

FLORIAN SCHOPP

**MOORING LINE FAILURE DETECTION OF
FLOATING PLATFORMS USING MOTION
PREDICTION BASED ON NEURAL NETWORKS**

São Paulo
2020

FLORIAN SCHOPP

**MOORING LINE FAILURE DETECTION OF
FLOATING PLATFORMS USING MOTION
PREDICTION BASED ON NEURAL NETWORKS**

Trabalho apresentado à Escola Politécnica
da Universidade de São Paulo para obtenção
do Título de Engenheiro Eletricista com
ênfase em Sistemas Eletrônicos.

São Paulo
2020

FLORIAN SCHOPP

**MOORING LINE FAILURE DETECTION OF
FLOATING PLATFORMS USING MOTION
PREDICTION BASED ON NEURAL NETWORKS**

Trabalho apresentado à Escola Politécnica
da Universidade de São Paulo para obtenção
do Título de Engenheiro Eletricista com
ênfase em Sistemas Eletrônicos.

Área de Concentração:

Electrical Engineering

Orientador:

Anna Helena Reali Costa

São Paulo
2020

ACKNOWLEDGMENTS

I would like to thank all those who supported me during my graduation here in Brazil, helping me to improve my Portuguese, explaining the exercises to me again when the language failed, and supporting me during the pandemic. In particular, my tutor Dr. Anna Reali, who made this work possible by offering me this topic and helping me not only during the implementation process but also to improve my Portuguese. Special thanks also goes to Amir Saad, whose support during his doctoral thesis enabled me to work efficiently under his supervision. Furthermore I am grateful for the help of all the members of the Digital Twins research team, who took the time to answer my questions and helped me with the structuring of the project.

Finally I would also like to thank the company Petrobras for opening up this research opportunity in an area where there is still a lot of research to be done and a lot of room for improvement. Special thanks goes to the FUSP for the financial support in form of a scholarship, that gave me the chance to focus more on the project.

ABSTRACT

Offshore platform oil extraction opens up new promising possibilities of extending the life span and usage of the scarce resource. With the new technology new problems of platform surveillance and security arise. The mooring systems give stability to the floating offshore platforms against environmental conditions, stabilizing the platform with mooring lines attached to the seabed. This work focuses on a novel approach using Neural Networks to predict the platform motion in different environmental conditions. Based on the predictions, different classifiers are then developed to detect mooring line failure in real time. The project helps in reducing the costs of maintenance and increases the security of the platforms.

Keywords – Neural networks, RNN, LSTM, offshore platform, mooring line, FPSO.

LIST OF FIGURES

1	Offshore risks of incidents (extracted from [1]).	18
2	Proposed general architecture: mooring line failure detection is based on the short-term prediction of platform motion.	21
3	Pipeline of the motion prediction approach to detect platform line failures.	25
4	Integrated proposed architecture consisting of three models. The predictor for predicting future platform motion, the comparator for calculating the difference between prediction and simulation and the classifier for identifying mooring line failure	25
5	The basic input and output structure of the neural network predicting the platform motion based on the past simulated platform motion	26
6	The basic input and output structure of the comparator calculating different error scores based on the difference between simulation and prediction .	27
7	The basic input and output structure of the classifier identifying mooring line failures based on different error scores calculated by the comparator . .	28
8	The figure shows the predictor module as part of the modular system structure	29
9	Generation of training and test data: Initially real data of environmental conditions are analyzed and sampled, feeding the simulator that then generates motion data for the specified platform.	30
10	Histograms showing the distribution of environmental variables recorded by a meteorological station located in the Campos Basin, RJ, with velocities split into its x and y components. The data are considered to follow a normal distribution, represented by a black line drawn on each histogram. .	32
11	Histogram of random selected cases used for training the predictors.	35
12	Scaled closest real life cases that are selected for training.	36
13	Training unit of a data window of 500 time steps, with 400 time steps being input (in blue) and 100 time steps being output (in orange).	37

14	Single Platform Motion Prediction of a predictor. Top: simulated platform motion. Middle: a test unit input (in blue) and a a test unit output (in orange). Bottom: the prediction (in orange) of a trained predictor for the test unit input (in blue).	38
15	Prediction Windows: the top graph shows a test with input from 0s to 400s (in blue) and output from 400s to 500s (in orange); the process is repeated in the subsequent graphs, with input from 100s to 500s and output from 500s to 600s and input from 200s to 600s, with output from 600 to 700s. The graph below shows the concatenation of the 3 predictions (from 400S to 700s) in orange, superimposed on the simulated curve (in blue).	39
16	Illustration of the histograms of the selected 6000 files for the MLP network (orange) against the histogram of all original 18000 files (blue).	40
17	Example of the MLP training curve of 5000 training environmental conditions (blue) and 1000 environmental conditions for validation (orange). The vertical axis represents the error and the horizontal axis represents the number of epochs. The error score decreases with an increasing number of epochs.	42
18	The MLP model has as input 600 seconds of time series of the features sway, surge, heave, roll, pitch, and yaw of the platform motion, and outputs 100 seconds of prediction of the three features sway, surge, and yaw.	43
19	The fully connected Multi-Layer Perceptron (MLP) architecture used, with 3600 input nodes, 300 output nodes, and three hidden layers with 7200, 3600, and 1800 nodes, respectively.	44
20	MLP prediction on a calm environmental condition with all mooring lines intact. Local x represents surge, local y represents sway, and $\sin(Z)$ represents yaw motion of the platform. The orange line is the MLP prediction and the blue line is the simulated platform motion.	45
21	A zoomed version of the MLP prediction on a calm environmental condition. Local x represents surge, local y represents sway, and $\sin(Z)$ represents yaw motion of the platform. The orange line is the MLP prediction and the blue line is the simulated platform motion.	46

22	Illustration of MLP prediction on a mild environmental condition. Local x represents surge, local y represents sway, and $\sin(Z)$ represents yaw motion of the platform. The orange line is the MLP prediction and the blue line is the simulated platform motion.	47
23	A zoomed illustration of MLP prediction with a mild environmental condition. Local x represents surge, local y represents sway, and $\sin(Z)$ represents yaw motion of the platform. The orange line is the MLP prediction and the blue line is the simulated platform motion.	47
24	Illustration of MLP prediction on a stormy environmental condition with all mooring lines intact. Local x represents surge, local y represents sway, and $\sin(Z)$ represents yaw motion of the platform. The orange line is the MLP prediction and the blue line is the simulated platform motion.	48
25	A zoomed illustration of MLP prediction on a stormy environmental condition with all mooring lines intact. Local x represents surge, local y represents sway, and $\sin(Z)$ represents yaw motion of the platform. The orange line is the MLP prediction and the blue line is the simulated platform motion.	48
26	Illustration of mooring line failure of L1 at approximately time step 5000. Local x represents surge, local y represents sway, and $\sin(Z)$ represents yaw motion of the platform. The orange line is the MLP prediction and the blue line is the simulated platform motion.	49
27	A zoomed illustration of Mooring line failure of line one. Local x represents surge, local y represents sway, and $\sin(Z)$ represents yaw motion of the platform. The orange line is the MLP prediction and the blue line is the simulated platform motion.	49
28	An illustration of mooring line failure of L9. Local x represents surge, local y represents sway, and $\sin(Z)$ represents yaw motion of the platform. The orange line is the MLP prediction and the blue line is the simulated platform motion.	50
29	Illustration of mooring line failure of L12 and L18 at approximately time step 5000. Local x represents surge, local y represents sway, and $\sin(Z)$ represents yaw motion of the platform. The orange line is the MLP prediction and the blue line is the simulated platform motion.	50

30	Illustration of the selected files histograms (orange) against the histogram of all 18000 files (blue) for the Long Short Term Memory (LSTM) network.	52
31	Illustration of the used LSTM architecture. In brackets are the number of training units, due to its variable size (depending on the number of selected environmental conditions) it is marked with "?", the second number represents the number of input steps and the last number the number of features	53
32	Training Curve of the first 700 epochs of the presented LSTM network using 1000 training environmental conditions (blue) and 200 validation environmental conditions (orange). The vertical axis represents the mean error and the horizontal axis represents the number of epochs. The error decreases steadily with number of epochs.	54
33	The LSTM model using the last 1000 seconds of the features surge, sway and yaw to predict 400 seconds of these three features.	55
34	Illustration of LSTM prediction of platform motion under a single environmental condition with all mooring lines intact. The simulated data are in blue and the predicted in green. Local x means surge, local y sway, and sin(Z) yaw.	55
35	Illustration of LSTM prediction on a single environmental condition with all mooring lines intact. The simulated data are in blue and the predicted in green.	56
36	Zoomed illustration of the LSTM prediction on a single environmental condition with all mooring lines intact. The simulated data are in blue and the predicted in green, with the topmost graph representing the Local x position (surge), the middle graph local y position (sway), and the bottom graph the sin(Z) angle (yaw).	57
37	Zoomed illustration of LSTM prediction on a single environmental condition with all mooring lines intact and rapid motion. The simulated data are in blue and the predicted in green, with the topmost graph representing the Local x position (surge), the middle graph local y position (sway), and the bottom graph the sin(Z) angle (yaw).	58

38	Illustration of LSTM prediction on a single environmental condition with a Mooring Line failure of Line 1 at 3500 seconds. The topmost graph representing the Local x (surge), the middle graph representing local y (sway), and the graph at the bottom representing $\sin(Z)$ (yaw).	59
39	Zoomed illustration of LSTM prediction on a single environmental condition with a Line failure of Mooring Line 1 at 3500 seconds. The topmost graph representing the Local x (surge), the middle graph representing local y (sway), and the graph at the bottom representing $\sin(Z)$ (yaw).	59
40	Illustration of different Mooring Line failures in different environmental conditions	60
41	Figure showing the comparator module as part of the modular system structure	63
42	In this plot are the actual motion (simulated data here, in blue), the predicted values (in green) and the difference between them (in red).	64
43	Prediction step by step. Each graph shows two 50s predictions concatenated in a 100s error window, and the respective difference (in green) between the predictions (in red) and the actual data (dashed in blue). The 50 second stride between one graph and another can be seen as time difference between the graphs.	65
44	Error Calculation. Top: predicted (orange) and actual (blue) motion values. Bottom: RMSE (green) median (orange) and mean(blue) error scores for each error window.	66
45	RMSE error score of features; surge, sway and yaw indexes for all environmental conditions in the test set. The red circles represent cases without mooring line failure and the blue circles represent cases with a mooring line failures. The x axis represents the error score for surge, the y axis represents sway and the z axis represents the error score for the yaw feature	69
46	Mean error score of features; surge, sway and yaw of indexes for all environmental conditions. The red circles represent cases without mooring line failure, and the blue circles represent cases with mooring line failure. The x axis represents the error score for surge, the y axis represents sway and the z axis represents the error score for the yaw feature	70

