Imperial College London

MSc AI Individual Project

Imperial College London

Department of Computing

Automation and Intelligent Control in High Performance Sailing Boats

Final Report

Submitted in partial fulfilment of the requirements for the MSc Degree in Artificial Intelligence of Imperial College London

Author:
Charles Metz (CID: 01825073)

Supervisor:
Dr Pedro Baiz
Second Supervisor:
Dr Eric Topham
Second Marker:
Prof Julie McCann

Date: September 4, 2020

Abstract

This study is a continuation of the work of Birk Ulstad, Roman Kastusik and Stanislas Hannebelle on the application of machine learning methods to the intelligent steering of sailing boats. The purpose of the study is to investigate models that reliably reproduce the behaviour of a sailing boat in its sea environment. These digital twins of the sailboat consist of timeseries forecasting models that predict the values of various variables that define the state of the boat for the following second. This allows a virtual simulation of how the angle of the boat's rudder affects the boat's course and its state, which is the basis for Reinforcement Learning algorithms to learn intelligent control of the rudder. Detailed background research provides an overview of relevant developments in the field of timeseries forecasting. The models investigated here are LSTM-based deep neural networks as well as models derived from first principles. The improvement of the architecture and hyperparameters of the models using Bayesian optimisation is discussed. A significant improvement of the models compared to a previous approach is achieved. While adequate model hyperparameters can be found for a given dataset of a given boat, it is found that they are not easily generalisable across different data collecting protocols. Finally, a framework with which to obtain and assess accurate forecasting models is proposed.

Keywords: Sailing, Autopilot, Digital Twin, Deep Learning, Recurrent Neural Networks, LSTM, Bayesian Optimisation, Timeseries Forecasting

Acknowledgements

I would like to thank Roman Kastusik and Dr Eric Topham, who during almost five months took time for daily online meetings. Despite otherwise very full agendas, they were always available for enquiries and made me benefit very much from their experience, ideas and constructive criticism. I am particularly grateful that they did so during an era marked by a crown-shaped predator, on top of which they allowed me to gain insight into what was happening at their company T-DAB. I would also like to thank Dr Pedro Baiz, who contributed his input and knowledge to the project on a weekly basis; his feedback proved valuable in many ways. I would also like to thank Stanislas Hannebelle: he took the time to explain various aspects of the project to me. Finally, I would like to thank Prof Julie McCann, who took the time to review this and another report and to answer several questions. All in all, I am extremely grateful that this project could take place despite the circumstances.

Contents

Chapter 1

Introduction

1.1 Motivation

Modern racing sailboats are masterpieces of engineering: from materials science to communication technology and aerodynamics, they combine state-of-the-art technology and science. One area seems to be somewhat excluded from this rapid development, namely that of sailing autopilots. During races they are estimated to take over 95% of the steering, but do so with about 80% of the performance of a human skipper. Hence, there is a large potential to reduce this discrepancy between man and machine by using novel machine learning (ML) methods. Reinforcement Learning (RL) is particularly interesting for this purpose, as it can theoretically result in algorithms that outperform human behaviour. However, it can only deliver satisfactory results if a satisfactory RL simulation environment is available to train it. Such a training environment corresponds to a time series forecasting model that is able to predict the next values of the measures that define the state of the sailboat and of its environment. Indeed, the accurate prediction of the latter variables allows to provide feedback to the RL algorithm about the consequences of its actions.
The present project is concerned with the optimisation of this RL simulation environment, i.e. with the elaboration of accurate time series forecasting models. This study is the fourth in a series of works dealing with ML solutions for sailing autopilots. A number of entities are involved in this ongoing project.

1.2 Entities involved in the project

In the following, the entities involved in the present work are presented.

- **T-DAB** is a London-based company specialising in data science and data engineering, offering solutions in a wide range of areas. It was established in 2017 and co-founded by Dr Eric Topham. Since 2019, the company has regularly offered students the opportunity to complete their master's theses within the company. Roman Kastusik, Birk Ulstad and Stanislas Hannebelle, who are regularly mentioned in the following, took advantage of this opportunity from January to June and April to September 2019, respectively. They were students at Imperial College London at the time and have contributed much of the work on this project to date.

- **WisConT** is a UK-Chinese joint venture specializing in data science consulting that works in collaboration with universities like Imperial College London. It was co-founded by Dr Pedro Baiz, the firm's CTO, who enables final-year students to conduct their master thesis as part of the JTR AI project.

- **Jack Trigger Racing (JTR)** is the company of professional skipper Jack Trigger. He regularly takes part in sailing races, such as the Route du Rhum in 2018. A long-term goal is to participate in the Vendée Globe, a single-handed race around the world, which is considered one of the most prestigious sailing races in the world. He regularly supplies T-DAB with new data, hence many of the datasets used in the following are based on him.

- **nke Marine Electronics** develops high-end instruments and technologies for sailing navigation, including autopilots. The French company is a provider of navigation equipment to many professional skippers, including Jack Trigger. Amongst other things, the cooperation with nke allows to address the technicalities of the on-board data collection and processing.

From here on, the collaboration between the four entities will be referred to as the ”JTR AI” project.

1.3 Outline

The remainder of this thesis is structured into the following parts:
Chapter 2: Background provides an overview of previous works on this project, as well as of the datasets available for those projects and the current one.
Chapter 3: Literature Review presents a systematic literature review of past and current developments in the fields which this thesis relates to, i.e. most notably that of the development of algorithms for autonomous sailing and of timeseries forecasting models. It ends with a presentation of the hypotheses that the present study aims to verify.
Chapter 4: Data presents the preprocessing steps applied to the available datasets and describes the characteristics of the resulting data, i.e. their statistical distribution and its implication for the present study.
Chapter 5: Models introduces the architectures of the deep learning models as well as the first-principle models that are investigated during the experimentations.
Chapter 6: Experiments presents the experimental approach pursued in this study. This includes namely the models trained, the datasets used for training, as well as the strategy and the metrics employed for assessing the performance of the models.
Chapter 7: Results and Evaluation discusses the performance of the models, both in absolute terms with respect to specific evaluation metrics, as well as in comparison to the performance of other models.
Chapter 8: Conclusion and Future Work presents the conclusions derived from the

presented findings. Furthermore, precise directions of further work are presented, in particular with respect to the integration of the elaborated simulation environment into existing RL algorithms.
Chapter 9: Ethical considerations discusses the ethical aspects of the present study and any work building upon it. In particular, issues relating to data privacy are considered. Furthermore, safety concerns in relation to the live deployment of ML algorithms on sailboats are presented.

Chapter 2

Background

The present study forms part of a series of works on JTR AI. Hence, in the following, an overview of the key points of previous works is provided in section 2.1. Furthermore, a first overview of the datasets available for JTR AI is presented in section 2.2. Finally, in sections 2.3 and 2.4, selected parts of previous works are presented in deeper detail where they are relevant for the present study.

2.1 Overview of previous work

As mentioned previously, Roman Kastusik, Birk Ulstad and Stanislas Hannebelle have previously worked on JTR AI. Their master theses lay the foundations of the project to date. The work of Roman Kastusik and Birk Ulstad provided in particular the basis for the parsing and exploitation of data logged during sailing races. This especially concerns the processing of different navigation logs from a format that is specific to the nke autopilots into a format that is compatible with the application of data analysis and ML methods. Furthermore, both students had the goal to train an ML-based sailboat autopilot using this data. This ML-based autopilot would steer the boat in a more intelligent manner than conventional autopilots that rely on classical control schemes. Training ML-based algorithms to ”steer the boat in a more intelligent manner” effectively means training these algorithms to compute a position of the rudder that would be similar to or better than the rudder angle a human sailor would set. Indeed, such a steering behaviour is preferable over the more ”rough” behaviour of traditional closed loop PID controllers that do not directly take into account information about wind, waves and other factors influencing the boat’s course and behaviour, which leads to their reduced performance. Two fundamentally different approaches can be employed to achieve that goal. They have been explored by Birk Ulstad, Stanislas Hannebelle and Roman Kastusik and are presented in the following.

2.1.1 Supervised learning approach

Birk Ulstad investigated a supervised learning approach: a model based on recurrent neural networks (RNNs) and long short-term memory (LSTM) networks was to learn the behaviour of the skipper (i.e. that of Jack Trigger) using data that had been logged during a number of boat races. The model was trained to receive inputs from environmental sensors indicating the physical state of the boat and the environment around it, based on which it had to return the same rudder angle as a human skipper would set in a given situation. Stanislas Hannebelle refined this approach in his master thesis, which started a little later, by refining the pre-processing of the data, by training models on different sub-samples of the available data, and by optimizing the hyperparameters of the models. Compared to the results of Birk Ulstad, this resulted in an improvement of the results. Various aspects of this work are relevant to the present project. They are described in more detail in section 2.3.

2.1.2 RL approach

Also described in detail below in section 2.4 is the approach of Roman Kastusik, which was fundamentally different from that of Birk Ulstad and Stanislas Hannebelle. Instead of a supervised learning approach, where the skipper's behaviour is only mimicked and can thus at best reach the performance of the data-generating skipper, Roman Kastusik used deep RL methods. In this approach, the model learns how to set the rudder to achieve the best possible performance. As explained in more detail in the appendix C.1, it does so by obtaining rewards or penalties for its behaviour. In theory, the model can learn behaviour superior to that of the human skipper.

Challenges in training a reliable RL simulation environment

However, a deep RL algorithm requires a reliable and as accurate as possible simulation environment of the boat at sea to learn the consequences of its steering behaviour and hence be able to learn optimal behaviour. This was a major challenge in Roman Kastusik's master thesis; while the RL agent converged in standard RL testing environments (further details in appendix C.1), it did not converge in the environment developed to simulate the sailboat at sea. This can be attributed to the fact that the amount of data available at that point in time of the JTR AI project was not sufficient to train the state estimator (cf. section 2.2). Furthermore, the datasets available at that point in time present a high degree of bias, as all of them originate from races whose routes go from the north-east to the south-west of the Atlantic Ocean. In other words, the state estimator is trained on rather specific sailing conditions. Moreover, the relatively small training time of the estimator might have been insufficient. Finally and most importantly, the unsatisfactory behaviour can be attributed to a more sophisticated model architecture and hyperparameters being required to capture the dynamics of the boat and its surroundings. Indeed, Roman Kastusik's study does not include an optimisation of the latter aspect but merely relied on a single, un-optimised model to provide the simulation environment, which was not fulfilled to a satisfactory degree. Hence, as described in more

detail in section 2.4, improvements to the simulation environment developed by Roman Kastusik are possible. Indeed, they constitute the major direction of work of the present project, as presented in the sections about the models investigated (5) and about the experimental strategy pursued in this study (6).
Hence, the present study consists in a refinement of the simulation environment developed by Roman Kastusik. In that light, previous approaches to refining the methods employed by Birk Ulstad and Roman Kastusik are worth considering. More precisely, it is worth taking the optimisations applied to the LSTMs of Birk Ulstad's supervised learning approach, and applying those optimisations to the present supervised problem for forecasting elements of the boat state more accurately.

2.1.3 Refinement of the existing approaches

It is worth noting that Stanislas Hannebelle's work consisted largely of a refinement of Birk Ulstad's works. This is especially true for the data pre-processing and selection of training data, which has been significantly improved by Stanislas Hannebelle and thus contributes to the increase of training data quality, whatever they are used for after this step. It also applies to the training of LSTMs in general, especially with respect to hyperparameter optimization. Stanislas Hannebelle worked this out in the context of a supervised learning problem for the prediction of an optimal rudder angle. This is different from the present study's main object of investigation, i.e. the improvement of boat state estimator. However, the pre-processing step to improve the data quality and the approach to optimize LSTMs lend themselves for an integration to this study, which is why the relevant parts of Stanislas Hannebelle's thesis are presented in more detail below in section 2.3.

In this light, the following sections first presents an overview of the available datasets, so that the reader is aware of the data to be used by any models. Subsequently, the parts of Stanislas Hannebelle's work that are relevant for the present study (data pre-processing, selection of data for model training and hyperparameter optimization) and the parts of Roman Kastusik's work important for this study (RL simulation environment) are presented. While the understanding of the RL-based approach is not indispensable for the present study, the interested reader can find an overview of it in appendix C.1.

2.2 Available datasets

In the following, the different boats for which datasets are available are presented. Subsequently, the datasets used by Roman Kastusik, Stanislas Hannebelle and Birk Ulstad are presented. Following this, datasets recently received by T-DAB - and that have not been used previously in JTR AI - are presented. Finally, tables 2.2 and 2.3 provide an overview of the provenance and the format of the datasets. In summary, these datasets essentially consist of key sailing measures (wind speed, boat speed etc.) recorded by different boat sensors during sailing races. A list of the measures available for this thesis is presented in table 2.4 and covered in more detail below.

For the sake of consistency, the denominations of the datasets and of features are kept identical as in previous works on JTR AI.

Single-handed and double-handed races It should be stressed that the datasets differ in that they were each generated while either a human skipper or an autopilot was in control of the boat. As will be seen in the following, this distinction is particularly relevant if a model is to be trained to imitate a human skipper using supervised learning: in this case, the training requires data generated by a human skipper. However, as will become apparent subsequently, this distinction is of little relevance if a state estimator is to be trained about the physics of the boat, i.e. to learn its real behaviour on sea independently of a human or an autopilot steering the boat. Indeed, in this case it is valuable to benefit from the ensuing increase in diversity of the data, but the data's being specifically generated by a human skipper or an autopilot is not of relevance.

2.2.1 Types of boats

The different datasets were recorded for different boats - an example of which can be found in fig. 2.1) - i.e. for

- Jack Trigger's **"Concise 8"**, belonging to the "Class 40" category of sailing yachts (cf. [1] for further information about this category of sailing yachts).
- Jack Trigger's **"Virgin Business Media" (VMB)**, also belonging to the "Class 40" category of sailing yachts.
- two unknown boats that are different from each other but belong to the same "IMOCA 60" category of boats (cf. [2] for the IMOCA 60 rules). In the following, both of these boats are denominated **"Unknown 1"** and **"Unknown 2"**.

Table 2.1 presents the technical aspects that define Concise 8, VMB and Unknown 1. For Unknown 2, no precise information is available at all and as will be seen in the following sections, it is not necessary for the present study. For VMB, only information about the sail area is available; for the other technical aspects, the upper limit is known from the Class 40 rules (cf. [1]). It should be emphasised that in spite of the presented numbers' corresponding to key characteristics of the boats, these mere numbers do not reflect that e.g. sails of the same sail area might differ substantially in their cut and hence their behaviour in wind. Another example is weight: it might be exactly identical by the absolute number for two boats, but be differently distributed in those two boats. Furthermore, it should be noted that the IMOCA 60-category Unknown 1 differs strongly from the Class 40-category Concise 8 and VMB concerning the size, weight and the sail area. This is in line with its belonging to the IMOCA 60 class, which essentially is a class of larger and bigger boats than the Class 40. However, while there are differences between the design of Concise 8, VMB and Unknown 1 and therefore their physical behaviour when sailed, it should be emphasized that all of them have been designed for the same type of offshore sailing along the same offshore and ocean routes, i.e. for the same conditions. Thus, notwithstanding the fundamental differences between the boats' dimensions and their behaviour when sailed, they present general similarities in their design; one could compare them to distant cousins from the same family. In this light, it is worth considering the different datasets available for the boats studied in the present work and the sailing conditions that they contain.

Technical aspect	**Measure**	**Concise 8**	**Virgin Media Business**	**Unknown 1**
Boat class		Class 40	Class 40	IMOCA 60
Weight [kg]		4500	≤ 4500	8200
Dimensions [m]	Length	12.19	≤ 12.19	18.28
	Width	4.5	≤ 4.5	5.94
	Draft	3.0	≤ 3.0	4.50
	Height	19	≤ 19	27
Sail Area [m^2]	Upwind	115	115	300
	Downwind	250	250	560

Table 2.1: Technical description of the three boats Concise 8, Virgin Media Business and Unknown 1. Where no precise information is available, upper limits are listed as found in the Class 40 design rules [1].

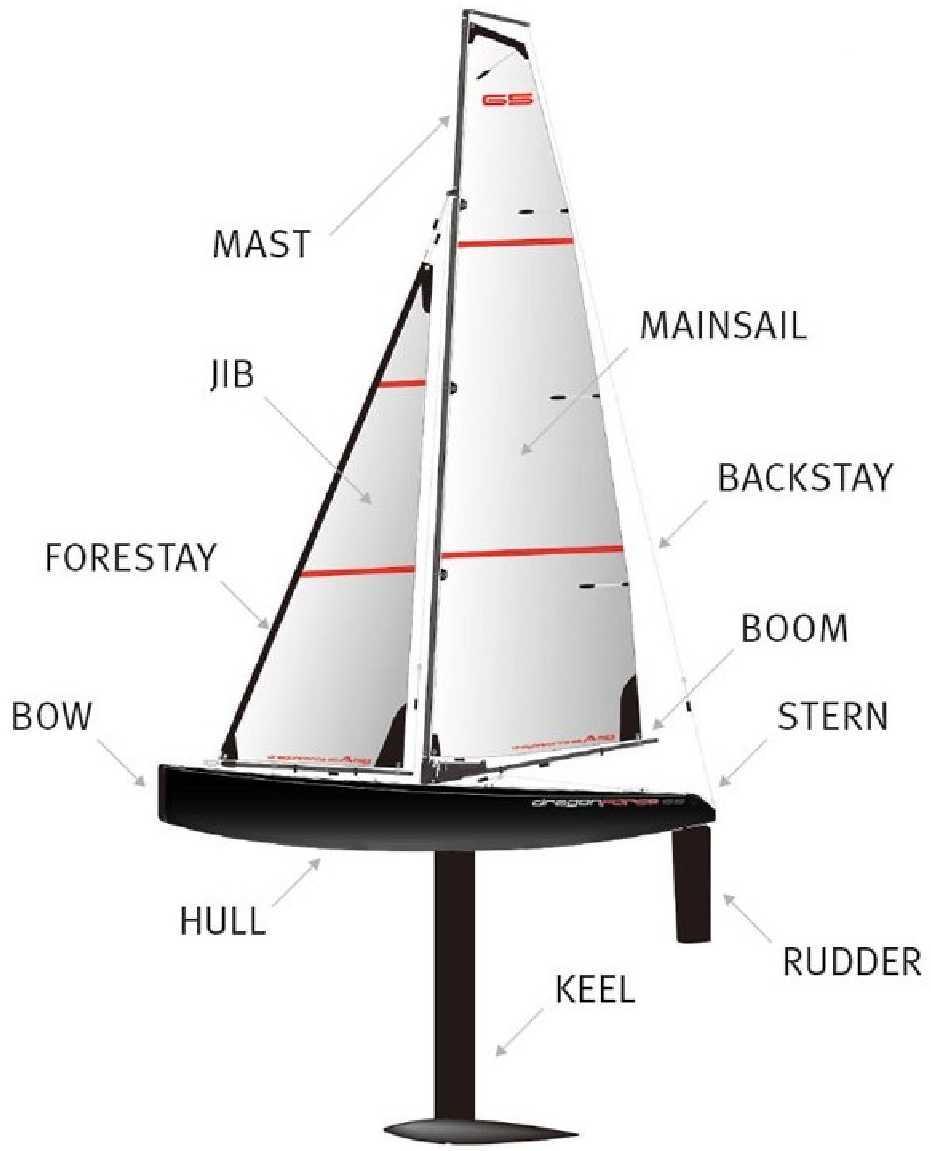


Figure 2.1: Example of a sailing yacht, from [3]

2.2.2 Concise 8

Four datasets are available for Concise 8. They differ not only in the sailing conditions in which they were recorded, but also in the format that was used for their logging.

Route du Rhum (nkz)

The Route du Rhum dataset (previously referred to as ”nkz” dataset) was recorded by JTR during parts of the Route du Rhum (RDR) race 2018 and sailed with the Concise 8. The dataset was recorded at 25 Hz sampling frequency using nke instruments and software. This was done in the proprietary format of nke, i.e. in the *”.nkz”* format. Roman Kastusik and Birk Ulstad spent a considerable time of their master theses on the conversion of these data into the .csv format for which multiple data processing tools and libraries exist. This was done using the nke software *LogAnalyser*. Due to the use of a flash drive not supported by nke's products during the time of the recording, this dataset is corrupted and data of the Route du Rhum race is only partly available. Further information on the used software, the procedure to convert the data from the .nkz to the .csv format as well as only parts of the data of this race being available can be found in Roman Kastusik's final report [4]. Finally, as the RDR race is single-handed, the autopilot was active during large parts of the race.

Route du Rhum (adrena)

The Route du Rhum (adrena) dataset (previously referred to as "adrena" dataset) was recorded by JTR during parts of the RDR race 2018 and sailed with the Concise 8, much like the nkz dataset. This dataset was generated by passing data of selected features at a frequency of 1 Hz to the navigation software *Adrena* running on the sailor's laptop. It was directly recorded in the .csv format and only when the autopilot was active, i.e. it does not contain any data of segments sailed by a human skipper. Details on the dataset can be found in [4].

DRHEAM (18 log)

The DRHEAM 18 dataset (referred to as "log" dataset in previous works) was recorded by JTR during the DRHEAM cup 2018, sailed with the Concise 8. It was recorded in a specific format that was used before nke introduced the .nkz format. The dataset in this .log format was transformedd into the .csv format by Roman Kastusik by using a specifically adapted parser. Crucially, it should be noted that records in this format are made at an inconsistent frequency. The consequences of this will be further discussed in section 2.3; further information on this format can be found in [4]. DRHEAM cup is a double-handed race, thus this dataset corresponds to a route mainly sailed by two human skippers, as opposed to the nkz and adrena datasets. Hence, this dataset is of relevance if a model is to be trained as a digital twin of a human skipper using supervised learning. As will be seen in the following, this is the reason why Stanislas Hannebelle made use of this dataset.

Atlantic

Also newly available is the Atlantic dataset, corresponding to the navigation log recorded during a delivery made by Jack Trigger in 2019. The delivery was from the port of Grenade to the port of Horta on the Azores Island. The recording was performed in the .nkz format at 25 Hz and has been converted entirely to the .csv format. Since this delivery was sailed solo by Jack Trigger, the autopilot was active during large parts of the trajectory.

2.2.3 Virgin Media Business (VMB)

One dataset from the VMB, a Class 40 boat different from, the Concise 8 is newly available.

DRHEAM 20

The DRHEAM 20 dataset was recorded by JTR during the DRHEAM cup 2020 and sailed with the VMB. The recording was performed in the .nkz format at 25 Hz. Due to an issue with the autopilot's recording similar to that encountered for the Route du Rhum (nkz) dataset (cf. 2.2.2), only 7 hours of data were saved during this multi-day race, all of which have been converted to the .csv format. As for DRHEAM 18, this dataset corresponds to a route mainly sailed by two human skippers, hence the data was largely generated by human skippers.

2.2.4 Unknown 1 and Unknown 2

Two datasets from two different IMOCA 60 boats are newly available.

transat_1 and transat_2

In early 2020, T-DAB received two datasets from nke that had been recorded during the Transat Jacques Vabre race in 2019. This data was not available for previous work on JTR AI. The data was recorded in the nke-proprietary format .nkz at 25 Hz and has been partly converted to the .csv format (cf. section 4.5 for further information on the conversion of this file to .csv). As opposed to the nkz, adrena and log datasets, these two datasets originate from two different sailing teams whose identity is unknown. Furthermore, whereas the previously mentioned datasets correspond to routes sailed by Class 40 boats, the two datasets dubbed transat_1 and transat_2 correspond to routes sailed by different boats both belonging the class IMOCA 60, i.e. boats with different characteristics. Finally, Transat Jacques Vabre is a double-handed race, hence the recorded data was largely generated by human skippers.

Boat (class)	Name	Race	Sailor	Year
Concise 8 (Class 40)	RDR (nkz)	RDR	Jack Trigger	2018
	RDR (adrena)	RDR	Jack Trigger	2018
	DRHEAM 18 (log)	DRHEAM cup	Jack Trigger	2018
	Atlantic	Delivery	Jack Trigger	2019
VMB (Class 40)	*DRHEAM 20*	DRHEAM cup	Jack Trigger	2020
Unknown 1 (IMOCA 60)	*transat_1*	Transat Jacques V.	Unknown	2019
Unknown 2 (IMOCA 60)	*transat_2*	Transat Jacques V.	Unknown	2019

Table 2.2: Overview of datasets available per boat. Datasets that are newly available for the present study are in italics.

Name	Original	Converted	Sampling frequency [Hz]	Length [h]
RDR (nkz)	.nkz	.csv	25	16
RDR (adrena)	.csv	.csv	1	306
DRHEAM 18 (log)	.log	.csv	Variable	64.5
transat_1	.nkz	.csv	25	383.5
transat_2	.nkz	.csv	25	387.5
Atlantic	.nkz	.csv	25	290.9
DRHEAM 20	.nkz	.csv	25	7

Table 2.3: Overview of the available datasets' formats and sampling frequencies. Datasets that are newly available for the present study are in italics.

2.3 Previous work by S. Hannebelle

While the present work focuses on improving the simulation environment of the deep RL algorithm developed by Roman Kastusik, Stanislas Hannebelle's work [5] (building on that of Birk Ulstad [6]) contains many aspects that are useful to the present work. The relevant parts of this previous work are presented in more detail in the following sections.

2.3.1 Data Pre-Processing

Re-Sampling Data to 25 Hz Stanislas Hannebelle used the DRHEAM 18 (log) dataset, as presented in section 2.2. The measures recorded in this dataset - wind speed, wind angle, position etc. - are presented in detail in table 2.4. In the log dataset, the time differences between the recordings of new states are not consistent; they range anywhere between 0.004 and 0.067 seconds. Moreover, many of these features come in the form of angles, which entails jumps from -180° to 180° respectively from 180° to -180°, as can be seen in fig. 2.2 (taken from Stanislas Hannebelle's master's thesis [5]). However, for the proper training of the ML models, timeseries data is needed which update at a consistent - and not at a constantly changing - frequency. For this reason Stanislas Hannebelle further developed a re-sampling algorithm used by Birk Ulstad, which re-samples the data to a constant frequency of 25 Hz. First, in order to account for the abrupt changes in the angular values (e.g. from -180° to 180°), angle values are replaced by their cosine and sine values. Subsequently, linear interpolation is used to resample the timeseries intervals to a constant 25 Hz. The pseudo code of this approach is shown in algorithm 1. Figures 2.3 and 2.4 serve as an illustration of this first pre-processing step.

Algorithm 1 Re-sampling log data to 25Hz Algorithm

```
1: procedure TO25HZ(log_csv_path,log_25Hz_csv_path)
2:     log ← read_csv(log_csv_path)
3:     for column ∈ set of angles in range [-180,180] or [0,360] do
4:         $log[column\_cos] \leftarrow cos(log[column])$
5:         $log[column\_sin] \leftarrow sin(log[column])$
6:         $log.drop(column)$
7:     end for
8:     log ← log.resample('00.04S').asfreq().interpolate('linear')
9:     for column ∈ set of angles in range [-180,180] or [0,360] do
10:        $log[column] \leftarrow sign(log[column\_sin])\dot{a}rccos(log[column\_cos])$
11:        $log.drop(column\_cos)$
12:        $log.drop(column\_sin)$
13:    end for
14:    log.save_csv(log_25Hz_csv_path)
15: end procedure
```

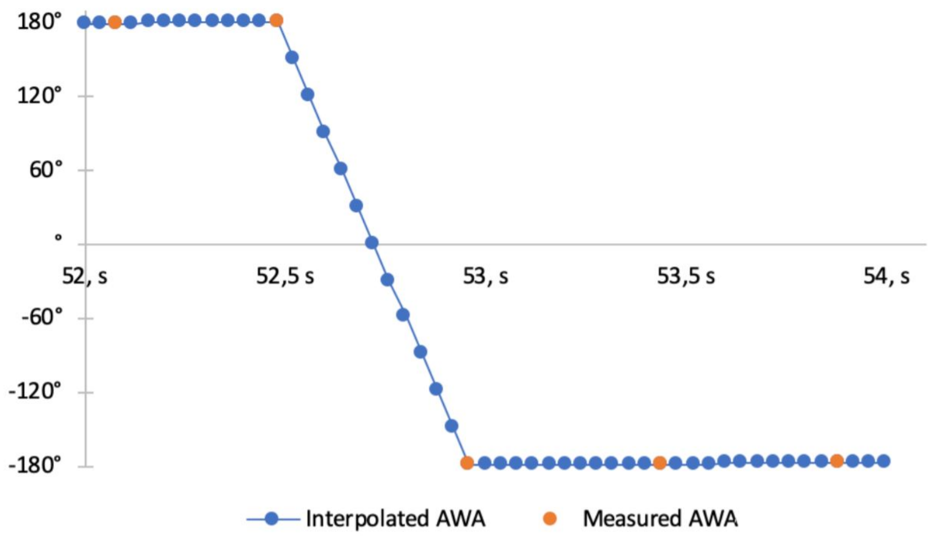


Figure 2.2: Measured and interpolated values of the Apparent Wind Angle, from [5]

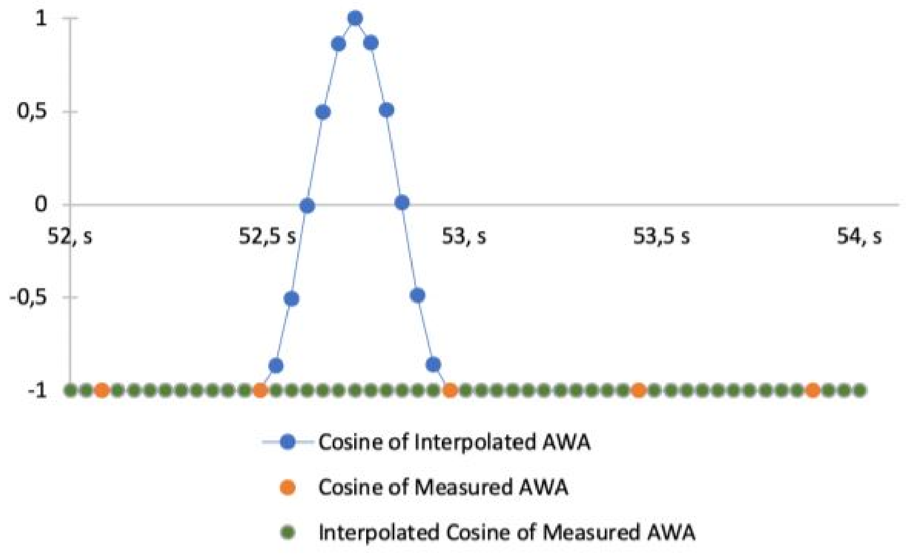


Figure 2.3: Cosines of the measured and interpolated values of the Apparent Wind Angle vs. interpolated cosines of the measured Apparent Wind Angle, from [5]

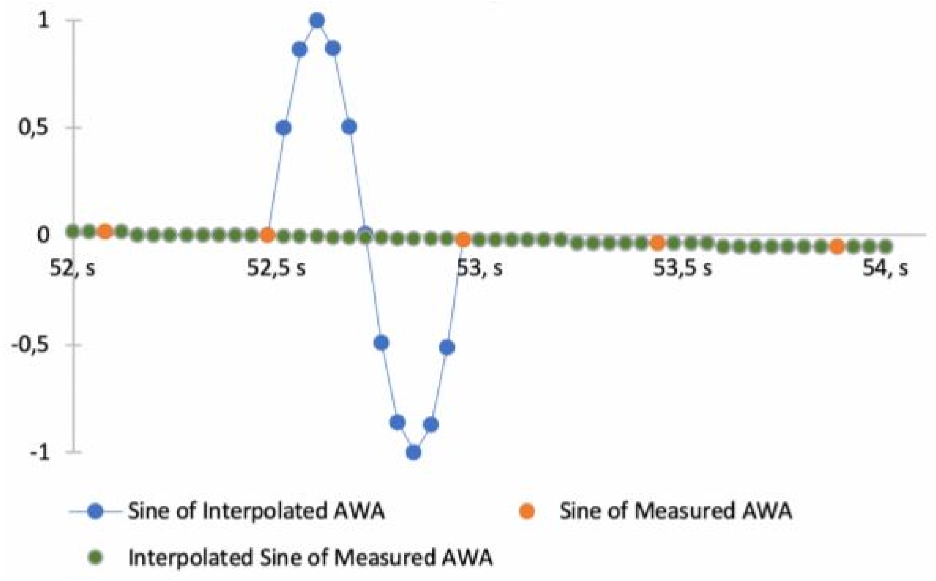


Figure 2.4: Sines of the measured and interpolated values of the Apparent Wind Angle vs. interpolated sines of the measured Apparent Wind Angle, from [5]

Feature Name	Description	Units	Source	Data Type	Range
Latitude	Global coordinate	[°]	GPS	float	[-90, 90]
Longitude	Global coordinate	[°]	GPS	float	[-180, 180]
TWS	True wind speed	[kts]	Derived	float	[0, 40.0]
TWD	True wind direction (global)	[°]	Derived (GPS)	float	[0, 360]
Current_speed	Speed of water current	[kts]	Derived (GPS)	float	[0,15.0]
Current_direction	Direction of water current (global)	[°]	Derived (GPS)	float	[0, 360]
Air_temp	Temperature of the air	[°C]	Measured	float	[0, 30.0]
Speed_ov_surface	Speed of the boat over the water	[kts]	Measured	float	[0, 25.0]
Speed_ov_ground	Speed of the boat over the ground	[kts]	Derived (GPS)	float	[0, 25.0]
VMG	'Velocity made good', speed towards wind direction	[kts]	Derived	float	[0, 25.0]
Heading_True	True heading relative to North	[°]	Derived (mag)	float	[0, 360]
Heading_ov_ground	True heading relarive to North accounting for Yaw	[°]	Derived (GPS)	float	[0, 360]
Pitch	Rotation around lateral axis of the boat	[°]	Measured	float	[-20, 20]
Roll	Rotation around longitudinal axis of the boat	[°]	Measured	float	[-60, 60]
Yaw	Rotation around vertical axis of the boat	[°]	Derived (GPS)	float	[-180, 180]
AWS	Apparent wind speed	[kts]	Measured	float	[0, 50.0]
AWA	Apparent wind angle (local)	[°]	Measured	float	[-180, 180]
TWA	True wind angle (local, awa accounting for boat motion)	[°]	Derived	float	[-180, 180]
Rudder (*nkz* and *log* data-sets only)	Angle of the rudder relative to neutral position	[°]	Measured	float	[-30, 30]

Table 2.4: Overview of data available for project, from [4]

Removal of Tacks

Motivation After their upsampling to a consistent frequency, a second preprocessing step is applied to the time series data. Indeed, a sailboat can move in different directions relative to the wind. This True Wind Angle between boat and wind determines the performance of the boat to a large extent, especially regarding the boat's attainable speed for a given True Wind Angle. This is illustrated by fig. 2.5 taken from [5], which represents the corresponding polar plot for the Concise 8, the boat sailed by Jack Trigger. In the figure it can be recognized that strong changes are taking place during the process of a tack, i.e. when the True Wind Angle passes 0°, as well as during the process of a gybe, i.e. when the True Wind Angle passes 180°. In both cases the main sail changes the side of the boat. Since the sailed course is significantly influenced by these tacks and gybes, in the following referred to as ”tacks” for easier reading, the performance of a skipper during a race depends crucially on them. Moreover, these are quite dangerous maneuvers, as the sails and the boom sweep over the boat and therefore change the balance on board considerably. This entails a significant potential for physical damage as well as representing a strong risk to the skipper. Furthermore, autopilot functions exist for this and are adequate. Moreover, these maneuvers are not critical to performance in offshore sailing, other than doing them safely so that the boat remains intact (which if it does not puts an end to the race or even threatens the skipper's life). Finally, all sorts of reasons can lead to the decision to tack, some relating for instance to strategy or safety, both of which are not available at this time to the autopilot. For these reasons, the model that imitates the skipper should not be trained using data containing tacks, but be optimised with regards to its performance under ”normal” sailing conditions.

