layered LSTM model architecture 20](#_Toc143462557)

[Figure 3‑6 Ticket creation flow process 31](#_Toc143462558)

[Figure 3‑7 Rolling input window for predictions 32](#_Toc143462559)

[Figure 4‑1 RW1 Training and Validation Loss & RMSE at 100 epochs 36](#_Toc143462560)

[Figure 4‑2 RW1 Training dataset Predictions (top) and Actual Data (bottom) 36](#_Toc143462561)

[Figure 4‑3 RW1 Validation dataset Predictions (top) and Actual Data (bottom) 37](#_Toc143462562)

[Figure 4‑4 RW2 Training and Validation Loss & RMSE at 100 epochs 38](#_Toc143462563)

[Figure 4‑5 RW2 Training dataset Predictions (top) and Actual Data (bottom) 38](#_Toc143462564)

[Figure 4‑6 RW2 Validation dataset Predictions (top) and Actual Data (bottom) 39](#_Toc143462565)

[Figure 4‑7 RW3 Training and Validation Loss & RMSE at 100 epochs 40](#_Toc143462566)

[Figure 4‑8 RW3 Training dataset Predictions (top) and Actual Data (bottom) 40](#_Toc143462567)

[Figure 4‑9 RW3 Validation dataset Predictions (top) and Actual Data (bottom) 41](#_Toc143462568)

[Figure 4‑10 RW4 Training and Validation Loss & RMSE at 100 epochs 42](#_Toc143462569)

[Figure 4‑11 RW4 Training dataset Predictions (top) and Actual Data (bottom) 42](#_Toc143462570)

[Figure 4‑12 RW4 Validation dataset Predictions (top) and Actual Data (bottom) 43](#_Toc143462571)

[Figure 4‑13 RW1 Test dataset Predictions (top) and Actual Data (bottom) 44](#_Toc143462572)

[Figure 4‑14 RW2 Test dataset Predictions (top) and Actual Data (bottom) 44](#_Toc143462573)

[Figure 4‑15 RW3 Test dataset Predictions (top) and Actual Data (bottom) 45](#_Toc143462574)

[Figure 4‑16 RW4 Test dataset Predictions (top) and Actual Data (bottom) 45](#_Toc143462575)

[Figure 4‑17 Survey results for detected threshold violation instances a) Reaction Wheel 1, b) Reaction Wheel 2, c) Reaction Wheel 3, d) Reaction Wheel 4 46](#_Toc143462576)

List of Tables

[Table 3‑1 Optimized Hyperparamters 18](#_Toc143462577)

[Table 3‑2 Reaction Wheel 1 architecture results of hyperparameter optimization 21](#_Toc143462578)

[Table 3‑3 Reaction Wheel 2 architecture results of hyperparameter optimization 22](#_Toc143462579)

[Table 3‑4 Reaction Wheel 3 architecture results of hyperparameter optimization 23](#_Toc143462580)

[Table 3‑5 Reaction Wheel 4 architecture results of hyperparameter optimization 24](#_Toc143462581)

[Table 3‑6 Reaction Wheel 1 Architectures Validation Loss and RMSE 27](#_Toc143462582)

[Table 3‑7 Reaction Wheel 2 Architectures Validation Loss and RMSE 28](#_Toc143462583)

[Table 3‑8 Reaction Wheel 3 Architectures Validation Loss and RMSE 29](#_Toc143462584)

[Table 3‑9 Reaction Wheel 4 Architectures Validation Loss and RMSE 30](#_Toc143462585)

[Table 4‑1 Selected model architecture for Reaction Wheel 1 35](#_Toc143462586)

[Table 4‑2 Selected model architecture for Reaction Wheel 2 37](#_Toc143462587)

[Table 4‑3 Selected model architecture for Reaction Wheel 3 39](#_Toc143462588)

[Table 4‑4 Selected model architecture for Reaction Wheel 4 41](#_Toc143462589)

[Table 4‑5 Database entry with critical operation information 47](#_Toc143462590)

[Table 4‑6 Database entry with procedure steps 47](#_Toc143462591)

List of Equations

[(1‑1) 4](#_Toc143462592)

[(1‑2) 4](#_Toc143462593)

[(1‑3) 4](#_Toc143462594)

[(1‑4) 4](#_Toc143462595)

[(1‑5) 5](#_Toc143462596)

[(1‑6) 5](#_Toc143462597)

[(1‑7) 5](#_Toc143462598)

[(1‑8) 5](#_Toc143462599)

[(1‑9) 5](#_Toc143462600)

[(1‑10) 5](#_Toc143462601)

[(3‑1) 19](#_Toc143462602)

List of Abbreviations

|  |  |
| --- | --- |
| IT | Information Technology |
|  |  |
|  |  |
|  |  |
|  |  |

# Introduction

introduce the problem and the research question.

the aim

the objective

the structure of the thesis

# Literature Review

Satellites are highly complex systems that are constituted of many different subsystems, each responsible for specific operations. All subsystems are equipped with a wide range of sensors, such as temperature, pressure, voltage or current sensors, that continuously take measurements. Those measurements are the telemetry that is being transmitted to the ground for monitoring and detailed analysis by the satellite operators, in order to determine the state of health of each subsystem and eventually of the entire satellite. Since the amount of data that is being generated and needs to be analysed is massive, there has been identified the need for automation either on board the satellites or on the ground. Since almost all telemetry that is being produced has temporal dependency and the measurements that are being taken are in series, time series forecasting could be applied.

## Time Series Forecasting

Time series forecasting is an analytical technique used to predict future values of a variable based on its historical data points, where each observation is recorded at specific time intervals. This technique finds applications across numerous domains, including finance, economics, healthcare, weather prediction, supply chain management, and more. The ability to make accurate predictions about future trends and patterns in time-dependent data is invaluable for decision-making, resource allocation, risk management, and strategic planning.

In time series forecasting, the primary goal is to develop models that can capture underlying patterns, seasonality, trends, and irregular fluctuations present in the data. Accurate forecasts enable organizations and stakeholders to anticipate market trends, optimize resource allocation, and mitigate potential risks, leading to improved operational efficiency and better-informed decisions.

## Recurrent Neural Networks

Traditional statistical methods for time series forecasting, such as ARIMA and exponential smoothing, often struggle to capture complex temporal dependencies and nonlinear relationships in data. This limitation prompted the exploration of more sophisticated techniques, leading to the rise of deep learning and specifically, Recurrent Neural Networks (RNNs).

RNNs are a class of neural networks uniquely designed to process sequences of data by maintaining a hidden state that captures information from previous time steps. However, vanilla RNNs have difficulties learning long-range dependencies due to the vanishing gradient problem. This limitation gave birth to a specialized type of RNN known as Long Short-Term Memory (LSTM).

Typical types of RNNs used for time series forecasting are GRUs, LSTM and transformers.

### GRU

Gated Recurrent Units (GRUs) are a type of gated RNN, similar in nature to LSTMs but with a simplified architecture. GRUs also incorporate gating mechanisms to control the flow of information, but they merge the cell state and hidden state, resulting in a more streamlined design compared to LSTMs. GRUs tend to have fewer parameters, which can make them computationally efficient and sometimes better suited for smaller datasets. While they might not capture long-range dependencies as effectively as LSTMs, GRUs have shown competitive performance in various time series forecasting applications.

A diagram of a diagram

Description automatically generated

Figure ‑ GRU cell structure

|  |  |
| --- | --- |
|  | (‑) |
|  | (‑) |
|  | (‑) |
|  | (‑) |

### LSTM

LSTM, as previously mentioned, is a specialized variant of Recurrent Neural Networks (RNNs) designed to overcome the challenges posed by vanishing gradients. LSTMs employ memory cells, gating mechanisms, and activation functions to effectively capture temporal dependencies in sequential data. This makes them well-suited for time series forecasting tasks, where patterns and relationships can span across different time steps. LSTMs have been widely adopted in various domains, ranging from finance and energy to healthcare and beyond. They excel in capturing both short-term fluctuations and long-term trends in time series data.

A diagram of a tank

Description automatically generated

Figure ‑ Vanilla LSTM cell structure

|  |  |
| --- | --- |
|  | (‑) |
|  | (‑) |
|  | (‑) |
|  | (‑) |
|  | (‑) |
|  | (‑) |

### Transformers

Transformers, originally designed for natural language processing tasks, have been adapted for time series forecasting with remarkable success. Unlike traditional RNNs, transformers process input data in parallel rather than sequentially, which can lead to faster training times. Attention mechanisms within transformers allow them to focus on relevant temporal relationships across various time steps, making them highly effective at capturing complex patterns in time series data.

In the context of time series forecasting, the "Transformer-XL" and "Temporal Fusion Transformers" are notable architectures that extend transformers to handle sequential data. These models take into account the order and temporal dependencies of the data, making them well-suited for tasks that involve both short-term and long-term forecasting.