47	Median error score of features; surge, sway and yaw of indexes for all environmental conditions. The red circles represent cases without mooring line failure, and the blue circles represent cases with mooring line failure. The x axis represents the error score for surge, the y axis represents sway and the z axis represents the error score for the yaw feature	71
48	Scatter plot of RMSE error score of features; The red circles represent cases without mooring line failures, and the blue circles represent cases with a mooring line failure. The x axis represents the error score for surge, the y axis represents sway and the z axis represents the error score for the yaw feature.	74
49	Scatter plot of Mean error score of features. The red circles represent cases without mooring line failure, and the blue circles represent cases with a mooring line failure. The x axis represents the error score for surge, the y axis represents sway and the z axis represents the error score for the yaw feature.	75
50	Scatter plot of Median error score of features; surge, sway and yaw of indexes for all environmental conditions. The red circles represent cases without mooring line failure, and the blue circles represent cases with a mooring line failure. The x axis represents the error score for surge, the y axis represents sway and the z axis represents the error score for the yaw feature.	76
51	Figure showing the classifier module as part of the modular system structure	78
52	Types of classifiers used.	78
53	K-nearest neighbour (KNN) classifier prediction on mooring line status using MLP predictor.	83
54	Decision Tree (DT) classifier prediction on mooring line status using MLP predictor.	85
55	Support Vector Classifier (SVC) prediction on mooring line status using MLP predictor.	86
56	K-nearest neighbour (KNN) prediction on mooring line status using LSTM predictor.	89

57	Decision Tree (DT) classifier prediction on mooring line status using LSTM predictor	91
58	Support vector classifier (SVC) prediction on mooring line status using LSTM predictor.	93
59	A single unit of a perceptron.	101
60	Sigmoid function plot.	102
61	ReLU function plot.	103
62	MLP	105
63	Basic RNN unit	107
64	A recurrent neural network (RNN)	108
65	long short term memory (LSTM) diagram	109
66	Cell state of a LSTM Cell	110
67	Forget gate of a LSTM Cell	111
68	Update gate of a LSTM Cell	112
69	Output gate of a LSTM Cell	112
70	Decision Tree structure	114
71	Binary linear Support Vector Machine. There are two classes of training observations, presented in black and gray. The hyper-plane is presented as a solid line separating the classes. Observations on the dotted lines (unfilled circles) are the support vectors.	117
72	Dynasim Interface source: User manual- Dynasim	119
73	platform Co-ordinate System Source: User manual- Dynasim	120
74	The main repository and a new branch [2].	126
75	The pull request concept on the <i>feature branch</i> git workflow [2].	126
76	Containerized applications.	127

LIST OF TABLES

1	Example of environmental conditions measurements	31
2	Necessary conditions with velocities split into its x and y components . . .	31
3	Environmental condition measurements scaled between 0 and 1	33
4	Randomly created scenarios that follow the normal distribution	33
5	Randomly created scenarios that follow the normal distribution, scaled between 0 and 1	34
6	Found real life scenarios that match the random scenario	34
7	Mean values and standard deviations of the variables of all cases and se- lected cases	35
8	Different MLP architecture compositions with 3600 input and 300 output nodes, and their respective error score.	43
9	3 environmental conditions selected.	44
10	Analyzed environmental conditions	56
11	RMSE MLP error scores	67
12	Median MLP error scores	68
13	Mean MLP error scores	68
14	LSTM: RMSE Error scores	72
15	LSTM: Mean error scores	72
16	LSTM: Median error scores	73
17	All simulated platform motions from Dynasim. Dates indicate which peri- ods of actual environmental conditions were used to generate the simulated data.	79
18	Balanced simulated platform motions	80
19	Example of Training data for the binary classification	81
20	Example of Test data for the binary classification	81

21	K-nearest neighbour (KNN) classifier metrics	82
22	Environmental condition of the 10 platform motions, the K-nearest neighbour (KNN) classifier classified wrongly.	84
23	Decision Tree (DT) classifier metrics	84
24	Environmental condition of the 8 platform motions the decision tree (DT) classifier classified wrongly.	85
25	Support vector classifier (SVC) metrics	87
26	Environmental condition of the 28 platform motions the Support Vector Classifier (SVC) classified wrongly.	88
27	Error rating of K-nearest neighbour (KNN) classifier based on LSTM predictor	90
28	Environmental condition of the 9 platform motions the K-nearest neighbour (KNN) classifier classified wrongly	90
29	Error rating of Decision Tree (DT) classifier based on LSTM predictor. . .	91
30	Environmental conditions of the 9 platform motions the Decision Tree (DT) classifier wrongly	92
31	Error rating of support vector classifier (SVC) based on LSTM predictor .	92
32	Environmental condition of the 12 platform motions the support vector classifier (SVC) classified wrongly.	94
33	Confusion Matrix for two classes	117
34	platform dimension and setup	121
35	Unscaled Dataset	124
36	Scaled Dataset	124

CONTENTS

1	Introduction	17
1.1	Detecting Mooring Failure	19
1.2	Objectives	20
1.3	Organization of the monography	21
2	Literature Review	22
3	Proposal	24
3.1	Motion Predictor	26
3.2	Comparator	27
3.3	Classifier	27
4	Predictor	29
4.1	Data preparation	29
4.1.1	Environmental condition selection	30
4.1.2	Training set creation	36
4.1.3	Testing set creation	38
4.2	MLP Predictor	39
4.2.1	Environmental condition selection	40
4.2.2	MLP Training	41
4.2.3	Model architecture	42
4.2.4	MLP Prediction	44
4.3	LSTM Predictor	51
4.3.1	Environmental condition selection	51
4.3.2	Model architecture	51

4.3.3	LSTM Training	54
4.3.4	LSTM Prediction	55
4.4	Discussion	61
5	Comparator	63
5.1	Error Score Calculation	64
5.1.1	Scatter plot visualization	66
5.2	MLP Error Scores	67
5.3	LSTM error Scores	72
5.4	Discussion	77
6	Classifier	78
6.1	Training set balancing	79
6.2	MLP classifier results	82
6.2.1	K-nearest neighbour (KNN) classifier result	82
6.2.2	Decision Tree classifier result	83
6.2.3	Support Vector Classifier (SVC) result	86
6.3	LSTM classifier results	88
6.3.1	K-nearest neighbour classifier results	89
6.3.2	Decision Tree classifier results	90
6.3.3	Support Vector Classifier (SVC) results	92
6.4	Discussion	94
7	Conclusion	97
	Appendices	99
A	Appendix	100
A.1	Machine Learning	100

A.1.1	Neural Network Principles	101
A.1.2	Multi- Layer perceptron	104
A.1.3	Recurrent Neural Networks	107
A.1.4	Long Short Term Memory	109
	Cell state	109
A.1.5	Classifiers	113
	K-nearest neighbor (KNN) classifier	113
	Decision tree (DT) classifier	113
	Support Vector Classifier (SVC)	115
A.1.6	Confusion matrix	117
A.2	Dynasim simulator	119
A.2.1	Coordinate systems	119
A.2.2	Platform model	121
A.2.3	Environmental Conditions	121
	Waves	121
	Wind	122
	Current	122
A.2.4	File Structure	123
A.3	Standardization of Measurements	123
A.4	Specifications	125
A.4.1	Git	125
A.4.1.1	Branch Workflow	125
A.4.1.2	Continuous integration	126
A.4.2	Containerization	127
A.4.3	Software specifications	128
A.4.3.1	Python and libraries	128

A.4.3.2 Website	129
A.4.4 System Specification	129
References	130

1 INTRODUCTION

Demand for alternative energy sources increased in the early 20th century as the world shifted from coal energy to hydrocarbon energy (crude oil). To accommodate this increasing demand the oil industries began finding ways to increase the production of crude oil both onshore and offshore. The discovery of large oil fields offshore presented the industry the challenge to safely and efficiently explore and extract the resources found. Floating platforms proved to be a viable solution [3]. Over the years various floating platforms have been developed, among them mobile offshore drilling units (MODU), floating production storage and offloading (FPSO) units, semi-submersibles platforms and tension leg platforms (TLP) [4]. These platforms differ in size, application, load holding capacity and water depth rating. The need of positioning these platforms in the desired location proved to be the most critical challenge. This marked the advent of mooring systems [3].

Mooring systems are used to provide stability to floating platforms against environmental conditions such as waves, currents and winds by anchoring the platform to the seabed at the desired location.

Early mooring systems were composed of ropes, chains and anchors. Over the years, mooring systems have evolved due to advancement in technologies and better materials, however it still follows the basic concept of the early mooring system composition. Mooring systems are one of the key components to ensure safety for the staff and the various operations carried out on the platforms, like drilling, production, and offloading. As these platforms venture into deeper water depth, the need for increased monitoring of the platform and its mooring system structure and integrity becomes critical. In Figure 1 the correlation between the risk of mooring system failure and water depth is represented graphically. It can be seen that the incident probability increases linearly with an increasing water depth.