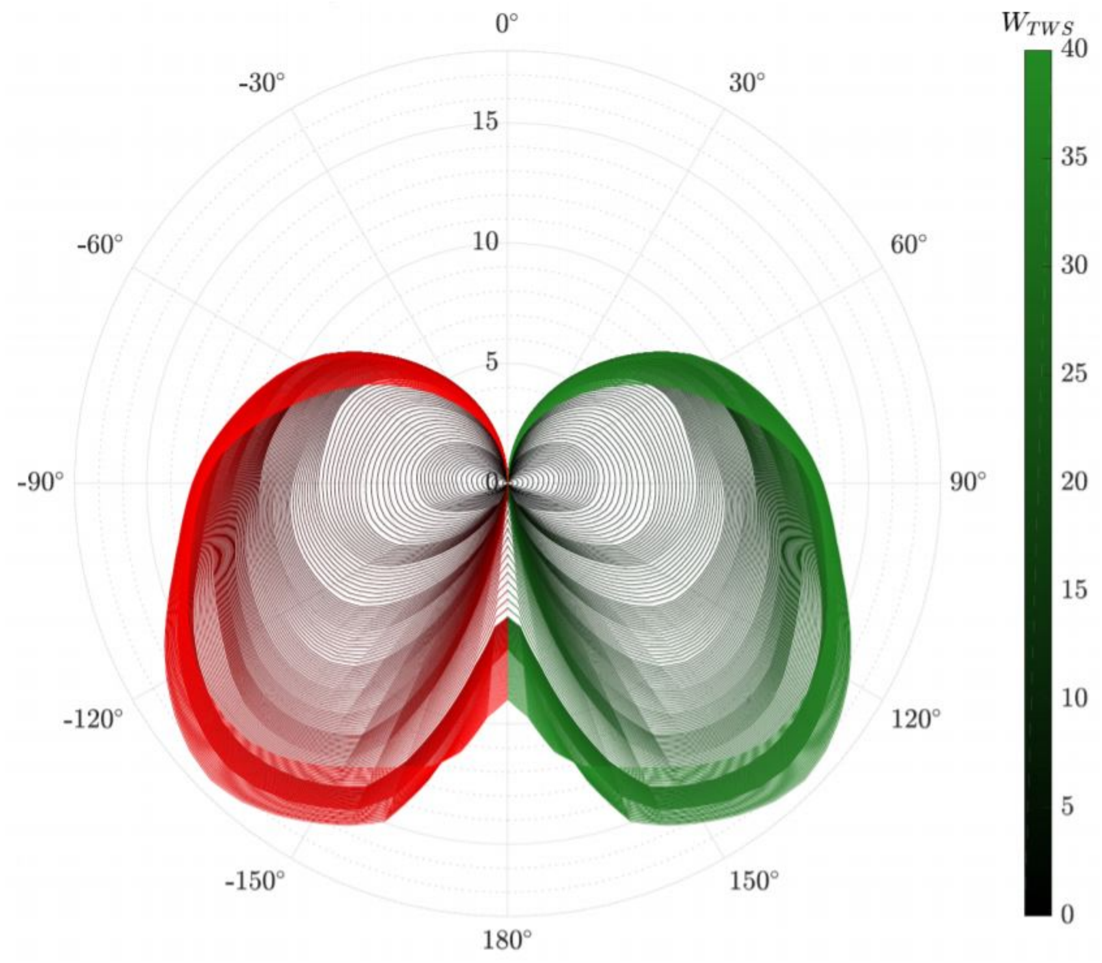


Figure 2.5: High resolution polar plot of the Concise 8, from [5]

Subdivision of data into segments Thus, Stanislas Hannebelle investigated models that recognize tacks, such that segments of the sailed course that contain them can be removed from the training data. To that end, the time series data was subdivided into successive sequences of 60 seconds. For every 60 s-long segment, the tack-identifying model should return the binary classification ”tack segment”/”no tack segment”. The selection of 60 s as segment duration stems from the duration of one tack maneuver, which typically takes a maximum of ca. 30 s.

Tack detection model The most reliable method was identified to be a decision tree, which receives as input the values of a selection of features and generates as output a prediction as to whether the considered segment contains a tack or not. More details about the inner workings of this tack detection model and the corresponding confusion matrix can be found in appendix B. For the present study, it is only of interest to be aware that a reliable tack-identifying model could be developed and can be used for tack detections for the newly available data.

2.3.2 Data Cleaning and Splitting

Two steps of cleaning In a first phase of data cleaning, not only the segments containing a tack were removed, but also the 60-second segment following each segment containing a tack. In fact, Jack Trigger had pointed out that it can take up to 60 seconds for the sailboat to return to its speed and optimal sailing conditions after a tack, i.e. to return to the conditions that the model should be trained on. In a second phase of data cleaning, the course of the boat was analysed in detail and segments with anomalies were identified and removed. Indeed, the dataset contained segments in which abnormal sailing conditions in the form of extremely low wind speeds and low boat speeds appeared. However, since Jack Trigger also stated that the model should imitate the boat at full speed and that low wind conditions were of little relevance. Indeed, while not necessarily appearing as ”outliers” statistically, data of these sections with low wind and/or boat speed does not capture the fact that they entail a very different physical behaviour of the boat. For instance, rocking of the boat can generate apparent wind speed even though there is no wind, simply because the mast moves the anemometer through the air as the boat rolls in the water. With these elements of the boat's physics being so different, these are not the primary conditions in which one would expect the autopilot to work. Hence, these ”abnormal” segments were removed. This second cleaning step was performed manually, i.e. by systematically examining the data for time windows where low wind and/or boat speed predominated (cf. [5]) and removing these time windows from the data retained for further use. This results in a dataset composed of different time series.

Resulting dataset Precisely, for the DRHEAM 18 (log) dataset, the data cleaning leads to 19 different time series. To illustrate, the first time series obtained is ”23rd July 2018 from 16:00 to 16:23”, the next one is ”23rd July from 16:25 to 16:36”, etc. For each of these time series, the first 60 % of the segments are concatenated and retained as training data. The next 20 % of the segments are concatenated to represent validation data, and the last 20 % are concatenated to represent test data. This allows the model to be trained, validated, and tested on different parts of the data. This approach takes into account that the conditions are not steady throughout the race, nor are they evenly temporally distributed across the race course. By composing the training, validation and testing datasets with data from different parts of the race, models can be trained and tested on sufficiently similar data. This differs from training the model on the first 60 % of the entire time series and validating and testing it on data from later parts of the sailed route, which would not take into account the uneven distribution of sailing conditions over the race.

2.3.3 Supervised Learning Process

The present study is concerned with the supervised learning problem of training reliable models to forecast the multiple variables that describe a boat's state. This problem has similarities with forecasting the rudder angle a human sailor would set. Hence, it is worth considering the approach taken to solving that problem.

Optimal Sampling Frequency and Input Length

As presented in section 2.3.1, the data is available in a resolution of 25 Hz after pre-processing. However, the model that is to predict the optimal rudder angle does not need to be trained at the maximum frequency of 25 Hz; sampling from the 25 Hz dataset by retaining e.g. only every fifth value allows to vary the data granularity of the input provided to the model. A further degree of freedom is the choice of the input length of the data, i.e. of the time window of which data is passed to the rudder-predicting model. Indeed, as will be elucidated in the literature review, the choice of the time window for which data is passed to the model heavily influences the model's performance. For these reasons, Stanislas Hannebelle investigated the effect of different sampling frequencies as well as of different time windows. Indeed, Birk Ulstad in his work used a much more complex model than Stanislas Hannebelle for the same rudder angle prediction task, and obtained the best results for a sampling frequency of 5 Hz with a time window of 25 s. However, it was found that this model led to severe overfitting (cf. Stanislas Hannebelle's thesis [5]) that can only be compounded when using an even more granular sampling frequency and an even longer time window. Stanislas Hannebelle therefore set these two values, 5 Hz and 25 s, as upper limits for the sampling frequency and the length of the time window. Furthermore, it was found that the skipper changes the position of the rudder at least once per second, so 1 Hz was retained as the lower limit for the frequency.
In this light, Stanislas Hannebelle trained and validated Birk Ulstad's model on the pre-processed data as described above for the frequencies {1 Hz, 5 Hz} and for time windows of the lengths {1s, 2s, 3s, 4s, 5s, 10s, 15s, 20s, 25s}. The retained optimal time windows were 5 s for a sampling frequency of 1 Hz and 2 s for a sampling frequency of 5 Hz. These two combinations were retained as sampling frequency and window length for the investigations that followed, and that were namely concerned with improving the architectures of the supervised learning models. To that end, an optimisation of the models' hyperparameters was performed.

Bayesian Optimisation of Hyperparameters

Interest for present study Using the previously mentioned pairs of sampling frequency and length of time window, Stanislas Hannebelle trained LSTMs and GRUs to predict the optimal rudder angle. The detail of these networks is of rather little interest here (and can be found in the original thesis [5]), since the present work focuses on predictions of the all of the boat's features given previous values of the boat's and the sea's features, while Stanislas Hannebelle focused on the prediction of the rudder angle only. It is much more the approach to the optimization of the hyperparameters of the models which is interesting for the present study. It is hence described in the following.

Approach The hyperparameters are divided into two classes: First, those which define the architecture of the model and second, those which determine the optimisation of the model. While Stanislas Hannebelle decided to retain *tanh* for the activation functions of the networks and not to experiment with other activation functions, he decided to optimise the following measures of the network architecture:

- Number of GRU respectively LSTM layers, to control the complexity of the model
- Number of units per GRU resp. LSTM layer, also to control the complexity of the model
- Dropout rate, to influence regularisation

Furthermore, it was decided to use the Adam optimiser because of its proven performance in optimisation tasks, and hence not to experiment with different solvers. However, it was decided to optimise the learning rate as it has a significant impact on the speed of the learning process. If these variables were now optimised using a grid search, numerous iterations would have to be run through to identify optimal values, which is costly in training time, compute resource, and financial resource. Furthermore, the dropout rate and the learning rate correspond to continuous values in an interval between certain lower and upper limits, which grid search does not take advantage of. Bayesian optimisation provides a remedy for both problems, and was therefore retained as an optimisation method. Again, the interested reader is referred to the original thesis for further information on this process [5].

2.4 Previous work by R. Kastusik

Stanislas Hannebelle's work aimed at refining and improving the pre-processing and hyperparameter optimisation employed by Birk Ulstad, which was a success. No such improvement was conducted for the forecasting model developed by Roman Kastusik. However, as outlined in the previous sections, an improvement of this forecasting model and more broadly of the RL simulation environment is vital for the progress of an RL-based autopilot. Hence, the following sections present in more ample detail the state estimator developed by Roman Kastusik. Subsequently, the deep RL model that he developed is presented to the extent relevant for the deeper comprehension of the state estimator's functioning.

2.4.1 State estimator

The overall data flow developed by Roman Kastusik is presented in figure 2.6, borrowed from his final report [4]. A state vector s_t, describing the estimated state of the sea and the boat at instant t, enters a deep deterministic policy gradient (DDPG) RL model. The latter outputs an action a_t that corresponds to the Rudder Angle at instant t that the deep RL model predicts to be best at that specific instant t. In the following, these vectors are presented in detail following the exact same denominations as Roman Kastusik used in his report (cf. [4]); this is done in order to ensure consistency and comparability between the different works on JTR AI.

The state vector s_t is composed of the state of the sea, the boat, and the rudder angle at time t, so

$$s_t = \begin{pmatrix} Sea\,State(t) \\ Boat\,State(t) \\ Rudder\,Angle(t) \end{pmatrix} = \begin{pmatrix} s_t^s \\ s_t^b \\ R_t \end{pmatrix} \tag{2.1}$$

where the sea state s^s is defined by

$$s_t^s = \begin{pmatrix} TWD \\ TWS \\ Current\,Direction \\ Current\,Speed \\ Air\,Temperature \end{pmatrix} \tag{2.2}$$

and the boat state s^b by

$$s_t^b = \begin{pmatrix} Speed\,over\,surface \\ Speed\,over\,ground \\ VMG \\ Heading_{true} \\ Heading\,over\,ground \\ Pitch \\ Roll \\ Yaw \\ AWS \\ AWA \\ TWA \end{pmatrix} \tag{2.3}$$

Finally, it is worth mentioning that as opposed to the supervised rudder angle prediction problem, the data used by Roman Kastusik were down-sampled to 1 Hz, with no other sampling frequencies being investigated. This choice was motivated by the fact that Roman Kastusik used the log, nkz and adrena datasets, the latter of which was available in a resolution of 1 Hz (cf. section 2.2).

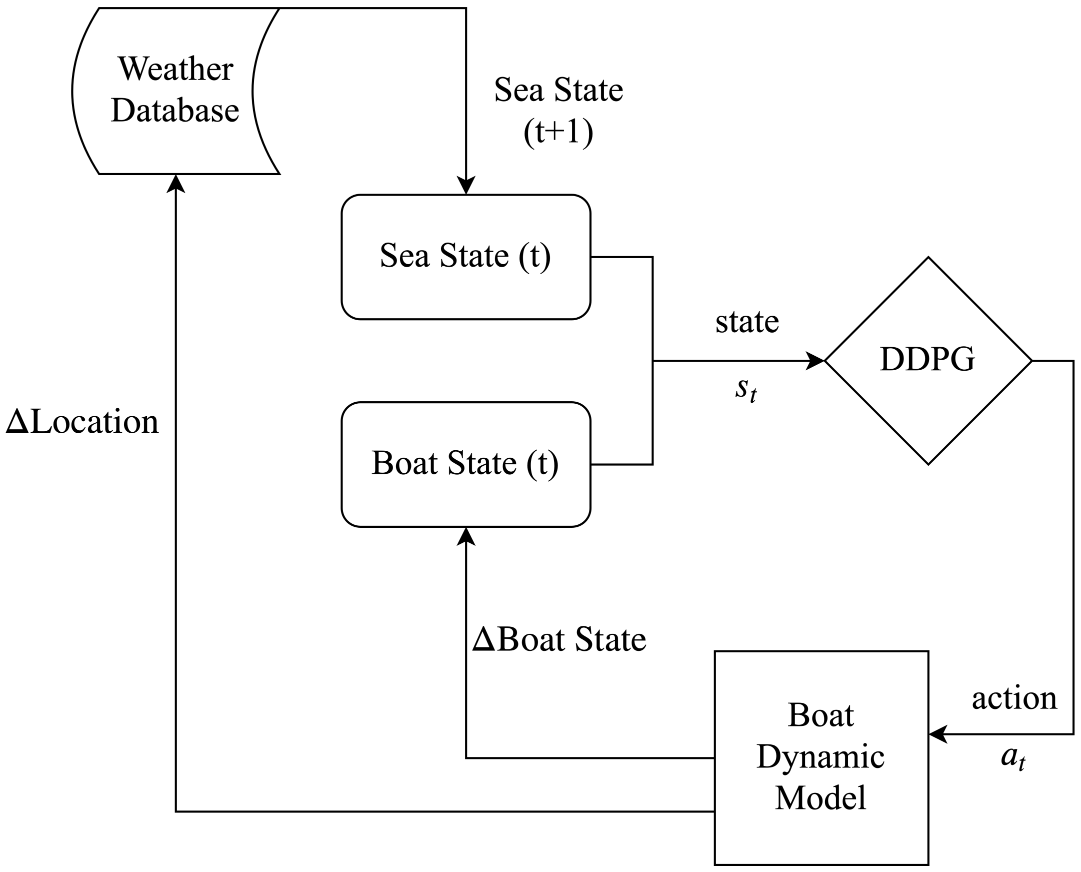
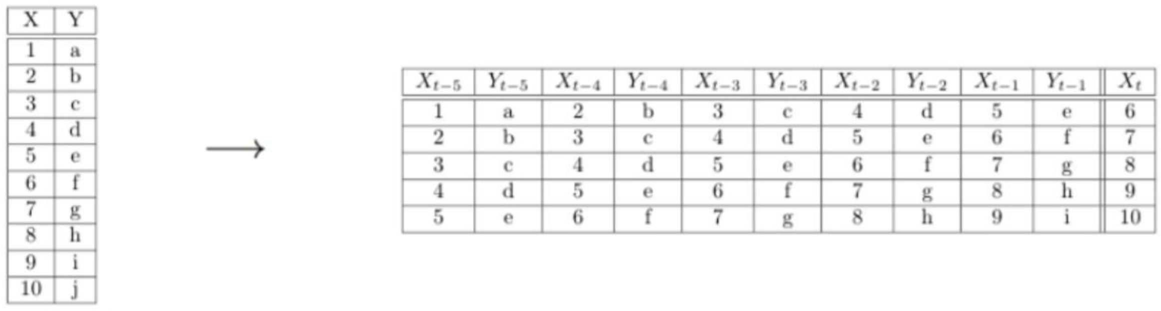


Figure 2.6: Overview of the data flow governing the boat and sea state; as presented in [4]

Model architecture Figure 2.8 depicts the architecture of the LSTM state estimator developed by Roman Kastusik. Table 2.5 presents the model hyperparameters in detail. The LSTM was trained to reduce the mean absolute error (MAE) of the predictions. The LSTM model was implemented using *keras*, for which the data needed to be re-arranged into a suitable format. Indeed, the data comes as timeseries, and a re-arrangement into the format required by *keras* was hence performed. This procedure is illustrated by figure 2.7. The LSTM hence learns to predict the *boat state* at instant t based on knowledge about the boat and sea state, i.e. a single-step multivariate forecast for s_b is performed. n_0 corresponds to the number of time steps before t for which data is taken into account for the forecast, i.e. the length of the time window. As can be seen in table 2.5, this was chosen to be $100s$. The reason for this choice is that this time window was estimated to capture large-scale changes like e.g. wave movements.

Metric	**Value**	**Description**
N layers	2	Number of LSTM layers in the network
N nodes	50	Number of LSTM units in every layer
Observed time n_0 [s]	100	Number of time-steps LSTM is fed through before making prediction
Dropout	0.35	Proportion of inputs to each of the layers dropped to avoid overfitting
Batch size	60	Number of examples run before gradient is updated
Shuffle	**True**	Shuffle training examples
Stateful	**False**	Stateful LSTM

Table 2.5: Model hyperparameters as presented in [4]



X	Y
1	a
2	b
3	c
4	d
5	e
6	f
7	g
8	h
9	i
10	j

$\longrightarrow$

X_{t-5}	Y_{t-5}	X_{t-4}	Y_{t-4}	X_{t-3}	Y_{t-3}	X_{t-2}	Y_{t-2}	X_{t-1}	Y_{t-1}	X_t
1	a	2	b	3	c	4	d	5	e	6
2	b	3	c	4	d	5	e	6	f	7
3	c	4	d	5	e	6	f	7	g	8
4	d	5	e	6	f	7	g	8	h	9
5	e	6	f	7	g	8	h	9	i	10

Figure 2.7: Illustration of the re-arrangement of the time series into the format required by keras; as presented in [5]

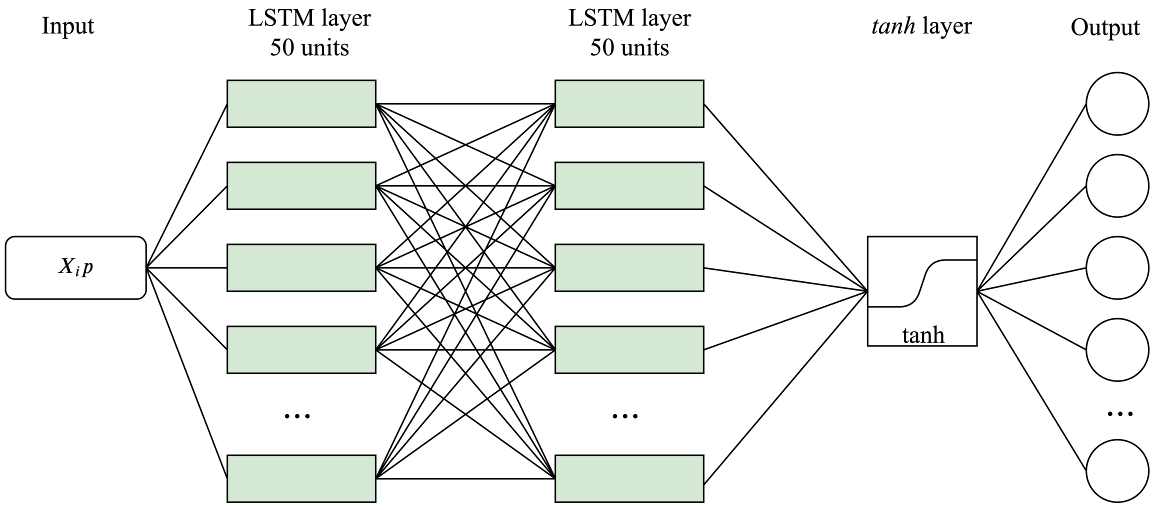


Figure 2.8: Architecture of the state estimator LSTM

Results The model architecture presented above was trained on two different datasets: RDR (nkz) (model 1), and RDR (nkz) *and* DRHEAM 18 (log) (model 2). The metric to be minimised by the models was the overall mean absolute error of all of the features, which had been normalised before (MAE and data normalisation as well as their relevance for the present study are treated in detail in sections 5.5 and 4.5.1). Both models were tested on the RDR (nkz) and DRHEAM18 (log) data. The resulting error metrics for models 1 and 2 are listed in table 2.6, as presented in [4]. Figures 2.9 and 2.10 illustrate the results for the two quantities speed over ground and sin(heading over ground).
While the interested reader is referred to the original study of Roman Kastusik [4], in summary, it can be stated that the predictions of the LSTM model (one model for all boat variables) was found to be of decidedly insufficient quality. Hence, the simulation environment in which the RL agent is to learn optimal rudder steering behaviour is not satisfactory either. However, ignoring these insufficiencies and aiming much more at the construction of a proof-of-concept integrating both the simulation environment and a deep RL agent, a first version of the latter was implemented. The reader is referred to appendix C.1 for further information on this RL algorithm.

Model	Test data	RMSE	MAE	std	Pearson’s correlation coefficient
model 1	nkz	$1.86 \cdot 10^{-3}$	0.03	$2.51 \cdot 10^{-2}$	0.90
model 1	log	$2.31 \cdot 10^{-1}$	0.30	$6.34 \cdot 10^{-2}$	0.24
model 2	nkz	$6.31 \cdot 10^{-2}$	0.18	$6.67 \cdot 10^{-2}$	0.60
model 2	log	$2.23 \cdot 10^{-1}$	0.36	$6.82 \cdot 10^{-2}$	0.48

Table 2.6: Error metrics obtained for model 1 and model2; as presented in [4]

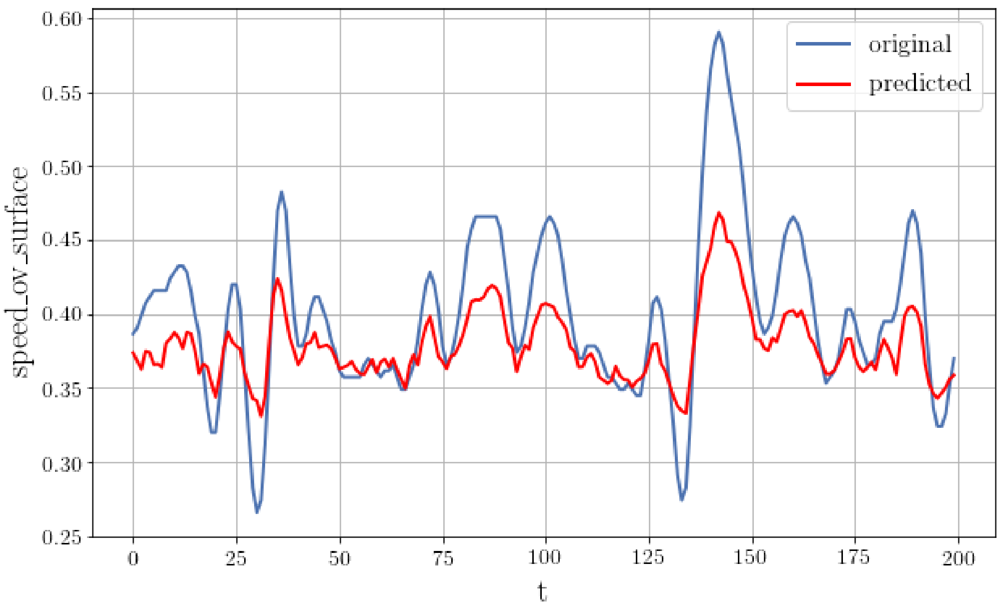


Figure 2.9: Prediction of speed over surface (model 2)

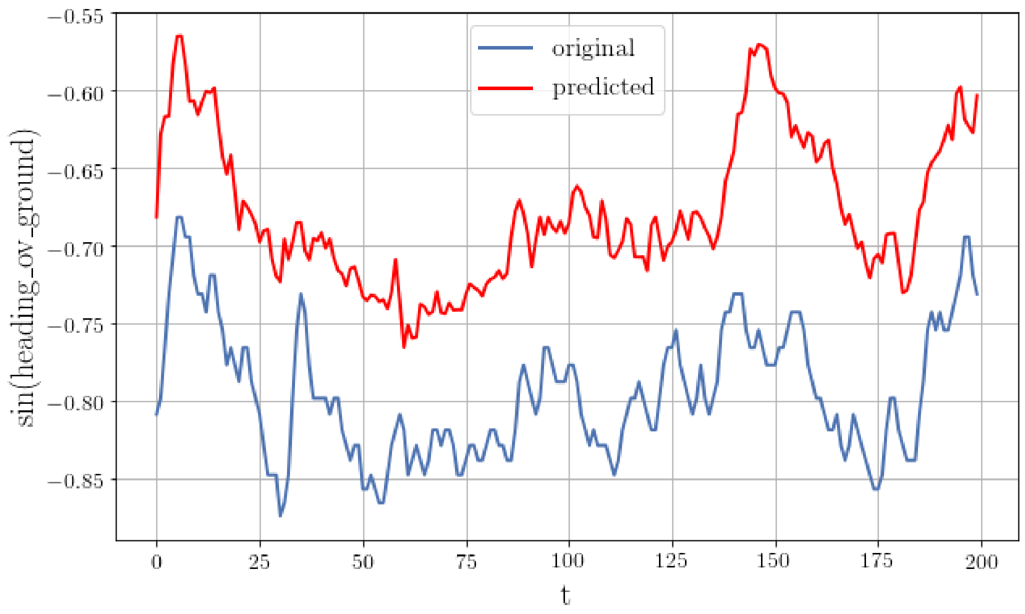


Figure 2.10: Prediction of sin(*heading over ground*) (model 2)

Chapter 3

Literature Review

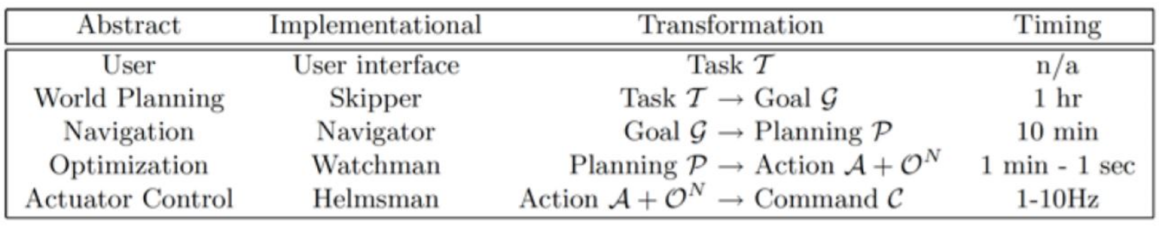
For the reasons mentioned in the previous sections, it should be obvious to the reader that an improvement of the simulation environment would result in significant progress, both in the results for the existing RL algorithm and in a possible further development of that algorithm. This task corresponds essentially to a multivariate timeseries forecasting problem. The corresponding literature is examined in more detail in the following. First, for completeness, an overview of research in the field of autonomous sailboats is provided. Subsequently, an overview of the development and current state-of-the-art of timeseries forecasting is provided.

3.1 Autonomous sailboats

3.1.1 RoboSail Project

In the early 2000s, a group led by Dr Pieter Adriaans carried out significant pioneering work in the field of autonomous sailing boats. In the so-called RoboSail Project, an attempt was made to transfer then novel AI methods to the control of sailing boats. More detailed information on this can be found in the corresponding publication [7] and on the project website [8]. The project was tested on a real sailing yacht, the "Syllogic Sailing Lab Open 40", and in real sea conditions and helped the team win the "Round Britain and Ireland" sailing race in 2002. It was also used on the sailboat "Kingfisher Open 60", which made it possible to cross the Atlantic and also achieved good results for this larger boat.

The AI system owed its success to the development of a hierarchy of tasks on board. A human user can give the system their expert knowledge; under them act four systems which simulate the Skipper, the Navigator, the Watchman and the Helmsman. Each of these systems covers different task areas and time scales, as presented in table 3.1, borrowed from [8].