A diagram of a decoder

Description automatically generated

Figure ‑ Architecture of Transformer forecasting model

The architecture of a transformer based timeseries forecasting model includes the following:

* Encoder and Decoder stacks: the encoder stack processes the historical time series data, and the decoder stack generates the future forecast.
* Encoder Input: time series data is first transformed into numerical representations that the Transformer can understand. Each data point in the time series is represented as a vector that includes relevant information, like the actual value, time step, and any other contextual features.
* Self-Attention: It allows the model to weigh the importance of each element in the input sequence with respect to all other elements. This captures both short-term and long-term dependencies in the sequence.
* Feed forward: After the self-attention layers, it includes feed forward neural networks for each position in the sequence. They transform the learned representations to capture more complex patterns.
* Multi Head Attention: The self-attention mechanism is often used in multiple "heads”, in order to capture different types of patterns and relationships. Each head focuses on a different aspect of the data, allowing the model to learn multiple dependencies simultaneously.
* Output layer: produces the final forecast for each future time step. The output can be a single scalar value or a vector representing multiple forecasted values.

## Model Building

To create a model for timeseries forecasting, there are several parameters that need to be taken into account and when combined result into many different architectures that have to be explored.

Those parameters are the number of layers within the model, as well as the type of layers and the number of neurons or units in each layer. In this project we chose to use the vanilla LSTM type of RNN for several reasons. First and foremost, LSTM cells have the ability to capture long-term dependencies in sequential data, like in our case, where we want to study the behavior of the reaction wheels of the satellite throughout a whole year. Secondly, LSTM have the advantage that they don’t need a complex model to be able to capture complex patterns within a large dataset. Additionally, when there are sufficient data for training, they tend to perform even better. Finally, by adding a number of layers of LSTM cells within the model both short and long term trends can be captured.

Another type of layer that was used is the Dense layer, which is the basic type of layer in machine learning.

The following figure displays a straightforward representation of the long short-term memory network. Typically, an LSTM network refers to an RNN with long short-term memory blocks. These blocks are interconnected recurrently and are responsible for maintaining memory. Each memory block contains memory cells that store the state, an input gate for determining what to learn, a forget gate for deciding what to discard, and an output gate for regulating modifications to the content. The critical distinction between LSTM and traditional RNN lies in the inclusion of these gates. The LSTM network utilizes these gates to make informed decisions about preserving relevant information, whereas the traditional RNN merely overwrites its contents at each time step during the modeling process.

A diagram of a flowchart

Description automatically generated

Figure ‑ LSTM cell architecture

Here Xt: input time step, ht: output, Ct: cell state, ft: forget gate, it: input gate, ot: output gate, Ĉt : internal cell state. Operations inside light red circle are pointwise.

A diagram of a output sequence

Description automatically generated

Figure ‑ LSTM memory network

In the beginning, the LSTM cells in the first layer receive the initial input sequence of length of the time-window that we are using . Each LSTM cell also maintains an internal or hidden state and a memory cell. These states are initialized before processing the sequence. The input sequence is processed one time step at a time, moving from the beginning to the end of the sequence. At each time step, the LSTM cell performs the following operations:

* The cell calculates how much of the new input Xt at the current time step should be added to the memory cell. This is controlled by the input gate, which is a sigmoid function that takes into account the current input Xt and the previous hidden state ht-1.
* The cell decides how much of the previous memory cell state Ct-1 should be forgotten or retained. This is determined by the forget gate, another sigmoid function that considers the previous hidden state and the current input.
* The cell updates its memory cell state by combining the information from the input gate it and the forget gate ft. This step involves element-wise multiplication and addition operations.
* The cell determines how much of the memory cell's current state should be output as the new hidden state ht. This is controlled by the output gate, which is one more sigmoid function based on the input Xt and the previous hidden state ht-1.
* The final hidden state ht is computed based on the output gate and the current memory cell state. This hidden state becomes the output of the current LSTM cell and is also passed as input to the next time step's LSTM cell in the same layer or to the corresponding LSTM cell in the next layer.

The output ht of each LSTM cell in a layer serves as input to the LSTM cells in the next layer. This layer-to-layer movement enables the network to capture increasingly complex temporal patterns and dependencies as the data progresses through the layers.

Other parameters that affect the performance of the model are the learning rate and the optimizer that is used and the input window.

The window size of the input data for an LSTM network refers to the number of time steps or observations that the LSTM considers as a single input sequence. In other words, it determines how much historical information the LSTM can utilize to make predictions or learn patterns in the data. The window size is a crucial hyperparameter in LSTM networks, and selecting an appropriate value depends on the nature of the data and the specific task at hand. A smaller window size may be suitable for tasks with rapid changes, while a larger window size might be more appropriate for tasks with long-term dependencies. In our case, we experimented with window sizes of 24, 48 and 72, which translates to using one, two or three days of data as input, since our data are hourly measurements.

The learning rate is a hyperparameter that controls the step size at which the optimizer adjusts the model's parameters during training. It determines how much the model should update its weights in response to the calculated gradients. A larger learning rate allows for faster convergence during training but may also lead to overshooting and instability. On the other hand, a smaller learning rate leads to slower convergence but provides more stable updates.

The optimizer that was used is Adam (Adaptive Moment Estimation). It has the ability to adapt the learning rate for each parameter in the model. Instead of using a fixed learning rate, it maintains separate adaptive learning rates based on the historical gradients. In this way it allows the model to converge faster and more efficiently.

## Hellas-Sat 3

Hellas-Sat 3 is a telecommunications satellite operated by Hellas-Sat, that provides coverage to Europe, the Middle East and South Africa, by utilizing Ka and Ku bands and it provides Fixed Satellite Services as well as Broadband Satellite Services. The satellite is placed in a geostationary orbit at 39° East longitude and was launched on 28th of June 2017.

Being a telecommunications satellite, Hellas-Sat 3 requires high pointing accuracy to be able to fulfil its mission and provide satisfactory services. Moreover, the pointing accuracy and the orbital position need to be maintained at all times.

Maintaining the orbital position, also known as stationkeeping, is achieved with a series of scheduled manoeuvres, that utilize the propulsion system of the spacecraft. It is composed by thrusters that are positioned all around the satellite with the aim to apply external force to it in all possible directions. The manoeuvres can be in a North, South, East and West direction.

The satellite is three-axis stabilized in order to achieve and maintain the high pointing accuracy. To achieve that there are four reaction wheels on board the satellite, that rotate to exchange angular momentum with the satellite body. In this way any external torque that is applied due to perturbations such as solar wind pressure or gravitational forces from the moon can be counter-balanced by spinning up or down the reaction wheels, with the result of applying the opposite torque and maintaining the attitude of the satellite.

The reaction wheels are placed in pyramidal configuration, so that they can contribute to the angular momentum of the spacecraft in all three axis. Additionally, by having 4 reaction wheels, there is redundancy aboard the spacecraft and in case of malfunction of one the wheels, the satellite can continue to provide services and fulfil its mission.

However, the momentum that can be stored in each reaction wheel is limited and defined by their properties, such as mass and moment of inertia. Therefore, there are limits to how fast they can spin before being in danger of breaking down. The operational limit for the reaction wheels aboard Hellas-Sat 3 is ±4500 rpm, since they can spin in both directions, clockwise and counterclockwise.

Hellas-Sat engineers and operators aim to keep the wheel speed well below the constructor’s limit, close to ±4000 rpm, so that there is a substantial margin in case of contingency.

When the wheel speed gets close or over the limit they are considered to be saturated and they may be unable to provide the necessary torque to perform attitude adjustments. To avoid that, wheel unloading needs to take place frequently.

Wheel unloading is the process of reducing the accumulated angular momentum in reaction wheels on a satellite or spacecraft. By using the propulsion system of the spacecraft, thrusters are fired in specific directions to generate torques that counteract the wheel's angular momentum. These thrusters are used to unload momentum while conserving the satellite's limited propellant and preserving the attitude of the satellite.

It may be performed as a standalone procedure, but it can also be combined with stationkeeping manoeuvres, so that fuel is preserved. Such is the case on Hellas-Sat 3. During the North/South and the East/West manoeuvres, the reaction wheels take advantage of the thruster firings to offload their momentum. However, because the manoeuvres take place every two weeks, they is a need to perform the procedure on its own, between the scheduled manoeuvres.

## Tools

A variety of tools were used in order to process and visualize the data , as well as for building machine learning models , optimizing and testing them.

We used Python programming language because it is widely popular for data processing tasks, easily accessible and most importantly there are any libraries that are compatible with it and expand its capabilities.

Also a MySQL database was created locally to perform testing, because the database for the tickets that Hellas Sat is using is of the same type and therefore this is were the results would have to be input.

Some of the python libraries that were used are the following:

* Pandas: To import the data that were provided in csv format, filter them and separate them into different sub-dataset that were useful for machine learning.
* Matplotlib: To visualize the data and plot the results.
* Tensorflow and Keras: To build machine learning models and test them. It is simple and user-friendly and requires no experience in machine learning for someone to build and test a model. Also, it allows to stack layers on top of each other, so that complex model architectures can be made.
* Kerastuner: to perform hyperparameter optimization on the built models. It is a library within tensorflow, that provides the tools to perform hyper parameter tuning on machine learning models that were built using keras. It has customizable search space to make it fit to one’s needs and includes multiple searching algorithms.
* Pymysql: to interact and import in a MySQL library the results for the created ticket. It has the capability not only to connect to a MySQL database, but also to execute MySQL queries from python.
* Numpy: to execute mathematical functions whenever necessary.