There are two types of mooring systems: disconnectable mooring systems and permanently connected mooring systems [5]. Disconnectable mooring systems are used in regions with cyclic loading environments, i.e. regions that experience repeated fluctuations of the

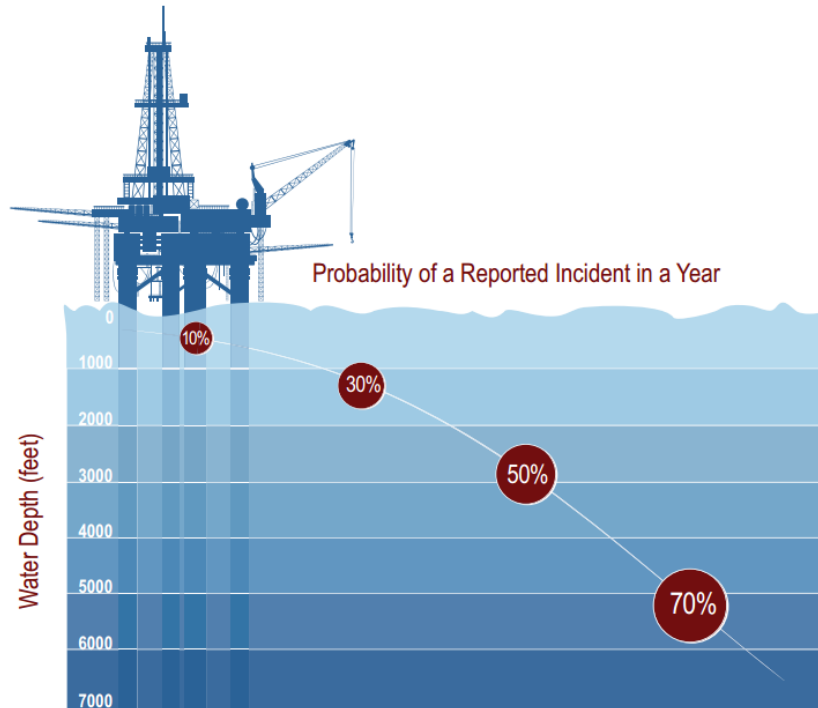


Figure 1: Offshore risks of incidents (extracted from [1]).

sea states: wind, wave and current. They are used to facilitate easy disconnection of the platform in the event of extreme weather conditions such as cyclones or other emergencies. These repeated sea state fluctuations increase the stress and strain exerted on offshore structural components and increase the probability of a component failure. The disconnectable mooring system usually consists of a turret with a detachable buoy (DTM buoy) which can be detached from a floating production unit (FPU) turret component and reattached when needed [6]. A turret is a device built directly into platforms for weather-vanning and anchoring the platform at a particular location. Weather-vanning refers to the ability of a platform to rotate freely in the direction of the environment disturbance [7].

Permanently connected mooring systems are used in regions where sea state fluctuations do not occur frequently. The stresses and strains caused by the sea states on the structural component of platforms are decreased. Most permanently connected FPSOs have a spread mooring configuration made up of four or more mooring lines [8].

Studies described in [9] and [10] have shown around 45% of mooring failures are either single line failures or multiple line failures which can be attributed to corrosion and fatigue. In some cases of single mooring line failure, additional mooring lines can be damaged due to increased load, stress and tension experienced by the remaining lines, since single line failure inadvertently increases the degradation rate of the mooring lines [9]. A report

in [11] indicated that 50% of the offshore platforms in the North Sea cannot monitor their mooring system in real-time, 33% of the platforms cannot measure offset from the no-load equilibrium, and 78% of the platforms lack a system that alerts in the event of a line failure. Therefore, when a mooring system gets compromised it can go unnoticed for a long period of time.

The company Petrobras has 40 proprietary platforms installed, which encompass 590 mooring lines. In the failure history, it was reported that the majority of failures (87%) occurs after 10 years of platform operation. This lifetime limit may increase, as the newer units have better designs that consider more fatigue factors and therefore extend the lines life expectancy. Most failures occur at the top of the line, where traction is greater, conditions are more severe, and corrosion is more pronounced.

Furthermore, failure of a mooring system can result in damage or loss of property, environmental pollution, personnel endangerment and depending on the severity of failure, in some cases oil production shutdown.

1.1 Detecting Mooring Failure

To address the occurrence of mooring failure, regular inspection of mooring systems is carried out using technologies like inclinometers, micro-remote operated vehicles (ROV), load shackles, visual inspections, positioning systems, and integrated monitoring systems.

Inclinometers are fitted on mooring lines to measure the mooring line angles. In a calm sea state mooring angles are measured and compared against mooring angles after turbulent weather has passed. Significant changes between the two measured angles indicate the possibility of a failure in a mooring line [12].

Load shackle with load cells are connectors used to link mooring lines. These load shackles monitor mooring line tension in real-time with the use of load cells [13]. There are different types of load shackles, for example wireless and traditional load shackles. Wireless load shackles require battery replacement at fixed intervals which serve as the energy source in deep waters. Both traditional and wireless shackles are affected by marine growth on mooring lines which occur naturally. Marine growth affects the tension measurement of the mooring line monitored by the load cell. Removal of marine growth is expensive and time consuming.

Micro-ROVs are also deployed to check the Mooring Lines when the inclinometer reading shows an offset. The use of micro-ROVs to inspect anchor lines helps eliminating

the need for full-size, expensive ROVs, or even offshore divers. The portability of the micro-ROVs makes it possible to store them on the platform or send them by helicopter, if needed.

These fault monitoring methods by ROVs or divers in which inspections are carried out at a fixed time interval have the disadvantage of not addressing the problem of real-time monitoring [14]. Failures of mooring components can occur after the inspection has been conducted, increasing the chance of detecting mooring system failures late.

Position monitoring of a platform can be achieved by using orbital satellites such as global position system (GPS). The platform is continuously monitored because the navigation system on board of the platform coupled with a computer is in constant communication with GPS satellites. The navigation systems periodically send the longitudinal and horizontal coordinates of the platform. Platforms drifting away from the perimeter of the watch circle most likely suffered mooring line failure [12].

Methods based on the installation of sensors on the lines – such as inclinometers and load shackles – are expensive and inefficient and require maintenance.

Recent improvements in technology present a solution for real-time monitoring and mooring system failure detection . Machine learning, a subgroup of artificial intelligence in which models are trained on a data driven approach without explicit instruction being provided, is a good alternative to be used for monitoring and mooring line failure detection. Data from sensors and inclinometers reading from the FPSO, GPS data, and weather data can be used to to implement machine learning algorithms for monitoring and detecting mooring line failures.

1.2 Objectives

Our proposal in this monography is to use an architecture as shown in Figure 2, with a predictor module that estimates the future platform motion based on the previous motion data of the platform. This predictor is a neural network trained with data that indicates motions without line breakage. The idea is that the predictor makes the future prediction of the motion considering the absence of failures, and that the sensors measure the actual motion performed by the platform. If there is a significant difference between the predicted and the measured value, this is notified by the status of the mooring line, which then indicates a line failure. We present here the development and comparison of two ML models based on neural networks for the predictor module, a Multi-layer

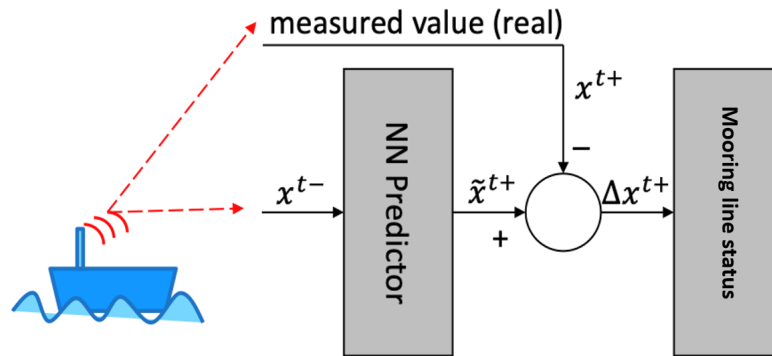


Figure 2: Proposed general architecture: mooring line failure detection is based on the short-term prediction of platform motion.

Perceptron (MLP) model and a Long Short Term Memory (LSTM) model, both capable of identifying when a mooring system of a platform is compromised.

1.3 Organization of the monography

This monography is structured as follows: Section 2 introduces related work. Section 3 explains in detail our proposal, while Section 4 details the predictor and its performance, Section 5 describes the comparator module and its results, and Section 6 describes the classifier and its performance in the experiments. Finally, Section 7 presents our conclusions and highlights future work.

2 LITERATURE REVIEW

Researchers in diverse domains are increasingly adopting machine learning (ML) techniques. The same approaches can also be adopted in the domain of mooring failure detection. In the following, we highlight articles which used ML techniques to monitor mooring system states.

Different input features can be used to classify Mooring Line system states. Frequency spectrum and autocorrelation functions can be extracted from the motion time series of the platform to determine mooring states shown by Tang [15]. The authors used an SVM algorithm to classify the status of a soft yoke single point mooring tower system. The SVM input was based on frequency spectrum and autocorrelation functions to determine failure of the mooring system. In conclusion the authors showed that frequency spectrum as well as autocorrelation functions were good indicators for mooring failure detection.

Prislin [16] proposed a novel concept with regards to the integrity of the mooring system. They implemented a system named Position Response Learning System (PRLS) that at its core uses multilayered perceptron. The PRLS system used Dynamic Global Position System (DGPS), meta-ocean data (waves, current, and wind), and inertial motion of the vessel (six degrees of freedom, 6DOF, see appendix A.2 for more information) as input features. The proposed system was able to provide two forms of output: classification-based output and regression-based output.

A MLP network in combination with kriging methods was used by Gumley [17] to predict when changes in mooring state occur. The models used meta-ocean data and GPS data as input to their models. The result showed both methods performed well in detecting changes in the mooring state. The output information for both methods was binary classification. Sidarta [18] also used a MLP model to detect mooring line failure by training the model to distinguish the normal drift period from damaged drift readings of the moored platform. The MLP was trained on simulated data from an in-house numerical simulation software named MLTSIM. Inputs to the MLP were GPS readings of the platform and the total mass of the FPSO platform. Furthermore, in their paper [19],

an MLP model was used to predict mooring line tension. 60-second time series of platform movements with one-second intervals were used as input features for the MLP to forecast 30 seconds of the platform mooring line tension.

A convolutional network (CNN) implemented by Jaiswal [20] could identify the horizontal position features of the platform and associate them with the mooring system states in different environmental conditions. Positional Data and obtained vessel motion were encoded in a single labeled image using a proprietary algorithm and were used as input to the CNN model. The image contains the statistics of the horizontal position parameters of the vessel and root mean square (RMS) values of the 6-DOF acceleration. The Output of the model was the status of the mooring lines.

Sireta [21] developed two machine learning models; a MLP and LSTM model, capable of detecting occurrences of mooring system failure. Two approaches were adopted to investigate which approach gave the best prediction results. These approaches were cross-correlation and auto-correlation. For the cross-correlation approach one degree of platform motion was used to predict the next time step of the same degree .i.e sway. For the auto-correlation approach 5 degrees of freedom (5 DOF) of the platform motion .i.e. surge, yaw, pitch, roll and heave out of the 6 DOF were used as input to predict sway motion of the platform, the remaining one degree (1 DOF) in this paper. Both models were trained using numerically simulated data and the trained model was given cases in which one mooring line breakage was presented for the prediction. RMSE error values between that of a failed mooring system and intact mooring system were monitored. Results of the work concluded both models were good at detecting mooring failure.