Abstract	Implementational	Transformation	Timing
User	User interface	Task $\mathcal{T}$	n/a
World Planning	Skipper	Task $\mathcal{T} \rightarrow$ Goal $\mathcal{G}$	1 hr
Navigation	Navigator	Goal $\mathcal{G} \rightarrow$ Planning $\mathcal{P}$	10 min
Optimization	Watchman	Planning $\mathcal{P} \rightarrow$ Action $\mathcal{A} + \mathcal{O}^N$	1 min - 1 sec
Actuator Control	Helmsman	Action $\mathcal{A} + \mathcal{O}^N \rightarrow$ Command $\mathcal{C}$	1-10Hz

Figure 3.1: Hierarchy used in the RoboSail project, table borrowed from [8]

3.1.2 Other research on autonomous sailboats

To the best of my knowledge, the RoboSail project is the only project with strong similarities to the JTR AI project. However, since the RoboSail project, different advances have been realised on aspects of strong relevance for autonomous sailboats.

One of these directions concerns the modelling of waves, which are an important component of the sailboat's environment and strongly influence the behaviour of the skipper. In fact, a human skipper tries to adjust the rudder angle such that the boat surfs on waves, i.e. such that the speed of the boat is increased by taking advantage of the waves. This also largely explains why human skippers perform better than classic autopilots, which essentially stick to a certain direction of the boat and refrain from making intelligent use of the waves. An intelligent autopilot would hence take advantage of the waves; to that end, it would have to have the relevant information or modelling at its disposal.

In this context Duz, Mak et al. have investigated the real time estimation of wave characteristics [9]. For this purpose, the performance of artificial neural networks was examined, which are trained to determine the wave height, the wave period and the wave angle (angle of the wave in relation to the boat) on the basis of a time-series of the 6 degrees of freedom (DOFs) of the boat (pitch, roll, yaw angles and latitude, longitude as well as vertical position). Further information is not provided to the models; they are "ship-agnostic". A multivariate LSTM-CNN and a Sliding Puzzle Network were investigated. Good results were obtained for the modelling of the wave angle and of the wave height; for the wave period the results remained improvable. It should be noted, however, that the input data included the vertical position of the boat, a variable that is not available in the data used in this thesis.

Shen, Wang et al. also present a model to describe wave sizes in their work [10]. However, the model used is highly simplified and assumes 4 DOFs to describe the boat state (roll and yaw angle, longitude and latitude); an exact description of the wave characteristics is not the primary goal of this work. Rather, it is the optimisation of an unmanned sailboat's speed using a first-principles approach to describe the boat's dynamics. On the basis of the latter, a feedforward and feedback control scheme is developed, by which the boat should reach the maximum speed in a given direction according to the speed polar diagram (cf. figure 2.5).

Another approach based on first-principles was presented by Deng, Zhang in [11]. A first principles model of a catamaran allowed them to optimize the path following of the catamaran, which essentially consists in the catamaran's following a number of waypoints (i.e. coordinates) that together constitute the route to be sailed. This differs from the approach explored by Roman Kastusik described in C.1 towards which the present study contributes, which essentially consists in an RL agent learning to be as close as possible or even ahead of a real boat for which data have been gathered, but not in following a route described by waypoints as closely as possible. Furthermore, the focus of Deng, Zhang et al.'s work is on optimizing the control system of the catamaran: a "robust fuzzy control scheme" is used to optimise the classical control scheme of the boat, which constitutes the central part of this work. While the work includes practical considerations like the saturation of actuators, the use of neural networks architectures is not further explored. Finally, Zhang et al. present in [12] "a waypoint-based path-following control for an unmanned robot sailboat". Again, a first-principles model is used and the study focuses on the optimization of the control system of the unmanned sailboat. However, within the closed-loop control system, Radial Basic Function Neural Networks (RBF-NNs) are used to optimize the structure and parameters control scheme. Again, machine learning methods are only used to improve a classical control loop of an autonomous sailboat, while their use for simulating the boat's environment is not investigated.

3.2 Timeseries Forecasting: Evolution and State of the Art

As mentioned previously, the present work is mainly concerned with accurately forecasting a boat's future states in a complex, noisy and dynamic environment. Timeseries forecasting is a broad field with many applications: engineering, finance and retail are only some of the areas where timeseries can be used to generate forecasting value by predicting system states, stock prices or sales figures. Accordingly, this field has long been the object of many research activities.

A good overview of the evolution of state-of-the-art methods in the field of timeseries forecasting can be found in the work of De Gooijer and Hyndman [13], who present the developments in the field from 1980 to 2006 and present multiple different approaches, including ARIMA, exponential smoothing and Kalman filters. Similarly, a good overview of the development of ML methods for timeseries forecasting until 2010 can be found in Ahmed, Atiya et al. [14], who describe the development of multilayer perceptrons, Bayesian neural networks, k-nearest neighbours and other methods for timeseries forecasting until 2010. Taieb, Bontempi et al. also review different methods for timeseries forecasting until 2012 [15], and study the (improving) effect of deseasonalisation on forecast accuracy.

While the above-mentioned studies primarily depict the development of timeseries forecasting until about 2010, Parmezan, Souza et al. present a comprehensive overview and evaluation of state-of-the-art models for timeseries forecasting ([16], 2019). The publication is particularly interesting for readers who want to get a good overview of both statistical methods (e.g. SARIMA) and machine learning algorithms (artificial neural networks, support vector machines, kNNs), and at the same time get a comprehensive introduction into the timeseries forecasting problem. Finally, Zheng authored a comprehensive report on trajectory data mining [17], providing a useful overview of problems, solutions and metrics applicable to data mining in the context of trajectory data. Although this study is motivated by and focused on trajectory data generated by mobile phones, it presents useful methods for trajectory outlier detection, trajectory uncertainty and trajectory classification.

3.3 Deep Learning for Timeseries Forecasting

As is apparent from the previous section, novel machine learning methods are increasingly present in the field of timeseries forecasting. Deep learning in particular is playing an increasingly prominent and successful role. In the following, recent developments and state-of-the-art deep learning approaches to the timeseries forecasting problem with particular relevance for the present project will be listed.

Abbasimehr, Shabani et al. present a model based on an LSTM network for demand forecasting [18]. While their model is of little relevance to the present project, the comprehensive literature review on timeseries forecasting with a special focus on RNN models is of particular interest. Similarly interesting is the publication by Sezer, Gudelek et al., who present a systematic overview of financial timeseries forecasting using deep learning techniques [19]. Despite the limitation to financial timeseries, the publication provides useful insights as it discusses MLPs, RNNs, Deep Belief Networks and deep RL in detail, all of which are methods that are also transferable to forecasting of other types of timeseries. As will be shortly touched upon in section 2.1, a relevant and much-noticed aspect of timeseries forecasting concerns the length of the time window which deep learning models are given as input. This is also of interest for the present paper, since the input time window is crucial for the performance of the model. In that light, Zhang, Wu et al. investigate the simultaneous input of four different lengths of a timeseries into an RNN model for an electric load forecasting task [20]. This method allows to capture phenomena that occur on different time scales. The authors of the study explicitly propose to apply the method in other fields and to test its application to bidirectional LSTMs (BLSTMs).

A paper by Xia, Song et al., who use a CNN-BLSTM architecture to predict the remaining useful life (RUL) of a turbofan based on timeseries data of turbofan sensor data, goes in this direction [21]. Using this method, they achieve results that are competitive with the state-of-the-art of that moment. To be more precise, several CNN-BLSTMs are combined in an ensemble framework, where each CNN-BLSTM is trained and validated for a different time window. This allows to capture phenomena with different time windows and to reflect them in the model. Figures 3.2 and 3.3 from the publication of Xia, Song et al. [21] illustrate the method.

The use of different time windows is taken to extremes by Baig, Ibal et al. who use adaptive time windows instead of static time windows as input to timeseries forecasting models [22], at least during training. Using this method, they achieve state-of-the-art results for a single-step uni-variate timeseries forecasting problem consisting in the estimation of resource utilisation of a data center.

Since multi-variate timeseries forecasting is relevant for the present project - the boat state s_b presented in 2.4 is a vector containing more than one value - the study by Du, Li et al. is of interest, which presents a ”novel temporal attention encoder-decoder model” based on BLSTMs [23]. It achieves good results when applied to real-world timeseries (air quality, power consumption, traffic). Finally, a publication by Bandara, Bergmeir et al. proposes to use clustered timeseries to train LSTMs on timeseries forecasting [24]. The clustering allows to construct different models for the different timeseries clusters and is designed to be applicable to not only LSTMs but other types of RNNs as well, which is found to augment the forecast accuracy.

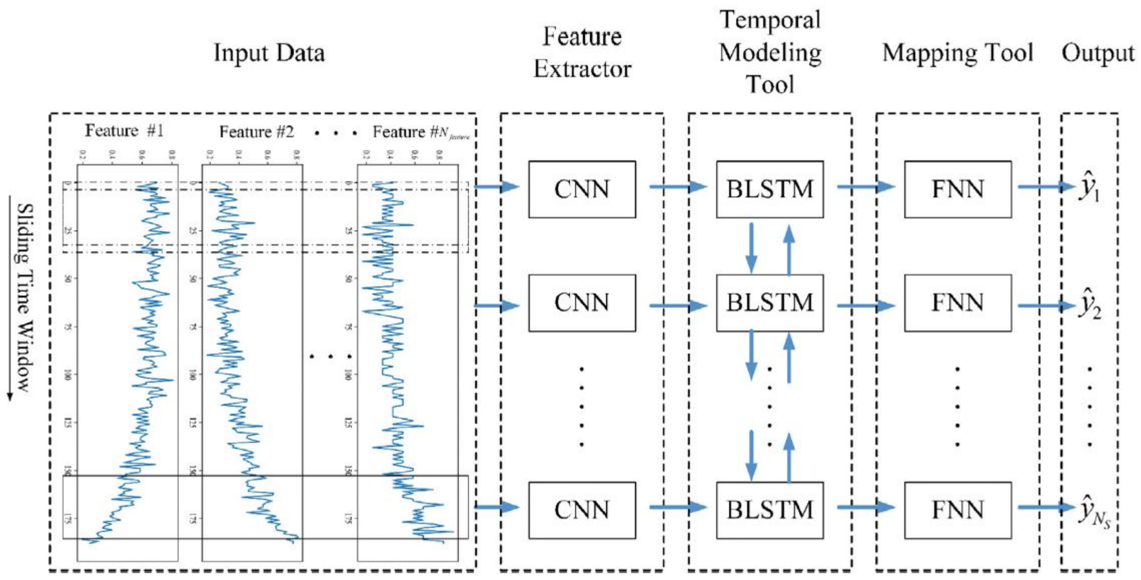


Figure 3.2: Framework of the CNN-BLSTM base model proposed by Xia, Song et al. [21]

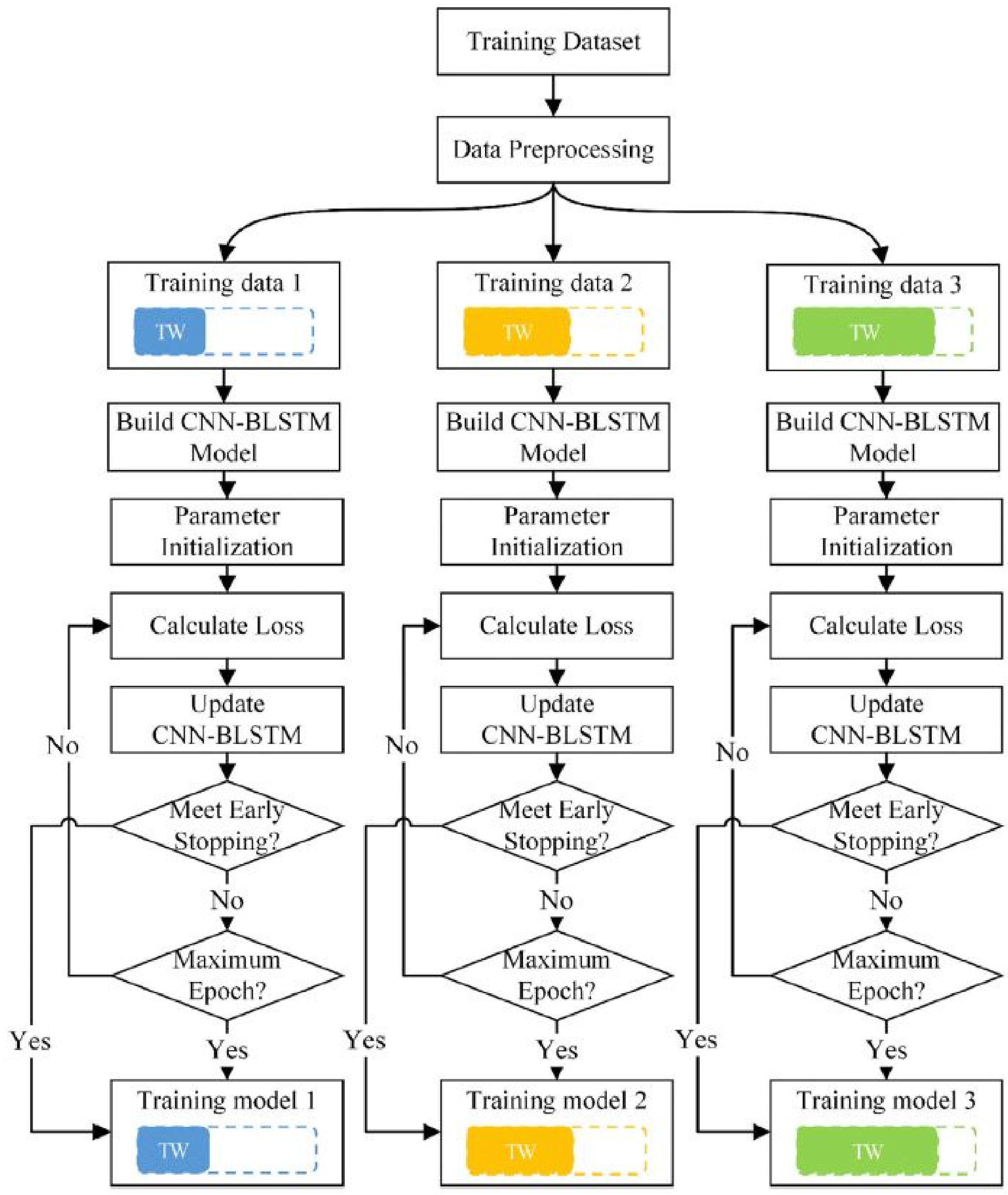


Figure 3.3: CNN-BLSTM training procedures with multiple time windows as proposed by Xia, Song et al. [21]

3.4 Generative Adversarial Networks for Timeseries Forecasting

The application of generative adversarial networks (GAN) in the field of timeseries is a relatively recent development, but offers interesting perspectives. These will be discussed in the following.

A milestone in this field was set by the paper of Esteban, Hyland et al. in which GANs are used for the generation of synthetic timeseries data [25]. Using multivariate medical timeseries data from an intensive care unit station, synthetic multivariate timeseries are generated. The generated data prove their worth by first training timeseries forecasting models on the synthetic timeseries, and then testing them successfully on the original, real timeseries (”Train on Synthetic, Test on Real” or TSTR). This approach is of particular interest in the medical field, where timeseries is needed for the training of medical staff, but is not readily available due to stringent privacy regulations.

This method has also been applied by Hartmann, Schirrmeister et al. who present modified Wasserstein GANs for the generation of synthetic electroencephalogram (EEG) timeseries [26]. In this paper, the GANs are also successfully used for data augmentation as well as restoration of corrupted data segments. Using a similar approach, Fekri, Ghosh et al. use GANs to generate synthetic energy consumption timeseries [27].

The use of GANs has thus also been introduced in the area of timeseries, but as described above mainly for the generation of synthetic timeseries. A paper that uses this approach to make predictions about future system states is that of Li, Chen et al. [28]. It presents a method to detect anomalies using GANs for multivariate timeseries. Using a GAN based on an LSTM-RNN architecture, the distribution of the multivariate timeseries of a number of sensors and actuators in a water treatment system under normal conditions is learned. Instead of discarding the discriminator after the training of the generator, in the production phase, the discriminator is used to detect anomalies in the timeseries.

Finally, Koochali, Schichtel et al. present a method to use GANs for single-step timeseries forecasting [29]. They present good results when applied to timeseries of nonlinear dynamic systems, especially when the data present strong noise. Figure 3.4 presents an overview of the proposed architecture; figure 3.5 presents the details of the generator and the discriminator architectures. From these figures, it can be seen that the generator learns to generate good ”fake” values for the next point in time of a given timeseries, and the discriminator learns to recognize these fakes. This trains a generator to make plausible fakes, i.e. good forecasts. Strong results on two different nonlinear dynamic systems are found. The authors explicitly suggest applying the method to multi-step problems as a direction of further work.

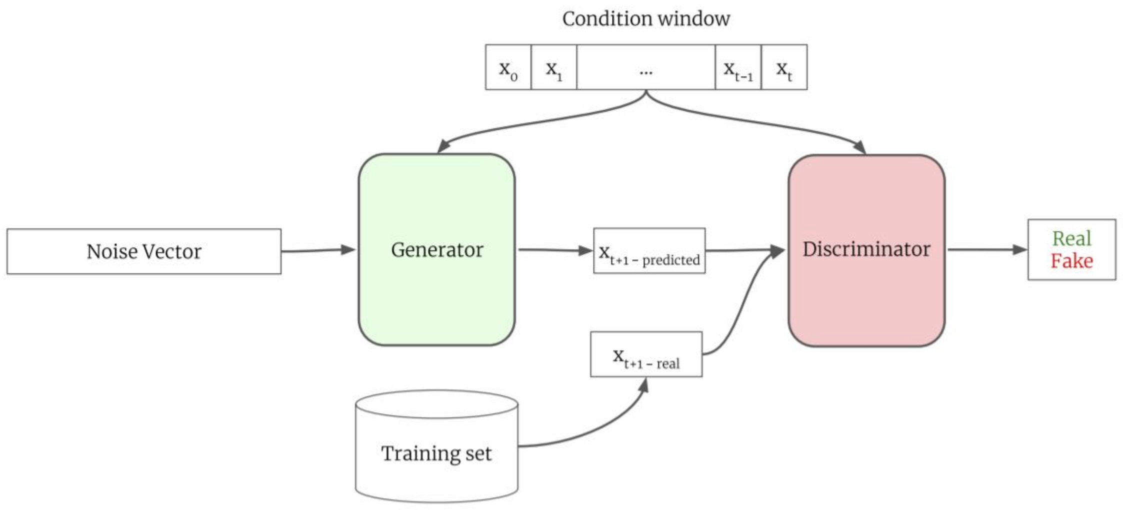


Figure 3.4: Overview of the architecture proposed by Koochali, Schichtel et al. [29]

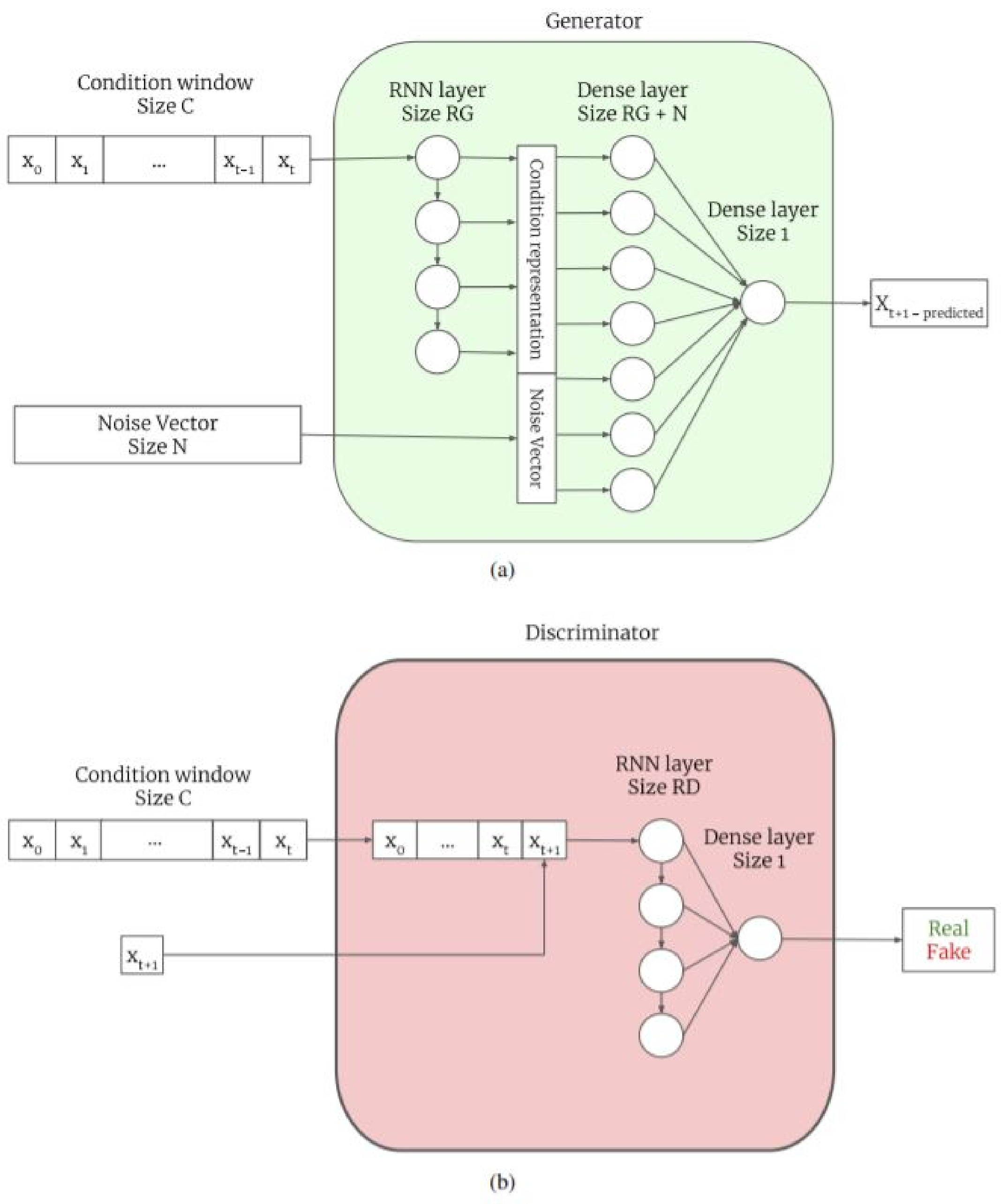


Figure 3.5: Detailed architectures of the generator (a) and the discriminator (b) as proposed by Koochali, Schichtel et al. [29]

3.5 Hybrid Models of Dynamic Systems

If first-principles models and ML models are presented separately in the previous sections, there is also an approach to combine first-principles methods with ML approaches to reliably forecast the behaviour of real-world dynamic systems. Roughly summarized, these so-called hybrid models aim to exploit the best of both worlds by making a timeseries forecast based on a first-principles model, which is then improved by ML models by the deviations from the values found in the real world.

Rasheed, San et al. present an overview of the evolution and current state-of-the-art of this approach to dynamic systems modeling in the broader context of ”digital twins” of physical systems [30]. Parish and Carlberg present a comprehensive and in-depth coverage of the mathematical details of hybrid models, especially with respect to error modeling [31]. An example for the concrete application of hybrid models is provided by Wu, Rincon et al., who embed a model based on physical principles into an RNN network structure to model nonlinear chemical processes [32]. Finally, Mohajerin and Waslander apply hybrid models for multi-step prediction of timeseries of two dynamic systems, namely a helicopter and a quadrocopter [33]. Furthermore, they present a novel method for an improved initialisation of RNNs.

3.6 Conclusion and resulting scope of the study

As seen in the previous sections, research into intelligent autopilots for sailboats has been very little explored. Although Adriaans et al. pioneered this domain in the early 2000s, no comparable comparable work has been conducted since. Even though autonomous sailboats have enjoyed further research, especially in the field of control, no comprehensive project like the one by Adriaans et al. has been undertaken since.

3.6.1 Conclusion

Motivation for choices

However, the supervised rudder prediction problem showed that one can achieve very good performance of single step uni-variate forecasting. Roman Kastusik’s work showed that RL could potentially be applied to the rudder prediction problem. Nonetheless, this remains unproven due to the unsatisfactory performance of the forecasting model. It follows that an improved simulation environment is a promising research area. This corresponds essentially to a single-step multivariate timeseries forecasting problem. It should be emphasised that the development of reliable forecasting models for a satisfactory RL simulation environment constitutes one major step in the advancement of the RL-based approach to this problem. However, it does not guarantee that the RL algorithm itself is satisfactory.

Possible directions of investigation

Different directions are possible to solve the single-step multi-variate forecasting problem and have been researched extensively. Especially the application of Deep Learning in this domain has been an object of strong interest recently. Additionally, novel approaches using GANs for forecasting hold potential. Finally, hybrid models have also been explored for this task.

Choice of direction of investigation

As deep learning and more specifically RNN architectures have recently delivered promising results for many and partly comparable cases, it was decided to analyse this approach in a first phase of the study. Moreover, since this field is currently being actively researched, the present study might also result in a contribution to this subject of interest. The investigation of GANs for forecasting could be of interest for future studies. Indeed, the field is currently being actively explored and interesting contributions could result if its methods were to be applied the problem at hand. Precisely, the applicability of GANs for forecasting could be investigated for the present problem, which is of higher complexity than the problems investigated in previous studies (notably Koochali, Schichtel et al. [29]). However, successful outcomes are less probable with this less explored approach than with the Deep Learning architectures, which have been researched quite extensively. Finally, hybrid models were not retained for further investigation. The degree of novelty of this field and thus a possible contribution to this research area would be rather insignificant.

3.6.2 Scope of the study

Following the conclusions from above, the overarching goal of the present study was defined as the development of a reliable RL simulation environment. This simulation environment effectively corresponds to a reliable forecasting model of the boat state features, as developed in a first, but decidedly unsatisfactory attempt by Roman Kastusik (cf. section 2.4.1).

Two aspects need to be addressed to that end. On the one hand, a data pipeline must be established such that the models can be trained, validated and tested on suitable data. Second, a strategy must be defined to develop the reliable forecasting models.

Data pipeline

The pipeline must

1. **convert** the data from their nke-proprietary format (.nkz) to a format for which data analysis and ML libraries are available (.csv).
2. **clean** the data from any corrupted and irrelevant recordings.
3. allow to consider the **distribution** and main characteristics of the data.
4. allow to **select** data that is conducive to the aim of identifying reliable forecasting models.
5. **preprocess** the data, i.e. apply any required mathematical transformations to it, as well as ensure its format corresponds to the format required by the forecasting models.

In summary, the pipeline must ensure that data is transformed from its raw .nkz state to a clean state in which it can be used by forecasting models.

Strategy to identify forecasting models

Using this data, the overarching goal of the present study can be pursued, i.e. the identification of reliable forecasting models. To that end, a progressive approach is chosen, consisting in the successive verification of the following hypotheses:

1. **Hypothesis 1: the performance of 1 model for n boat state features, as used by Roman Kastusik, can be improved by optimising the model's hyperparameters.**
   On the basis of the satisfying results of the Bayesian optimisation in the rudder prediction problem (section 2.3.3), Bayesian optimisation is employed for this step. It aims at clarifying whether the original model can be improved at all, as well as to assess whether the optimised model's accuracy is satisfactory.

2. **Hypothesis 2: training n separate forecasting models for n boat state features yields more accurate predictions than training 1 model for n features.**
   The underlying rationale is that a single model for n features must optimise the predictions for all features at once, i.e. it optimises towards one overall error metric. In contrast, training n models separately for n features allows each model to be trained only on those dynamics that matter for "its" predictions. This could potentially result in more accurate forecasting models, which is the focus of this study.

3. **Hypothesis 3: if a model's hyperparameters lead to inaccurate predictions, removing one dense layer improves the prediction accuracy.**
   Indeed, not the same level of complexity may be required to be captured by each model; architectural changes might be able to take this into account. Precisely, as has been described by Talathi et al. [34], dense layers can directly influence the level of complexity that a model might take into account.

4. **Hypothesis 4: Where they can be used, deterministic models achieve better performance than LSTM-based models.**
   Some boat state features behave according to formulae derived from definitions or physical first principles. These formulae might be able to predict values better than models based on deep learning, as they encapsulate the true physical relations between features. Indeed, the overarching goal of the present study consists in the identification of reliable forecasting models. These do not need to be deep learning models *per se*.

5. **Hypothesis 5: Where they can be used, deterministic models serve as an indicator of a dataset's inaccuracies.**
   As mentioned in the previous point, certain features are defined as direct and unambiguous functions of other features. One can hence compute these features' values by deterministic models and compare them to the "true" values

recorded in the datasets. Any deviations between the two should be an indicator of inaccuracies in that dataset.

6. **Hypothesis 6: Hyperparameters of forecasting models that are accurate for a specific data format can be used to train accurate forecasting models for the same boat, but using data of another data format.**
   Indeed, the Concise 8 (DRHEAM 18) dataset was recorded in an old data format, while the new data arrives in a different format (.nkz; cf. 2.2). If this holds true, one can optimise a model's hyperparameters for a given boat's data coming in a specific data format, and simply re-use them to train models for data coming in a new format. This would avoid hyperparameter optimisations, which can be lengthy and computationally costly. In other words, hyperparameters leading to good results for Concise 8 (DRHEAM 18) could be re-used to train models on the Concise 8 (Atlantic) dataset, which comes in the .nkz format.

7. **Hypothesis 7: Hyperparameters of forecasting models that are accurate for a specific data format can be used to train accurate forecasting models using data from a different boat, coming in another data format.**
   This hypothesis corresponds to an extension of the previous one: if it holds true, no hyperparameter optimisation needs to be conducted for data from a new boat coming in a different format. In other words, hyperparameters resulting in reliable predictions for Concise 8 (DRHEAM 18) could be re-used to train models on the Unknown 1 (transat_1) dataset, which comes in the .nkz format.

A number of experiments can be set up to verify each of these hypotheses. They all contribute towards the main goal of the present study, consisting in the identification of reliable forecasting models for the boat state's features.

Chapter 4

Data

The aim of the present study consists in developing accurate forecasting models of the features that define the boat state as listed in section 2.4.1, as well as to conduct experiments with respect to the performance of these models for different boats and datasets. This chapter presents the data available for this purpose.

4.1 Gathering

As described in section 2.2, new navigation logs have become available in the nke-proprietary format .nkz, in which data from boat sensors is recorded at 25 Hz. Datasets to be received in future iterations of JTR AI are going to be in this specific format. The conversion of these files into .csv files that are usable by standard programming libraries is done with the software LogAnalyser and was described by Roman Kastusik in his thesis ([4]). This section describes the unforeseen challenges that this data presented, which is always a risk with real world data.

4.1.1 Challenges encountered

Description of main challenge However, a persistent problem occurred with the conversion of the new, .nkz-based files to the usable .csv format. Indeed, for the latitude and longitude features, the conversion resulted in their values corresponding only to a very reduced set of values, as with a precision of 16 decimal places were expected and necessary. An example of a sailed trajectory consisting of these truncated latitude and longitude values is given in Appendix D.

Implications for the present study Only at the beginning of August 2020 and after a lengthy exchange with nke could the correct conversion procedure be established. This unforeseen challenge added considerable overhead, affecting the scope of the study. It should be emphasised that the resolution of this issue was not only of primordial importance for the present study, but also for future iterations of JTR AI, as there is no other way to obtain the navigation logs from the nke autopilots.

Further challenges It can generally be stated that the logging of the navigation data by the nke autopilot is not consistent between boats. This concerns the number of features recorded, which can vary from recording to recording. Furthermore, the order in which the features are logged can also change from file to file. Hence, atop of requiring substantial computational and time resources, the conversion from .nkz to .csv files is tedious and requires multiple manual interventions.

4.1.2 Implications on conversion of datasets

Entirely converted datasets The datasets for which pre-processing and conversion was successful are

- Atlantic
- RDR
- DRHEAM 20

They were hence entirely converted. This was also performed in view of future iterations of JTR AI for which this data is likely to be valuable.

Partially or un-converted datasets However, the conversion process was particularly time consuming for the following datasets:

- transat_1 (Unknown 1)
- transat_2 (Unkown 2)

The conversion required multiple interventions to convert the .nkz files to .csv. Hence, in an effort to limit the resources devoted to this data gathering aspect of the study, it was decided to convert only a part of this data.

Determination of necessary dataset volume The overall objective of the present study should be kept in mind, which consists in the identification of hyperparameters of reliable forecasting models for boat state features. This includes an assessment as to whether these hyperparameters are adequate for models trained on different data formats (i.e. old DRHEAM 18 format vs. new .nkz-based format). It also includes the assessment as to whether the hyperparameters are adequate for models trained on datasets of the same format, but of different boats (e.g. Concise 8 and Unknown 1). For the latter, it is sufficient to have one other boat than Concise 8; this boat was chosen to be Unknown 1 (transat_1). Furthermore, for these analyses it is sensible to use comparable datasets, where comparability includes the mere volume of the datasets. Hence, it suffices to convert only as much of transat_1 as is needed to have a dataset length that is comparable to the length of cleaned data available for DRHEAM 18's, i.e. 64.5 hours (as inherited from previous iterations of JTR AI, cf.