# Methodology

In this chapter we will discuss the dataset that was used and how they were processed in order to be fit to use for training the models. Afterwards the process flow of how the models were built and assessed will be demonstrated.

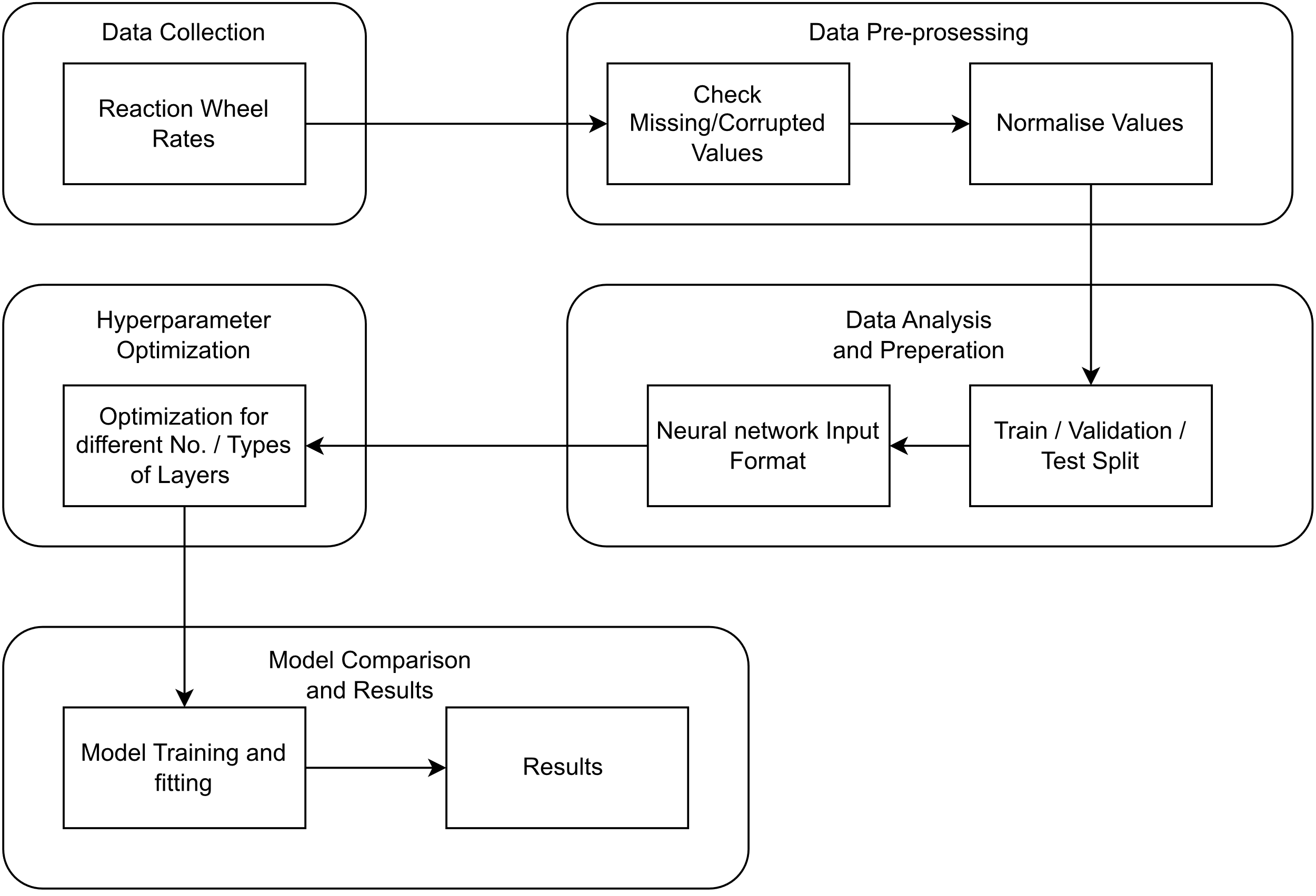


Figure ‑ Process diagram

## Data Collection

The data that were used were provided by Hellas-Sat and collected through their telemetry system for Hellas-Sat 3. The rates of all four reaction wheels were provided for a timespan of one year, from 1-1-2022 01:00 UTC to 31-12-2023 23:00 UTC. The measurements that were provided were the hourly rpm of each wheel, which was calculated by taking the mean of the minimum and the maximum value in a 60 minute window. For each wheel the dataset that was used contained 8749 measurements. Each reaction wheel has each own telemetry: AW1010R is RW 1, AW2010R is RW 2, AW3010R is RW3 and AW4010R is RW4.

|  |  |
| --- | --- |
| a)  A blue line graph with numbers and a white background  Description automatically generated | b)  A blue line graph with numbers and a white background  Description automatically generated |
| c)  A blue line graph with numbers and a white background  Description automatically generated | d)  A blue line graph with numbers and a white background  Description automatically generated |

Figure ‑ Reaction Wheel rates Jan 2022 - Dec 2022, a) Reaction Wheel 1, b) Reaction Wheel 2, c) Reaction Wheel 3, d) Reaction Wheel 4

Some noticeable features on the plots of the reaction wheel rates are the following:

* All reaction wheels follow a similar pattern across the year.
* The amplitude of the range in each season varies. The maximum value is smaller around April and September, while it is larger around January, July and November.
* In November all reaction wheels are being forced to rotate in the opposite direction from their prior direction. This is done on purpose by the operators for maintenance purposes.
* The sudden decrease in the rates of the wheels is either because a stationkeeping maneuver is performed, or because of a manual wheel unloading. When a stationkeeping maneuver is performed, automatically a wheel unloading is performed, so that the satellite can take advantage of the thrust and the torques that are applied on it and become more fuel efficient.

## Data Pre-Processing

The data were provided in CSV format and had to be converted into a dataframe, with the corresponding columns of date and time, telemetry code, and value in rpm. All data were scanned for missing or corrupted values and an additional step that had to be taken is that we took the absolute values of the measurements. This is related to the fact that the reaction wheels have the ability to rotate both clockwise and counter-clockwise and at all times two of the wheels are spinning CW and two CCW. For purposes of maintenance, once per year the satellite operators make each wheel spin in the opposite direction and therefore we had to filter out of the dataset this transition, in order to avoid confusing the model during the training, because this is not a normal behavior of the wheels that the model needs to be able to predict.

Additionally, during the process of training, experimenting and building the models we came to the conclusion that all values needed to be normalized with respect to the maximum value that the rate of the wheel is allowed to reach, as advised by the constructor and Hellas-Sat. In this way better performance of the models was achieved.

The necessity for normalization can be demonstrated with the following example. The same model was used for training and validation two times. It included 1 LSTM layer with 64 units, 2 Dense layers with 128 units each and a Dense layer with 1 unit which is the output and a learning rate of 0.001 with Adam optimizer. On the first iteration the values were not normalized, while on the second they were. In the following figures, the results of the training on the validation can be seen. We notice that on the non-normalized predictions in Figure 3-3, the predictions are cut-off when they reach higher values, while in the normalized model in the Figure 3-4 we see patterns in the predictions in the higher values, that mimic those of the actual data, that are being plotted below them.

A picture containing text, screenshot, line, font

Description automatically generated

Figure ‑ Non-normalized predictions for example LSTM model

A picture containing text, screenshot, line, font

Description automatically generated

Figure ‑ Normalized predictions for example LSTM model

## Data Analysis and Preparation

Moreover, before the data was fed to the models they had to be split into three subsets: Training, Validation and Testing. For training purposes 70% of the original dataset was used, for validation 20% and for testing 10%. The resulting subsets contained 6124 datapoints for training, 1750 for validation and 802 for testing.

Additionally, the data were reshaped in the appropriate way to be used as input in the models. LSTM models expect a 3D shaped input in the format of [batch size, time steps, features]. Batch size is the number of samples in each batch, time steps is the number of steps in each sequence and features is the number of measurable properties.

## Hyperparameter Optimization

In Table 3-1 all hyperparameters are listed, as well as the range of their values that were explored.

Table ‑ Optimized Hyperparamters

|  |  |
| --- | --- |
| Hyperparameter | Range |
| No. Layers | [1, 2, 3, 4, 5, 6] |
| Type of Layers | [Dense, LSTM] |
| No. LSTM units | [32, 64, 128, 192, 256] |
| No. Dense neurons | [32, 64, 128, 192, 256] |
| Learning Rate | [0.1, 0.01, 0.001] |
| Optimizer | [Adam] |
| Window Size | [24, 48, 72] |

For a more complete exploration of the possible architectures, a tool for hyperparameter optimization was created using kerastuner in pyhton that utilized Hyperband optimization, so that our research would be more extensive and would include as many candidate architectures as possible.