Machine learning algorithms used in the offshore sector include multilayer perceptron (MLP), convolutional neural network (CNN), support vector machine (SVM) and long short term memory (LSTM). Since most ML studies in the literature used MLP [16–18, 22], this project investigates its limitations and capabilities. Considering the time series nature of the problem, a type of network widely used in other areas are the recurrent neural networks, more specifically LSTM networks, which are also investigated here (see Appendix A.1.4 for details). MLP as well as LSTM networks were implemented and analyzed in the present project, evaluating their efficiency and effectiveness in detecting mooring line failure.

3 PROPOSAL

As it was outlined in the introduction, mooring systems are a safety critical component of floating structures, since a system failure can lead to platform drifting and, in extreme cases lead to oil spillage. This section proposes a solution which makes it possible to monitor mooring systems and detect failure without the necessity of preventive inspections.

The study is based on the assumption that there exists a regular motion pattern, that can be used to identify mooring line failures, which results in abnormalities within this pattern. The regular motion pattern makes it possible to predict future platform motions based on previous motions. Abnormalities can then be identified by calculating the difference between predicted and simulated motions.

The study presented in this monography is intended as a proof of concept using simulated data for binary mooring line failure identification. In some other phase of the project to be developed in the future by other researchers, the classification module will ideally be able to identify the fault line based on the actual readings of the GPS sensors and of the inertial measurement unit (IMU) of the platform.

Figure 3 shows the necessary steps for the approach, that also form the outline of the project. The first step is the selection of environmental conditions best representing the entirety of all measured conditions. These conditions are then simulated by the Dynasim simulator (see Appendix A.2) creating platform motions based on the environmental condition and the platform model. By using these simulations predictor models can then be developed to predict the platform motions using past platform motions as input. Based on the difference between simulated and predicted motions error scores can be calculated representing the disparity between them. Finally binary classifiers can be trained using the error scores as basis to identify mooring line failure occurrences.

The system can therefore be separated into 3 modules as illustrated in figure 4. A combination of Dynamic Global Position System (DGPS) and IMU measurements (surge,

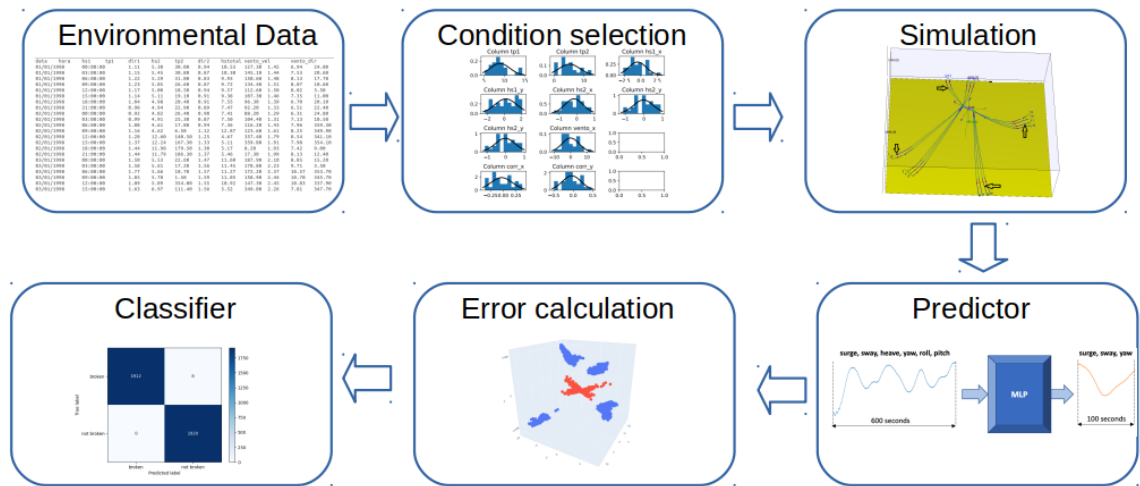


Figure 3: Pipeline of the motion prediction approach to detect platform line failures.

sway, heave, roll, pitch, and yaw) gives 6 degrees of freedom (6DoF) of the platform motion and will be used as inputs to the proposed system. The system output is a binary identification of mooring line failures.

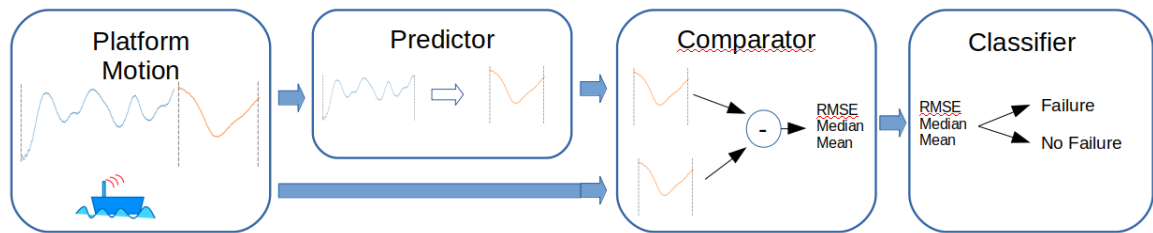


Figure 4: Integrated proposed architecture consisting of three models. The predictor for predicting future platform motion, the comparator for calculating the difference between prediction and simulation and the classifier for identifying mooring line failure

The three modules are:

1. Predictor: A neural network predicting the platform motion based on the past simulated platform motion. The system is presented in the chapter 4.
2. Comparator: A module to calculate error scores based on the difference between the prediction and recorded data as measure of disparity . The module is presented in chapter 5.
3. Classifier: Module for mooring line failure identification based on the calculated error scores of the comparator. The module is presented in chapter 6.

The following section emphasize on the individual modules.

3.1 Motion Predictor

Two neural networks are developed, a feed-forward multi-layer perceptron network (MLP) and a long short term memory recurrent network (LSTM), for simplicity here we refer to both models as predictors. These predictors are used to predict the platform motion by using the last two to six minutes of previous platform motion. The basic principle can be seen in figure 5.

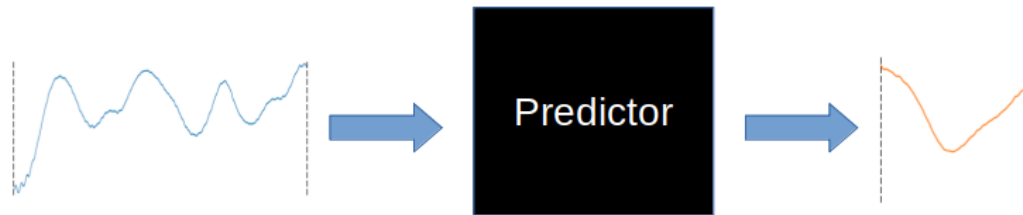


Figure 5: The basic input and output structure of the neural network predicting the platform motion based on the past simulated platform motion

Platform motions are simulated using the Dynasim software to train the predictors, MLP and LSTM, to predict platform motions based on past platform motions.

In order to train the predictors, training sets must be created, that match the real life scenario as close as possible. To create these training sets the data needs to be prepared. Environmental cases that best reflect the real life scenario must be selected for simulation and adequate time frames of prediction times and input times must be chosen.

In order to achieve the objectives, the following steps are carried out in this project:

1. Obtaining real data on environmental conditions to guide the generation of simulated data for training and testing the proposal;
2. Collecting synthetic data for training and testing of the developed ML solutions. Data are generated by using real environmental condition readings and a model of a FPSO as inputs to a hydro-dynamical numerical simulator (Dynasim) whose output are platform motions in relation to different environmental conditions. Long term simulation is executed. Line breaking events are executed at known time moments to generate test data for the ML-based solutions.
3. Development of the 2 ML-based solutions to analyze the breakage detection of the

mooring system;

4. Training and validation of the 2 solutions, using data generated by the simulator under normal conditions (without breaking lines).

3.2 Comparator

The comparator compares the predicted motion with the simulated motion and calculates error scores based on the difference. For system reliability the error scores should not rely on single predictions but on multiple predictions over a longer period of time, to verify that the difference is due to a Line failure and not faulty sensor data. Therefore the Comparator creates error windows which consist of two consecutive predictions. For each error window the comparator calculates the Root mean square error, the mean error and the median error, to further reduce the influence of outliers or faulty sensor data. The 3 different error scores are then fed to the classifier. Figure 6 shows the inputs and outputs of the comparator.

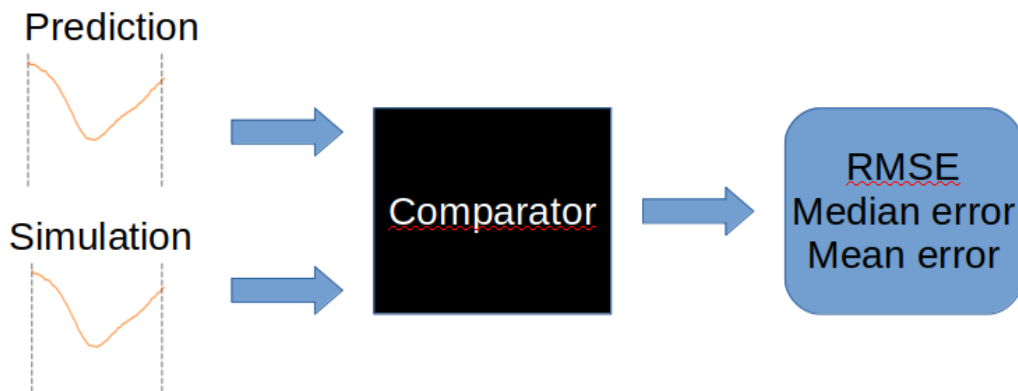


Figure 6: The basic input and output structure of the comparator calculating different error scores based on the difference between simulation and prediction

3.3 Classifier

Based on the different error scores calculated by the Comparator different classifier models are developed. The classifiers take the different error scores as input for Mooring Line Failure identification. The identification in this first phase of the project is binary

identification of line failures. In the last part of the chapter the different classifiers are compared against each other to identify the best suited classifier. Figure 7 shows the implemented classifier with its input and output.



Figure 7: The basic input and output structure of the classifier identifying mooring line failures based on different error scores calculated by the comparator

4 PREDICTOR

As described in the proposal the predictor module is the first of the three main modules of the system. It predicts short term platform motion based on previous recorded platform motion. The predictor is the first module of the data pipeline as it can be seen in figure 8. The prediction is done using two different Machine Learning algorithms, MLP and LSTM. The necessary Data preparation for training the models is discussed in section 4.1. After the data is prepared the implemented MLP structure is presented in section 4.2 and the implemented LSTM structure is presented in section 4.3, showing the network structure as well as the prediction results. The last part of this chapter compares the two implemented models based on their results.