2.2). The number of hours that is necessary to obtain a comparable length after cleaning was found to be 67 hours, as is listed in table 4.1. The already briefly mentioned cleaning process of the converted data is explained in the next section.

4.2 Cleaning

Irrelevant data As explained in detail in section 2.3.1, it is of advantage not to train an ML algorithm (whether supervised or RL-based) on segments with certain undesirable characteristics. These include

- tack maneuvers
- abnormal conditions which are irrelevant for racing (e.g. low wind speed)
- corrupted recordings with incomplete data
- segments where the boat was anchored or in the harbour, i.e. without any sailing being performed

Motivation Indeed, JTR AI focuses on improving the behaviour of the autopilot only when it is engaged in normal sailing. Hence, data with the mentioned characteristics is not of relevance for the present study and needs to be removed from the dataset.

Cleaning procedure Hence, following the approach explained above in section 2.3.2, the converted data was subdivided into segments of 60 seconds. The data was examined for the mentioned undesirable characteristics, segment by segment. If such characteristics were detected, the corresponding segment was noted and not retained for further use. The result is shown in table 4.1. The following observations can be made:

- For the Concise 8 (Atlantic) dataset of Concise 8, 228.7 hours are left after cleaning, which is much more than required a priori.
- 16 more hours of data from Concise 8 is available from RDR.
- 63.3 hours of the new converted data remain for Unknown 1, which is only 1.2 hours less than the number of hours recorded for DRHEAM 18 (64.5 hours) and hence of a comparable length.
- For VMB (DRHEAM 20) only 6 hours remain, because the boat was in a harbour during one of the recorded 7 hours.

At this point it should be emphasised that in Roman Kastusik's, the available data did not pass through these pre-processing steps. Hence, his state estimator was also trained on segments containing tacks or low wind speeds, which is not desirable for the reasons mentioned in section 2.3.1. This is a first clear difference to Roman

Kastusik's approach.

After this cleaning of irrelevant segments, a look at the data provides information about the material available, which finally allows to make a selection of the datasets to be used.

Boat	**Name**	**Original length [h]**	**Converted length [h]**	**Cleaned length [h]**
Concise 8	RDR (nkz)	16	16	16
	DRHEAM 18 (log)	64.5	91	64.5
	Atlantic	290.9	290.9	228.7
VMB	DRHEAM 20	7	7	6
Unknown 1	transat_1	383.5	67	63.3

Table 4.1: Overview of the original length, converted length and the cleaned length of the datasets available for the different boats.

4.3 Distribution

The goal of the present study consists in identifying hyperparameters of reliable forecasting models of the boat state features. Furthermore, it is attempted to assess whether these hyperparameters are adequate for different formats of the datasets and for models trained on different boats.

Motivation The selection of those data on which these forecasting models are trained and of those on which their performance is tested is a cardinal step in this process. Indeed, reliable forecasting models should capture the physics of the boat reliably for as broad a distribution of sailing conditions as possible.

Features describing the physical environment Only some features are relevant to describe the physical environment and state of the boat. Indeed, some features are partially redundant to others and some correlate strongly to others (e.g. boat and wind speed under normal conditions). Therefore, 4 features were selected that capture the essential elements of the environment as experienced by the boat. In consultation with Dr. Eric Topham, internal supervisor of the study with extensive sailing experience, these features were found to be:

- **True Wind Speed (TWS)**: important for the lift generated from the airflow over the sails, as well as affecting the sea state.
- **True Wind Angle (TWA)**: the angle at which the wind hits the boat in relation to its orientation when the boat would be stationary. It allows to characterize the angle of the wind in relation to the boat at any given moment.

- **Apparent Wind Angle (AWA)**: characterizes the angle at which the wind effectively hits the boat when the boat is in motion.
- **Pitch**: describes the inclination of the boat's bow; numerous high and low values for pitch hence indicate an agitated sea that would raise and lower the nose of the boat. Out of the pitch, roll and yaw angles that describe the attitude of the boat, pitch is the best for approximating the sea state. Indeed, roll can be affected by wind which can be independent of the sea state, and yaw can be affected by the helmsman steering the boat in a choice of direction unrelated to the sea state.

The distribution of these features' values is described in figures 4.1 to 4.4 for the cleaned datasets. The following observations can be made:

- What is presented in visual form here corresponds to what can already be seen in table 4.1, namely that the datasets differ in their sheer volume: DRHEAM 20 contains 6 clean hours, while Concise 8 (Atlantic) has almost 229 clean hours.
- Accordingly, the longer datasets cover a greater variety of conditions. This is reflected by the fact that for the shown key features, Concise 8 (Atlantic) covers values from almost all areas and also contains e.g. the rare moments where very high True Wind Speeds prevail (fig. 4.1), while DRHEAM 20 contains only the values specifically found during the 6-hour recording period.
- Finally, it can be noted that in fig. 4.2, True Wind Angles are also found in suboptimal angles close to -180° and 180° (suboptimal according to the polar plot 2.5). This can be explained by the fact that Concise 8 (Atlantic) was a single-handed delivery, meaning that the autopilot was switched on over long distances, while the other datasets were recorded during double-handed races, where a person was usually behind the tiller (cf. section 2.2). The standard autopilot just follows a given compass heading without adapting its behaviour to the TWA like a human would. Indeed, if the true wind direction remains changed anfd the boat heading remains fixed, then the boat will experience a change in true and apparent wind angles resulting in the sailing under suboptimal wind angles.

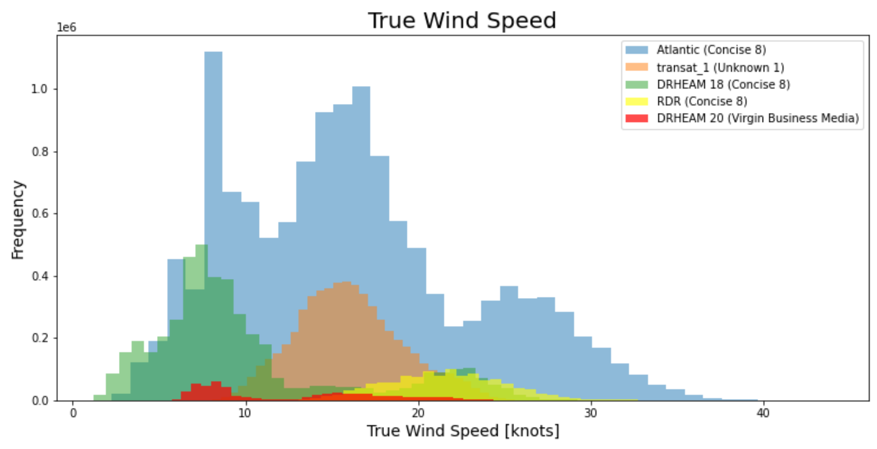


Figure 4.1: Distribution of True Wind Speed (TWS) in the available datasets.

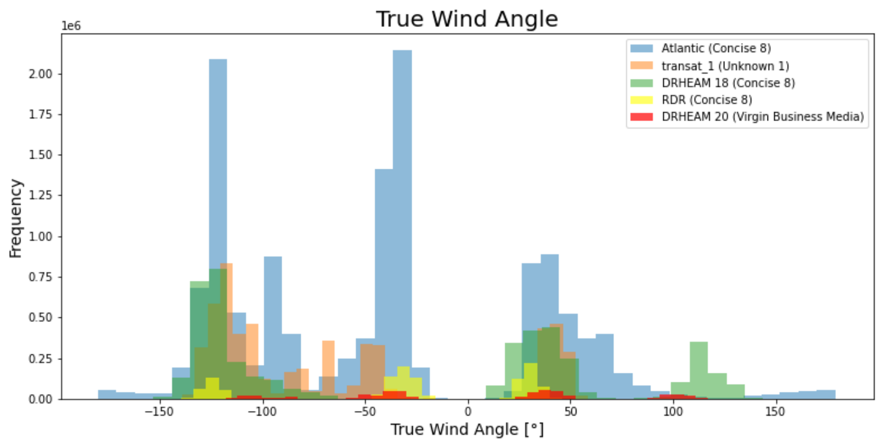


Figure 4.2: Distribution of True Wind Angle (TWA) in the available datasets.

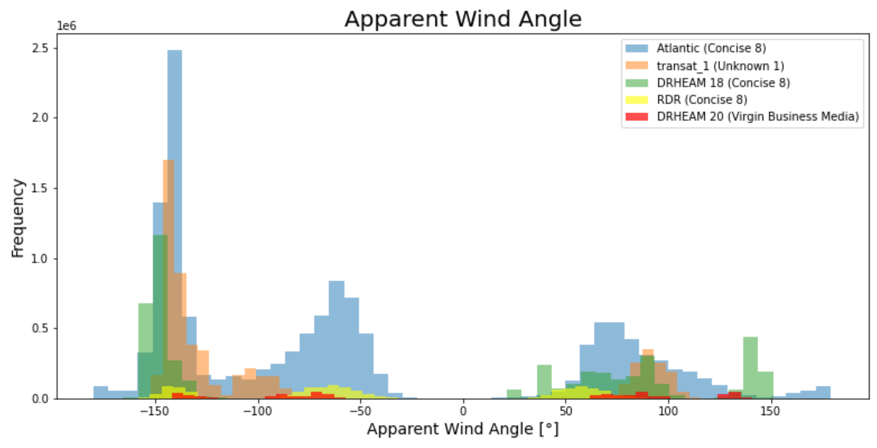


Figure 4.3: Distribution of Apparent Wind Angle (AWA) in the available datasets.

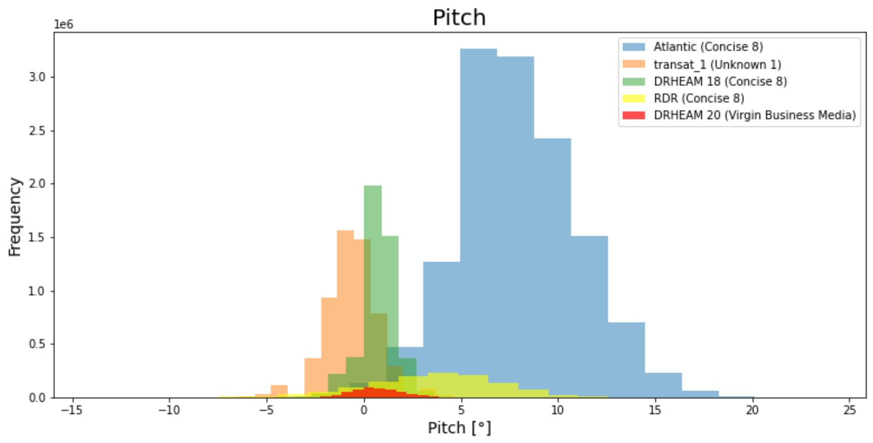


Figure 4.4: Distribution of Pitch in the available datasets.

4.4 Selection

As stated in the introduction to the present chapter, the goal of data gathering, cleaning and pre-processing is to generate datasets which are comparable in length to the DRHEAM 18 ”ersatz” dataset.

4.4.1 Choice of datasets

Unknown 1 Since only one dataset is available for the boat Unknown 1 (cf. table 4.1), the entire converted and cleaned dataset Unknown 1 (transat_1) is logically used. With 63.3 hours of converted and cleaned data, the dataset has a length that is comparable to DRHEAM 18.

Concise 8 A different picture emerges for Concise 8, since in addition to the very long Concise 8 (Atlantic) dataset, the COncise 8 (RDR) and Concise 8 (DRHEAM 20) dataset are also available. As explained earlier, DRHEAM 20 is not retained due to its inconsiderable length. Furthermore, as explained in the previous section, Concise 8 (Atlantic) covers a much wider range of sea states; virtually everything that is included in RDR is already included in Concise 8 (Atlantic). Chapter 6 will explain in more detail why it is of interest to train the models on datasets which show a certain diversity; for the understanding of the present section, it suffices to state that this allows the models to generalize their capabilities for as many different states as possible. Accordingly, the Concise 8 (Atlantic) dataset was retained as the dataset for Concise 8 and RDR was excluded from further use in this study.

4.4.2 Data splitting

Method For the splitting of the selected data into training, validation and testing datasets, the method described in 2.3.2 was used. It should be briefly recalled that this method consists of considering contiguous segments between two anomalies or between two tacks, and for each of those segments assign the first 60% of the data to the training subset, the next 20% to the validation subset and the last 20% to the testing subset, as opposed to taking this split just once over the *entire* dataset. Fig. 4.5 to 4.11 show the resulting distribution of the previously mentioned key sea state features for the dataset Unknown 1 (transat_1), now divided into training, validation and testing subsets.

Resampling for large datasets However, performing this step for Concise 8 (Atlantic) does not affect its consisting of almost 229 (cleaned of tacks and anomalities) hours, which is considerably more than the ca. 65 hours of data which are deemed necessary at all. Hence, only a subsample of the training, validation and testing splits was retained with a total duration of 63.3 hours, i.e. exactly the same amount as is available for Unknown 1. Hence, after splitting the data into training, validation and testing subsets just as for the transat_1 dataset, the subsets were each divided into 5 parts, respectively. For each of the subsets, only the first 27.7 % from each of these 5 parts were retained and concatenated, so that the desired 63.3 hours of total training, validation and testing data were obtained. Fig. 4.6 to 4.12 show the distribution of the subsets that were retained.

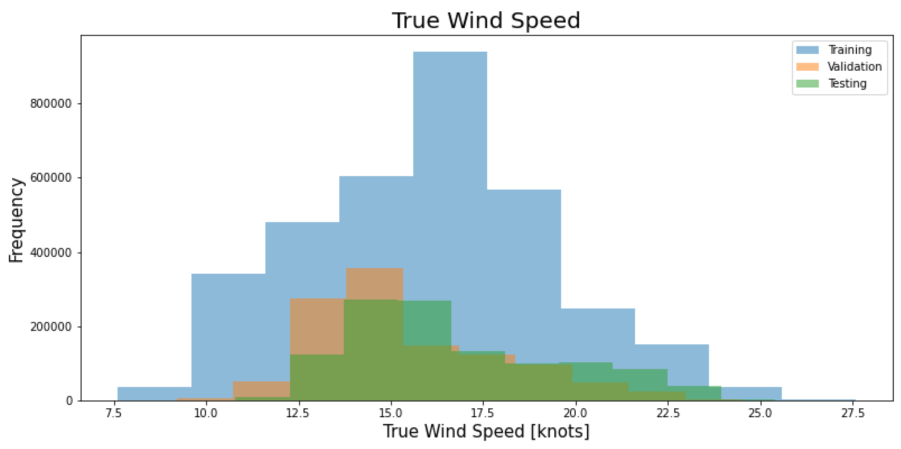


Figure 4.5: Distribution of True Wind Speed in **Unknown 1 (transat_1)**.

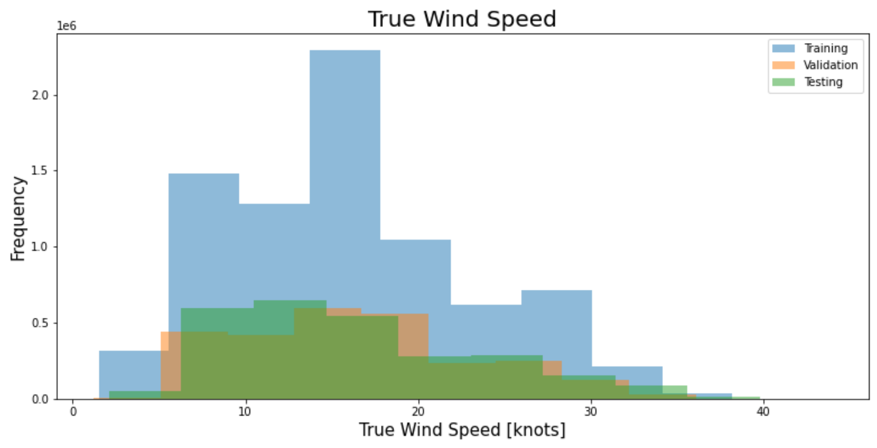


Figure 4.6: Distribution of True Wind Speed in **Concise 8 (Atlantic)**.

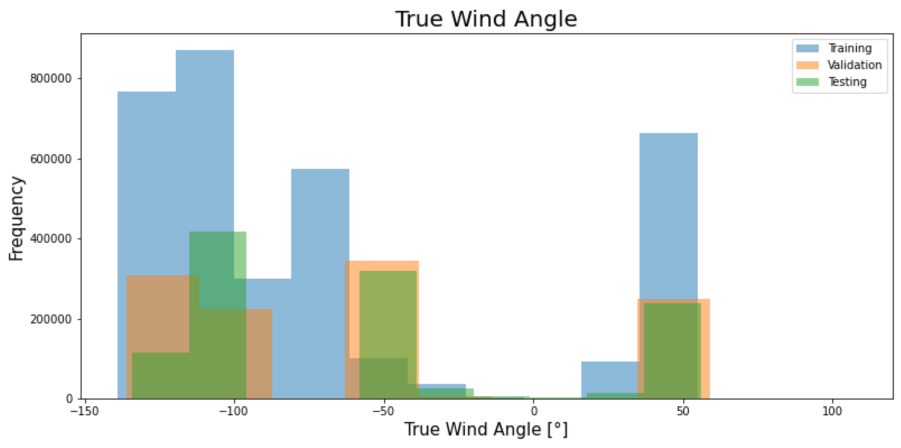


Figure 4.7: Distribution of True Wind Angle in **Unknown 1 (transat_1)**.

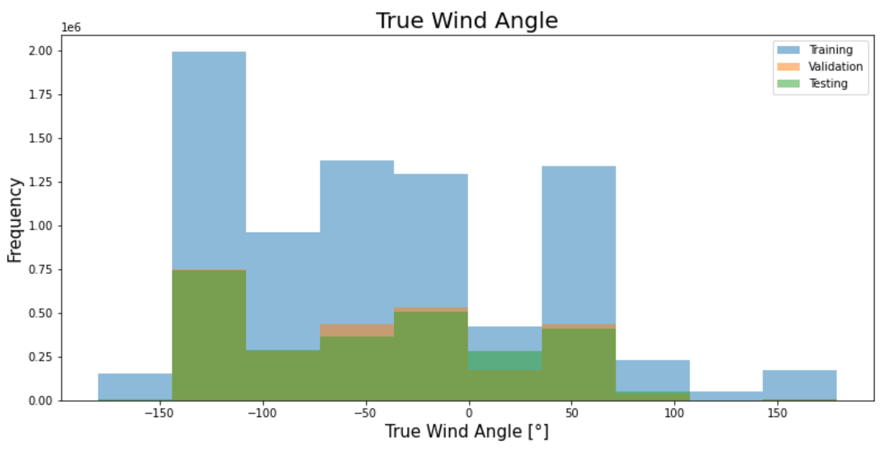


Figure 4.8: Distribution of True Wind Angle in **Concise 8 (Atlantic)**.

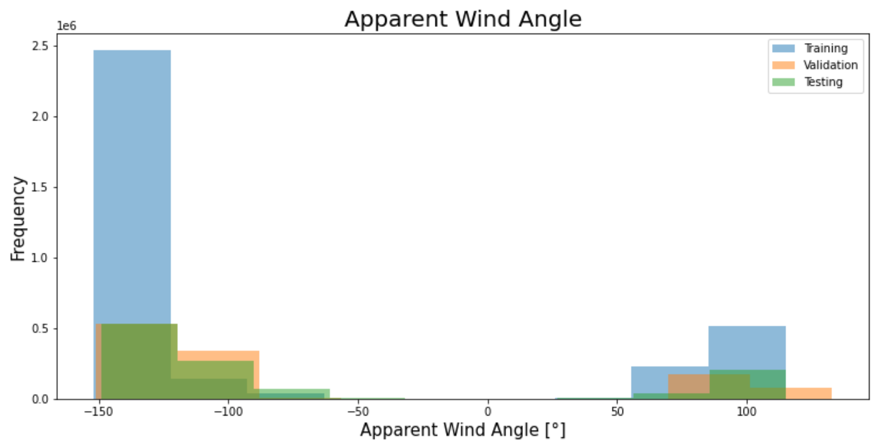


Figure 4.9: Distribution of Apparent Wind Angle in **Unknown 1 (transat_1)**.

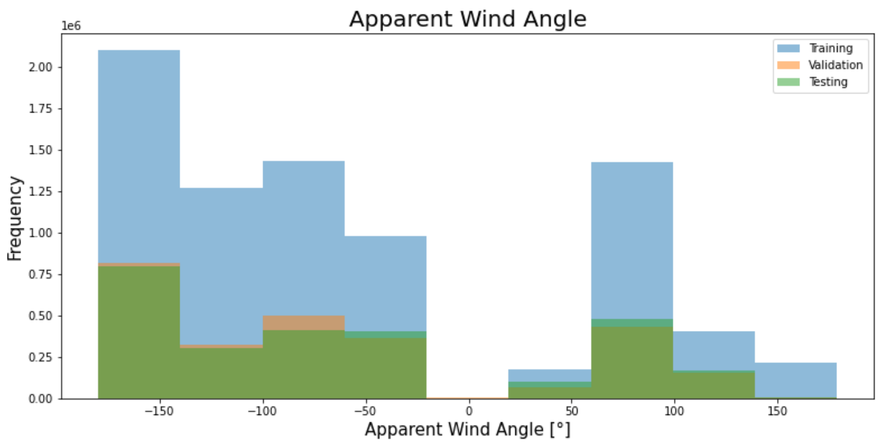


Figure 4.10: Distribution of Apparent Wind Angle in **Concise 8 (Atlantic)**.

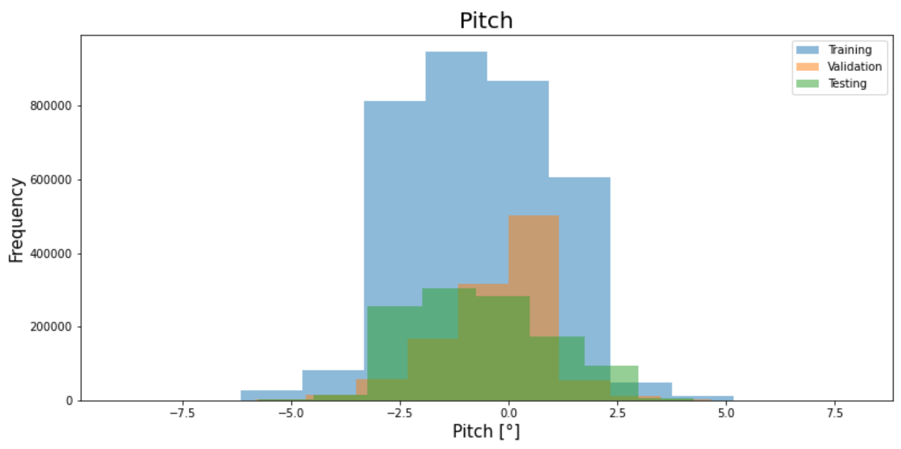


Figure 4.11: Distribution of Pitch in **Unknown 1 (transat_1)**.

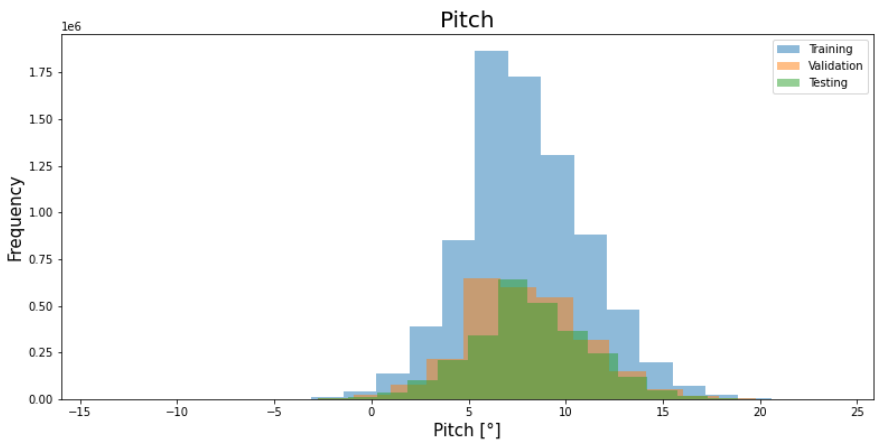


Figure 4.12: Distribution of Pitch in **Concise 8 (Atlantic)**.

4.5 Preprocessing

The data, cleaned and selected as described in the previous sections, must still be prepared before it can be fed into any models. This concerns firstly their content, i.e. mathematical transformations applied to the data, and secondly the form, i.e. re-arranging the data into a format that is compatible with the common ML-libraries.

4.5.1 Normalisation

Motivation Typically, the values of features are subject to rescaling before entering deep neural networks. This normalisation step allows to reduce the effects that the features' different value ranges (cf. table 2.4) have on the performance of the neural networks employed. Furthermore, if a single model is to be trained to predict multiple values, the error metric that it is trained to optimise consists of the average of the different feature's prediction errors. Hence, if the models were to be trained on unscaled data, the value of this overall error metric would be severely dependent on features whose value range can be relatively high (e.g. True Wind Direction), which would overtrump the effect of features that can only attain relatively low values (e.g. Current Speed).

Transformations Table 4.2 provides an overview of the preprocessing applied to the values of each feature, here can correspond to one of three transformations:

- As motivated and explained in detail in section 2.3.1 above, the **sine and cosine** of the angle features is computed if their range covers [-180°,180°] or [0°,360°], effectively reducing their range to [-1,1].
- Features that can only take on positive values (e.g speed) are subject to Min-Max normalisation, i.e. $x_{normalised} = \frac{x_{original} - x_{min}}{x_{max} - x_{min}}$. The normalised values hence lie in the range [0,1].
- Features whose values might be either positive or negative (e.g. rudder angle, whose values are contained in the range [-45°,45°]) are subject to Max-Abs normalisation, i.e. $x_{normalised} = \frac{x_{original}}{max(|x_{min}|,|x_{max}|)}$. This leads to the normalised values lying in the interval [-1,1].

4.5.2 Rearranging the Timeseries

As explained in detail in section 2.4.1, the timeseries data needs to be re-arranged into a specific format in order to enter the keras models. This step is conducted as the last pre-processing intervention. While it has been described extensively in the mentioned paragraph 2.4.1, a technical remark is opportune at this point.

Previous approach and disadvantages In previous iterations of JTR AI, the re-arranging of the data was done using a custom-made function (cf. [4], [5], [6]). This is error-prone and leads to the local runtime running out of memory for large datasets.

Updated approach and advantages Hence, in the present study this step was conducted using the "Timeseries Generator" class of TensorFlow. It relies on a generator that produces batches of timeseries data on which the models can be trained batch by batch. It has been specifically designed to transform timeseries data and to cope with large amounts of them. Apart from removing memory issues and error sources, the use of generators during model training allows for multi-processing, speeding up the training process. The interested reader finds further information in the online TensorFlow documentation [35]. The use of this class might be of special interest for future iterations of JTR AI.

Feature Name	Range	Preprocessing
Latitude	[-90, 90]	Max-Abs
Longitude	[-180, 180]	Cos and Sin
TWS	[0, 40.0]	Min-Max
TWD	[0, 360]	Cos and Sin
Current_speed	[0,15.0]	Cos and Sin
Current_direction	[0, 360]	Cos and Sin
Air_temp	[0, 30.0]	Min-Max
Speed_ov_surface	[0, 25.0]	Min-Max
Speed_ov_ground	[0, 25.0]	Min-Max
VMG	[0, 25.0]	Min-Max
Heading_True	[0, 360]	Cos and Sin
Heading_ov_ground	[0, 360]	Cos and Sin
Pitch	[-20, 20]	Max-Abs
Roll	[-60, 60]	Max-Abs
Yaw	[-180, 180]	Cos and Sin
AWS	[0, 50.0]	Min-Max
AWA	[-180, 180]	Cos and Sin
TWA	[-180, 180]	Cos and Sin
Rudder	[-30, 30]	Max-Abs

Table 4.2: Preprocessing applied to the features.

Chapter 5

Models

The aim of the present study is to develop reliable timeseries forecasting models for those features that define the state of the boat, which is a supervised learning task. These forecasting models constitute the simulation environment for an RL agent that is to learn to intelligently set the rudder angle. Hence, the only boat variable that is not to be predicted by any supervised model and for which no model is investigated here is that of the rudder angle, as this feature is to be learnt by an RL agent. Different forecasting models are conceivable to predict the boat state features. The present chapter presents the general structure and functioning of these models.

5.1 1 Model for n Features

Design A single model can be trained to predict the next values of all of the boat state's features. The single model receives as input the values of the n_0 time steps preceding the moment for which it is to make a prediction. The model returns a vector of all predicted feature's values. This model and the hyperparameters that govern it have been described and illustrated extensively in the background research above (2.4.1).

Advantages It is worth noting that the tuning of these hyperparameters allows to adapt the model's behaviour and hence to optimise its performance for a given task. As described in detail in the background research above (2.3.3), Bayesian optimisation allows to improve the hyperparameters. The main advantage of training 1 model for n features resides in this optimisation only being required for 1 model. Indeed, as the optimisation of hyperparameters can prove lengthy and tedious, conducting it for one model can be preferable over performing it for multiple models. This latter approach of training multiple models for timeseries forecasting problem at hand is described subsequently.

5.2 n Models for n Features

Design The boat state consists of multiple features, each of which needs to be predicted in timeseries prediction problem at hand. An alternative to training a single model to predict n features' values is to train n models separately. In this framework, each model is trained to specifically predict only one variable's next state; i.e. one model to predict the true wind speed, one model for apparent wind angle, etc. The general structure of these models and the hyperparameters that can be varied are identical as for the 1 model for n features, except that a single value is returned instead of a vector constituting the predicted boat state. Hence, each of the n models returns only one value. In turn, the n models together returning n values constitute the RL simulation environment.

Advantages One advantage of this approach resides in the fact that the n models only need to be optimised with respect to their respective features, instead of being optimised to predict all features at once. This reduces the complexity the models must take into account.

5.3 Deterministic models

A radically different approach can also be adopted to compute the values of features. Indeed, the navigation logs contain directly measured values as perceived by the sensors on board, e.g. Apparent Wind Angle (AWA), Apparent Wind Speed (AWS) etc. On the other hand, they contain values derived from these measured values, such as the True Wind Angle (TWA) and the True Wind Speed (TWS). This derivation is done directly on board by the software of the nke autopilot and stored as such in the navigation logs. Table 2.4 lists feature by feature which features are directly measured and which are derived.

5.3.1 Motivation

Hence, it is possible to derive the values of certain features directly from the values of other features. This can be interesting for a number of reasons:

1. **Alternative to forecasting models with poor performance:** if no accurate forecasting model can be identified for a given feature, it might be computed from a number of other predicted values that can be accurately predicted. That way, a much more accurate estimate can still be computed, despite there not being an accurate LSTM-based model. An example of this is the TWA, which is defined as a function of other features.

2. **Evaluating the accuracy of data:** If features are defined as direct functions of each other, the calculated values should in theory exactly match the recorded

values, if they are computed on the basis of recorded, ”true” values (instead of which one could also use predicted values, if one wants to make a prediction). In this sense, any discrepancies between the calculated and recorded values are an indicator of imperfections in the recorded values, because in the case of exactly recorded values both should be exactly the same. This is of importance as the sensors on board can potentially be poorly or mis-calibrated. Comparing the recorded vs. the ”true” values recorded by the sensors and calculating the corresponding error measures allows for a type of ground truth against which to assess how strong such mis-calibrations potentially are.

3. **Speed:** this approach is potentially faster than providing inputs to a trained LSTM-based model, performing the large number of calculations that that model consists of, and only then having an output. Indeed, if relying partly on first-principles formulae for the simulation environment, only some values would need to be predicted using the computationally heavier models, while the rest can be quickly determined using computationally light formulas. It should be noted that the speed of the predictions is not important in this study, since the goal is only to develop a reliable RL simulation environment. However, in other applications, the speed at which predictions can be made may play a role. This will be explained in chapter 8.

5.3.2 Formulae

The exact derivation of the first-principles formulae will not be described here, as this essentially corresponds to what has been described in the theses of Birk Ulstad [6], Roman Kastusik [4] and in various online resources, e.g. [36]. Here we will be content to merely state the formulae for each of these features.