Hyperband's key idea is to quickly identify and eliminate poorly performing configurations, focusing resources on the most promising ones. This approach allows Hyperband to explore a wide range of hyperparameter configurations more efficiently than traditional grid search.

It starts by sampling a large number of random hyperparameter configurations and evaluates them for a small number of epochs, which is referred to as the "initial" number of epochs. After each epoch, it discards the poorly performing configurations and retains the top-performing configurations. The retained configurations are then allocated more resources (epochs) and trained further.

The process continues, progressively reducing the number of configurations but increasing the number of epochs allocated to each configuration at each round. Eventually, only the best configuration remains and is trained for a large number of epochs.

In summary, Hyperband does not exhaustively test all possible combinations of parameters. Instead, it combines random sampling with early elimination of underperforming configurations to find the best hyperparameter configuration within a specified resource budget.

In Hyperband, the number of configurations tested at each round is determined by a formula:

|  |  |
| --- | --- |
|  | (‑) |

n : the number of configurations at the current round

ceil : function that rounds a number up to the nearest integer

R : the maximum number of epochs specified for the search

eta : the factor parameter

s : the round number (starting from 0)

The factor parameter in the Hyperband tuner controls the resource allocation and termination rate of the hyperparameter search. It determines the ratio between the number of configurations to evaluate at each round of successive halving. Its purpose is to control the aggressive elimination of poorly performing configurations and the allocation of more resources to promising configurations.

Although the reaction wheels are similar to their characteristics a model for each one had to be created, because they are positioned in different spots within the spacecraft and therefore they experience and are affected by the perturbations in a unique manner each. In the following tables we can see the results of hyperparameter optimization of different architectures for each reaction wheel. As we can see, the results vary among the reaction wheels for similar architectures.

The architecture that was chosen was a multi-layered LSTM or “stacked” LSTM. In a “stacked” LSTM network, like the example shown in Figure 3-5, all input data are inserted in the first LSTM layer to generate hidden states, which are then used as inputs in the next LSTM layer and so on, unlit we get to the output layer. By adding more hidden layers, we add greater model depth and can observe the problem in different time scales. After analyzing the data for the reaction wheel rates, it was observed that there are some patterns emerging with different frequencies. A daily, weekly, bi-weekly and monthly periodicity are observable as well as a fluctuation in the maximum value that is reached in each epoch throughout the year. Therefore, with the aim to capture these dependencies in the model, several “stacked” architectures were tested with different number of hidden layers. The results for all 4 reaction wheels from the hyperparameter optimization regarding the number of layers, units and trainable parameters are displayed in Tables 3-2 to 3-5 below.



Figure ‑ Multi-Layered LSTM model architecture

Table ‑ Reaction Wheel 1 architecture results of hyperparameter optimization

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Network | Window Size | Layers | Output Shape | No. of Parameters | Total Parameters | Trainable Parameters |
| RNN | (24, 1) | LSTM | (None, 32) | 7296 | 7329 | 7329 |
|  |  | Dense | (None, 1) | 33 |  |  |
| RNN | (48, 1) | LSTM | (None, 256) | 312320 | 312577 | 312577 |
|  |  | Dense | (None, 1) | 257 |  |  |
| RNN | (48, 1) | LSTM | (None, 64) | 28928 | 45825 | 45825 |
|  |  | Dense | (None, 256) | 16640 |  |  |
|  |  | Dense | (None, 1) | 257 |  |  |
| RNN | (24, 1) | LSTM | (None, 64) | 16896 | 41857 | 41857 |
|  |  | Dense | (None, 128) | 8320 |  |  |
|  |  | Dense | (None, 128) | 16512 |  |  |
|  |  | Dense | (None, 1) | 129 |  |  |
| RNN | (48, 1) | LSTM | (None, None, 128) | 90624 | 119937 | 119937 |
|  |  | LSTM | (None, 32) | 20608 |  |  |
|  |  | Dense | (None, 256) | 8448 |  |  |
|  |  | Dense | (None, 1) | 257 |  |  |
| RNN | (72, 1) | LSTM | (None, None, 192) | 203520 | 273537 | 273537 |
|  |  | LSTM | (None, 64) | 65792 |  |  |
|  |  | Dense | (None, 64) | 4160 |  |  |
|  |  | Dense | (None, 1) | 65 |  |  |
| RNN | (72, 1) | LSTM | (None, None, 64) | 35072 | 245121 | 245121 |
|  |  | LSTM | (None, None, 32) | 12416 |  |  |
|  |  | LSTM | (None, 192) | 172800 |  |  |
|  |  | Dense | (None, 128) | 24704 |  |  |
|  |  | Dense | (None, 1) | 129 |  |  |
| RNN | (72, 1) | LSTM | (None, None, 64) | 35072 | 267585 | 267585 |
|  |  | LSTM | (None, None, 192) | 197376 |  |  |
|  |  | LSTM | (None, 32) | 28800 |  |  |
|  |  | Dense | (None, 64) | 2112 |  |  |
|  |  | Dense | (None, 64) | 4160 |  |  |
|  |  | Dense | (None, 1) | 65 |  |  |

Table ‑ Reaction Wheel 2 architecture results of hyperparameter optimization

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Network | Window Size | Layers | Output Shape | No. of Parameters | Total Parameters | Trainable Parameters |
| RNN | (24, 1) | LSTM | (None, 256) | 287744 | 288001 | 288001 |
|  |  | Dense | (None, 1) | 257 |  |  |
| RNN | (48, 1) | LSTM | (None, 64) | 28928 | 28993 | 28993 |
|  |  | Dense | (None, 1) | 65 |  |  |
| RNN | (48, 1) | LSTM | (None, 128) | 90624 | 94785 | 94785 |
|  |  | Dense | (None, 32) | 4128 |  |  |
|  |  | Dense | (None, 1) | 33 |  |  |
| RNN | (24, 1) | LTSM | (None, 64) | 16896 | 50049 | 50049 |
|  |  | Dense | (None, 256) | 16640 |  |  |
|  |  | Dense | (None, 64) | 16448 |  |  |
|  |  | Dense | (None, 1) | 65 |  |  |
| RNN | (48, 1) | LTSM | (None, None, 192) | 185088 | 382721 | 382721 |
|  |  | LTSM | (None, 128) | 164352 |  |  |
|  |  | Dense | (None, 256) | 33027 |  |  |
|  |  | Dense | (None, 1) | 257 |  |  |
| RNN | (72, 1) | LTSM | (None, None , 192) | 203520 | 233409 | 233409 |
|  |  | LTSM | (None, 32) | 28800 |  |  |
|  |  | Dense | (None, 32) | 1056 |  |  |
|  |  | Dense | (None, 1) | 33 |  |  |
| RNN | (72, 1) | LSTM | (None, None, 32) | 13440 | 217217 | 217217 |
|  |  | LSTM | (None, None, 192) | 172800 |  |  |
|  |  | LSTM | (None, 32) | 28800 |  |  |
|  |  | Dense | (None, 64) | 2112 |  |  |
|  |  | Dense | (None, 1) | 65 |  |  |
| RNN | (72, 1) | LSTM | (None, None, 128) | 102912 | 538625 | 538625 |
|  |  | LSTM | (None, None, 192) | 246528 |  |  |
|  |  | LSTM | (None, 128) | 164352 |  |  |
|  |  | Dense | (None, 128) | 16512 |  |  |
|  |  | Dense | (None, 64) | 8256 |  |  |
|  |  | Dense | (None, 1) | 65 |  |  |

Table ‑ Reaction Wheel 3 architecture results of hyperparameter optimization

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Network | Window Size | Layers | Output Shape | No. of Parameters | Total Parameters | Trainable Parameters |
| RNN | (24, 1) | LSTM | (None, 32) | 7296 | 7329 | 7329 |
|  |  | Dense | (None, 1) | 33 |  |  |
| RNN | (48, 1) | LSTM | (None, 256) | 312320 | 312577 | 312577 |
|  |  | Dense | (None, 1) | 257 |  |  |
| RNN | (48, 1) | LSTM | (None, 128) | 90624 | 107265 | 107265 |
|  |  | Dense | (None, 128) | 16512 |  |  |
|  |  | Dense | (None, 1) | 129 |  |  |
| RNN | (24, 1) | LSTM | (None, 128) | 66560 | 149121 | 149121 |
|  |  | Dense | (None, 256) | 33024 |  |  |
|  |  | Dense | (None, 192) | 49344 |  |  |
|  |  | Dense | (None, 1) | 193 |  |  |
| RNN | (48, 1) | LSTM | (None, None, 256) | 312320 | 706817 | 706817 |
|  |  | LSTM | (None, 192) | 344832 |  |  |
|  |  | Dense | (None, 256) | 49408 |  |  |
|  |  | Dense | (None, 1) | 257 |  |  |
| RNN | (72, 1) | LSTM | (None, None, 192) | 203520 | 536449 | 536449 |
|  |  | LSTM | (None, 192) | 295680 |  |  |
|  |  | Dense | (None, 192) | 37056 |  |  |
|  |  | Dense | (None, 1) | 193 |  |  |
| RNN | (72, 1) | LSTM | (None, None, 192) | 203520 | 417537 | 417537 |
|  |  | LSTM | (None, None, 32) | 28800 |  |  |
|  |  | LSTM | (None, 192) | 172800 |  |  |
|  |  | Dense | (None, 64) | 12352 |  |  |
|  |  | Dense | (None, 1) | 65 |  |  |
| RNN | (72, 1) | LSTM | (None, None, 64) | 35072 | 353825 | 353825 |
|  |  | LSTM | (None, None, 32) | 12416 |  |  |
|  |  | LSTM | (None, 256) | 295936 |  |  |
|  |  | Dense | (None, 32) | 8224 |  |  |
|  |  | Dense | (None, 64) | 2112 |  |  |
|  |  | Dense | (None, 1) | 65 |  |  |