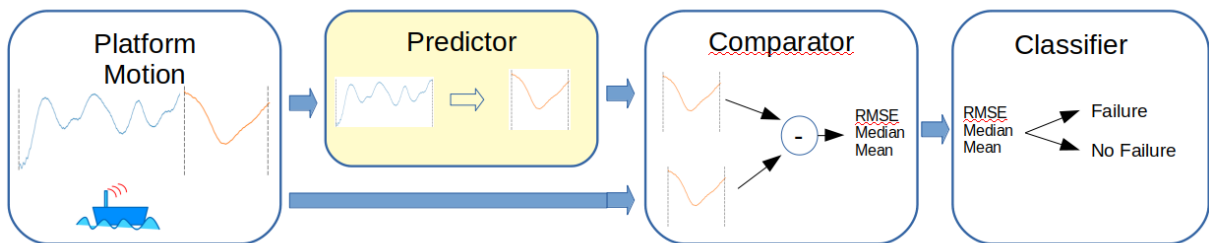


Figure 8: The figure shows the predictor module as part of the modular system structure

4.1 Data preparation

This section outlines the necessary steps to prepare the data for training the predictors. Section 4.1.1 describes the important step of choosing the right data for training since poorly chosen data can lead to slow learning rates. Then Section 4.1.2 explains the creation of training sets, which prepares data arrays that can be understood by the predictors.

The general pipeline for this phase of data preparation is shown in Figure 9. Environmental Data was initially collected in the region of interest and an analysis of the

data distribution was carried out. Then, a subset of the data was sampled and fed to the Dynasim simulator, generating the platform motion based on the environmental condition and the platform structure without failures in the mooring lines. Testing data with and without mooring line failures was then created for all available cases. Whenever pertinent, the data was standardized for values between 0 and 1.

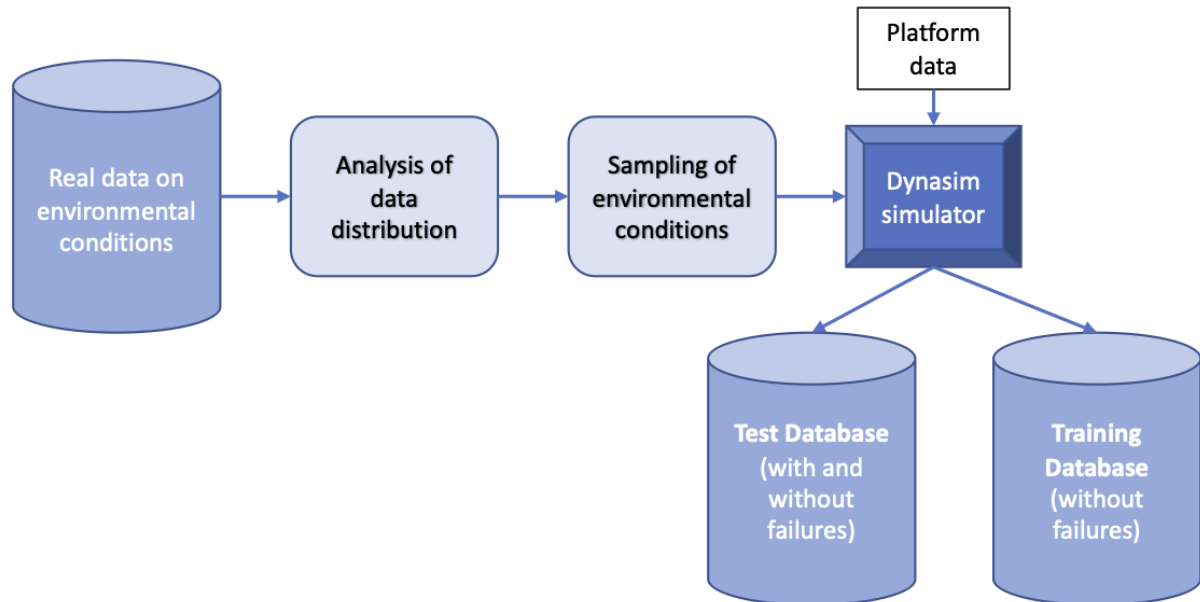


Figure 9: Generation of training and test data: Initially real data of environmental conditions are analyzed and sampled, feeding the simulator that then generates motion data for the specified platform.

4.1.1 Environmental condition selection

Environmental conditions have been recorded by a weather station located in Campos Basin (Bacia de Campos) of Rio de Janeiro (RJ), Brazil from 2003 until 2009 in intervals of 3 hours and have been exported in a text file, totaling 18000 different environmental conditions. A data extract of these environmental conditions can be seen in Table 1. Four different dimensions were captured: wave, swell, wind and current. Wave and Swell are characterized by their height (hs), peak to peak time (tp), and direction (dir). Index 1 is used for short frequency waves (hs1, tp1, dir1) and index 2 indicates long frequency waves also known as swell (hs2, tp2, dir2). Wind and current are characterized by their speed (vel) and direction (dir). As can be seen in Table 1 there exist many measurement points with missing measurements.

To resolve the missing data, non-existent swells are considered to be 0 since it is likely that they were too weak to be measured. If other characteristics are unknown, the

Table 1: Example of environmental conditions measurements

data	hora	hs1	tp1	dir1	hs2	tp2	dir2	hstotal	vento_vel	vento_dir	corr_vel	corr_dir
01/01/03	00:00	1.11	5.30	30.0	0.94	10.53	127.3	1.45	6.94	24.0	NaN	NaN
01/01/03	03:00	1.15	5.45	30.8	0.87	10.30	145.1	1.44	7.53	20.6	NaN	NaN
01/01/03	06:00	1.22	5.29	31.0	0.83	9.93	150.6	1.48	8.13	17.7	NaN	NaN
01/01/03	09:00	1.23	5.05	26.6	0.87	9.72	134.4	1.51	8.07	10.6	NaN	NaN
01/01/03	12:00	1.17	5.00	18.5	0.94	9.57	112.6	1.50	8.02	3.5	NaN	NaN
...
31/12/09	12:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
31/12/09	18:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
31/12/09	21:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
01/01/09	00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

environmental condition is discarded from the further analysis.

In order to avoid angles to indicate directions, the speed was divided into its x-component and y-component. The x-component is calculated by

$$x = v * \cos(\phi), \quad (4.1)$$

and the y-component by

$$y = v * \sin(\phi), \quad (4.2)$$

where v is the velocity and ϕ is its direction. For the wave and the swell, their angles are multiplied by the height. The result can be seen in Table 2 and in the histograms of Figure 10.

Table 2: Necessary conditions with velocities split into its x and y components

tp1	tp2	hs1_x	hs1_y	hs2_x	hs2_y	vento_x	vento_y	corr_x	corr_y
7.10	3.74	0.91	1.47	-0.60	-0.09	-4.21	-3.49	-0.10	0.05
7.61	5.62	1.59	-0.55	-0.30	-1.04	-7.46	-0.18	0.13	0.02
7.62	7.94	-1.10	-1.31	1.14	0.44	-3.37	8.89	-0.14	-0.02
8.23	9.44	2.12	-0.58	-0.94	0.80	-0.83	10.52	-0.15	-0.01
...
7.17	8.55	-2.45	0.51	0.25	0.56	11.99	3.87	0.55	0.08
7.46	8.07	-0.45	-2.52	-0.81	0.27	11.24	1.94	0.45	-0.31
7.35	0.00	0.81	-2.41	0.00	0.00	-7.02	-7.41	0.10	-0.53

Finally, data are then scaled so that their values are in the range between 0 and 1. For each variable, the minimum and maximum values are calculated and the data are

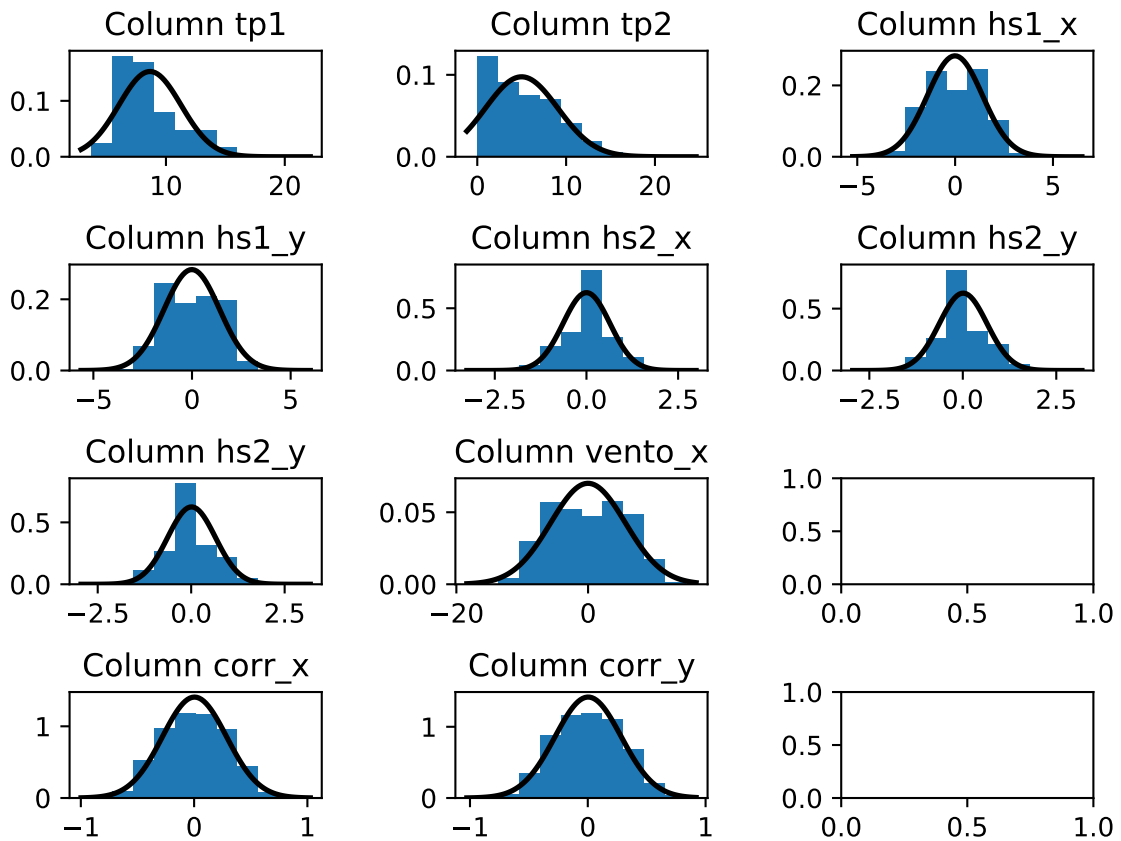


Figure 10: Histograms showing the distribution of environmental variables recorded by a meteorological station located in the Campos Basin, RJ, with velocities split into its x and y components. The data are considered to follow a normal distribution, represented by a black line drawn on each histogram.

then standardized by:

$$X_{out} = \frac{X_{in} - Min}{Max - Min}, \quad (4.3)$$

where Min is the minimum value, Max is the maximum value, X_{in} is the original value, and X_{out} is the scaled value. Further details on this data standardization technique can be found in the Appendix A.3. The result can be seen in Table 3.