Definitions TWS and TWA are by their very definitions unambiguous functions of other features:

- **True Wind Angle (TWA):**

$$TWA_{t+1,det} = arccos\left(\frac{AWS_{t+1} \cdot cos(AWA_{t+1}) - Speed\_ov\_ground_{t+1}}{TWS_{t+1}}\right) \tag{5.1}$$

- **True Wind Speed (TWS):**

$$\begin{aligned} TWS_{t+1,det} = \quad & \Big(AWS_{t+1}^2 + Speed\_ov\_ground_{t+1}^2 \\ & - 2 \cdot AWS_{t+1} \cdot Speed\_ov\_ground_{t+1} \cdot cos(TWA_{t+1})\Big)^{\frac{1}{2}} \end{aligned} \tag{5.2}$$

It should be noted that TWA is a boat state feature as defined in section 2.4.1 above. It can hence be used in light of the first motivation stated above, i.e. to assess whether it is theoretically possible to predict a given feature’s value by using other features. It can also be used with a view on the second motivation stated above, i.e. to assess the accuracy of a dataset by comparing the computed and the recorded values. On the other hand, as defined in section 2.4.1, TWS is a sea state feature and not a boat state feature. It can hence only be used to assess the accuracy of the dataset, as the present study is concerned with identifying forecasting models for boat state features only.

Derivations For the latitude and longitude features, another first-principles based method is used to derive the position of the boat in the next moment. Indeed, based on the boat's equations of motion, the position of the boat at the next instant $t + 1$ can be calculated from certain values available at the current instant t. While the formulae governing the relationships of TWA and TWS and the other features are ”absolute truths” that emanate from the definitions of these measures, the formulae of latitude and longitude being derived from equations of motion means that they are prone to slight deviations from the true recorded value. However, in the following, for the sake of conciseness both categories will be referred to as ”first-principles” or ”deterministic” formulae.

Definitions :

- **Latitude:**

$$\begin{aligned} Latitude_{t+1,det} = \quad & arcsin\Big(sin(Latitude_t) \cdot cos(Speed\_ov\_ground_t/R) \\ & + cos(Latitude_t) \cdot sin(Speed\_ov\_ground_t/R) \\ & \cdot cos(Heading\_ov\_ground_t)\Big) \end{aligned} \tag{5.3}$$

- **Longitude:**

$$\begin{aligned} Longitude_{t+1,det} = \quad & Longitude_t \\ & + arctan\Big(sin(Heading\_ov\_ground_t) \\ & \cdot sin(Speed\_ov\_ground_t/R) \cdot cos(Latitude_t) \\ & + cos(Speed\_ov\_ground_t)/R) \\ & - sin(Latitude_t) \cdot sin(Latitude_{t+1})\Big) \end{aligned} \tag{5.4}$$

where the abbreviations for features are used as defined in table 2.4, and det indicates that a value is computed deterministically.

5.4 Prediction Time Horizon

At this point it should be noted that the models, just like in Roman Kastusik's study, are to predict the boat variables in the next second. In other words, at a given moment t, a prediction is made of the boat state in the next instant $t + 1s$.

Choice of a time horizon Since the data was sampled at 25 Hz, i.e. every 0.04 seconds, it would also be conceivable to predict the boat variables for any $t+n\cdot 0.04s$, where n is a strictly positive integer. Nonetheless, the ultimate goal of JTR AI is to develop an algorithm that outputs an advantageous rudder angle that can be set by the autopilot hardware in the physical world. Now the live implementation of all of collecting input data, calculating an adequate rudder angle and physically setting that rudder angle on board takes time. Indeed, this was the reason for one of the major conclusions of a previous iteration of JTR AI [5], which consisted of the recommendation to focus on predictions for $t + 1s$ instead of $t + 0.20s$.

Limitations and implications However, as explained in the background research 2.3, $1s$ is also the most generous time horizon which should be used, as a human sailor is roughly estimated to change the rudder angle at least once per second. For these reasons, the prediction time horizon for the rudder prediction RL algorithm was retained at 1 Hz, i.e. 1 second. The RL simulation environment - i.e. the time-series forecasting models discussed in the following - was hence also designed for that time horizon, and the data needed to train, validate and test these models re-arranged to 1 Hz according to the procedure described in section 4.5.2.

5.5 Evaluation Metrics

Motivation The performance evaluation of the models is carried out using the mean absolute error and the root mean squared error. Indeed, they are two evaluation metrics commonly used in the machine learning community and allow to assess the performance of the models relatively quickly. While Roman Kastusik also relied on these two metrics to describe his model's performance, the results are not directly comparable: in the mentioned thesis, different data was used and a different preprocessing was carried out. In particular, in Roman Kastusik's work, data was not examined for tacks or abnormal segments.

Definitions Finally, these two performance metrics are defined as:

- Mean absolute error (MAE):
  $MAE = \frac{1}{n\_samples} \sum_{i=1}^{n\_samples} |y_{true,i} - y_{predicted,i}|$

- Root Mean Squared Error (RMSE):
  $RMSE = \frac{1}{n\_samples} \sum_{i=1}^{n\_samples} (y_{true,i} - y_{predicted,i})^2$

Now that there is clarity about both the available data and the models to be trained and tested, the next chapter will explain the experiments that can be conducted with them.

Chapter 6

Experiments

The overarching goal of this study consists in the identification of reliable forecasting models for the boat state features. This section presents the experiments conducted in view of that goal.

Overview The high-level approach adopted to resolving problem at hand can be subdivided into different steps, namely:

1. For a given boat's dataset coming in a specific format, investigate whether accurate forecasting models can be identified. In the present study, Concise 8 (DRHEAM 18) was retained for this step due to its being available early on in the project.

2. Test whether the hyperparameters that are adequate for that boat and that dataset's format is also adequate for

   - a dataset from the same boat, but in a different format, i.e. Concise 8 (Atlantic).
   - a dataset from a different boat and in a different format, i.e. Unknown 1 (transat_1).

In the following sections, this approach is sub-divided into more fine-grained steps.

Motivation At this point it is worth explaining again why models with exactly the same hyperparameters as found for DRHEAM 18 are trained on the new data. The aim is to find out whether the hyperparameters identified for DRHEAM 18 from the old data format are also adequate for new, .nkz-based data formats for the same boat (Concise 8) and for a different boat (Unknown 1). In fact, if the latter was the case, this would mean that in order to generate a boat simulation environment, all that is needed is a cleaned dataset that allows the model to learn different weights, but does this with the known hyperparameters. If this is not the case, it would mean that for each new boat the architecture and hyperparameters would have to be optimised, which is a much more complex and time-consuming process than training models with a once-and-for-all identified architecture along an established pipeline. This is of particular relevance as additional data from new boats is likely to be available in future iterations of JTR AI.

Note about environment in which the experiments were conducted A cloud infrastructure (Azure Machine Learning) was available for the experiments, with various different computing capacities (virtual machines) and correspondingly different computing speeds. Most of the calculations were performed on a private laptop with an Intel Core i9-9980HK processor. All computing times that are indicated in the following were recorded for the mentioned computer.

6.1 1 Model for n features

6.1.1 Motivation and Hypothesis

In a first approach, it was sought to improve the model that Roman Kastusik had developed during his work on JTR AI and that was found to be inaccurate (cf. section 2.4.1). This model consisted in 1 model for n features, consisting of 2 LSTM layers followed by 1 dense layer, as laid out in more detail in the background research 2.4. No optimisation of the hyperparameters was performed for this model. Hence, this first step is concerned with verifying

Hypothesis 1: the performance of 1 model for n features can be improved by optimising the model's hyperparameters.

6.1.2 Experiments

Hence, the first experiment to conduct consists in optimising the model's hyperparameters. As laid out in the introduction of this chapter and summarised below in table 6.2,

- this **optimisation** is performed for the Concise 8 (DRHEAM 18) dataset that was recorded in a format that differs from the new, .nkz-based datasets.
- the single model for n features is then **trained** on the Concise 8 (DRHEAM 18) training subset and tested on its testing subset.
- the performance of the model is **evaluated** by considering the resulting testing error metrics as well as samples of the predicted vs. the true values.

In the following sections, the approach for the optimisation of the hyperparameters is presented.

6.1.3 Observed time window

A manual hyperparameter optimisation was conducted with respect to the observed time, i.e. the number of time-steps whose values are fed into the LSTM layers, or in other words the number of time steps that the model is ”looking back”.

Choice of time window Indeed, after deliberations with the project’s supervisors, of whom Dr Eric Topham has extensive sailing experience, it was assumed that 60 seconds of input would most probably be enough to capture the dynamics at sea (wind, waves, boat dynamics), while the original 100 seconds would be unnecessarily much.

Advantages of the chosen time window This has the considerable advantage that the network’s weights only need to be trained for 60 inputs instead of 100, which in principle should entail a greatly reduced training time for the model to converge. This is of particular importance as long training times can constitute a major bottleneck for the progress of studies like the present one.

Potential for optimisation Naturally, the systematic and analytical optimisation of the input time window could be conducted. The literature presented above with e.g. [21] shows that this is an important area of interest currently. However, the goal of the present study consists of the identification of hyperparameters that define accurate forecasting models, as opposed to the analytical optimisation of the optimal input time window length. Hence, in a first approach, it was refrained from performing an analytical optimisation of this hyperparameter.

6.1.4 Search space

The hyperparameters to be optimised are listed below in table 6.1. Before enunciating any considerations as to how to optimise their values, it needs to be defined in what ranges to optimise these values.

Reasons for original model’s poor performance The main explanation for the original hyperparameters’ resulting in unsatisfactory model performance is expected to reside in its inability to capture the complex dynamics of the present problem. Indeed, if a single model is to predict n features for the highly dynamical system represented by a sailboat, it must be able to capture the complexity of that task.

Dimensions of search space Hence, a relatively large search space should be chosen, in which combinations of hyperparameters are possible that take this complexity into account. The search space that was adapted is hence characterised by generous dimensions, as can be seen in table 6.1. For instance, the batch size can vary from 64 to eight times that value. Again, this large search space results from the fact that the prediction problem is considered rather complex and that any optimisation method should be allowed to take this complexity into account.

Hyperparameter	**Minimum**	**Maximum**	**Original**
Batch size	64	512	60
Dropout rate	0.2	0.7	0.40
Learning rate exponent	-5	-1	-3
Number of epochs	20	150	30
Number of layers	1	6	2
Number of nodes	10	51	50
Observed time n_0 [s]	*N.A.*	*N.A.*	*100*

Table 6.1: Search space for the hyperparameters of 1 model for n features, optimised and original hyperparameters.

6.1.5 Bayesian Optimisation

Given the relatively large number of hyperparameters to be optimised, their optimisation by use of human experience as for the observed time window is out of the question. Hence, these hyperparameters were optimised by conducting a systematic optimisation; namely by performing a Bayesian optimisation. This method and its advantages are presented above in the background research 2.3.3. It shall be recalled that its benefits reside in

- being **computationally efficient**, especially for large search spaces. This makes it preferable over other optimisation schemes like grid search, which requires many more iterations of models being trained and validated on different combinations of hyperparameters, making it much more computationally costly.
- being able to **optimise continuous hyperparameters** (e.g. learning rate). Again, this stands in contrast to grid search, in which the models can only be trained and validated on discrete values.

Given this relatively large range of the search space, 25 steps of random exploration followed by 5 steps of Bayesian optimisation are conducted. This allows to effectively investigate a large proportion of the search space.

6.2 N models for n features: Model 1

6.2.1 Motivation and Hypothesis

As presented in detail in section 5.2, training n models separately to predict n features has the potential to yield better predictions. Indeed, each model only being required to learn the behaviour of ”its” feature is likely to yield more accurate than if a single model has to capture all features’ behaviour together, with accuracy potentially suffering from the complexity of the task. Hence, this second step is concerned with verifying

Hypothesis 2: training n separate models for n features yields more accurate predictions than training 1 model for n features.

6.2.2 Experiments

Logically, this step is performed with the Concise 8 (DRHEAM 18) dataset, enabling comparability with the single model for n features. As summarised below in table 6.2, this approach consists in

- **defining one set of hyperparameters** that the n models are going to have, named ”Model 1”.
- **training** n models with Model 1 hyperparameters for the n features.
- **evaluating** the performance of the models by computing the testing error metrics and comparing them with the results obtained with 1 model for n features. Furthermore, considering plots of samples of the predicted vs. true values allows to analyse the behaviour of the models and recognise poor behaviour (e.g. offset).

The choices for the Model 1 hyperparameters are explained in the following.

6.2.3 Model 1 hyperparameters

Motivation With unlimited time and computational resources available, the determination of adequate hyperparameters for Model 1 would consist in the systematic optimisation of the hyperparameters of each individual model. However, since these lengthy optimisations were out of the question in the present project, it was decided to train n models with the same hyperparameters in a first approach. If the hyperparameters entail unsatisfactory results for specific features, then these hyperparameters can still be optimised in a second iteration.

Choice of hyperparameters In a first approach, the same hyperparameters as optimised for the single model for n features were retained, with one exception. Indeed, these hyperparameters are optimised for the same dynamics of the same boat but for the prediction of multiple features. They can hence be taken as a valid starting point.

Additional dense layer The Model 1 hyperparameters contain only one difference to the set of hyperparameters that was optimised for 1 model for n features. Indeed, given the complexity of the present problem, one can argue that there only being one final dense layer does not take into account the complexity of the highly dynamical and nonlinear problem at hand. However, in a deep learning model, dense layers with relu activation functions increase a model's ability to be trained on nonlinearities (cf. [34]). Hence, an additional dense layer with a relu activation function was integrated before the final dense layer for Model 1.

6.3 N models for n features: Model 2

6.3.1 Motivation and Hypothesis

Model 1 corresponds to one a standard set of hyperparameters for all n features. However, for some features, the additional dense layer of Model 1 can also be detrimental to the model's performance. Indeed, for those features which are not characterised by nonlinearities to the same degree as other features, it may be counterproductive to increase the model's ability to capture nonlinearities (cf. [34]). This leads to the following

Hypothesis 3: if Model 1 hyperparameters lead to unsatisfactory results for a feature, removing one dense layer improves the prediction accuracy.

The set of hyperparameters identical to Model 1, but with only one final dense layer is dubbed "Model 2" in the following.

6.3.2 Experiments

Again, this step is performed with the Concise 8 (DRHEAM 18) dataset, enabling comparability with the single model for n features. As summarised below in table 6.2, this approach consists in

- **defining one set of hyperparameters** that the models are going to have, named "Model 2".
- **training** the models for the selection of features.
- **evaluating** the performance of the models by computing the testing error metrics and comparing them with the results obtained in the previous steps. Furthermore, considering plots of samples of the predicted vs. true values allows to analyse the behaviour of the models and recognise trends in the behaviour.

Finally, while Model 2 is also a classical LSTM-based model, a radically different approach can also be taken to computing the values of certain features. This is explained in the next section.

6.4 Deterministic models

6.4.1 Motivation and hypotheses

As explained in detail above in section 5.3, deterministic models potentially present a number of advantages. Importantly, they might allow deriving some features' values directly from other features. These models are either a type of absolute truths (in the case of formal definitions) or derived from equations of motion. They are hence expected to result in better performance than LSTM-based models, which essentially have to learn the behaviour encapsulated in these formulae. This leads to the following

Hypothesis 4: Where they can be used, deterministic models achieve better performance than LSTM-based models.

However, as laid out in detail in the mentioned section 5.3, the datasets themselves might contain imperfections resulting from e.g. sensor mis-calibration. This leads to the following

Hypothesis 5: Where they can be used, deterministic models serve as an indicator of a dataset's inaccuracies.

The viability of deterministic models is to be tested both for the old data format of Concise 8 (DRHEAM 18), as well as for the new, .nkz-based data from Concise 8 (Atlantic) as well as Unknown 1 (transat_1). Indeed, in view of the first hypothesis, this allows to assess whether these models can be applied to both data formats. Moreover, in view of the latter hypothesis, this enables to compare the quality of the different datasets in function of their formats.

6.4.2 Experiments

The approach is summarised below in table 6.2; it consists in

- **computing** the values of TWA and TWS at moment t from the true values of other features at moment t according to the deterministic formulae. Indeed, if perfectly accurate forecasting models were to be identified for the other features and TWA and TWS were to be computed from them, this constitutes the benchmark performance that the prediction *via* direct derivation could achieve. Moreover, it allows to assess the quality of the data at hand, as the computed TWA and TWS should in theory exactly correspond to the true values.
- **computing** the values of Latitude and Longitude at moment $t+1$ from the true values at moment t. This allows to assess whether the first-principles model is a viable alternative to the LSTM-based models.
- **evaluating** the performance of the models by computing the testing error metrics and comparing them with the results obtained in the previous steps (ob-

viously only for Concise 8 (DRHEAM 18)). Moreover, the consideration of these error metrics for TWA and TWS allows to assess the quality of the data at hand. Furthermore, considering plots of samples of the predicted vs. true values allows to analyse the behaviour of the models and recognise trends in the behaviour.

6.5 Transferability of model hyperparameters between boats and datasets

6.5.1 Motivation and hypotheses

The overarching goal of the present study consists in the identification of reliable forecasting models of the boat state features. The experiments from the previous steps are designed to do so for one specific boat (Concise 8) and one specific dataset of a specific format (DRHEAM 18). As explained in detail in the introduction of this chapter, once these models have been identified for this specific combination of boat and dataset, it is of crucial interest to see whether the hyperparameters of these models are transferable. This leads to the following

Hypothesis 6: Hyperparameters of forecasting models that are accurate for Concise 8 (DRHEAM 18), i.e. the old data format, can be used to train accurate forecasting models using data from the same boat, but in the new, .nkz-based format (Concise 8 (Atlantic)).

This allows to evaluate the transferability of hyperparameters between different dataset formats. In this light, another test can be made, namely

Hypothesis 7: Hyperparameters of forecasting models that are accurate for Concise 8 (DRHEAM 18), i.e. the old data format, can be used to train accurate forecasting models using data from a different boat and in the new, .nkz-based format (Unknown 1 (transat_1)).

This allows to evaluate the transferability of hyperparameters between different boats with different dataset formats.

6.5.2 Experiments

The verification of these hypotheses can be made by conducting the following experimental steps, as summarised in table 6.2:

- For each boat state feature, identify the best-performing model for Concise 8 (DRHEAM 18).
- For Concise 8 (Atlantic) and Unknown 1 (transat_1), respectively:
  - **train** n models for n features using the most accurate model's hyperparameters (from DRHEAM 18).
  - **evaluate** the performance of the models by computing the testing error metrics and comparing them with the results obtained in the previous steps. Furthermore, considering plots of samples of the predicted vs. true values allows to analyse the behaviour of the models and recognise trends in the behaviour.

Experiment 1: Optimise, train and test 1 model for n features
For the Concise 8 (DRHEAM 18) dataset:
- Define search space in which hyperparameters of 1 model for n features should be optimised
- Perform Bayesian optimisation of hyperparameters
- Compare optimised hyperparamaters to hyperparameters of original 1 model for n features
- Train 1 model for n features with optimised hyperparameters
- Test the trained model, compare with results obtained with original model
Experiment 2: Train and test n separate models for n features
For the Concise 8 (DRHEAM 18) dataset:
- Define 1 specific set of hyperparameters for the n models to be trained (Model 1)
- Train n separate models (all with Model 1 hyperparameters) for all of the n features
- Evaluate for which features Model 1 is not satisfactory
Experiment 3: Tune, train and test selection of separate models
For the Concise 8 (DRHEAM 18) dataset:
- Define 1 specific set of hyperparameters for those models for which Model 1 hyperparameters were not satisfactory (Model 2)
- Train separate models (all with Model 2 hyperparameters) for that selection of features
- Evaluate for which features Model 2 is not satisfactory
Experiment 4: Test deterministic models
For the Concise 8 (DRHEAM 18),Concise 8 (Atlantic) and Unknown 1 (transat_1) datasets, respectively:
- For features for which deterministic models apply: compute values as function of other features
- Compare true and computed values to assess performance of deterministic models as well as accuracy of dataset
Experiment 5: Train and test n models for n features on new data
For the Concise 8 (Atlantic) and Unknown 1 (transat_1) datasets, respectively:
- For each feature:
- If no deterministic model can be used, train model with hyperparameters that are most adequate according to DRHEAM 18 results

Table 6.2: Overview of the experimental approach.

Chapter 7

Results and Discussion

In the following, the results of the experiments are presented and discussed.

7.1 1 Model for n Features

7.1.1 Results

The following sections present the results of the optimisation, training and testing of 1 model for n features. They correspond to the experimental steps laid out in section 6.1 and summarised under Experiment 1 in table 6.2.

Bayesian Optimisation

The Bayesian optimisation required a total runtime of 6.5 days. Table 7.1 presents the Bayesian-optimised hyperparameters vs. the original hyperparameters. The optimisation results in the same number of LSTM layers as the original model, namely 2. All other hyperparameters change with respect to the original set of values.

Hyperparameter	Optimised	Original
Batch size	88	60
Dropout rate	0.387	0.40
Learning rate exponent	-3.21	-3
Number of epochs	146	30
Number of LSTM layers	2	2
Number of LSTM nodes	35	50
Overall MAE	0.048	0.062
Overall RMSE	0.085	0.101

Table 7.1: Optimised and original hyperparameters of 1 model for n features. Overall MAE and RMSE are with respect to the normalised feature values. Investigations conducted with **Concise 8 (DRHEAM 18)**.

Training and Testing

Using the optimised hyperparameters as presented in the previous section, 1 model for n features was trained on the DRHEAM 18 training and validation subsets, which took a total of 12 hours. As can be seen in table 7.1, the 1 model for n features with optimised hyperparameters leads to an improved performance in comparison to the original model trained by Roman Kastusik: the overall MAE of the normalised values of the features is reduced from 0.062 to 0.048, and the corresponding RMSE is reduced from 0.101 to 0.085.

7.1.2 Discussion

It is found that the performance of the original model can be improved by conducting a hyperparameter optimisation. Thus, the hypothesis laid out in section 6.1 is confirmed.

Limitations of error measures However, the reader should be aware that as mentioned in section 5.5, the performance metrics obtained here cannot be directly compared to the ones obtained for the original model, as the latter relies on different training, validation and testing subsets. Hence, when performing comparisons with this previous iteration of JTR AI, the values should only be considered as approximate indicators of an overall improvement.

Bayesian optimisation It can be noted that the optimised hyperparameters result in the same number of LSTM layers as the original architecture had. However, the number of LSTM nodes per layer is optimised from 50 to 35. This suggests that the model only needs to learn a reduced number of LSTM nodes and corresponding weights to capture the temporal dynamics of the boat (cf. [37] for further information on LSTM's abilities to capture temporal patterns). This implies that the original model exceeded the number of necessary nodes. Moreover, it is noted that the optimised number of epochs increases nearly five-fold from 30 to 146, and that the optimised batch size increases from 60 to 88. This indicates that the model needs to be trained longer and with more data as was the case with the original model. Hence, the original model was designed to capture an unnecessary level of temporal complexity in its 50 LSTM nodes, while not allowing for enough training in terms of both epochs and batch size.

Limitations of trained model Moreover, the achieved optimisation does not suffice to train a reliable forecasting model. Indeed, figs. 7.1 to 7.4 show an excerpt of the course of the true values vs. the course of the predicted values for a selection of features. It can be noted that in spite of the overall MAE and RMSE improving with the optimised architecture, the predictions made by the forecasting model do not accurately follow the true values. Indeed, if one considers the mentioned figures, one can note that the 1 model for n values generally captures the up and down movements of the features' patterns. This pattern can be found similarly for all features; the presented selection of features and of the time window serves only as an example.

Explanations for observed performance Furthermore, it can be noted that the following of the true values does not happen on the correct scale, as well as with a strong offset. This suggests that the part of the model that allows to learn the changes in time of the values, i.e. the LSTM layers, is able to capture the movements in time of the values. Hence, the deficiency of the model is likely to reside in the part of the model that transforms the LSTM's output into the predicted value, i.e. the final dense layer with tanh activation function (cf. fig. 2.8 for the setup of the forecasting model).

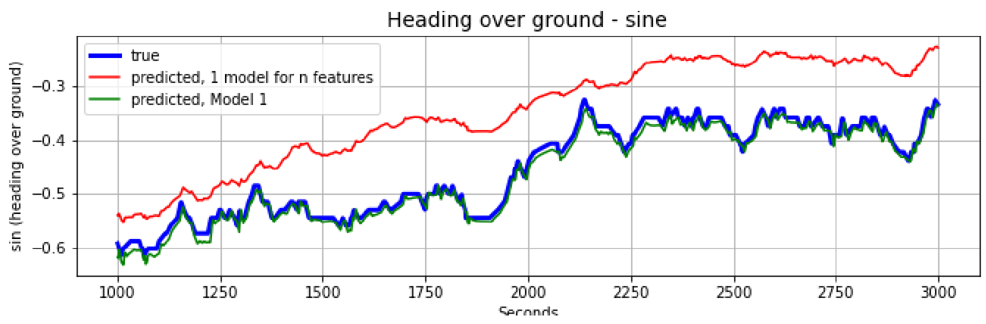


Figure 7.1: Predictions for the sine of heading over ground (testing subset of **DRHEAM 18, Concise 8**).

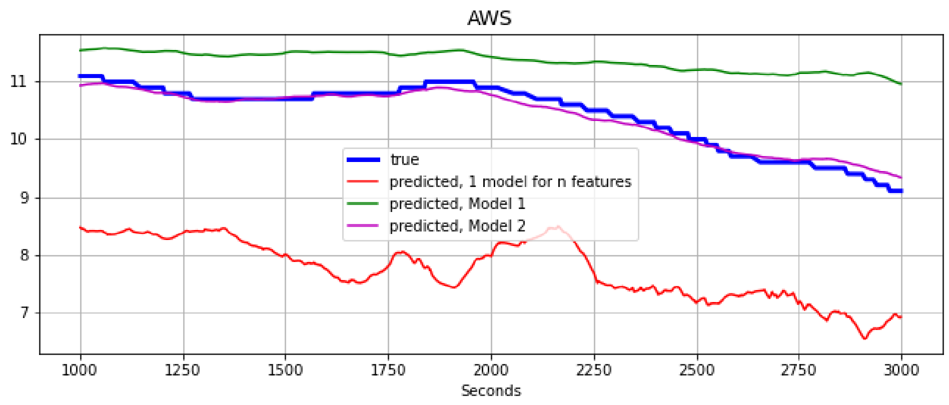


Figure 7.2: Predictions for apparent wind speed (testing subset of **DRHEAM 18, Concise 8**).

Feature Name	Model	MAE	RMSE
Longitude - cos	1 Model for n fatures	0.002	0.002
	Model 1	0.002	0.002
	Deterministic	$\mathbf{5.783 \cdot 10^{-7}}$	$\mathbf{5.445 \cdot 10^{-7}}$
Longitude - sin	1 Model for n fatures	0.029	0.035
	Model 1	0.027	0.032
	Deterministic	$\mathbf{5.233 \cdot 10^{-7}}$	$\mathbf{6.200 \cdot 10^{-7}}$
Speed_ov_surface	1 Model for n fatures	1.510	2.389
	Model 1	**1.200**	**1.846**
Speed_ov_ground	1 Model for n fatures	1.530	1.938
	Model 1	**0.820**	**1.154**
VMG	1 Model for n fatures	1.230	1.938
	Model 1	**0.520**	**0.866**
Heading_True - cos	1 Model for n fatures	0.042	0.072
	Model 1	**0.017**	**0.045**
Heading_True - sin	1 Model for n fatures	0.029	0.054
	Model 1	**0.017**	**0.038**
Heading_ov_ground - cos	1 Model for n fatures	0.065	0.112
	Model 1	**0.037**	**0.076**
Heading_ov_ground - sin	1 Model for n fatures	0.029	0.076
	Model 1	**0.020**	**0.044**
Yaw - cos	1 Model for n fatures	0.017	0.061
	Model 1	**0.007**	**0.051**
Yaw - sin	1 Model for n fatures	0.073	0.097
	Model 1	**0.040**	**0.053**
TWA - cos	1 Model for n fatures	0.065	0.091
	Model 1	**0.026**	**0.036**
	Deterministic	0.746	0.938
TWA - sin	1 Model for n fatures	0.078	0.116
	Model 1	**0.021**	**0.034**
	Deterministic	0.745	0.938

Table 7.2: Test performance for the **DRHEAM 18 dataset (Concise 8)**, part 1.

Feature Name	Model	MAE	RMSE
AWA - cos	1 Model for n fatures	0.106	0.128
	Model 1	**0.028**	**0.037**
AWA - sin	1 Model for n fatures	0.088	0.146
	Model 1	**0.021**	**0.037**
Latitude	1 Model for n fatures	0.755	0.883
	Model 1	0.835	0.955
	Model 2	0.466	0.574
	Deterministic	$\mathbf{1.643 \cdot 10^{-5}}$	$2.121 \cdot 10^{-5}$
Pitch	1 Model for n fatures	0.720	**0.950**
	Model 1	0.770	1.006
	Model 2	**0.641**	1.058
Roll	1 Model for n fatures	2.940	4.047
	Model 1	3.750	4.951
	Model 2	**1.390**	**2.002**
AWS	1 Model for n fatures	2.340	3.136
	Model 1	2.310	2.905
	Model 2	**1.220**	**1.748**

Table 7.3: Test performance for the **DRHEAM 18 dataset (Concise 8)**, part 2.

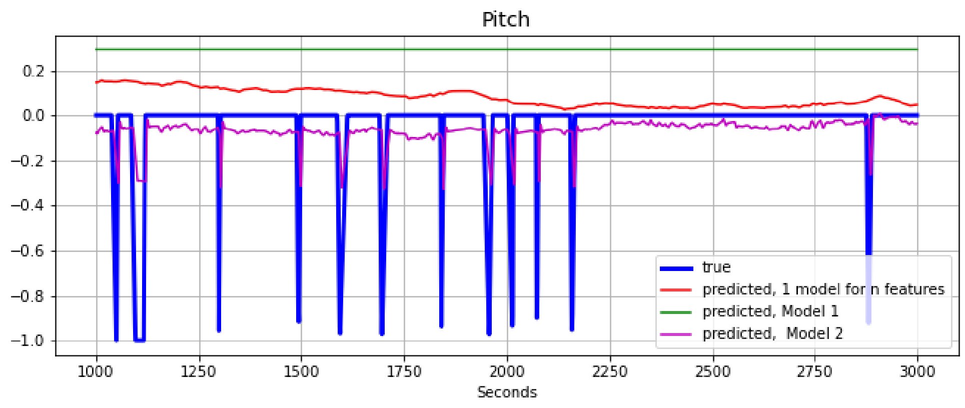


Figure 7.3: Predictions for pitch (testing subset of **DRHEAM 18, Concise 8**).

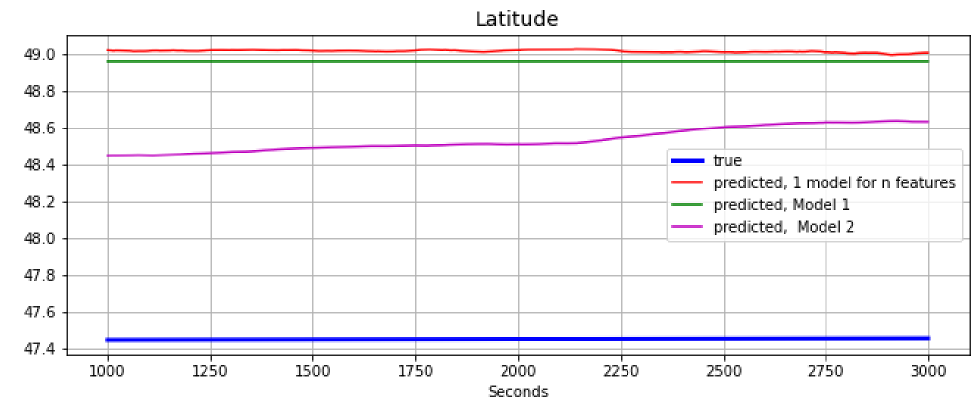


Figure 7.4: Predictions for latitude (testing subset of **DRHEAM 18, Concise 8**).

7.2 N Models for n Features: Model 1

The following sections present and discuss the results of the training and testing of n separate models for n features. They correspond to the experimental steps laid out in section 6.2 and summarised under Experiment 2 in table 6.2.

7.2.1 Results

N models were trained for n features using the optimised hyperparameters and architectures with one additional dense layer as described in section 6.2 and as summarised in table 7.4. This took ca. 10.5 hours per feature, resulting in a total training time of 19 features $\cdot 10.5 \frac{h}{feature} = 199.5h$ for the training of all models.