Table ‑ Reaction Wheel 4 architecture results of hyperparameter optimization

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Network | Window Size | Layers | Output Shape | No. of Parameters | Total Parameters | Trainable Parameters |
| RNN | (24, 1) | LSTM | (None, 192) | 166656 | 166849 | 166849 |
|  |  | Dense | (None, 1) | 193 |  |  |
| RNN | (48, 1) | LSTM | (None, 32) | 10368 | 10401 | 10401 |
|  |  | Dense | (None, 1) | 33 |  |  |
| RNN | (48, 1) | LSTM | (None, 256) | 312320 | 328833 | 328833 |
|  |  | Dense | (None, 64) | 16448 |  |  |
|  |  | Dense | (None, 1) | 65 |  |  |
| RNN | (24, 1) | LSTM | (None, 128) | 66560 | 79393 | 79393 |
|  |  | Dense | (None, 32) | 4128 |  |  |
|  |  | Dense | (None, 256) | 8448 |  |  |
|  |  | Dense | (None,1) | 257 |  |  |
| RNN | (48, 1) | LSTM | (None, None, 128) | 90624 | 148481 | 148481 |
|  |  | LSTM | (None, 64) | 49408 |  |  |
|  |  | Dense | (None, 128) | 8320 |  |  |
|  |  | Dense | (None, 1) | 129 |  |  |
| RNN | (72, 1) | LSTM | (None, None, 192) | 203520 | 281985 | 281985 |
|  |  | LSTM | (None, 64) | 65792 |  |  |
|  |  | Dense | (None, 192) | 12480 |  |  |
|  |  | Dense | (None, 1) | 193 |  |  |
| RNN | (72, 1) | LSTM | (None, None, 256) | 336896 | 764289 | 764289 |
|  |  | LSTM | (None, None, 64) | 82176 |  |  |
|  |  | LSTM | (None, 256) | 328704 |  |  |
|  |  | Dense | (None, 64) | 16448 |  |  |
|  |  | Dense | (None, 1) | 65 |  |  |
| RNN | (72, 1) | LSTM | (None, None, 192) | 203520 | 396865 | 396865 |
|  |  | LSTM | (None, None, 128) | 164352 |  |  |
|  |  | LSTM | (None, 32) | 20608 |  |  |
|  |  | Dense | (None, 128) | 4224 |  |  |
|  |  | Dense | (None, 32) | 4128 |  |  |
|  |  | Dense | (None, 1) | 33 |  |  |

## Model Comparison

The next step from hyperparameter optimization was to assess the resulting models for each type of architecture. Assessing machine learning models is an important step in evaluating their performance and to determine whether they are suitable for our project. The metrics that were used for the evaluation are Root Mean Squared Error, Loss and Validation Loss.

Root Mean Squared Error (RMSE) is a metric used to measure the accuracy of a model’s predictions compared to the actual values. It calculates the square root of the average of squared differences between predicted values and actual values. A lower RMSE indicates better performance of the model.

Loss refers to the objective function used during the training process. In LSTM models it is usually Mean Squared Error. During training the model tries to minimize this loss, making the predictions closer to the actual values. Monitoring the loss during training allows you to understand how well the model is learning from the data.

Validation loss is computed using the same loss function as during training, but it is calculated on the validation dataset. This dataset is not used for training the model, and its purpose is to evaluate how well the model generalizes to unseen data. The validation loss helps you detect overfitting, when the model performs well on the training data but poorly on new data.

In the following Tables 3-6 to 3-9 are displayed the results from all the models that were trained regarding the validation loss and the Root Mean Squared Error. As we can see there is no absolute correlation between number of layers, window size and lower validation error and RMSE.

Generally having a greater window size, which means more data as input used to make the next prediction, results in better performance, but not always. There are cases where a 48-hour window has a better performance than a 72-hour window with the same model architecture.

Additionally, more layers don’t necessarily mean a better predicting performance for the models. This means that adding more complexity to the structure of the model is not mandatory to get good results. Simpler models with just two LSTM layers and two Dense layers are enough for some of the wheels’ models to be able to capture the complexity of the data in a satisfactory level.

Finally, it is important to note that not all reaction wheels models had the best performance with the same architecture. That is indicative of the fact that each reaction wheel is affected by environmental effects in a unique way and since each model is trained on dedicated data for each reaction wheel, it is expected to have different results.

Table ‑ Reaction Wheel 1 Architectures Validation Loss and RMSE

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Network | Window Size | Layers | Val. Loss | RMSE |
| RNN | (24, 1) | LSTM | 0.010 | 0.100 |
|  |  | Dense |  |  |
| RNN | (48, 1) | LSTM | 0.015 | 0.122 |
|  |  | Dense |  |  |
| RNN | (48, 1) | LSTM | 0.00087 | 0.029 |
|  |  | Dense |  |  |
|  |  | Dense |  |  |
| RNN | (24, 1) | LSTM | 0.0013 | 0.036 |
|  |  | Dense |  |  |
|  |  | Dense |  |  |
|  |  | Dense |  |  |
| RNN | (48, 1) | LSTM | 0.00092 | 0.030 |
|  |  | LSTM |  |  |
|  |  | Dense |  |  |
|  |  | Dense |  |  |
| RNN | (72, 1) | LSTM | 0.00086 | 0.029 |
|  |  | LSTM |  |  |
|  |  | Dense |  |  |
|  |  | Dense |  |  |
| RNN | (72, 1) | LSTM | 0.00089 | 0.030 |
|  |  | LSTM |  |  |
|  |  | LSTM |  |  |
|  |  | Dense |  |  |
|  |  | Dense |  |  |
| RNN | (72, 1) | LSTM | 0.00083 | 0.029 |
|  |  | LSTM |  |  |
|  |  | LSTM |  |  |
|  |  | Dense |  |  |
|  |  | Dense |  |  |
|  |  | Dense |  |  |

Table ‑ Reaction Wheel 2 Architectures Validation Loss and RMSE

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Network | Window Size | Layers | Val. Loss | RMSE |
| RNN | (24, 1) | LSTM | 0.00158 | 0.040 |
|  |  | Dense |  |  |
| RNN | (48, 1) | LSTM | 0.00214 | 0.046 |
|  |  | Dense |  |  |
| RNN | (48, 1) | LSTM | 0.00131 | 0.036 |
|  |  | Dense |  |  |
|  |  | Dense |  |  |
| RNN | (24, 1) | LTSM | 0.00099 | 0.031 |
|  |  | Dense |  |  |
|  |  | Dense |  |  |
|  |  | Dense |  |  |
| RNN | (48, 1) | LTSM | 0.00147 | 0.038 |
|  |  | LTSM |  |  |
|  |  | Dense |  |  |
|  |  | Dense |  |  |
| RNN | (72, 1) | LTSM | 0.00150 | 0.039 |
|  |  | LTSM |  |  |
|  |  | Dense |  |  |
|  |  | Dense |  |  |
| RNN | (72, 1) | LSTM | 0.00167 | 0.041 |
|  |  | LSTM |  |  |
|  |  | LSTM |  |  |
|  |  | Dense |  |  |
|  |  | Dense |  |  |
| RNN | (72, 1) | LSTM | 0.00160 | 0.040 |
|  |  | LSTM |  |  |
|  |  | LSTM |  |  |
|  |  | Dense |  |  |
|  |  | Dense |  |  |
|  |  | Dense |  |  |