Using the resulting data-set, a subset of environmental conditions was drawn using the normal distribution as a probability density function. For each feature one value was drawn from this density function. The randomly picked values are then put together to form a random scenario. This procedure was repeated the number of times necessary to create a subset of all environmental conditions with a similar normal distribution. An example can be seen in Table 4.

Table 3: Environmental condition measurements scaled between 0 and 1

tp1	tp2	hs1_x	hs1_y	hs2_x	hs2_y	vento_x	vento_y	corr_x	corr_y
0.19	0.16	0.53	0.62	0.42	0.46	0.40	0.41	0.44	0.56
0.22	0.24	0.59	0.43	0.47	0.29	0.30	0.52	0.56	0.54
0.22	0.34	0.34	0.36	0.72	0.56	0.43	0.81	0.42	0.52
0.26	0.40	0.64	0.43	0.36	0.62	0.51	0.86	0.41	0.53
0.28	0.44	0.20	0.53	0.71	0.49	0.89	0.58	0.51	0.62
...
0.20	0.36	0.21	0.53	0.57	0.58	0.91	0.65	0.79	0.58
0.21	0.34	0.40	0.24	0.38	0.53	0.89	0.58	0.74	0.35
0.21	0.00	0.52	0.25	0.52	0.48	0.31	0.29	0.55	0.23

Table 4: Randomly created scenarios that follow the normal distribution

tp1	tp2	hs1_x	hs1_y	hs2_x	hs2_y	vento_x	vento_y	corr_x	corr_y
6.89	9.96	1.99	0.24	0.10	-0.34	6.18	-4.88	0.51	0.65
10.07	4.71	3.43	-1.17	0.40	-0.74	5.67	1.04	-0.08	0.29
14.65	6.50	0.85	-1.27	0.17	-0.58	-5.71	-0.48	-0.13	0.51
5.38	5.80	-0.78	-1.79	1.25	0.35	-11.94	-8.47	0.10	0.11
...
5.69	3.37	0.72	0.52	-0.23	0.53	8.19	-10.03	-0.24	0.45
8.51	8.67	2.77	-1.31	-0.19	0.03	5.14	1.51	0.39	0.04
7.00	8.10	0.88	-0.13	0.46	0.49	6.14	1.30	-0.37	0.09
9.00	5.37	0.61	0.27	-0.47	0.47	-0.99	-5.78	-0.11	-0.68
11.40	3.01	0.82	2.26	-0.25	0.83	-7.51	3.40	-0.00	-0.07

This normal distributed cases are then scaled using the same calculation described previously (Equation 4.3 as detailed in Appendix A.3). The result can be seen in Table 5 and in histograms in Figure 11.

Based on this randomly created scenarios the closest measured scenarios can be found by calculating the difference for every feature individually and summing up the absolute differences. The real life scenario with the smallest resulting difference is considered the best fit for the randomly generated scenario and is selected. The scaled closest real life cases are shown in Table 6 and in histograms of Figure 12.

Table 5: Randomly created scenarios that follow the normal distribution, scaled between 0 and 1

tp1	tp2	hs1_x	hs1_y	hs2_x	hs2_y	vento_x	vento_y	corr_x	corr_y
0.18	0.42	0.63	0.51	0.54	0.42	0.73	0.37	0.77	0.90
0.36	0.20	0.76	0.37	0.59	0.35	0.71	0.56	0.45	0.70
...
0.30	0.23	0.50	0.51	0.44	0.56	0.50	0.34	0.43	0.15
0.44	0.13	0.52	0.70	0.48	0.63	0.29	0.63	0.49	0.49

Table 6: Found real life scenarios that match the random scenario

tp1	tp2	hs1_x	hs1_y	hs2_x	hs2_y	vento_x	vento_y	corr_x	corr_y
0.15	0.32	0.59	0.44	0.61	0.38	0.73	0.31	0.76	0.78
0.35	0.25	0.59	0.45	0.54	0.37	0.73	0.56	0.55	0.65
0.44	0.20	0.48	0.32	0.55	0.33	0.33	0.48	0.42	0.84
0.19	0.33	0.39	0.23	0.65	0.43	0.21	0.25	0.53	0.66
0.31	0.18	0.29	0.46	0.42	0.37	0.51	0.34	0.68	0.15
...
0.20	0.32	0.66	0.53	0.49	0.60	0.80	0.28	0.44	0.76
0.29	0.16	0.57	0.41	0.42	0.48	0.70	0.55	0.71	0.55
0.11	0.32	0.57	0.52	0.61	0.51	0.80	0.56	0.27	0.60
0.25	0.21	0.55	0.60	0.56	0.64	0.48	0.32	0.50	0.20
0.47	0.20	0.55	0.65	0.48	0.65	0.32	0.50	0.49	0.51

The mean values and respective standard deviations of the actual data with the sampled data are then compared to evaluate the data that feed the simulator (Table 7).

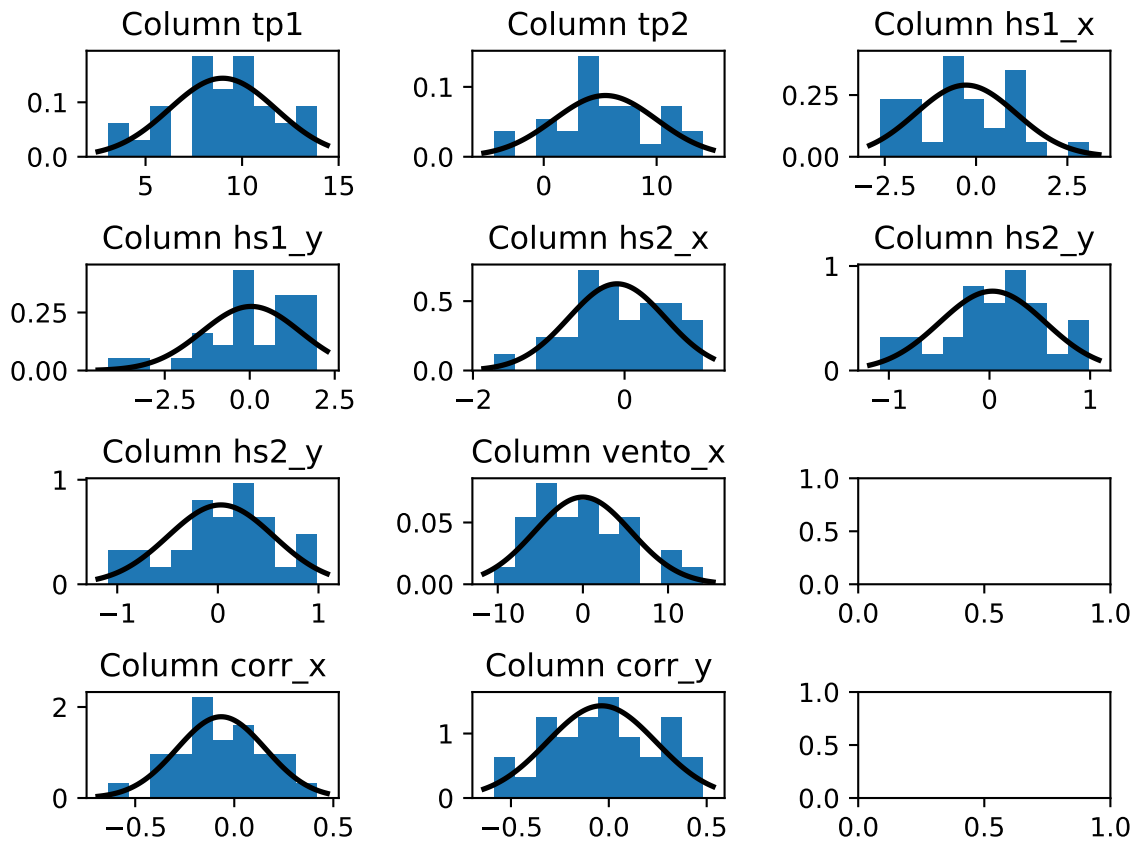


Figure 11: Histogram of random selected cases used for training the predictors.

Table 7: Mean values and standard deviations of the variables of all cases and selected cases

tp1	tp2	hs1_x	hs1_y	hs2_x	hs2_y	vento_x	vento_y	corr_x	corr_y
Mean value and standard deviations of each column for all cases									
8.65	4.99	-0.01	-0.01	0.00	0.01	-0.01	0.04	0.00	0.00
2.62	4.09	1.41	1.41	0.64	0.64	5.69	5.69	0.28	0.28
Mean value and standard deviations of each column for selected cases									
8.29	5.46	0.37	-0.27	0.22	0.08	0.18	-0.40	0.01	0.07
1.84	3.03	1.30	1.26	0.57	0.61	6.58	5.02	0.30	0.31

Table 7 shows that the mean value and the standard deviations of the randomly drawn scenarios show little difference in comparison to all the cases, meaning that the drawn subset is a good representation of all measured cases.