Error metrics The resulting values for MAE and RMSE are listed in tables 7.2 and 7.3. They are visualised in figs. 7.5 and 7.6. When comparing the prediction accuracy obtained with Model 1 vs. the accuracy obtained with 1 Model for n features, the average change of the features' MAE is -34.25 %, while the average change for RMSE is -32.57 %. It can be seen that both the MAE and the RMSE improve for all features, the only exceptions are

- Latitude
- Pitch
- Roll

Furthermore, only a marginal improvement can be reached for the MAE of AWS (reduction of 1.28%).

7.2.2 Discussion

Accuracy of predictions Except for the 4 listed features, the improved error metrics suggest a greatly improved performance of the n models for n features in comparison to the 1 model for n features. Indeed, this is substantiated when considering fig. 7.1. In the figure, it can be observed how the original and the predicted values strongly agree for the example of the feature ”Heading over Ground”. This level of performance is also observed for all other features not listed above, i.e. satisfactory forecasting models can be achieved for these features. For the features listed above, fig. 7.2 to 7.4 show by way of example that the performance of the Model 2 hyperparameters is by no means satisfactory. Indeed, the predicted values hardly follow the tendencies of the true ones, i.e. they do not even capture the up and down movements to a satisfactory degree.

Explanations for observed performance It can be observed that except for four features, the hyperparameters of Model 1 allow to capture the temporal patterns of the features. They do so without the offset observed for 1 model for n features, as well as to scale. This is attributed to the fact that the n models can each tune their weights to the behaviour of a single feature instead of being trained on predicting n features simultaneously. Furthermore, this is attributed to the additional dense layer being able to capture more nonlinear behaviour and the non-temporal aspects of the model (cf. [34]), which was observed to be a problem in the evaluation of the previous experiment. A more detailed description of the effects of an additional dense layer is laid out in 6.2.

Expressiveness of results It should be noted that with the present experimental setup, it cannot be quantified separately what the individual contributions are of the additional dense layer as well as of the separate training of n models. As explained in section 6.2 and as is clear from the nearly 200 hours required to train the models, this was not a priority in light of the temporal and computational resources of the present study. Moreover, the overall aim of the present study consists in the identification of reliable state estimators rather than in the systematic study of incremental changes to the hyperparameters. Hence, while one can note that the training of n separate models with one additional dense layer leads to greatly improved forecasting models, the separate contributions of these two changes cannot be quantified here.

In summary, the training of n separate models for n features yields more accurate predictions than the single model for n features. The hypothesis laid out in section 6.2 is thus confirmed. However, it is noted that for a reduced number of features, the separate models do not reach a satisfactory performance.

Hyperparameter	Model 1	Model 2
Number of LSTM layers	2	2
Batch size	88	88
Dropout rate	0.387	0.387
Learning rate exponent	-3.21	-3.21
Number of epochs	146	146
Number of Dense layers	2	1
Number of nodes	35	35
Activation function	tanh	tanh

Table 7.4: Hyperparameters of model 1 and model 2.

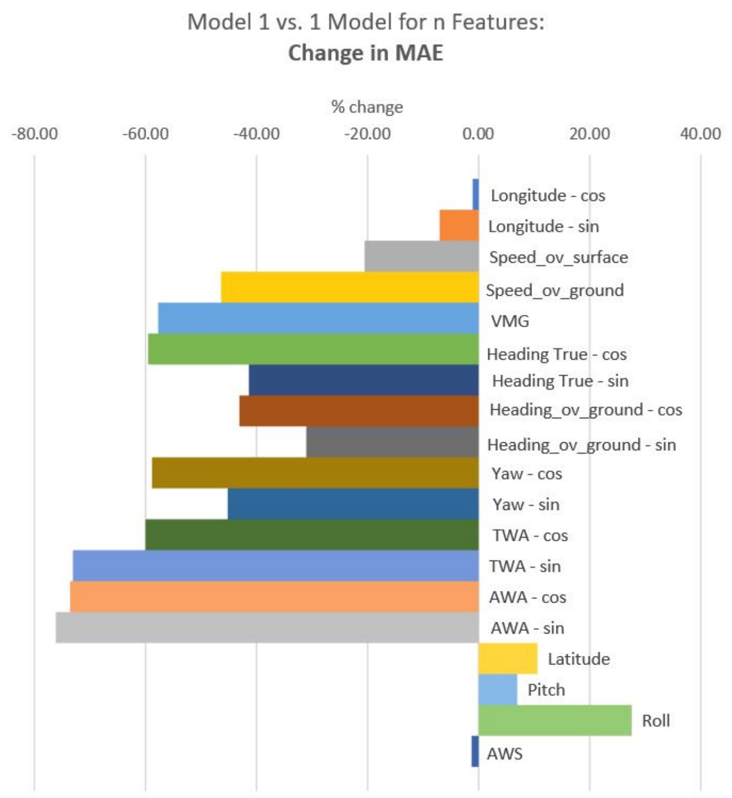


Figure 7.5: Change in MAE of Model 1 vs. 1 Model for n features (testing subset of **DRHEAM 18, Concise 8**).

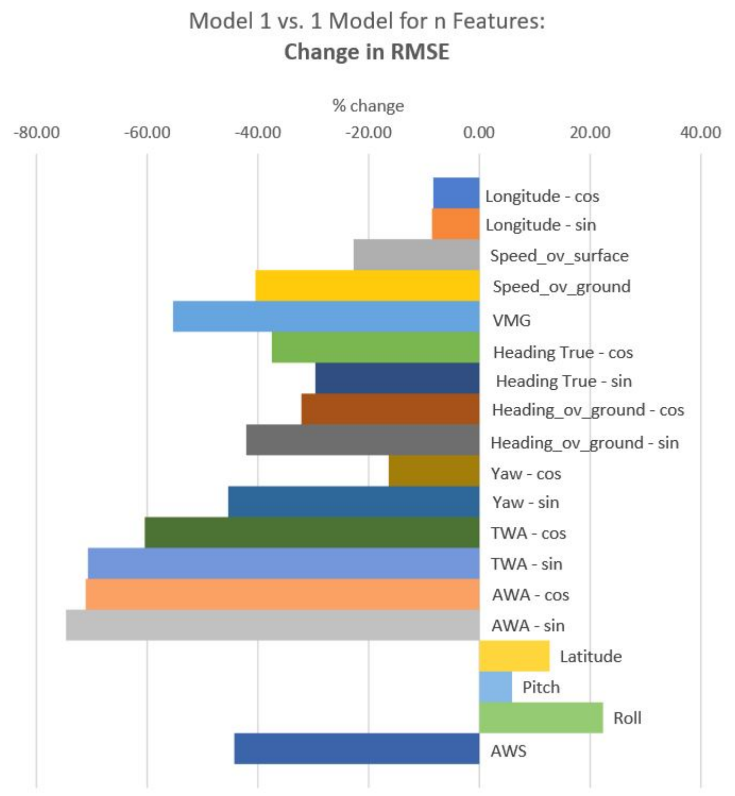


Figure 7.6: Change in RMSE of Model 1 vs. 1 Model for n features (testing subset of **DRHEAM 18, Concise 8**).

7.3 N Models for n Features: Model 2

The following sections present and discuss the results of the tuning, training and testing of separate models for a selection of features. They correspond to the experimental steps laid out in section 6.3 and summarised under Experiment 3 in table 6.2.

7.3.1 Results

Model 2, which differs from Model 1 only in that it has one instead of two final dense layers (cf. section 6.3), was trained on those features for which Model 1 has proven to be insufficient. Its set of hyperparameters is summarised in table 7.4. The training took ca. 9.5 hours per feature.

Error metrics As can be observed in table 7.3 and in figs. 7.7 and 7.8, the MAE and the RMSE decrease with Model 2 for all of the 4 concerned features. The only except is the RMSE for Pitch, which is more elevated than for 1 model for n features.

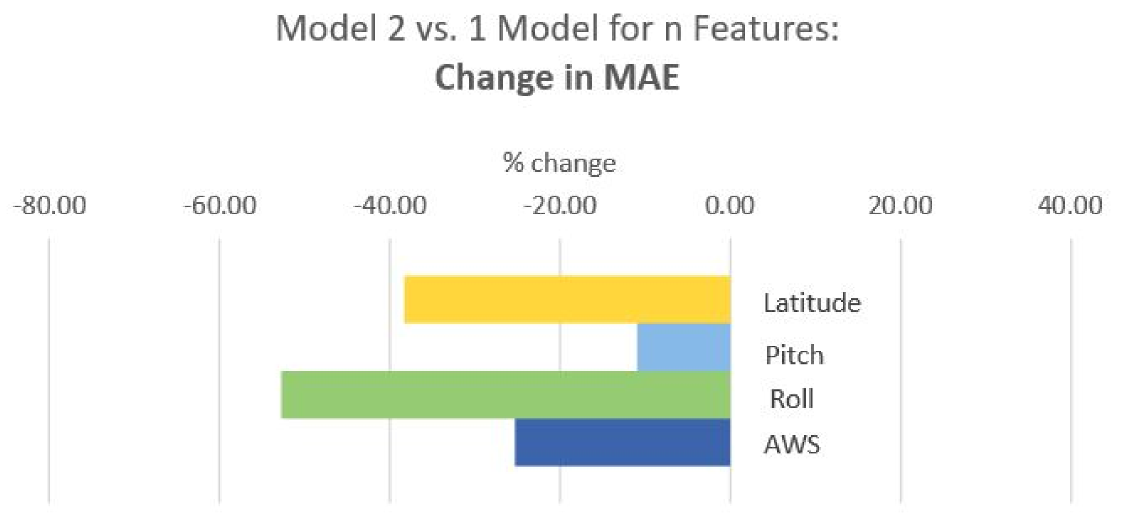


Figure 7.7: Change in RMSE of Model 2 vs. 1 Model for n features (testing subset of **DRHEAM 18, Concise 8**).

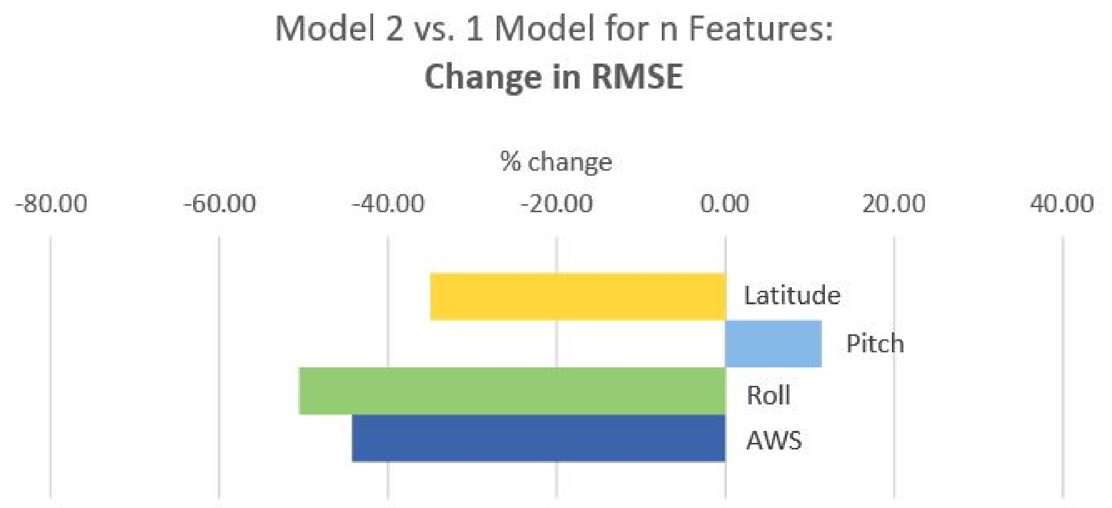


Figure 7.8: Change in RMSE of Model 2 vs. 1 Model for n features (testing subset of **DRHEAM 18, Concise 8**).

7.3.2 Discussion

Training time The training time per model is inferior to that observed for Model 1 (9.5 hours vs. 10.5 hours). This is coherent with the fact that fewer weights need to be trained without the 35 dense nodes from the removed dense layer.

Performance of models The performance of Model 2 for AWS and Roll is satisfactory, as emanates from the decreasing error metrics (cf. fig. 7.7 and 7.8) as well as from fig. 7.2, presenting an excerpt for AWS. Removing one dense layer hence improved the performance of the forecasting models with respect to Model 1. This is attributed to the fact that the additional dense layer with relu activation function of Model 1 leads to the model training for an excessive level of nonlinearity (cf. [34]). This effect is reduced by reducing the number of dense layers to one. Moreover, for Pitch and Latitude, the picture is tainted: from fig. 7.3 it is obvious that the trained model is able to predict the course of Pitch, but does not do so to scale. Moreover, considering the discrepancy between true and predicted values in fig. 7.4, it is is evident that the hyperparameters of model 2 do not deliver a reliable prediction model.

Observations for Pitch For Pitch, strong doubts can be expressed as to whether the pitch recorded in the DRHEAM 18 dataset can be considered accurate. Indeed, as can be seen in fig. 7.3, it remains at 0° to change only rarely and abruptly. This corresponds to a boat that would pitch hardly at all and when it does, very strongly at once. However, the inclination of a boat changes constantly when sailed, even if perhaps only slightly. As will be seen in the following sections, this stands in opposition to the new, .nkz-based data, where Pitch is subject to strong changes (cf. fig. 7.21). The latter hence reflect the real conditions much more accurately. For these reasons, no further model for pitch was investigated in a first phase until experiments with the new, .nkz-based data.

Observations for Latitude Considering fig. 7.4, it can be stated that none of the trained and tested models is satisfactory to predict Latitude. The predictions of the models deviate strongly from the true values, not only on the shown excerpt, but also concerning MAE and RMSE. While the detailed results can be found in tables 7.2 and 7.3, one can note that e.g. the MAE of 0.466 degrees latitude corresponds in the more frequently used units to 51.73 kilometers, which is obviously not satisfactory. However, in the first phase of experimentation, we refrained from investigating alternative models to achieve better predictions of latitude. Indeed, in an attempt to optimise time and computation resources, it was decided to first consider the performance of deterministic models for Latitude before entering any hyperparameter optimisations for the Latitude model.

In summary, for those features where Model 1 hyperparameters lead to unsatisfactory results, removing one dense layer improves the prediction accuracy. The hypothesis laid out above in section 6.3 can hence be confirmed. However, for Pitch and Latitude, the improved accuracy is not at a satisfactory level.

7.4 Deterministic Models

The following sections present and discuss the results of the testing of deterministic models for a selection of features. They correspond to the experimental steps laid out in section 5.3 and summarised under Experiment 4 in table 6.2.

At this point, it should be emphasised that the following sections are presented with a rare shift from a boat-centric point of view to a dataset-centric perspective. Indeed, as laid out in 5.3, first-principles models can be used to assess the accuracy of a dataset. Such an evaluation is hence inherently focused on datasets, and not on boats.

7.4.1 Results

Error metrics The resulting error metrics are presented in tables 7.2 and 7.3 for DRHEAM 18, table 7.6 for Atlantic and table 7.7 for transat_1. For DRHEAM 18, it can be observed that the predictions made for both Latitude and Longitude present over 99.9 % percent lower MAE and RMSE than obtained with the best LSTM-based models. For Atlantic and transat_1, the error metrics are of a comparable low order of magnitude. For the computation of TWS from other features' true values, the error metrics are in the same range: DRHEAM 18 presents an MAE of 0.461 knots, Atlantic an MAE of 0.410 knots and transat_1 an MAE of 0.230 knots. Finally, for TWA (split into cos and sin), both error metrics are ca. 30 times higher for DRHEAM 18 than recorded for the best-performing LSTM model. For Atlantic and transat_1, the MAE and RMSE are in a comparable range as for DRHEAM 18.

7.4.2 Discussion

Performance for TWS The results presented above indicate that for both data formats, the recordings for TWS are in strong accordance with the unambiguous formulae by which the different features are defined. Plots 7.9 to 7.11 illustrate the ability of the deterministic model to closely follow the true values of TWS. However, as the MAE and the RMSE are not exactly zero as they should by definition, it should be noted that the datasets cannot be considered as absolute ground truths. This is attributed to the sensors on board being mis-calibrated as well as to sensor drift that might occur in the rather harsh environment of the sea.

Performance for TWA As indicated by the error metrics TWA, the recordings are hardly in accordance with the deterministic formulae for TWA. Fig. 7.12 to 7.14 display this for an illustrative excerpt. In other words, it is found that the TWA data cannot be considered as absolute truth.

Performance for Latitude and Longitude Finally, for Latitude and Longitude, it was observed that the error metrics can be substantially reduced with respect to all other models for DRHEAM 18. Furthermore, it has been noted that they remain in a comparable order of magnitude for Atlantic and transat_1. Hence, the first-principles formulae derived from the equations of motion provide a reliable forecasting model for these features.

Illustration of Latitude and Longitude predictions Fig. 7.15 and 7.16 illustrate these results on a purely illustrative basis, i.e. without constituting any reference by which to judge the models' performance. For a given starting point, the following positions are calculated by using each time

- the previous latitude and longitude, computed from the first-principles formulae
- values of the other features as required by the first-principles formula (true values as recorded in the dataset)

The resulting trajectories follow the originally recorded path relatively closely for the given, purely illustrative excerpt of 5 minutes of data. Even though at first glance the calculated and recorded end positions seem to differ considerably for DRHEAM 18, the deviation between the coordinates of the two final points for the given example corresponds to 13.35 m over a total true . For Atlantic (final deviation of 2.17) and transat_1 (final deviation of 3.83 m), the performance for these purely illustrative excerpts is comparable. Fig. 7.16 illustrates this behaviour for transat_1.

Hence, in summary, considering the True Wind Angle and DRHEAM 18, the deterministic model does not provide better performance than LSTM-based models. The first hypothesis laid out in section 5.3 is hence partially refuted. Indeed, for TWS, Latitude and Longitude, the deterministic models are superior to the LSTM-based models. Hence, while it cannot be generally stated that the deterministic models always outperform LSTM-based models, for some features they do. Finally, when considering e.g. fig. 7.14 and the corresponding error metrics, the second hypothesis made in section 5.3 can be confirmed: the deterministic models serve as indicators of a dataset's inaccuracies.

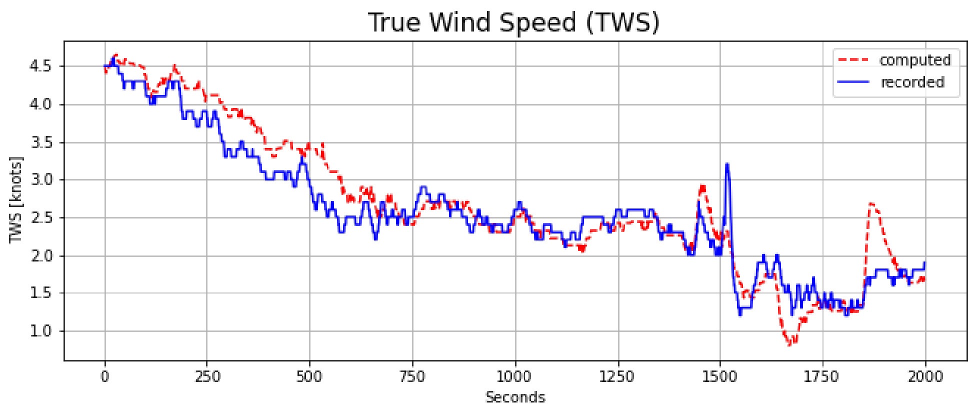


Figure 7.9: Computed vs.true values for True Wind Speed (testing subset of **DRHEAM 18, Concise 8**).

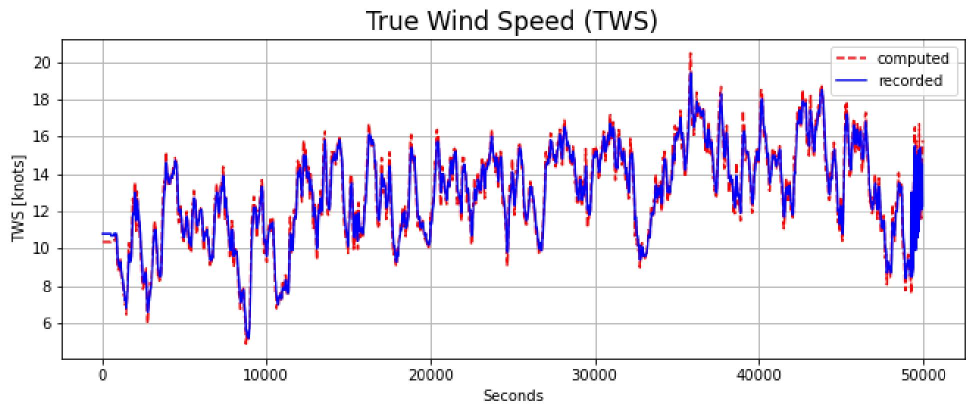


Figure 7.10: Computed vs.true values for True Wind Speed (testing subset of **Atlantic, Concise 8)**.

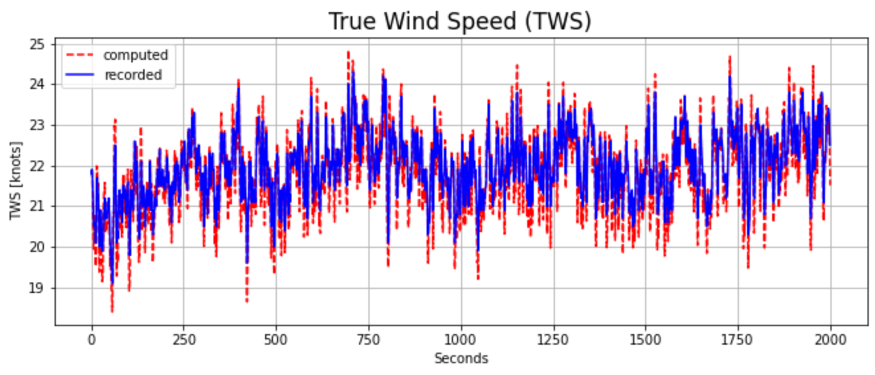


Figure 7.11: Computed vs.true values for True Wind Speed (testing subset of **transat_1, Unknown 1**).

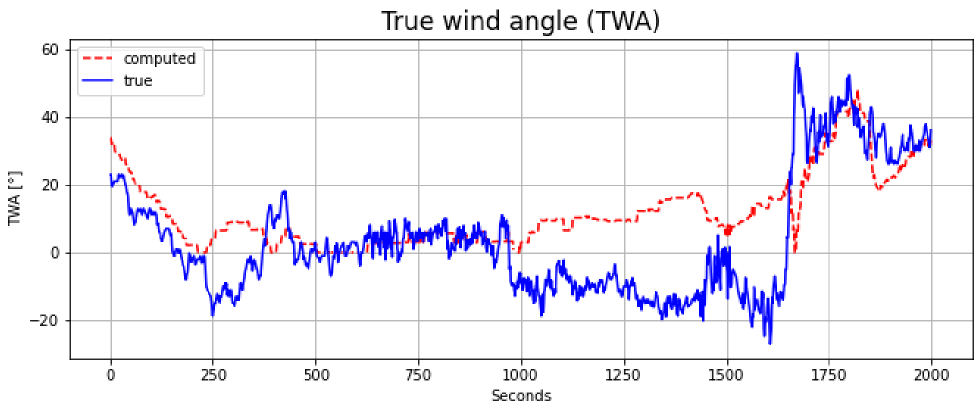


Figure 7.12: Computed vs.true values for True Wind Angle (testing subset of **DRHEAM 18, Concise 8**).

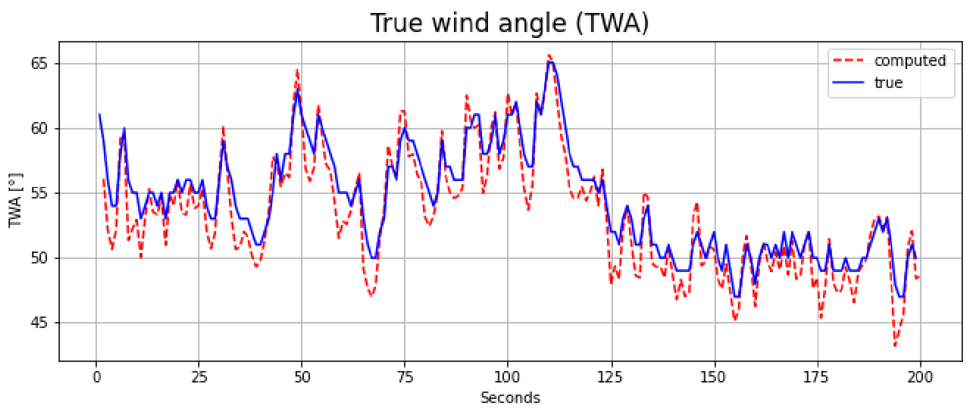


Figure 7.13: Computed vs.true values for True Wind Angle (testing subset of **Atlantic, Concise 8**).

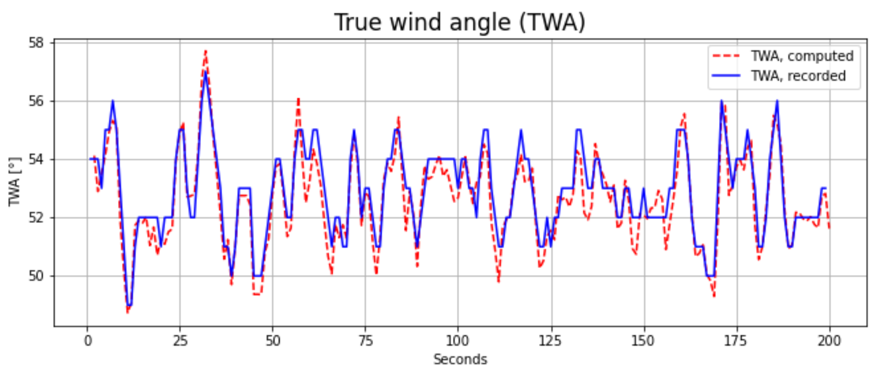


Figure 7.14: Computed vs.true values for True Wind Angle (testing subset of **transat_1, Unknown 1**).

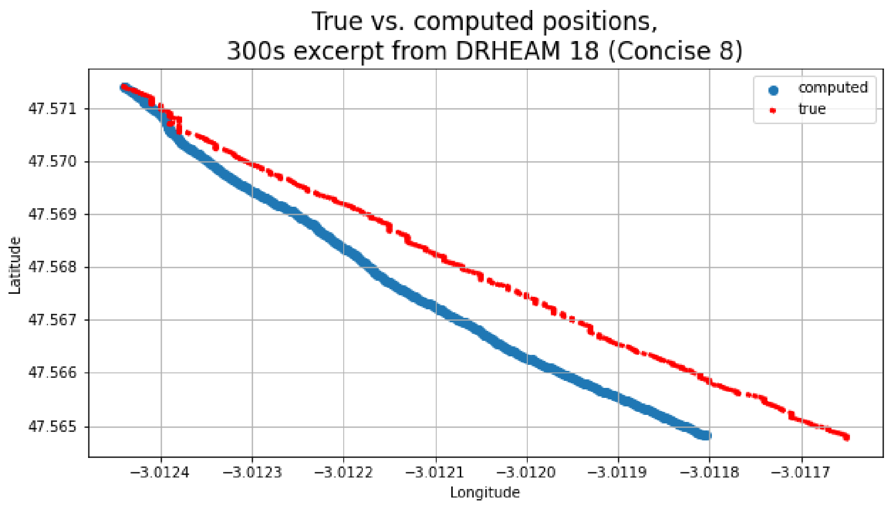


Figure 7.15: Computed vs.true latitude and longitude values for a 5 minutes excerpt (**DRHEAM 18, Concise 8**). The starting point is in the upper left corner, from where the boat was sailed to the lower right.

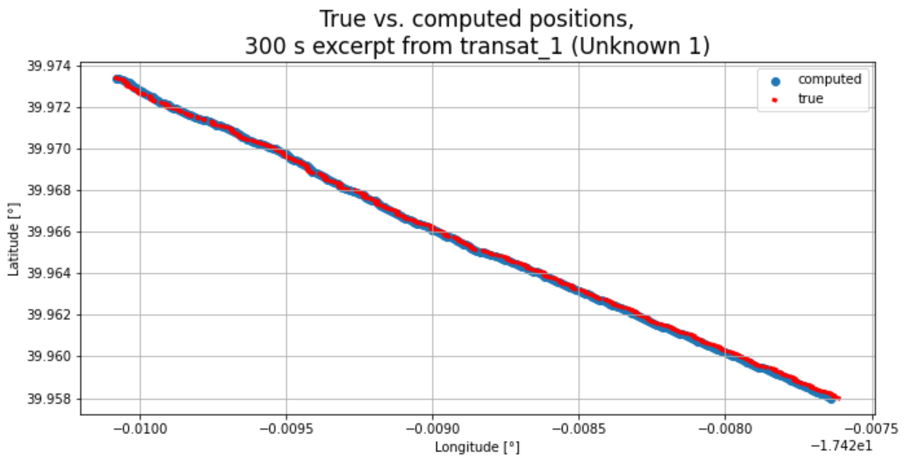


Figure 7.16: Computed vs.true latitude and longitude values for a 5 minutes excerpt (testing subset of **transat_1, Unknown 1**). The starting point is in the upper left corner, from where the boat was sailed to the lower right.

Feature Name	Retained model	Deterministic model
Longitude - cos	Deterministic	Yes
Longitude - sin	Deterministic	Yes
Speed_ov_surface	Model 1	
Speed_ov_ground	Model 1	
VMG	Model 1	
Heading_True - cos	Model 1	
Heading_True - sin	Model 1	
Heading_ov_ground - cos	Model 1	
Heading_ov_ground - sin	Model 1	
Yaw - cos	Model 1	
Yaw - sin	Model 1	
AWA - cos	Model 1	
AWA - sin	Model 1	
TWA - cos	Model 1	Yes
TWA - sin	Model 1	Yes
Latitude	Deterministic	Yes
Pitch	Model 2	
Roll	Model 2	
AWS	Model 2	

Table 7.5: Best-performing models for the boat state features according to the **Concise 8 (DRHEAM 18)** dataset.

7.5 Transferability of models between boats and datasets

The following sections present and discuss the results of the training and testing of n separate models for n features on new data. They correspond to the experimental steps laid out in section 6.5 and summarised under Experiment 5 in table 6.2.

The previous experimental steps served to identify optimised forecasting models for Concise 8 (DRHEAM 18). The models retained as best for the various features are summarised in table 7.5. As previously presented, very accurate results were obtained with the deterministic models for Latitude and Longitude. Hence, in contrast to all other features, no LSTM-based models were retained for these features.

7.5.1 Concise 8 (Atlantic)

Results

Error metrics Table 7.6 presents the error metrics that result from this experimental step. Fig. 7.17 visualises the difference between the MAE values recorded for Atlantic and the ones obtained for DRHEAM 18 with the same model hyperparameters. A similar situation holds for the RMSE. The visualisation is performed only for LSTM-based models, as the previous section already presents and discusses the deterministic models.

Discussion

Performance The performance of the models is generally not comparable with that found for DRHEAM 18. Both the MAE and the RMSE increase in comparison to the values obtained for DRHEAM 18 for nearly all features, as can be seen in fig. 7.17 for the MAE. A few exceptions can be observed; e.g. for AWS, the MAE and RMSE (1.211 and 1.484) are below the values found for the models trained and tested on DRHEAM 18 (1.220 and 1.748). The satisfactory performance for AWS is illustrated by the course of the predicted vs. true values as shown in fig. 7.20. However, as becomes apparent from fig. 7.17, for the vast majority of features the quality of the predictions differs considerably from those determined for DRHEAM 18. Fig. 7.19 illustrates this for feature ”Heading over ground - sine”. While the predicted values follow the pattern of the true values, a strong and consistent offset between both courses can be noticed. Finally, fig. 7.21 illustrates a similar behaviour for Pitch. While the mentioned figures are excerpts to illustrate the behaviour of the models, the same observations regarding poor performance hold for the other features.

In summary, the hyperparameters that result in accurate forecasting models for Concise 8 (DRHEAM 18) cannot be simply transposed identically to the training of forecasting models for the same boat, but with a different data format (Concise 8 (Atlantic)). The first hypothesis laid out in section 6.5 is hence refuted.