Table ‑ Reaction Wheel 3 Architectures Validation Loss and RMSE

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Network | Window Size | Layers | Val. Loss | RMSE |
| RNN | (24, 1) | LSTM | 0.00136 | 0.037 |
|  |  | Dense |  |  |
| RNN | (48, 1) | LSTM | 0.00178 | 0.042 |
|  |  | Dense |  |  |
| RNN | (48, 1) | LSTM | 0.00116 | 0.034 |
|  |  | Dense |  |  |
|  |  | Dense |  |  |
| RNN | (24, 1) | LSTM | 0.00093 | 0.030 |
|  |  | Dense |  |  |
|  |  | Dense |  |  |
|  |  | Dense |  |  |
| RNN | (48, 1) | LSTM | 0.00120 | 0.035 |
|  |  | LSTM |  |  |
|  |  | Dense |  |  |
|  |  | Dense |  |  |
| RNN | (72, 1) | LSTM | 0.00123 | 0.035 |
|  |  | LSTM |  |  |
|  |  | Dense |  |  |
|  |  | Dense |  |  |
| RNN | (72, 1) | LSTM | 0.00125 | 0.035 |
|  |  | LSTM |  |  |
|  |  | LSTM |  |  |
|  |  | Dense |  |  |
|  |  | Dense |  |  |
| RNN | (72, 1) | LSTM | 0.00127 | 0.036 |
|  |  | LSTM |  |  |
|  |  | LSTM |  |  |
|  |  | Dense |  |  |
|  |  | Dense |  |  |
|  |  | Dense |  |  |

Table ‑ Reaction Wheel 4 Architectures Validation Loss and RMSE

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Network | Window Size | Layers | Val. Loss | RMSE |
| RNN | (24, 1) | LSTM | 0.00129 | 0.036 |
|  |  | Dense |  |  |
| RNN | (48, 1) | LSTM | 0.00195 | 0.044 |
|  |  | Dense |  |  |
| RNN | (48, 1) | LSTM | 0.00111 | 0.033 |
|  |  | Dense |  |  |
|  |  | Dense |  |  |
| RNN | (24, 1) | LSTM | 0.00083 | 0.029 |
|  |  | Dense |  |  |
|  |  | Dense |  |  |
|  |  | Dense |  |  |
| RNN | (48, 1) | LSTM | 0.00123 | 0.035 |
|  |  | LSTM |  |  |
|  |  | Dense |  |  |
|  |  | Dense |  |  |
| RNN | (72, 1) | LSTM | 0.00120 | 0.035 |
|  |  | LSTM |  |  |
|  |  | Dense |  |  |
|  |  | Dense |  |  |
| RNN | (72, 1) | LSTM | 0.00127 | 0.036 |
|  |  | LSTM |  |  |
|  |  | LSTM |  |  |
|  |  | Dense |  |  |
|  |  | Dense |  |  |
| RNN | (72, 1) | LSTM | 0.00126 | 0.035 |
|  |  | LSTM |  |  |
|  |  | LSTM |  |  |
|  |  | Dense |  |  |
|  |  | Dense |  |  |
|  |  | Dense |  |  |

## Ticket Creation

Once the models for each reaction wheel are built and trained, we can make predictions for each one, to detect if their rate is going to exceed a specific threshold and then create a ticket in the Spacecraft Control Centre with the necessary information, so that the engineers, operators and flight dynamics team can act accordingly. The flow diagram below, Figure 3-6, explains the procedure in which the ticket will be created and the steps that need to take place.

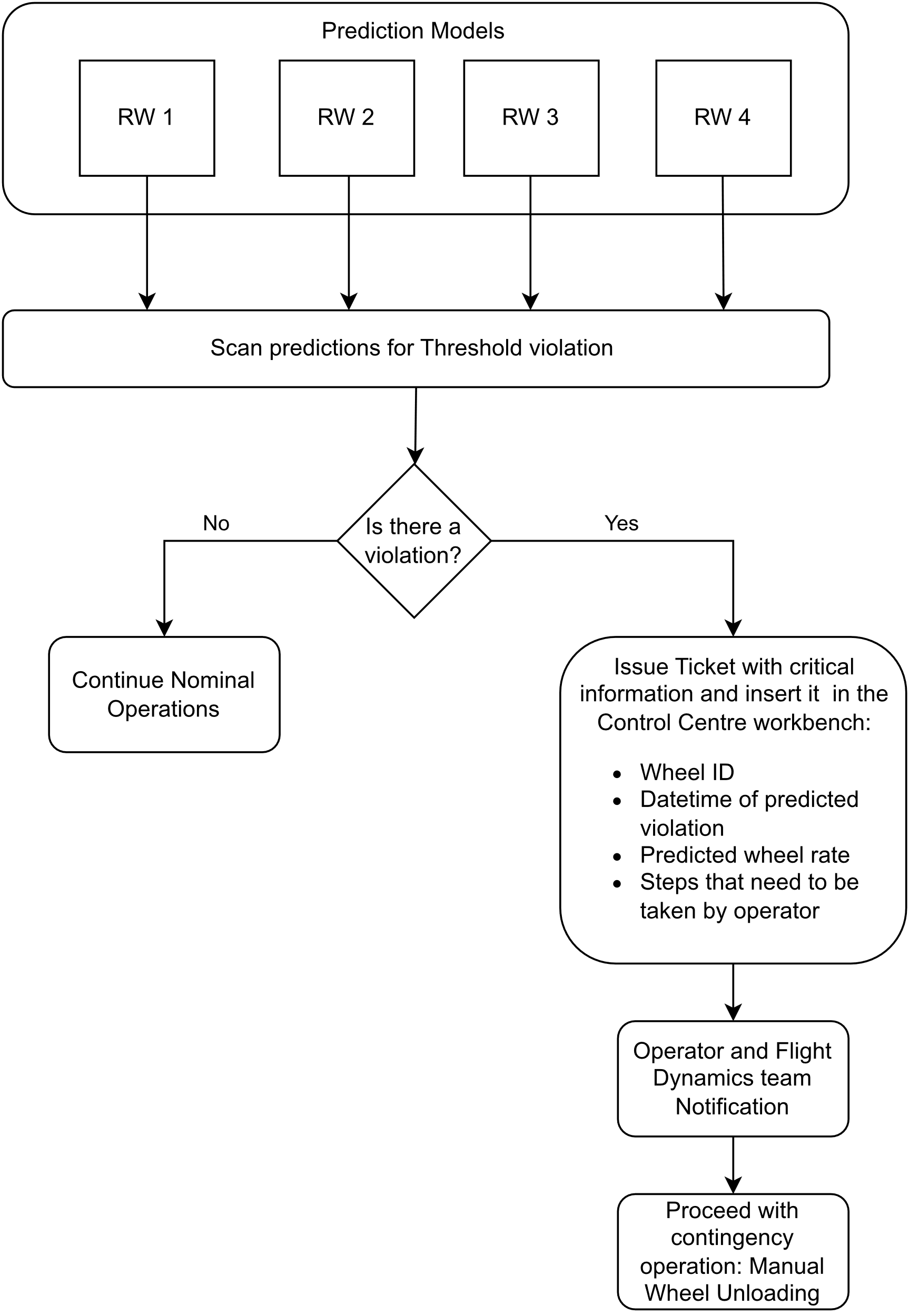


Figure ‑ Ticket creation flow process

Firstly, the predictions are created for each Reaction Wheel for a certain period. Then the results need to be examined, in order to assess if any of the predictions exceed a set threshold.

To make predictions a rolling input window system was created, as shown in Figure 3-7. To start making the predictions the hourly Reaction Wheel rates from the last 72 hours are needed. Then using these as input data a prediction is made for the next hour, which can be then used along side with the last 71 hours, with the aim to predict the next hour. This procedure can be followed another 70 times, 71 in total, so that we can get a prediction in total for the next 71 hours, approximately 3 days. No further predictions should be made, because they would be based only on already predicted values and no real data.