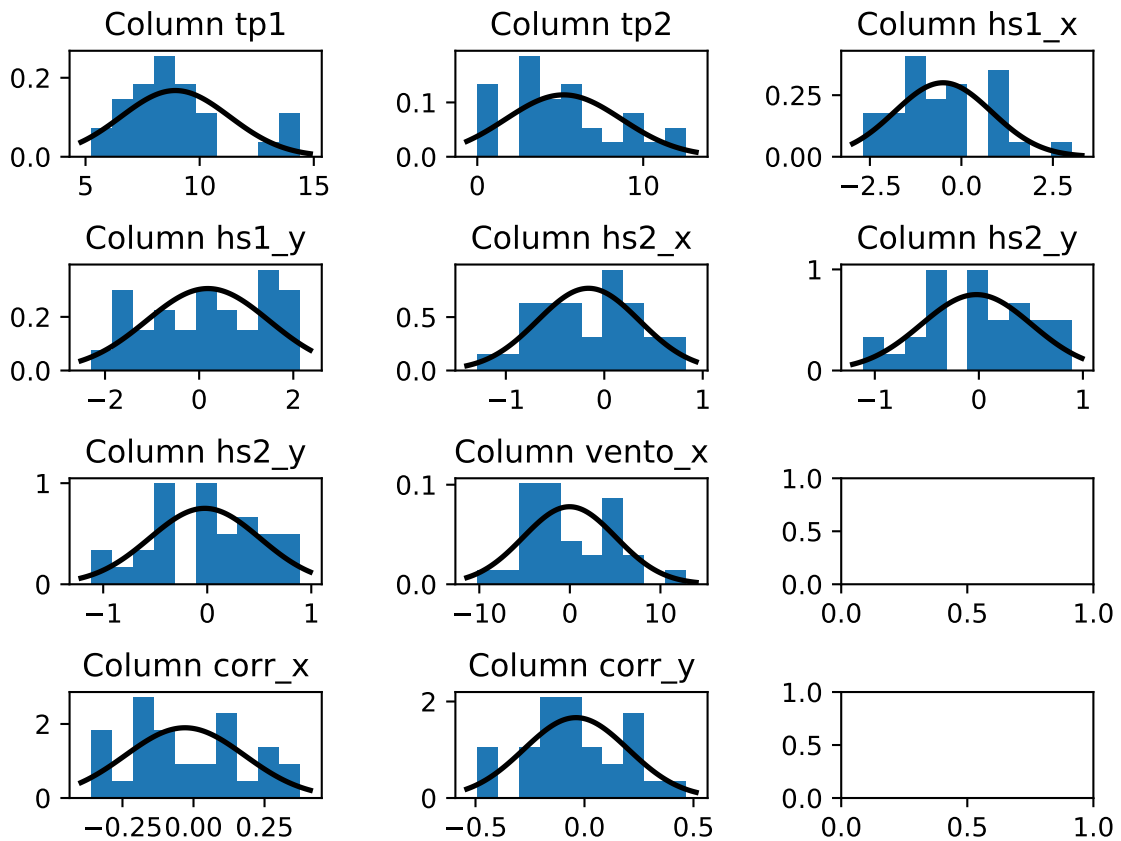


Figure 12: Scaled closest real life cases that are selected for training.

4.1.2 Training set creation

After suitable environmental conditions are sampled that best represent the environmental conditions of the region, the next step is to make the simulated data suitable to train the predictors. Since the used predictors follow the supervised learning paradigm, training sets must be created, which consist of a set of input and output data.

For each of the selected environmental conditions (see Figure 11), Dynasim was used to simulate 3 hours of platform motion. The environmental conditions are indexed by their line number, which equals the index attributed to the 6 degrees of freedom (DoF) motion data generated by Dynasim. Firstly parts of the time series are sampled that characterize the platform motion generated by the simulator for each environmental condition. Each of these parts is called a data window and corresponds to a training unit. The size of the data windows is a system parameter and is kept fixed for each predictor. Data windows are independent of each other and can be drawn from completely random points within the three-hour data regarding the same environmental condition. Likewise, data windows are sampled from other selected environmental conditions, and the set of all these training

units forms a training set for a neural network. Depending on the machine learning (ML) model to be trained, the size of these data windows as well as the 6DoF motion variables (here called features) can vary. The Features are sway, surge, yaw, heave, roll, and pitch (see Appendix A.2 for details).

A training unit is composed of the pair $\langle \text{input} - - \text{output} \rangle$, which is defined by a partition of the data window. Typical window ranges representing input data are between 200 and 500 time steps, and between 50 and 200 time steps of output data. The size of the input and output are also system parameters. For training a network, the size of the input and output data series, and the features used must be kept fixed for a specific predictor. For example, the training set for a certain network could consist of data from training units of 500 time steps, each unit divided as follows: The first 400 time steps of each feature representing the input data, and the last 100 time steps representing the output data of each feature, as shown in Figure 13. In our case, 1 time step corresponds to 1 second.

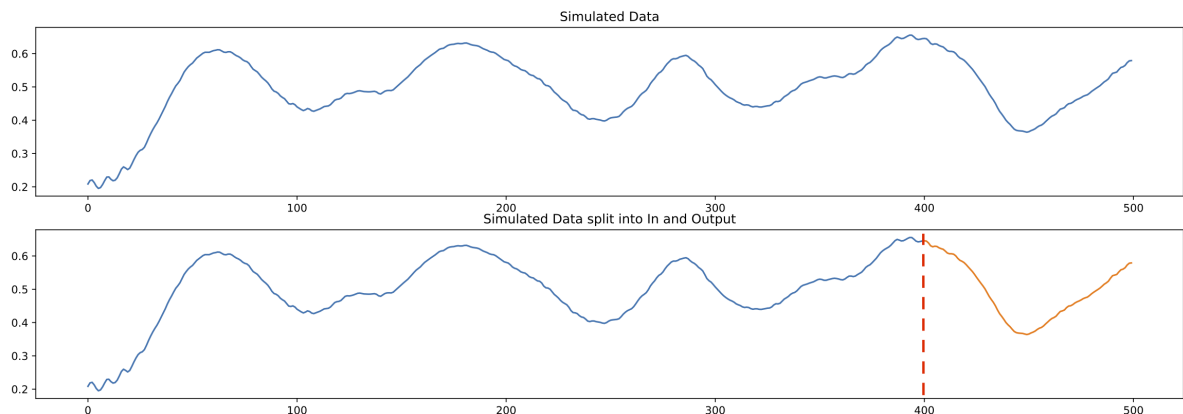


Figure 13: Training unit of a data window of 500 time steps, with 400 time steps being input (in blue) and 100 time steps being output (in orange).

The process of generating training units consists of sliding the data window on the time series generated by Dynasim, with strides defined as system parameters. In our proposal, MLP and LSTM models use different strides (see Section 4.2 and Section 4.3 for details). It is important to note that the data windows can overlap in the sampling process.

After the simulated data are split into several training units, they are bundled into so-called batches. In our proposal, 1 batch consists of 32 training units. Section 4.2 and Section 4.3 explain in detail how these batches can then be fed to the predictors for training.

4.1.3 Testing set creation

Similarly to the training set creation in section 4.1.2, the testing sets are then prepared. Figure 14 shows a data window of the simulated platform motion on top. This data window is then partitioned into a test unit with an input and output, as shown in the middle graph. The bottom graph shows the prediction of the predictor for this motion feature (in orange). Afterwards the predicted motion will be compared to the simulated motion.

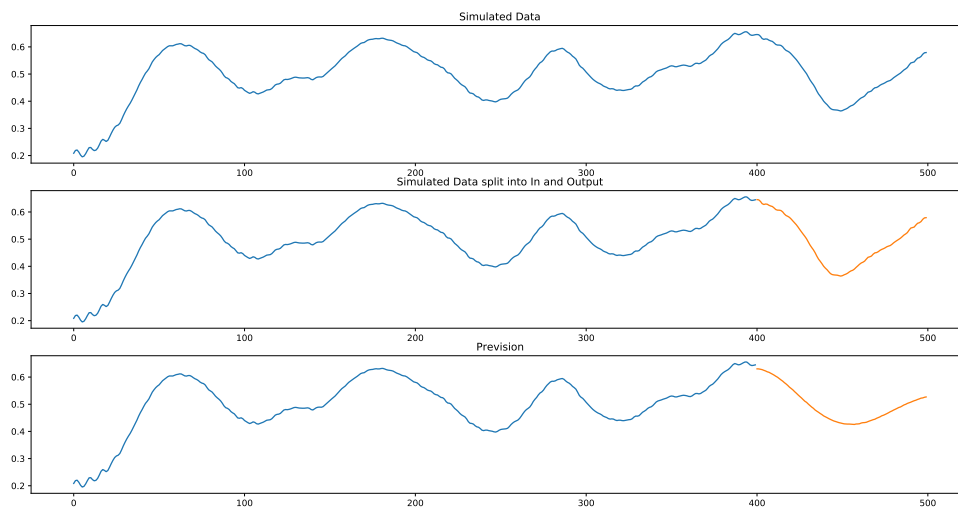


Figure 14: Single Platform Motion Prediction of a predictor. Top: simulated platform motion. Middle: a test unit input (in blue) and a test unit output (in orange). Bottom: the prediction (in orange) of a trained predictor for the test unit input (in blue).

The same procedure is repeated for the whole test set. It is important to note that the test sets consist of simulated motion data with and without mooring line failure.

The inputs of the testing units are fed to the trained predictor, who predicts the platform motion based on the seen input. These outputs of the trained predictors, which are temporal predictions of motion features, are then concatenated with each other as it can be seen in Figure 15. In this figure, three consecutive predictions are shown. The simulated as well as the predicted motion are shown. The last graphic shows the concatenation of the predictions for three test unit inputs. Each test unit is defined by a prediction window, as shown in Figure 15, having the same dimensions as the data window. The prediction window stride is a system parameter, which defines the time range of interest on the simulated data curve that is used for the prediction (the input) and for the calculation of the error (the output). In the figure, the input consists of 400

seconds (in blue), the output consists of 100 seconds (in orange), with a stride of 100 seconds (red curve that increases at the beginning of the graph).