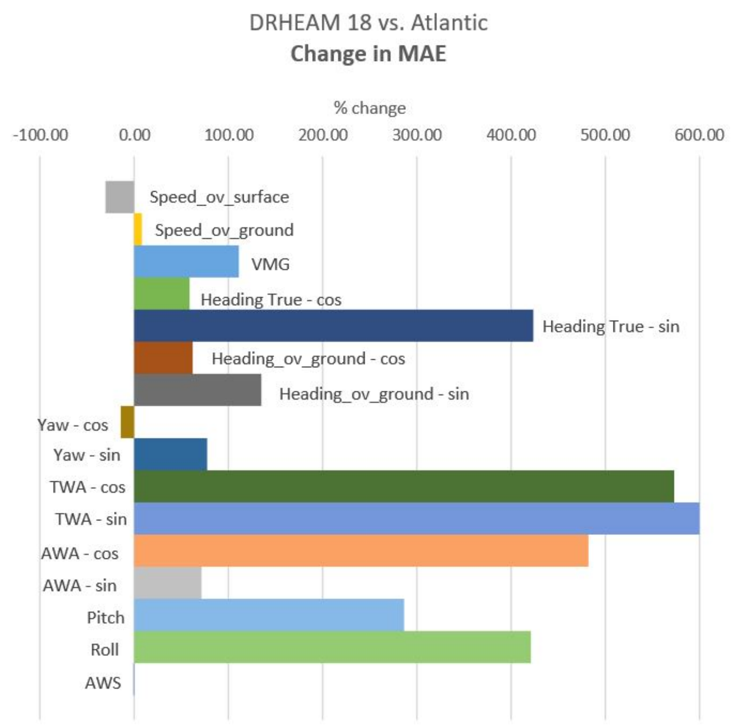


Figure 7.17: Change in MAE of Atlantic vs. DRHEAM 18 (both for Concise 8). x-axis cut off at 600 % (% change for TWA - sin: 3271.43).

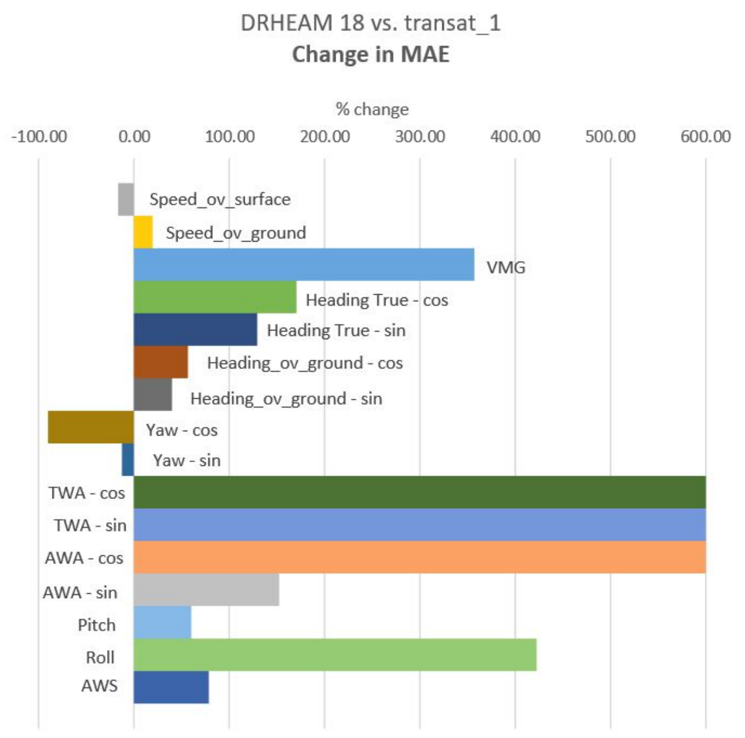


Figure 7.18: Change in MAE of transat_1 (Unknown 1) vs. DRHEAM 18 (Concise 8). x-axis cut off at 600 % (% change for TWA - cos: 1723.08, TWA - sin: 819.05, AWA - cos: 721.43).

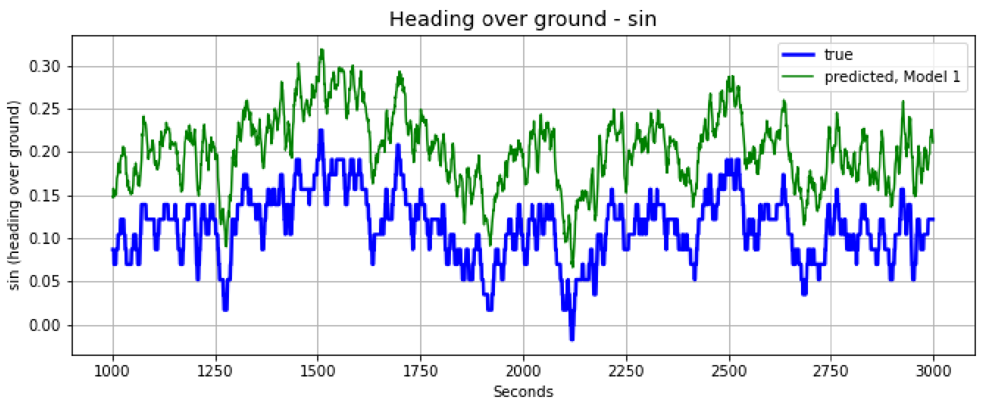


Figure 7.19: Predictions for the sine of heading over ground (testing subset of **Atlantic, Concise 8**).

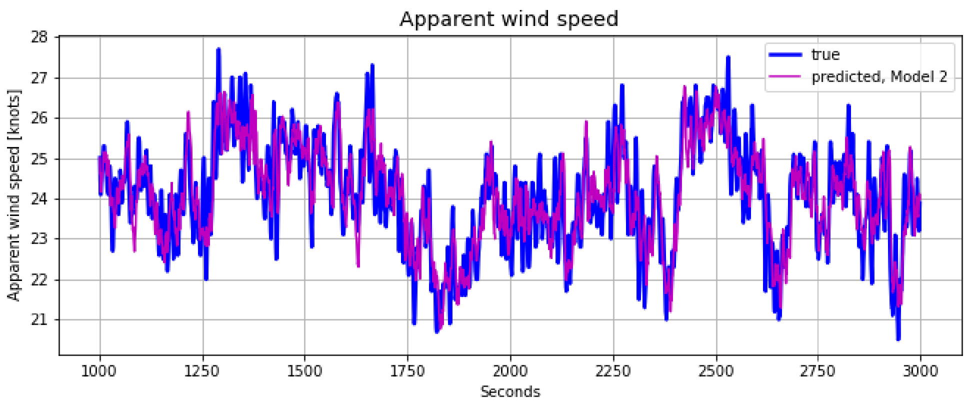


Figure 7.20: Predictions for apparent wind speed (testing subset of **Atlantic, Concise 8**).

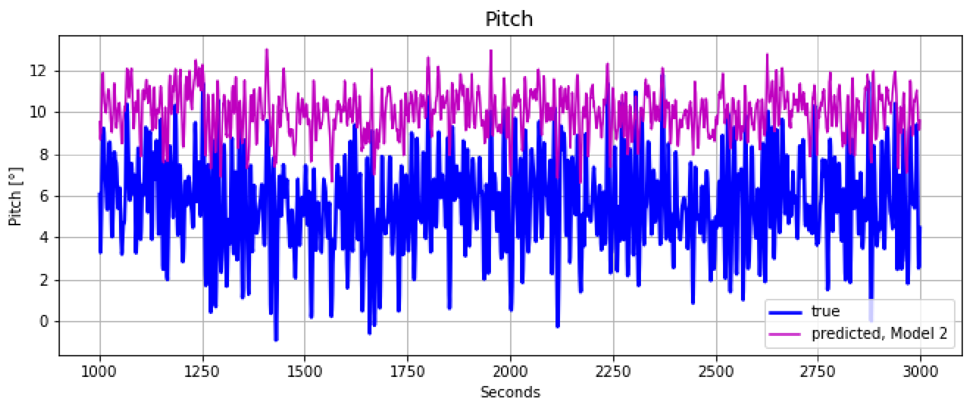


Figure 7.21: Predictions for pitch (testing subset of **Atlantic, Concise 8**).

Feature Name	Retained model	MAE	RMSE
Longitude - cos	Deterministic	$1.398 \cdot 10^{-7}$	$1.647 \cdot 10^{-7}$
Longitude - sin	Deterministic	$7.889 \cdot 10^{-8}$	$9.316 \cdot 10^{-8}$
Speed_ov_surface	Model 1	0.834	0.986
Speed_ov_ground	Model 1	0.887	1.095
VMG	Model 1	1.098	1.378
Heading_True - cos	Model 1	0.027	0.034
Heading_True - sin	Model 1	0.089	0.112
Heading_ov_ground - cos	Model 1	0.060	0.072
Heading_ov_ground - sin	Model 1	0.047	0.062
Yaw - cos	Model 1	0.006	0.009
Yaw - sin	Model 1	0.071	0.087
TWA - cos	Model 1	0.175	0.251
	Deterministic	0.746	0.939
TWA - sin	Model 1	0.0373	0.0560
	Deterministic	0.708	0.938
AWA - cos	Model 1	0.163	0.183
AWA - sin	Model 1	0.036	0.054
Latitude	Deterministic	$3.994 \cdot 10^{-5}$	$4.033 \cdot 10^{-5}$
Pitch	Model 2	2.477	2.993
Roll	Model 2	7.242	8.452
AWS	Model 2	1.211	1.484

Table 7.6: Test performance for the **Concise 8 (Atlantic)** dataset.

7.5.2 Unknown 1 (transat_1)

Results

Error metrics Table 7.7 presents the error metrics resulting from this final experimental step. Fig. 7.18 visualises the difference between the MAE recorded for transat_1 and for DRHEAM 18 with the same model hyperparamaters. It can be observed that the MAE increases for all features except for Speed over Surface, cosine of Yaw and sine of Yaw.

Discussion

Performance Similarly to the results obtained for Atlantic, the performance of the models is generally not comparable to the one found for DRHEAM 18. By considering fig. 7.18, one can observe that the MAE increases for nearly all features; the same holds for the RMSE. As observed for Atlantic, for some features, the error metrics are comparable or improve with respect to the metrics observed for DRHEAM 18. Among these is the cosine of Yaw, for which the MAE passes from 0.007 to 0.0007 and the RMSE from 0.061 to 0.001. However, when considering fig. 7.18, it is evident that this is an exception and that the majority of the hyperparameters that are adequate for DRHEAM 18 are not for transat_1. Furthermore, figs. 7.22 to 7.24 again reveal the same pattern that can be observed as for Atlantic (Concise 8): while the models capture the time element of the patterns rather consistently, i.e. follow the up and down movements of the true values, they do so with a strong offset and not to scale.

Similarities with Atlantic By comparing figs. 7.17 and 7.18, one can observe that the change in for Concise 8 (Atlantic) and transat_1 (Unknown 1) is negative or close to zero for a number of features. Interestingly, this selection of features is similar for both datasets, namely for the features Speed over Surface, Speed over Ground and cosine of Yaw.

Implications of similarities This means that for some features, the hyperparameters found for DRHEAM 18 are adequate. For these features, the hyperparameters do generalise to data recorded for the same boat in a different data format (Concise 8 (Atlantic)), as well as to data recorded for different boats in a different data format (Unknown 1 (transat_1)). Hence, it cannot be excluded that at least for some features, optimised hyperparameters might be transferred between data formats and boats.

In summary, the hyperparameters that result in accurate forecasting models for Concise 8 (DRHEAM 18) cannot be simply transposed identically to the training of forecasting models for a different boat with a different data format (Unknown 1 (transat_1)). The second hypothesis laid out in section 6.5 is hence refuted.

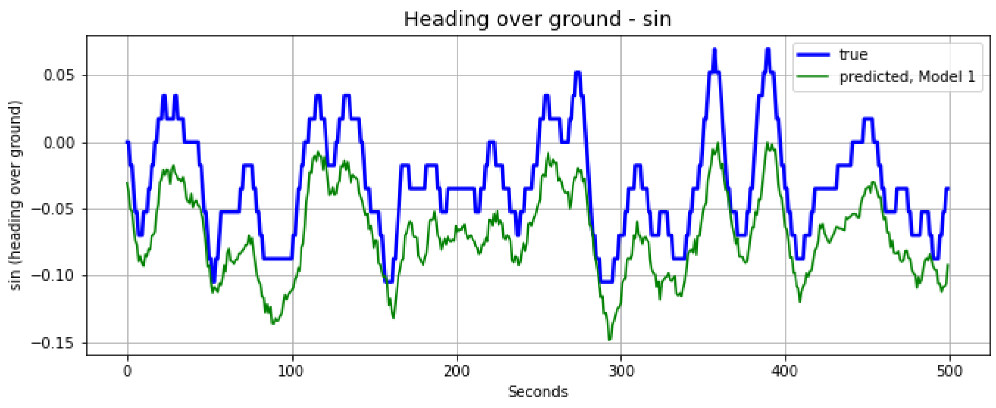


Figure 7.22: Predictions for the sine of heading over ground (testing subset of **transat_1, Unknown 1**).

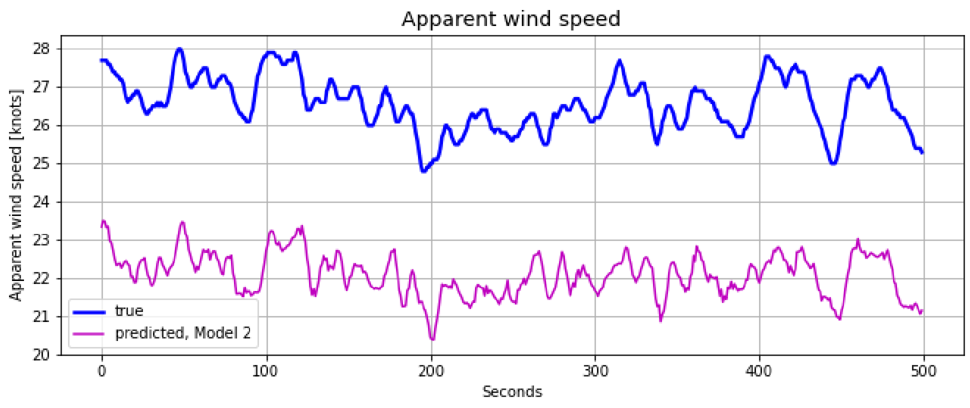


Figure 7.23: Predictions for apparent wind speed (testing subset of **transat_1, Unknown 1**).

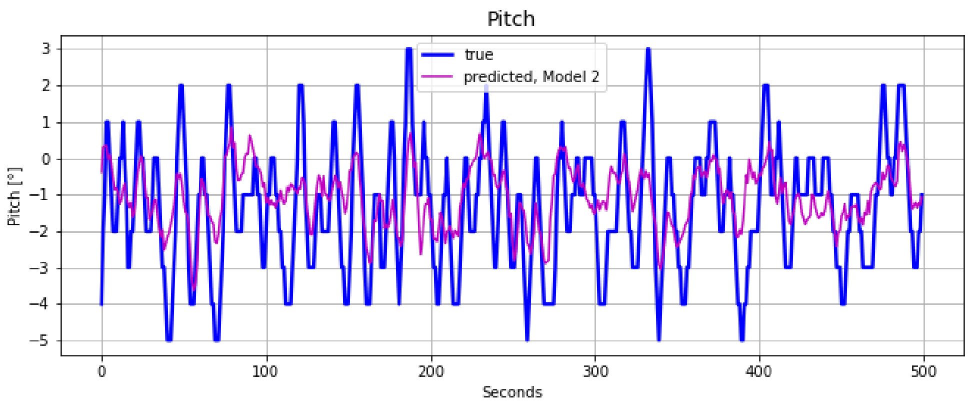


Figure 7.24: Predictions for pitch (testing subset of **transat_1, Unknown 1**).

Feature Name	Retained model	MAE	RMSE
Longitude - cos	Deterministic	$1.384 \cdot 10^{-7}$	$1.762 \cdot 10^{-7}$
Longitude - sin	Deterministic	$3.811 \cdot 10^{-7}$	$4.743 \cdot 10^{-7}$
Speed_ov_surface	Model 1	1.002	1.152
Speed_ov_ground	Model 1	0.981	1.145
VMG	Model 1	2.377	2.875
Heading_True - cos	Model 1	0.046	0.076
Heading_True - sin	Model 1	0.039	0.056
Heading_ov_ground - cos	Model 1	0.058	0.092
Heading_ov_ground - sin	Model 1	0.028	0.041
Yaw - cos	Model 1	0.0007	0.001
Yaw - sin	Model 1	0.035	0.045
TWA - cos	Model 1	0.474	0.656
	Deterministic	0.369	0.390
TWA - sin	Model 1	0.193	0.273
	Deterministic	0.549	0.564
AWA - cos	Model 1	0.230	0.272
AWA - sin	Model 1	0.053	0.080
Latitude	Deterministic	$4.838 \cdot 10^{-5}$	$4.977 \cdot 10^{-5}$
Pitch	Model 2	1.027	1.281
Roll	Model 2	7.263	8.947
AWS	Model 2	2.178	2.816

Table 7.7: Test performance for the **Unknown 1 (transat_1)**.

Chapter 8

Conclusion and Future Work

The overarching goal of this thesis consists in developing reliable forecasting models for features that describe the state of a sailing boat. The results presented above allow to draw a number of conclusions. Moreover, they allow to determine a clear framework with which to train reliable forecasting models and hence generate a reliable RL simulation environment.

8.1 Conclusion

In the present study, it was shown that it is possible to identify reliable forecasting models for the features that define a sailboat's state. It was found that training separate models individually for all the features is an expedient method to that end. However, it was shown that the hyperparameters of these separate models do not generalise across different navigation recording systems. Finally, it could not be conclusively determined whether for a given data format, the hyperparameters of forecasting models generalise across different boats. Different conclusions can be drawn from this.

1. As a first conclusion, Bayesian optimisation improves the performance of a single forecasting model for n features (7.1). In comparison to the original, un-optimised model, the reduction of the overall MAE is of -22.58%, while the overall RMSE is reduced by 15.84% However, the optimised single model does not achieve an accurate level of prediction performance. This is illustrated below in fig. 8.1, showing an excerpt of one of the features' true and predicted values. The poor performance attributed to the fact that a single network cannot take into account the complexity of the task at hand.

2. Using the Concise 8 (DRHEAM 18) dataset, it has been shown that it is possible to train and test n reliable models for n features, and that they result in more accurate predictions than a single model for n features (7.2). Furthermore, it is possible to do so with a relatively small amount of cleaned data (64.5 hours for DRHEAM 18). Only one exception holds, namely for Pitch (7.3.1). However, this was not investigated further as it is found that the data available for Pitch is highly likely to be corrupted. Finally, for the problem at hand, tuning the

models by removing a dense layer proves successful (7.3) as it allows to adapt the model's ability to capture the level of nonlinearities effectively at hand ([34]). This can be observed in fig. 8.1 below, where the tuning of a first model (Model 1) leads to better results (Model 2).

3. It is found that for some features, deterministic models that rely on first-principles formulae can be used instead of LSTM-based models (7.4). They outperform the latter for latitude and longitude (99.97% improvement or more over best-performing LSTM models) and are a viable alternative to them. Moreover, the use of deterministic formulae allows to conclude that the data at hand does not represent the absolute physical truth and that it should be used with caution. Indeed, the values of some features should be exactly derivable from those of other features, as they are linked by unambiguous formulae and are partly calculated directly in that way in the autopilot software. It is hence found that these formulae can serve as an indicator of a dataset's inaccuracies. However, when verifying whether the recorded data is coherent with the unambiguous relationships that govern this data, it was found that these computed values are not exactly identical with the recorded values. As an example, for the Atlantic dataset, the values of AWS are computed by using the originally recorded "true" data. The computed values for AWS show an MAE of 0.410 knots to the originally recorded values for AWS, i.e. the formula is not exactly verified. This inaccuracy is attributed to the mis-calibration of the autopilot sensors.

4. It is found that hyperparameters that lead to accurate results for a given data format do not generalise to other data formats (7.5). This was shown by using data that was recorded for the same boat, but relied on a different collecting system (Concise 8). It was also shown for data that was recorded for different boats using different collecting systems (Concise 8 and Unknown 1).

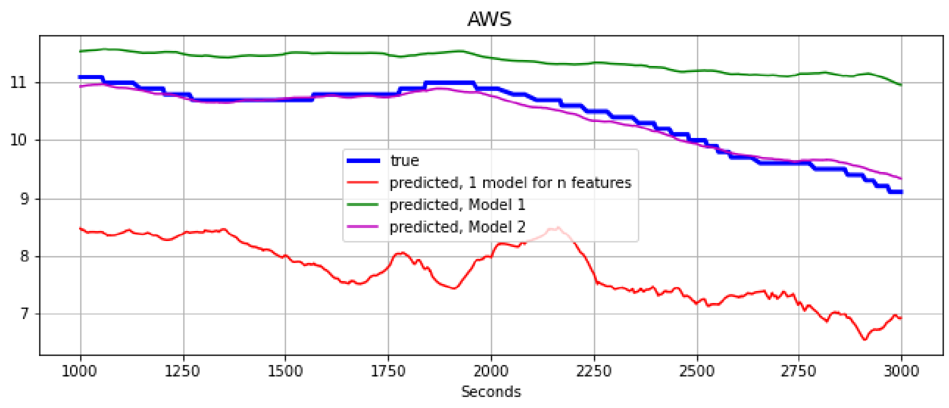


Figure 8.1: Predictions for apparent wind speed (testing subset of **DRHEAM 18, Concise 8**).

Hence, in summary, reliable forecasting models of the boat state features could be identified in the present study. However, they could only be established for a specific boat for which data is at hand in a specific data format (Concise 8 (DRHEAM 18)). This data format corresponds to an old recording protocol.

All of the new data available for this and future iterations of JTR AI is going to be in a modern .nkz format. Hence, the logical next step of JTR AI consists in optimising and training forecasting models for different boats (Concise 8 and Unknown 1) using datasets in the .nkz format. This allows to assess the transferability of model hyperparameters across different boats rather than across different data formats. This investigation constitutes the logical next step in reaching the overarching goal of the present study, i.e. creating a reliable RL simulation environment by identifying accurate forecasting models of the boat state's features. From this, a framework for future work in JTR AI can be derived. It will be presented subsequently.

8.2 Future Work

8.2.1 Framework to create a reliable RL simulation environment

Motivation

The logical next step in JTR AI consists in the identification of reliable forecasting models for the boats state features. In particular, it would be of interest to identify architectures and hyperparameters that allow the training of adequate forecasting models for any new .nkz dataset from any unseen boat.

Computational cost This would save the long and computationally intensive step of identifying optimal models using Bayesian optimisation each time data from a new boat becomes available. Indeed, the Bayesian optimisation of 1 model for n features alone required 6.5 days of runtime (cf. section 6.1).

Generalisability Besides the identification of generalisable architectures and hyperparameters, it would be of interest to investigate whether there are generalisable models which, once trained, might be applicable to other boats of the same class (e.g. a model trained on Concise 8 (Class 40) which would be applicable to VMB (Class 40)), or which are trained on hybrid datasets and might be applicable to other boats. An example of the latter would be a model trained on Concise 8 (Class 40) and Unknown 1 (IMOCA 60) which would be adequate for Unknown 2 (IMOCA 60). This would be the "holy grail" of forecasting models for JTR AI, as the resulting RL simulation environment would be suitable for any boat, which would mean that one would only need to focus on developing the RL algorithms. Indeed, this would significantly increase the scalability and commercial viability of any solutions, as one would not need to re-iterate through all optimisation steps each time an algorithm would need to be developed for a new boat.

Framework

The framework to conduct this in a systematic manner is presented below in table 8.1. It is based on the conclusions drawn from the present study and composed of the following steps:

1. **individually optimise n models for n features**. In 7.1, it has been shown that Bayesian optimisation can be used to improve a forecasting model. Moreover,
   - it was found that the number of dense layers following the LSTM layers influences the behaviour of the forecasting models (6.2 and 6.3). Hence, the search space should include the number and width of dense layers of the network.
   - this should be done for all features except Latitude and Longitude. Indeed, in section 7.4 it was found that the deterministic models can be used for accurate predictions with the available datasets.
   - the optimised hyperparameters should be compared to identify whether certain features require similar or even identical hyperparameters. This allows for extensions of research into the transferability of hyperparameters across the models for different boats.
2. **train the models with the optimised hyperparameters.**
3. **test the trained models**
   - on the testing subset of the dataset they were trained on, such that its performance can be assessed.
   - on the testing subsets of the other datasets to assess whether models trained on one boat generalise to other boats.

Data to be used

The described investigation should be conducted with the Concise 8 (Atlantic), the Unknown 1 (transat_1), and a hybrid dataset. The hybrid dataset should be sampled from training, validation and testing data from the Concise 8 (Atlantic) and Unknown 1 (transat_1) datasets. Indeed, the use of a hybrid dataset allows to assess whether there are features for which models can be trained on data from different boats and still result in accurate predictions. In other words, this helps to assess whether there are features for which models can be trained boat-independently. As explained in the first part of this section, this would be beneficial by reducing the computational effort to train accurate forecasting models.

Phase 1: Model optimisation
For each of the three boats: - For Latitude and Longitude, retain deterministic models - For all other features: - define search space for Bayesian optimisation, including number and width of dense layers - run n Bayesian optimisations of n LSTM-based models
Phase 2: Model training
For the n features for which no deterministic models can be used: train the n LSTM-based models with optimised hyperparameters
Phase 3: Model testing and evaluation
For each of the n LSTM-based models, test performance on testing subset of: - boat on whose training and validation subsets the model was trained on - other boats

Table 8.1: Framework to train adequate forecasting models for all features, inspect their transferability and to apply them for RL. Boats involved: Concise 8 (Atlantic), transat_1 (Unknown 1), hybrid boat (Atlantic + Unknown 1).

8.2.2 Further directions of work

Integration of forecasting models into RL framework

The development of reliable forecasting models for boat state features is synonymous with the development of a reliable RL simulation environment. Hence, once this development has taken place as laid out in detail in the previous section, the logical continuation would consist in the utilisation of the developed forecasting models in the RL environment developed in a previous iteration of JTR AI. This would be of interest to

- test whether the RL algorithm developed by Roman Kastusik remains unsatisfactory even when it is trained using a reliable simulation environment.

- use the simulation environment to further develop the RL algorithm.

It should be mentioned that a large amount of the time available for this individual project was spent on improving the RL framework itself (structure, reduction of calculation steps, speed, reliability, but not the algorithms themselves). This purely technical work includes the implementation of a very easy integration of new models and data into the RL framework. In other words, the optimised forecasting models would only need to be plugged into the RL framework and the performance of the RL agent could be assessed swiftly.

Investigation of other forecasting models

A second direction of work would focus on the forecasting models. In fact, if substantially more computing power was available than was for the present iteration of JTR AI, various other models could be tested for the forecasting task. The extensive background research in section 3 presents the various directions that are possible.

GANS for forecasting Particularly, an interesting and novel approach would consist in investigating the application of GANs for forecasting, a field that has been pioneered in recent years and shown reliable performance for nonlinear data as available in this study (e.g. in [25], [26], [29]).

Hybrid models Moreover, the application of hybrid models as investigated in [30], [31] and [33] could be of interest. Indeed, in section 7.4, it has been shown that deterministic models can be applied to compute the values of certain features. At the same time, it has been seen that these deterministically computed values deviate from the recorded values. Indeed, the computed values for AWS present an MAE of 0.410 knots and an RMSE 0f 0.520 knots with respect to the originally recorded values. However, both error metrics should theoretically be exactly zero, as explained in section 6.4. As laid out in the mentioned publications and in the literature review of the present study (3.5), hybrid models allow to account for deviations between deterministic models and physical reality. This makes the application of hybrid models an interesting direction of further work for JTR AI.

Explainability of forecasting models

Finally, investigating the explainability of the forecasting models would constitute a valuable further direction of work.

Motivation Indeed, in section 7, one can only observe and hypothesise as to why the performance of forecasting models is satisfying or unsatisfying. However, no well-founded quantitative statements can be made about the influence of the individual features on the predictions made by a model. The deeper understanding resulting of the forecasting models would not only be useful to optimise the hyperparameters and architectures of the models (which would correspond to a type of ”model debugging”). As mentioned in the following chapter on ethical considerations (9), it would also be relevant to understand in which situations the forecasting models are not reliable and where thus the RL agent cannot be trained reliably to steer the boat. In the event of a possible deployment of an autopilot resulting from further iterations of JTR AI, this would allow to avoid dangerous situations where the autopilot would not be up to its task of steering the boat safely.

Context Finally, research activity of explainability in the context of ML is currently strongly increasing. This recent upsurge has been documented and described among others by Arrieta et al. [38]. Fig. 8.2 from the mentioned publication visualises this interest. Moreover, frameworks for explainable ML and AI are available e.g. with the SHAP library [39] and Google's Explainable AI tools [40]. However, these frameworks are not specifically designed for problems with timeseries of dynamical systems. Hence, the proposed direction of further work could result in valuable contributions to this field of research, considering that many ML problems involve timeseries data and that explainability is desirable for these problems.

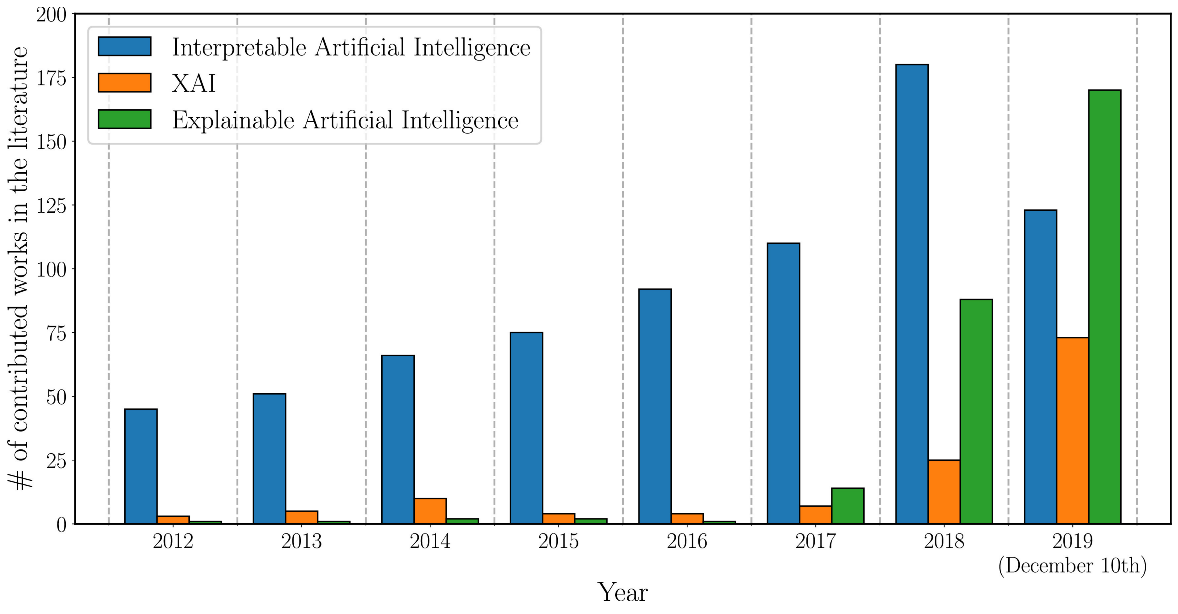


Figure 8.2: "Evolution of the number of total publications whose title, abstract and/or keywords refer to the field of XAI during the last years", as in [38].

Chapter 9

Ethical considerations

In the following, ethical considerations regarding the present project are conducted in accordance with the ethics checklist published by the Department of Computing at Imperial College London. The completed checklist can be found in A.