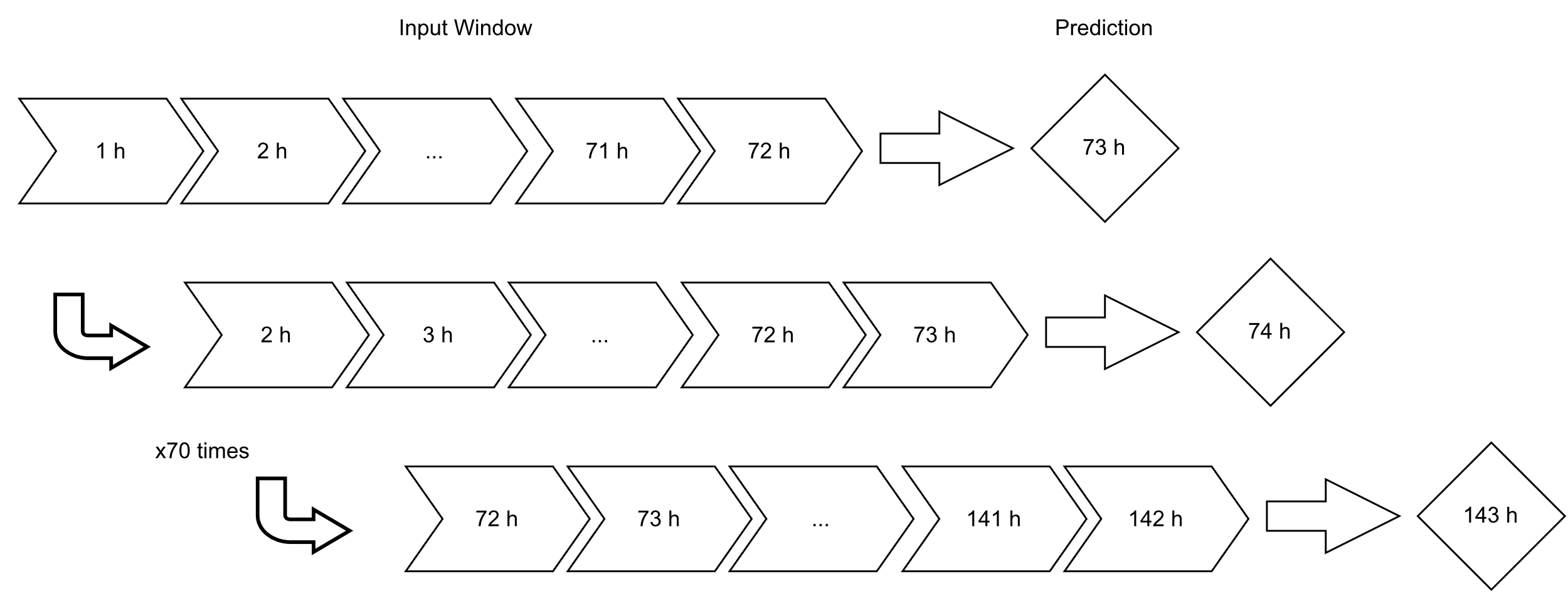


Figure ‑ Rolling input window for predictions

If none of the wheels exceed the set limit, then no action needs to be taken and operations can proceed nominally. However, if the predictions indicate that one of the wheels will surpass the limit at a specific time, a ticket needs to be created in the operation center workbench, so that the operators can be notified that a manual wheel unloading will need to take place.

To achieve that a process of scanning the prediction results was created, which identifies which wheel will exceed the limit and when, and then creates a ticket with the appropriate information on the workbench, by creating an entry in the right tables in its database.

A ticket is produced only for the most imminent threshold violation in one of the reaction wheels. This is due to the fact that when a wheel unloading takes place on the spacecraft, all four wheels are being unloaded at the same time. Therefore, if the procedure is executed for the most imminent limit violation chronologically, then afterwards the rates of all the wheels are being affected and the predictions that were made before become invalid, as well as any alarms that the threshold will be reached.

The information that is input is the date of the procedure, which is the date that the rate will pass the threshold, the predicted rate of the wheel, which wheel will it be and the steps of the procedure that will need to be followed.

The procedure that the operator will have to execute is 4000CA02 and includes 11 steps which are also included in the ticket that is produced and displayed on the workbench:

• Step 1: Initialize procedure 4000CA02.

• Step 2: Verify Observability. Verification of the housekeeping packet that is needed for the operation. This is done to ensure that all telemetry that is needed for the operation are downlinked. Need to activate the table DUMP NM\_DMP for the operation follow-up and in particular the duty cycle parameters checking.

• Step 3: Deactivate automatic wheel unloading. The automatic wheel unloading is disabled to avoid the random execution of the wheel unloading during the procedure.

• Step 4: Configure wheels unloading in the direction (East, West, North , South) that has been decided by the flight dynamics team. The maneuver direction shall be set to select the thruster set used during the wheel unloading. If it is not provided, the procedure will be rejected

• Step 5: Activate wheels unloading. Wheel unloading starts as soon as telecommand 3002FHA is received on board. While wheel unloading has started, procedure execution has to be continued in order to monitor wheel unloading behavior.

• Step 6: Monitor wheels unloading. The wheel unloading is check thanks to specific telemetry: spacecraft attitude, wheel angular momentum, on-control thruster valves status and temperature. If these telemetry are not correct during the wheel unloading the operator has to abort the procedure.

• Step 7: Verify end of wheels unloading. The thrusters history is recorded to know wheel unloading fuel consumption.

• Step 8: Reset boost parameters. The boost parameters must be reset after the boost to avoid performing any operational mistake.

• Step 9: Re-activate automatic wheels unloading. Set back on automatic wheel unloading, which was deactivated in the beginning of the procedure.

• Step 10: Deactivate observability. Come back to the default spacecraft telemetry plan by stopping NM\_DM table DUMP.

• Step 11: End procedure.

The ticket shall be marked as unverified, so that the flight dynamics team can process the information that is given and decide in what direction the maneuver for the unloading will be performed and if there is a more efficient time to execute the procedure than the one that is predicted. Also in consideration needs to be taken whether there are any other procedures that have to be executed on the same time and if the wheel unloading needs to be prioritized.

# Results

## Model Selection

According to the results from the training of the models that we mentioned in section 3.5, the following architectures were chosen for the predictions of each reaction wheel. In all cases, the input window that was chosen was 72 hours, even though in some cases a 48-hour input window resulted in slightly lower validation losses and RMSE. This choice was made, with the aim to make trustworthy predictions for the longest period of time possible, by using the rolling window method that was described in section 3.6.

### Reaction Wheel 1

A structure of three LSTM layers followed by 3 Dense layers was the most efficient, with validation loss 0.00083 and RMSE 0.029. The first LSTM layer is composed of 64 units, the second LSTM composed of 192 units, while the third composed of 32 units. The following first and second Dense layers have 64 neurons each and the final Dense layer is the output layer and has just 1 neuron.

Table ‑ Selected model architecture for Reaction Wheel 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Network | Window Size | Layers | Output Shape | Val. Loss | RMSE |
| RNN | (72, 1) | LSTM | (None, None, 64) | 0.00083 | 0.029 |
|  |  | LSTM | (None, None, 192) |  |  |
|  |  | LSTM | (None, 32) |  |  |
|  |  | Dense | (None, 64) |  |  |
|  |  | Dense | (None, 64) |  |  |
|  |  | Dense | (None, 1) |  |  |

The model was trained for 100 epochs and the results for the training and validation data, as well as the losses and the RMSE, are displayed in Figures 4-1 to 4-3.

A graph of a training loss and validation loss

Description automatically generated

Figure ‑ RW1 Training and Validation Loss & RMSE at 100 epochs

A blue lines with white text

Description automatically generated with medium confidence

Figure ‑ RW1 Training dataset Predictions (top) and Actual Data (bottom)

A graph of blue lines

Description automatically generated with medium confidence

Figure ‑ RW1 Validation dataset Predictions (top) and Actual Data (bottom)

### Reaction Wheel 2

A structure of two LSTM layers followed by two Dense layers was the most efficient, with validation loss 0.0015 and RMSE 0.039. The first LSTM layer is composed of 192 units and the second LSTM composed of 32 units. The following Dense layer has 32 neurons and the final Dense layer is the output layer and has just 1 neuron.

Table ‑ Selected model architecture for Reaction Wheel 2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Network | Window Size | Layers | Output Shape | Val. Loss | RMSE |
| RNN | (72, 1) | LTSM | (None, None , 192) | 0.0015 | 0.039 |
|  |  | LTSM | (None, 32) |  |  |
|  |  | Dense | (None, 32) |  |  |
|  |  | Dense | (None, 1) |  |  |

The model was trained for 100 epochs and the results for the training and validation data, as well as the losses and the RMSE, are displayed in Figures 4-4 to 4-6.

A graph of a training loss and validation loss

Description automatically generated

Figure ‑ RW2 Training and Validation Loss & RMSE at 100 epochs

A graph of blue lines

Description automatically generated with medium confidence

Figure ‑ RW2 Training dataset Predictions (top) and Actual Data (bottom)

A graph of blue lines

Description automatically generated with medium confidence

Figure ‑ RW2 Validation dataset Predictions (top) and Actual Data (bottom)

### Reaction Wheel 3

A structure of three LSTM layers followed by two Dense layers was the most efficient, with validation loss 0.00125 and RMSE 0.035. The first LSTM layer is composed of 192 units, the second LSTM composed of 32 units, while the third composed of 192 units. The Dense layers has 64 neurons and the final Dense layer is the output layer and has just 1 neuron.

Table ‑ Selected model architecture for Reaction Wheel 3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Network | Window Size | Layers | Output Shape | Val. Loss | RMSE |
| RNN | (72, 1) | LSTM | (None, None, 192) | 0.00125 | 0.035 |
|  |  | LSTM | (None, None, 32) |  |  |
|  |  | LSTM | (None, 192) |  |  |
|  |  | Dense | (None, 64) |  |  |
|  |  | Dense | (None, 1) |  |  |

The model was trained for 100 epochs and the results for the training and validation data, as well as the losses and the RMSE, , are displayed in Figures 4-7 to 4-9.

A graph of a training loss and validation loss

Description automatically generated with medium confidence

Figure ‑ RW3 Training and Validation Loss & RMSE at 100 epochs

A blue lines with white text

Description automatically generated with medium confidence

Figure ‑ RW3 Training dataset Predictions (top) and Actual Data (bottom)

A graph of blue lines

Description automatically generated with medium confidence

Figure ‑ RW3 Validation dataset Predictions (top) and Actual Data (bottom)

### Reaction Wheel 4

A structure of two LSTM layers followed by two Dense layers was the most efficient, with validation loss 0.0015 and RMSE 0.039. The first LSTM layer is composed of 192 units and the second LSTM composed of 32 units. The following Dense layer has 32 neurons and the final Dense layer is the output layer and has just 1 neuron.

Table ‑ Selected model architecture for Reaction Wheel 4

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Network | Window Size | Layers | Output Shape | Val. Loss | RMSE |
| RNN | (72, 1) | LSTM | (None, None, 192) | 0.0012 | 0.035 |
|  |  | LSTM | (None, 64) |  |  |
|  |  | Dense | (None, 192) |  |  |
|  |  | Dense | (None, 1) |  |  |

The model was trained for 100 epochs and the results for the training and validation data, as well as the losses and the RMSE, , are displayed in Figures 4-10 to 4-12.

A graph of a training loss and validation loss

Description automatically generated with medium confidence

Figure ‑ RW4 Training and Validation Loss & RMSE at 100 epochs

A graph of blue lines

Description automatically generated with medium confidence

Figure ‑ RW4 Training dataset Predictions (top) and Actual Data (bottom)

A graph of blue lines

Description automatically generated with medium confidence

Figure ‑ RW4 Validation dataset Predictions (top) and Actual Data (bottom)

## Predictions

Using the abovementioned models for each reaction wheel, we can make predictions for their future speed rates, so that we can subsequently survey them to check if any of the wheels will need unloading.

To create predictions to showcase the functionality of the ticket creation and the whole pipeline, we used the test dataset, because it is constituted of data that any of the models have never interacted with. In this way we can also assess the performance of the models in never-before-seen data.

The threshold that was set is 4000 rpm. Figures 4-13, 4-14, 4-15 and 4-16 display the predictions that were made for each reaction wheel respectively by using their developed moedls, when their test dataset was used as an input.

A graph of blue lines

Description automatically generated with medium confidence

Figure ‑ RW1 Test dataset Predictions (top) and Actual Data (bottom)

A graph of blue lines

Description automatically generated with medium confidence

Figure ‑ RW2 Test dataset Predictions (top) and Actual Data (bottom)

A graph of blue lines

Description automatically generated with medium confidence

Figure ‑ RW3 Test dataset Predictions (top) and Actual Data (bottom)

A graph showing the results of a performance

Description automatically generated with medium confidence

Figure ‑ RW4 Test dataset Predictions (top) and Actual Data (bottom)

After surveying the predictions for each reaction wheel, the results thar are shown in Figure 4-17, show that all of them at some point will surpass the limit that was set and almost all of them more than a single time.

|  |  |
| --- | --- |
| a) | b) |
| c) | d) |

Figure ‑ Survey results for detected threshold violation instances a) Reaction Wheel 1, b) Reaction Wheel 2, c) Reaction Wheel 3, d) Reaction Wheel 4

However, only the first instance time-wise is important, because if a wheel unloading is performed at that point, all wheels will offload their momentum and reduce their rates. Therefore, any predictions that were made for after that time are obsolete. As we can see from the survey results, the first instance of limit violation is going to happen at reaction wheel 2, in 210 hours from the moment of creating the predictions or at the date that is calculated in J2000 format.

## Ticket

Having acquired the critical information of which reaction wheel is predicted to surpass the threshold and when, we create a ticket on the operator’s workbench to notify them about the situation and make the aware of the procedure they will have to execute. In the ticket are also included the steps of the procedure they will have to follow.

In Tables 4-5 and 4-6, we can see the inputs on the MySQL database that were created, with the critical information for the procedure and the steps that the operator has to take, to execute it successfully.

Table ‑ Database entry with critical operation information

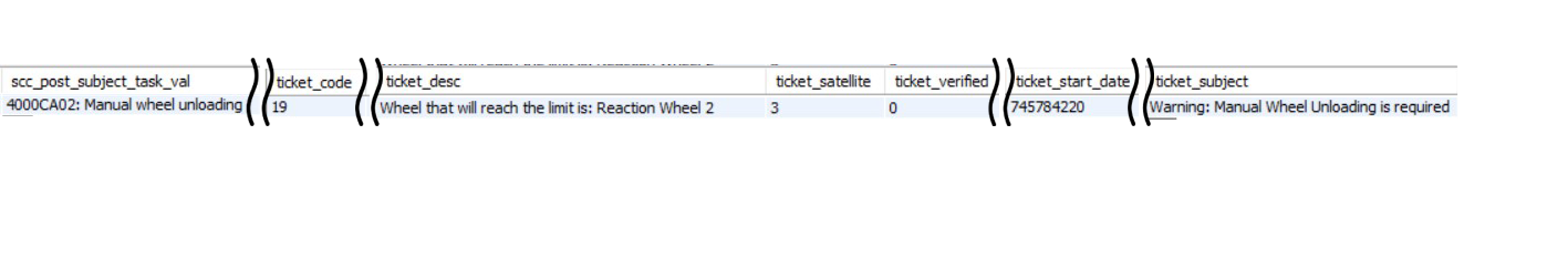


Table ‑ Database entry with procedure steps

A screenshot of a computer

Description automatically generated

## Current Solution

Currently there are no systems or solutions implemented in Satellite Control Center for Hellas-Sat 3 that can provide predictions regarding the rates of the reaction wheels, or any other kind of telemetry. With that being said, in case of a contingency or if a trend of increasing rates is observed by one of the operators, nothing can be done proactively except for waiting to see how the trend will evolve and take action afterwards, hoping that it is not too late.

By implementing our solution, accurate and trustworthy predictions can be made for the reaction wheels’ rates up to three days ahead and a ticket is created providing important information to the operators and notifying them about the situation. It is a reasonable amount of time for engineers to analyse the situation and estimate the status of the spacecraft, in order to investigate if there are any contingencies and make the correct decisions. The system can be used to get a first assessment of the evolution of any alarming rates, but can also be used repeatedly as the critical moment comes closer, in order to get even more precise predictions.

During a demonstration of our solution to Hellas Sat satellite engineers, we received the following feedback: “*Being able to predict how the rates of the reaction wheels will evolve, in the next hour and even better for the following days is a very powerful tool. We recently had a contingency on one of the other satellites, where the rates of the reaction wheels had reached the operational limit that we have imposed. We could not take any action and just had to wait to see how the rates will evolve in the next hour and then plan to execute a contingency procedure if it would be necessary. Had we had the tool that you have developed, we would be able to have foreseen this situation and plan accordingly beforehand. Even if we hadn’t done that, we would be able to use the tool in the last minute, to make predictions for the following hours and execute a contingency procedure before it is too late. Thankfully the rates of the reaction wheels started decreasing and no further action needed to be taken.*”

# Conclusions

## Summary

## Limitations

## Future work

References

Click here to enter reference list.

Appendices

Ethical approval letter

A letter with a blue circle and text

Description automatically generated with medium confidence

Appendix guidance (Appendix title (use Heading 7))

Appendix section (use Heading 8)

Appendix subsection (use Heading 9)

Creating captions in appendices

If you have chosen to include chapter numbers in your captions, then follow the instructions given here to apply the same format to the captions in your appendices. This section explains how to caption the figures and tables in your Appendices, assuming that Heading 7 is numbered “Appendix A” and that the Figures and Tables are going to be labelled ‘Figure A-1’, ‘Figure A-2’, ‘Table B-1’ etc.

You will have to create new, separate labels that look like the ‘Figure’ and ‘Table’ labels you used in the main body of your thesis.

1. Select the **References** tab on the Ribbon then click on **Insert Caption**
2. Click **New Label**. Type **Figure\_Apx** then click **OK**
3. You now have two labels for figures, called **Figure** and **Figure\_Apx**  
   Repeat for table captions.
4. In the **Caption** box, type your caption text
5. Click **Numbering**. Tick **Include chapter numbering** and choose **Heading 7** from the drop-down list of styles and click **OK** twice
6. Your caption should look something like this:

**Figure\_Apx A‑1 This is the caption text for a Figure in the Appendix**

1. Delete the extraneous ‘\_Apx’ from the caption label so it reads:  
   **Figure A‑1 This is the caption text for a figure in the appendix**.  
   **TIP:** Instead of deleting each ‘\_Apx’ individually use **Find & Replace** to modify all the labels at once.

Creating Lists of Figures and Tables for Appendices

This template already includes a List of Figures and a List of Tables, however you will have to create two new lists for the ‘Figure\_Apx’ and the ‘Table\_Apx’ labels.

1. Place the insertion point on a blank row after the existing List of Figures
2. Select the **Insert Table of Figures** command on the **References** tab of the Ribbon
3. Set the **Caption Label** box to ‘**Figure\_Apx**’ and click **OK**  
   **Note:** Word will put a single blank line between the original list and the new list preventing it from appearing as one seamless list. However, if you select the blank paragraph between the tables you can hide it by opening the Font dialog box from the Home tab and selecting **Hidden**.
4. Click after the List of Tables and repeat for the Caption Label ‘Table\_Apx’.

Appendix title (use Heading 7)

Appendix section (use Heading 8)

Appendix subsection (use Heading 9)