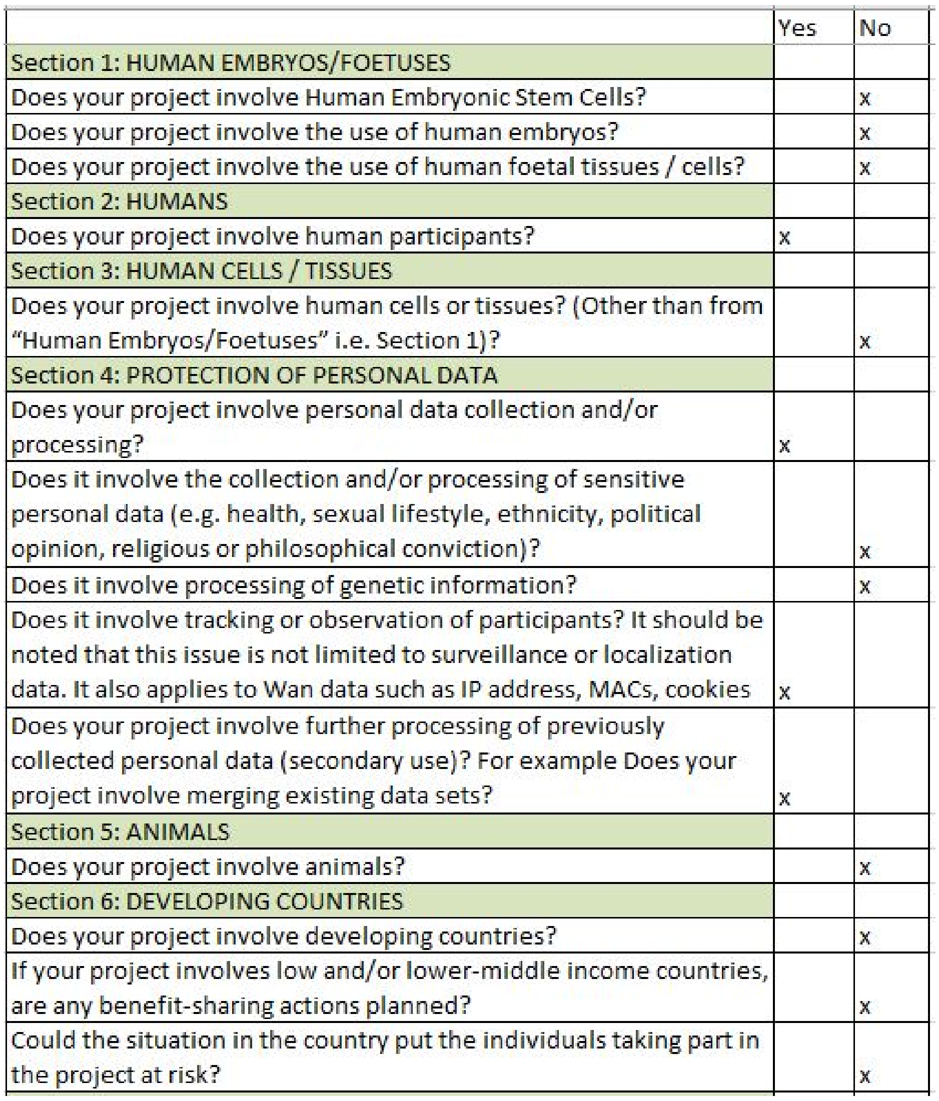
- **Humans involved.** The project involves human participants at the moment where the autopilot is deployed on a real boat. As the project is conducted in cooperation with JTR, the first humans that are likely to be involved involved are Jack Trigger and any crew members or guests on boat. If the autopilot proves to be successful and is deployed to other boats, all persons on boats that are guided by this autopilot and/or persons in the surroundings of the boat (e.g. boats navigating in proximity to the boat using the autopilot) are potentially affected by the autopilot. Moreover, given the black box nature of large parts of the models studied in the present work, a deployment to real-world settings on boats should only be conducted after thorough testing of the technology. This should include, but is not limited to, real tests in real-world settings where the autopilot's guiding the boat could be interrupted at any moment by a human closely monitoring the autopilot's performance and the boat state; imposing limits on the rudder angle movement, s.t. the system is prevented from performing any extreme rudder angle changes following the model's output; and implementing a framework that allows to interpret the model's output, i.e. to explain its behaviour.

- **Protection of personal data.** The data of Jack Trigger's races is stored within the cloud services of T-DAB. The access to these services is password-protected, and so is the access to the private laptop used to connect to these services. Other personal data, i.e. navigation logs provided by NKE and originating from other users of NKE technologies, are stored on the same protected services. This data is anonymous, i.e. no information on the identities of these users is available. Furthermore, no efforts whatsoever are made to re-construct the identities of these individuals. Finally, when using the cloud services, the choice of location is set to ”UK South”. The personal data do hence not leave the UK, nor the EEA.

- **Dual use.** The project has an exclusive civilian applicaton focus.
- **Legal issues.** A non-disclosure agreement with nke is in place from previous projects on JTR AI, according to which no data provided by nke shall be published. Hence, apart from the data being stored on protected cloud services, the code is stored on a private repository as opposed to a public one. While parts of the code might be published at later stages of the JTR AI project, this is to be done such that the provided data and information on how it was recorded by nke hardware and/or software cannot be inferred from the published code. Furthermore, the sailing autopilot is designed with the explicit aim of reaching high performance in automated sailing during races. Hence, before using the autopilot in any race, it must be checked whether the algorithms and capabilities of the used autopilot are in accordance with the rules of the race. In practice, this means that the autopilot must fulfil the class rules of the class it belongs to. In particular, JTR's Concise 8 belongs to the Class40, a sailboat class used in many races. As of April 2020, the Class40 informs in its class rules that "a plan to limit all the main components involved in the pilot (Inertial navigation system, processor, autopilot computer and related licenses, excluding wind and speed sensors) is being drafted for submission to a vote by a AGE in 2020 and an application in 2021." ([1]). In light of these developments, the rules applicable to sailboat classes that are admitted to races should be checked before using the autopilot in any race. Finally, the General Data Protection Regulation (GDPR), applicable in the UK, has to be respected. More specifically, in the present project, the responsibilities as a controller in the sense of GDPR apply, i.e. compliance with the data protection principles as listed in article 5 of the GDPR must be observed and individuals' rights have to be respected. The latter includes that Jack Trigger must be in a position to exercise his rights "regarding their personal data, including the rights of access, rectification, erasure, restriction, data portability, objection and those related to automated decision-making" ([41]). A reliable communication between T-DAB and Jack Trigger ensures that these rights can be observed throughout the project. Finally, the security of the personal data must be ensured. As mentioned above, the security of the personal data is provided via its storage on a protected cloud service.

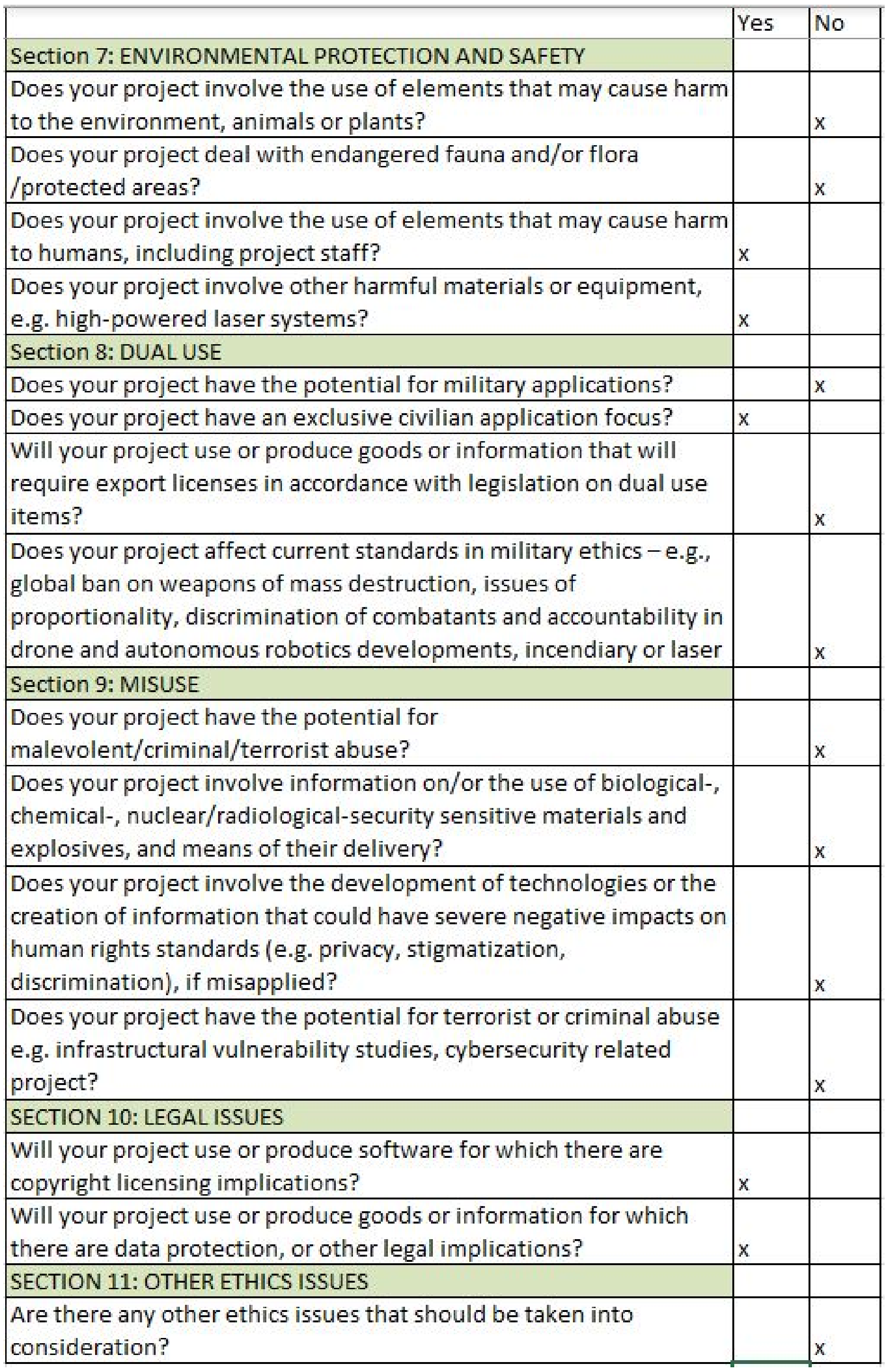
Appendix A

Ethics checklist



	Yes	No
Section 1: HUMAN EMBRYOS/FOETUSES		
Does your project involve Human Embryonic Stem Cells?		x
Does your project involve the use of human embryos?		x
Does your project involve the use of human foetal tissues / cells?		x
Section 2: HUMANS		
Does your project involve human participants?	x	
Section 3: HUMAN CELLS / TISSUES		
Does your project involve human cells or tissues? (Other than from "Human Embryos/Foetuses" i.e. Section 1)?		x
Section 4: PROTECTION OF PERSONAL DATA		
Does your project involve personal data collection and/or processing?	x	
Does it involve the collection and/or processing of sensitive personal data (e.g. health, sexual lifestyle, ethnicity, political opinion, religious or philosophical conviction)?		x
Does it involve processing of genetic information?		x
Does it involve tracking or observation of participants? It should be noted that this issue is not limited to surveillance or localization data. It also applies to Wan data such as IP address, MACs, cookies	x	
Does your project involve further processing of previously collected personal data (secondary use)? For example Does your project involve merging existing data sets?	x	
Section 5: ANIMALS		
Does your project involve animals?		x
Section 6: DEVELOPING COUNTRIES		
Does your project involve developing countries?		x
If your project involves low and/or lower-middle income countries, are any benefit-sharing actions planned?		x
Could the situation in the country put the individuals taking part in the project at risk?		x

Figure A.1: Ethics checklist provided by the Department of Computing of Imperial College, part 1.



	Yes	No
Section 7: ENVIRONMENTAL PROTECTION AND SAFETY		
Does your project involve the use of elements that may cause harm to the environment, animals or plants?		x
Does your project deal with endangered fauna and/or flora /protected areas?		x
Does your project involve the use of elements that may cause harm to humans, including project staff?	x	
Does your project involve other harmful materials or equipment, e.g. high-powered laser systems?	x	
Section 8: DUAL USE		
Does your project have the potential for military applications?		x
Does your project have an exclusive civilian application focus?	x	
Will your project use or produce goods or information that will require export licenses in accordance with legislation on dual use items?		x
Does your project affect current standards in military ethics – e.g., global ban on weapons of mass destruction, issues of proportionality, discrimination of combatants and accountability in drone and autonomous robotics developments, incendiary or laser		x
Section 9: MISUSE		
Does your project have the potential for malevolent/criminal/terrorist abuse?		x
Does your project involve information on/or the use of biological-, chemical-, nuclear/radiological-security sensitive materials and explosives, and means of their delivery?		x
Does your project involve the development of technologies or the creation of information that could have severe negative impacts on human rights standards (e.g. privacy, stigmatization, discrimination), if misapplied?		x
Does your project have the potential for terrorist or criminal abuse e.g. infrastructural vulnerability studies, cybersecurity related project?		x
SECTION 10: LEGAL ISSUES		
Will your project use or produce software for which there are copyright licensing implications?	x	
Will your project use or produce goods or information for which there are data protection, or other legal implications?	x	
SECTION 11: OTHER ETHICS ISSUES		
Are there any other ethics issues that should be taken into consideration?		x

Figure A.2: Ethics checklist provided by the Department of Computing of Imperial College, part 2.

Appendix B

Cleaning of abnormal segments and of segments containing tacks

B.1 Tack detection model

The most reliable method was identified to be a decision tree, which receives as input the difference of the following measures between times t and $t + 30s$: True Wind Angle, Rudder Angle, Magnetic Heading and Roll Angle. The tack-recognizing decision tree reached a recall rate of 100 % and a precision rate of 75 %, as well as an F10-score of 99.67%. The F-10 score was chosen as an evaluation metric by Stanislas Hannebelle as it allows to put a stronger importance on the recall rate as on the precision rate, which is desirable as a maximum of tacks should be identified by the tack detection model, cf. his final report for further information [5]. The confusion matrix of the model is displayed in table B.1.

	True Tack	True No-Tack
Predicted as Tack	True Positives: 6	False Positives: 2
Predicted as No-Tack	False Negatives: 0	True Negatives: 1449

Table B.1: Confusion matrix of the decision tree classifier; for details, cf. [5]

Appendix C

Reinforcement Learning Framework

C.1 Deep RL agent

As laid out in the earlier vchapters of this report, the focus of this thesis lies on identifying a reliable model that forecasts boat states, i.e. on the determination of a robust simulation environment for a deep RL algorithm. The further development of the latter is not part of the present project, and is therefore not described in detail. For in-depth information about the deep RL algorithms employed, the interested reader is referred to Roman Kastusik's final report [4]. However, the detailed description of the reward function developed and used by Roman Kastusik is worth describing in more detail here.

Rationale The underlying idea consists in training the RL agent by selecting a random entry from the dataset as the agent's initial state (i.e. including its location). Following this, the location lying τ minutes ahead in the originally sailed track is retrieved from the dataset and used as waypoint, i.e. the position that the RL agent should be as close to as possible. Any time the state of the RL agent is updated, the state of the waypoint is updated to the next state according to the originally recorded navigation log. The RL agent's objective would then consist in going from start to finish by being as close to the waypoint as possible. Finally, in order to prevent a high rate of inputs that would effectively correspond to sudden rudder movements and hence high drag, the rate of inputs needs to be penalized.

Reward function Following these observations, the reward function was defined as

$$r = \lambda e^{\mathbf{x_t^T} \mathbf{\Lambda} \mathbf{x_t}} - \zeta \dot{R} \tag{C.1}$$

where

- $\mathbf{x_t^T}$ = [speed over surface(t); speed towards way point(t)], polar speed(TWA,TWS) $\cdot$ [1; 1] is the error of the states relative to the waypoint reference
- $\mathbf{\Lambda}$ is a matrix of weights given to each of the states in x

- $\lambda, \zeta \in [0, 1]$ are arbitrary scaling factors

Hence, the developed reward function incentivises the RL agent to be as close as possible to the waypoint ahead. Hence, in theory, the RL agent can learn steering behaviour that leads to faster sailing than the original sailor's sailing.

Challenges encountered The developed RL algorithm was validated in simulation environments provided by Open AI Gym [42], namely in the two classical environments of continuous control 'MountainCarContinuous-v0' and 'Pendulum-v0', which proved that the developed RL algorithms are robust. However, the application to the sailing simulation environment as described in 2.4.1, i.e. as defined by the LSTM model supposed to simulate the boat's behaviour, was not satisfactory. If one compares the originally sailed route with the route "sailed" by the trained RL-algorithm, one can see in figure C.1 that these differ strongly from each other: the RL-algorithm predicts rudder angles which result in a constant turning of the boat. This is in no way satisfactory. Due to the good performance of the RL algorithm in the two Open AI gym environments and the still improvable results of the simulation environment as presented in section 2.4.1, it is assumed that this is mainly due to the unsatisfactory performance of the simulation environment, i.e. of the timeseries forecasting model. In addition, it only makes sense to optimise the RL algorithm (e.g. its hyperparameter or its reward function) if a simulation environment is available that reflects the real conditions of the boat reliably. This led to the decision to focus the present work on the development of a reliable simulation environment. In this light, the next chapter presents an in-depth literature review of developments in the domain of autonomous sailing and timeseries forecasting.

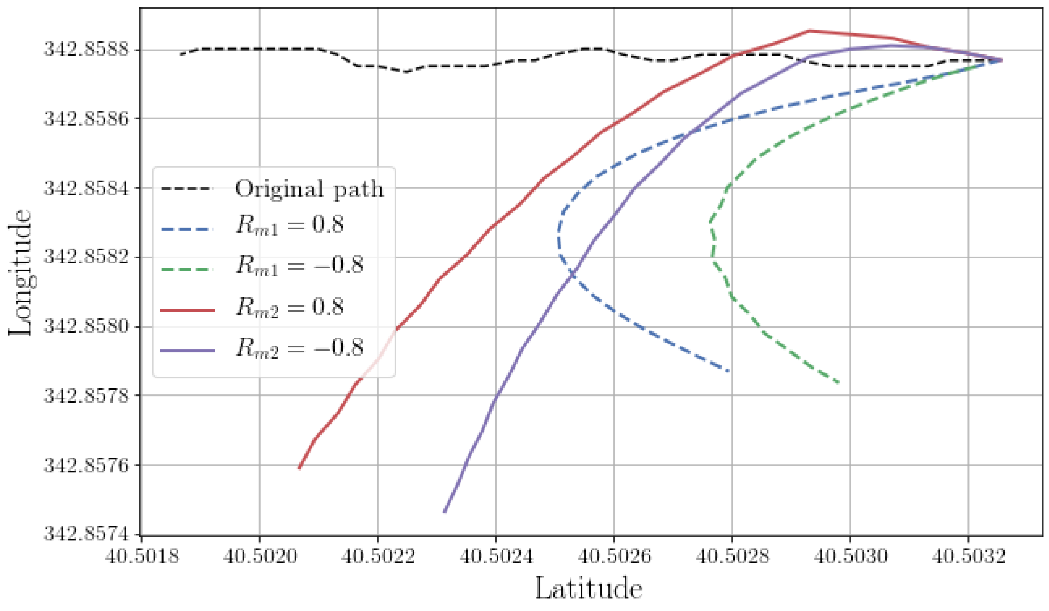


Figure C.1: Paths generated by the deep RL algorithm using models 1 and 2 vs. original path

Appendix D

Conversion from nkz to csv

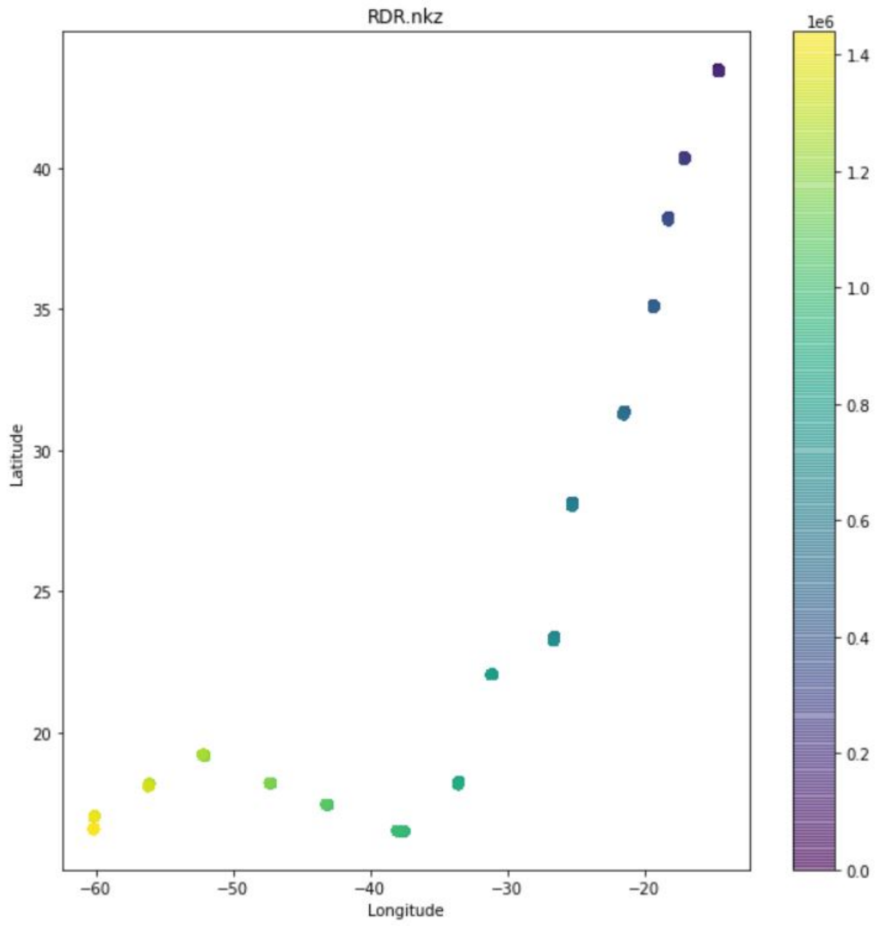


Figure D.1: Truncated latitude and longitude values for the RDR dataset. A properly converted dataset would present itself with multiple latitude and longitude values, resulting in a continuous sailing trajectory instead of disparate points.

Bibliography

[1] Class40 2020 class rules, 2020. URL `https://www.class40.com/modules/kameleon/upload/1classrules2020v3.pdf`. pages 12, 13, 106

[2] Imoca class rules, accessed 25.05.2020. URL `https://www.imoca.org/en/imoca/class-rules`. pages 12

[3] Play to sail website, accessed on 26.08.2020. URL `https://www.dockstahavet.se/blog/basic-yachting-terminology`. pages 14

[4] R. Kastusik, P. Baiz, and E. Topham. Automation and intelligent optimisation in high performance sailing boats, 2019. pages 15, 20, 25, 27, 28, 29, 30, 45, 56, 60, 110

[5] S. Hannebelle, P. Baiz, and E. Topham. Automation and intelligent optimisation in high performance sailing boats, 2019. pages 17, 18, 19, 21, 22, 23, 24, 25, 29, 56, 62, 109

[6] B. Ulstad, P. Baiz, E. Topham, and I. Scattergood. Automation and intelligent optimisation in high performance sailing boats : Supervised learning approach, 2019. pages 17, 56, 60

[7] Martijn L Van Aartrijk, Claudio P Tagliola, and Pieter W Adriaans. Ai on the ocean: the robosail project. In ECAI, pages 653–657. Citeseer, 2002. pages 32

[8] Robosail website, accessed 24.05.2020. URL `http://robosail.com/sailingteam/`. pages 32, 33

[9] Bulent Duz, Bart Mak, Remco Hageman, and Nicola Grasso. Real time estimation of local wave characteristics from ship motions using artificial neural networks. 09 2019. pages 33

[10] Zhipeng Shen, Saisai Wang, Haomiao Yu, and Chen Guo. Online speed optimization with feedforward of unmanned sailboat via extremum seeking without steady-state oscillation. Ocean Engineering, 189:106393, 2019. ISSN 0029-8018. doi: https://doi.org/10.1016/j.oceaneng.2019.106393. URL `http://www.sciencedirect.com/science/article/pii/S0029801819305475`. pages 33

[11] Yingjie Deng, Xianku Zhang, Guoqing Zhang, and Chenfeng Huang. Parallel guidance and event-triggered robust fuzzy control for path following of autonomous wing-sailed catamaran. **Ocean Engineering**, 190:106442, 2019. ISSN 0029-8018. doi: https://doi.org/10.1016/j.oceaneng.2019.106442. URL `http://www.sciencedirect.com/science/article/pii/S0029801819305906`. pages 34

[12] Guoqing Zhang, Jiqiang Li, Bo Li, and Xianku Zhang. Improved integral los guidance and path-following control for an unmanned robot sailboat via the robust neural damping technique. **Journal of Navigation**, 72(6):1378–1398, 2019. doi: 10.1017/S0373463319000353. pages 34

[13] Jan G. De Gooijer and Rob J. Hyndman. 25 years of time series forecasting. **International Journal of Forecasting**, 22(3):443 – 473, 2006. ISSN 0169-2070. doi: https://doi.org/10.1016/j.ijforecast.2006.01.001. URL `http://www.sciencedirect.com/science/article/pii/S0169207006000021`. Twenty five years of forecasting. pages 34

[14] Nesreen K. Ahmed, Amir F. Atiya, Neamat El Gayar, and Hisham El-Shishiny. An empirical comparison of machine learning models for time series forecasting. **Econometric Reviews**, 29(5-6):594–621, 2010. doi: 10.1080/07474938.2010.481556. URL `https://doi.org/10.1080/07474938.2010.481556`. pages 34

[15] Souhaib Ben Taieb, Gianluca Bontempi, Amir F. Atiya, and Antti Sorjamaa. A review and comparison of strategies for multi-step ahead time series forecasting based on the nn5 forecasting competition. **Expert Systems with Applications**, 39(8):7067 – 7083, 2012. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2012.01.039. URL `http://www.sciencedirect.com/science/article/pii/S0957417412000528`. pages 34

[16] Antonio Rafael Sabino Parmezan, Vinicius M.A. Souza, and Gustavo E.A.P.A. Batista. Evaluation of statistical and machine learning models for time series prediction: Identifying the state-of-the-art and the best conditions for the use of each model. **Information Sciences**, 484:302 – 337, 2019. ISSN 0020-0255. doi: https://doi.org/10.1016/j.ins.2019.01.076. URL `http://www.sciencedirect.com/science/article/pii/S0020025519300945`. pages 35

[17] Yu Zheng. Trajectory data mining: An overview. **ACM Trans. Intell. Syst. Technol.**, 6(3), May 2015. ISSN 2157-6904. doi: 10.1145/2743025. URL `https://doi.org/10.1145/2743025`. pages 35

[18] Hossein Abbasimehr, Mostafa Shabani, and Mohsen Yousefi. An optimized model using lstm network for demand forecasting. **Computers & Industrial**

Engineering, 143:106435, 2020. ISSN 0360-8352. doi: https://doi.org/10.1016/j.cie.2020.106435. URL `http://www.sciencedirect.com/science/article/pii/S0360835220301698`. pages 35

[19] Omer Berat Sezer, Mehmet Ugur Gudelek, and Ahmet Murat Ozbayoglu. Financial time series forecasting with deep learning : A systematic literature review: 2005–2019. **Applied Soft Computing**, 90:106181, 2020. ISSN 1568-4946. doi: https://doi.org/10.1016/j.asoc.2020.106181. URL `http://www.sciencedirect.com/science/article/pii/S1568494620301216`. pages 35

[20] Bing Zhang, Jhen-Long Wu, and Pei-Chann Chang. A multiple time series-based recurrent neural network for short-term load forecasting. **Soft Computing**, 22:4099 – 4112, 2018. doi: https://doi.org/10.1007/s00500-017-2624-5. URL `https://link.springer.com/article/10.1007%2Fs00500-017-2624-5`. pages 35

[21] Tangbin Xia, Ya Song, Yu Zheng, Ershun Pan, and Lifeng Xi. An ensemble framework based on convolutional bi-directional lstm with multiple time windows for remaining useful life estimation. **Computers in Industry**, 115: 103182, 2020. ISSN 0166-3615. doi: https://doi.org/10.1016/j.compind.2019.103182. URL `http://www.sciencedirect.com/science/article/pii/S0166361519303987`. pages 35, 36, 37, 66

[22] Shuja ur Rehman Baig, Waheed Iqbal, Josep Lluis Berral, and David Carrera. Adaptive sliding windows for improved estimation of data center resource utilization. **Future Generation Computer Systems**, 104:212 – 224, 2020. ISSN 0167-739X. doi: https://doi.org/10.1016/j.future.2019.10.026. URL `http://www.sciencedirect.com/science/article/pii/S0167739X19309203`. pages 36

[23] Shengdong Du, Tianrui Li, Yan Yang, and Shi-Jinn Horng. Multivariate time series forecasting via attention-based encoder–decoder framework. **Neurocomputing**, 388:269 – 279, 2020. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2019.12.118. URL `http://www.sciencedirect.com/science/article/pii/S0925231220300606`. pages 36

[24] Kasun Bandara, Christoph Bergmeir, and Slawek Smyl. Forecasting across time series databases using recurrent neural networks on groups of similar series: A clustering approach. **Expert Systems with Applications**, 140: 112896, 2020. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2019.112896. URL `http:`

//www.sciencedirect.com/science/article/pii/S0957417419306128. pages 36

[25] Cristóbal Esteban, Stephanie L. Hyland, and Gunnar Rätsch. Real-valued (medical) time series generation with recurrent conditional gans, 2017. pages 38, 103

[26] Kay Gregor Hartmann, Robin Tibor Schirrmeister, and Tonio Ball. Eeg-gan: Generative adversarial networks for electroencephalograhic (eeg) brain signals, 2018. pages 38, 103

[27] Mohammad Navid Fekri, Ananda Mohon Ghosh, and Katarina Grolinger. Generating energy data for machine learning with recurrent generative adversarial networks. **Energies**, 13(1):130, Dec 2019. ISSN 1996-1073. doi: 10.3390/en13010130. URL http://dx.doi.org/10.3390/en13010130. pages 38

[28] Dan Li, Dacheng Chen, Jonathan Goh, and See kiong Ng. Anomaly detection with generative adversarial networks for multivariate time series, 2018. pages 38

[29] Alireza Koochali, Peter Schichtel, Andreas Dengel, and Sheraz Ahmed. Probabilistic forecasting of sensory data with generative adversarial networks – forgan. **IEEE Access**, 7:63868–63880, 2019. ISSN 2169-3536. doi: 10.1109/access.2019.2915544. URL http://dx.doi.org/10.1109/ACCESS.2019.2915544. pages 38, 39, 40, 42, 103

[30] A. Rasheed, O. San, and T. Kvamsdal. Digital twin: Values, challenges and enablers from a modeling perspective. **IEEE Access**, 8:21980–22012, 2020. pages 41, 103

[31] Eric J. Parish and Kevin T. Carlberg. Time-series machine-learning error models for approximate solutions to parameterized dynamical systems. **Computer Methods in Applied Mechanics and Engineering**, 365:112990, 2020. ISSN 0045-7825. doi: https://doi.org/10.1016/j.cma.2020.112990. URL http://www.sciencedirect.com/science/article/pii/S0045782520301742. pages 41, 103

[32] Zhe Wu, David Rincon, and Panagiotis D. Christofides. Process structure-based recurrent neural network modeling for model predictive control of nonlinear processes. **Journal of Process Control**, 89:74 – 84, 2020. ISSN 0959-1524. doi: https://doi.org/10.1016/j.jprocont.2020.03.013. URL http://www.sciencedirect.com/science/article/pii/S095915241930825X. pages 41

[33] N. Mohajerin and S. L. Waslander. Multistep prediction of dynamic systems with recurrent neural networks. **IEEE Transactions on Neural Networks**

and Learning Systems, 30(11):3370–3383, Nov 2019. ISSN 2162-2388. doi: 10.1109/TNNLS.2019.2891257. pages 41, 103

[34] Sachin S. Talathi and Aniket Vartak. Improving performance of recurrent neural network with relu nonlinearity, 2015. pages 43, 69, 80, 84, 99

[35] Tensorflow timeseries generator, accessed on 20.08.2020. URL `https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/sequence/TimeseriesGenerator`. pages 56

[36] Apparent wind angle wikipedia page, accessed on 27.07.2020. URL `https://rb.gy/jnjffv`. pages 60

[37] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. **Neural Computation**, 9(8):1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735. URL `https://doi.org/10.1162/neco.1997.9.8.1735`. pages 75

[38] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador Garcia, Sergio Gil-Lopez, Daniel Molina, Richard Benjamins, Raja Chatila, and Francisco Herrera. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. **Information Fusion**, 58:82 – 115, 2020. ISSN 1566-2535. doi: https://doi.org/10.1016/j.inffus.2019.12.012. URL `http://www.sciencedirect.com/science/article/pii/S1566253519308103`. pages 103, 104

[39] Shap library, accessed on 26.05.2020. URL `https://shap.readthedocs.io/en/latest/`. pages 104

[40] Google explainable ai framework, documentation webpage, accessed on 28.08.2020. URL `https://cloud.google.com/explainable-ai`. pages 104

[41] Guide to the general data protection regulation (gdpr), accessed 22.05.2020. URL `https://ico.org.uk/for-organisations/guide-to-data-protection/guide-to-the-general-data-protection-regulation-gdpr/controllers-and-processors/what-does-it-mean-if-you-are-a-controller/`. pages 106

[42] Open ai gym, accessed on 24.05.2020. URL `https://gym.openai.com/`. pages 111