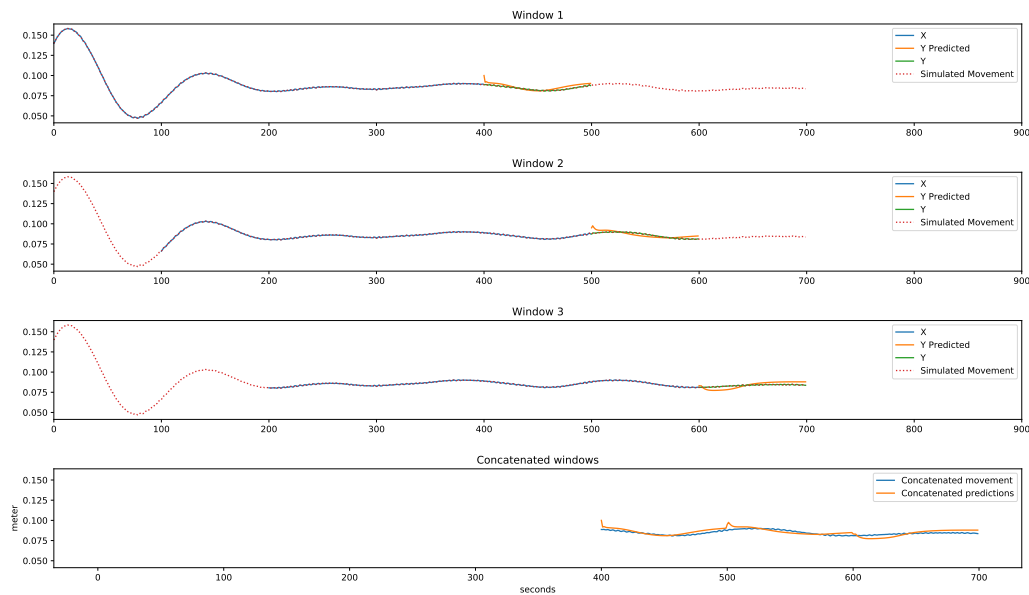


Figure 15: Prediction Windows: the top graph shows a test with input from 0s to 400s (in blue) and output from 400s to 500s (in orange); the process is repeated in the subsequent graphs, with input from 100s to 500s and output from 500s to 600s and input from 200s to 600s, with output from 600 to 700s. The graph below shows the concatenation of the 3 predictions (from 400S to 700s) in orange, superimposed on the simulated curve (in blue).

The stride of the prediction window is never bigger than the prediction time range, therefore there exists always a predicted data for every time step.

If the stride is smaller than the prediction time range, there can exist multiple predictions for one time step. In this case the mean value is calculated between all available predictions of that time step, resulting in a single prediction for every time step.

4.2 MLP Predictor

In this section, a Multi-layer Perceptron (MLP) neural network (see Appendix A.1.2 for details) is implemented and results of the implemented MLP are provided. The section is split into different parts representing the consecutive steps necessary for the implementation of the MLP predictor. Section 4.2.1 describes the environmental conditions, used for training and validation of the MLP. Next, the system parameters of the data window like input and output size and stride for the training and validation units are defined. A description of DoF features used as input and output for the MLP model is provided. The best model architecture found that was able to predict the platform motion most

accurately is also provided. Finally, the results achieved with the predictions made by the MLP model are presented.

4.2.1 Environmental condition selection

A good training data-set is critical to the success of the models ability to learn. Using the same procedure as explained in section 4.1.1 a subset of 5000 environmental conditions for training and another 1000 conditions for validation were selected for the proposed MLP model. Figure 16 shows the histogram of the selected subset and the histogram of all measured conditions. It can be seen that the file selection reflects the real-life distribution of measured environmental conditions.

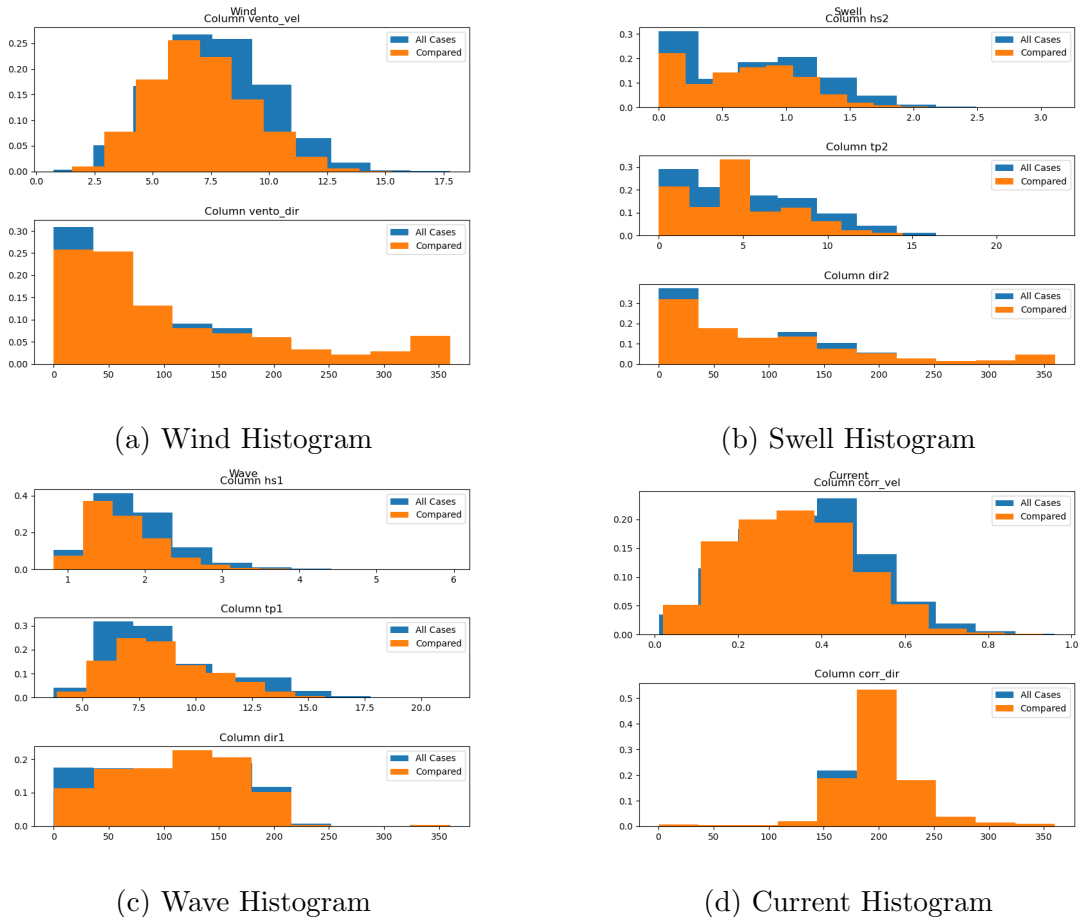


Figure 16: Illustration of the histograms of the selected 6000 files for the MLP network (orange) against the histogram of all original 18000 files (blue).

4.2.2 MLP Training

A neural network learns by the iterative process of gradient descent whose objective function is reducing the error between a networks prediction and the actual target value. The learning process is done using back-propagation algorithm which uses gradient descent to minimize the networks error. The gradient descent algorithm uses the training data units to modify the weights and biases of a network [23] (see Appendix A.1.2 for details on neural network learning process). For example, a training set can consist of 1000 training units and the gradient descent algorithm can be instructed to used the entire 1000 units to make a gradient descents step. This is known as batch gradient descent or gradient descent. Each gradient consists therefore of the entire training set.

An epoch is known as the process of training the model on the entire training set. If the model is trained for 10 epochs for example, the training set is passed through the network 10 times and 10 gradient descent steps are done using the entire set.

In cases where there are too much training units in a set which can not be fed in its entirety into the memory of the system the set can be split into batches, containing a subset of the training units. This is known as mini-batch gradient descent. For example, a training set that consists of 1000 training units can be split into 4 batches of 250 units. The batches are then fed to the network individually, resulting in a gradient step for each individual batch. In this scenario a single epoch is completed 4 gradient steps. Meaning that 10 epochs would need 40 gradient steps to complete the training of the network [23].

Apart from the training data, validation data is used by the model to validate the training progress of the network. To be sure that the validation set and training set are independent, the validation set uses different environmental conditions than the training set. The model modifies its weights and biases by comparing the computed output with the given output, trying to minimize the difference between them. This is continuously done for each batch until a stop criteria is met, like a maximum number of epochs or a diminishing accuracy of the validation data, since the model is likely to be over-fitting at this point (see Appendix A.1.2 for more on over-fitting).

In this project, the MLP training set used was composed of 5000 different environmental conditions. Each environmental condition consisted of 10,000 training units, adding up to 50,000,000 training units. A single epoch was completed in 1,562,500 gradient descent steps. The validation data was composed of 1000 environmental conditions. Each environmental condition consisted of 10000 validation units, adding up to a total of 10,000,000 validation units. The training process stopped at epoch 3000 when the model started to

over-fit.

The proposed network predicts 100 seconds based on the last 600 seconds of horizontal platform motion , i.e., each unit consisted of 700 seconds , with an input of 600s and an output of 100s.

Figure 17 shows the training curve of the first 400 epochs. The blue line shows the Mean error, based on the difference between the computed output and the expected output for the training set. The orange line shows the mean error of the validation set. As expected the accuracy of the training set is monotonically improving during the first 400 epochs. It can be seen that the accuracy of the validation set is also increasing steadily, indicating a balanced training set and a good learning process.

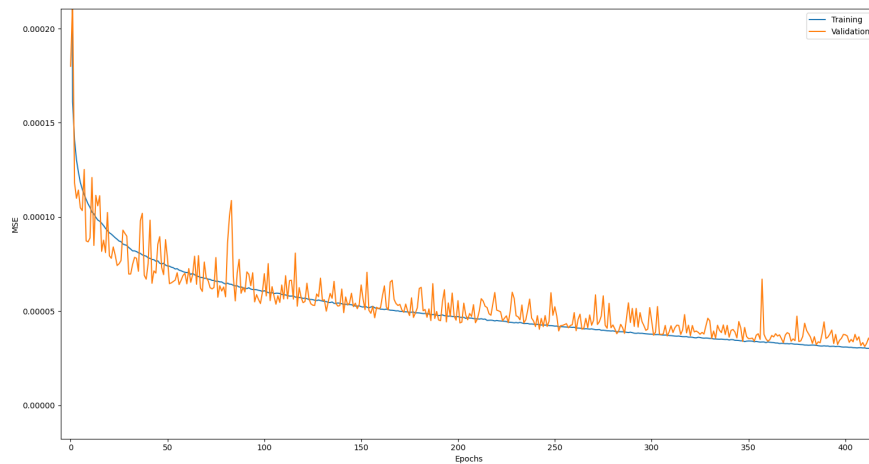


Figure 17: Example of the MLP training curve of 5000 training environmental conditions (blue) and 1000 environmental conditions for validation (orange). The vertical axis represents the error and the horizontal axis represents the number of epochs. The error score decreases with an increasing number of epochs.

4.2.3 Model architecture

To find the best MLP model capable of learning the complex motions of the platform, MLP models with different layer compositions and activation functions were tried. Four different MLP model compositions are presented in Table 8 and the MLP model with the best error score, highlighted in bold was considered to be the best. The numbers in the model architecture column refer to the number of nodes in the layer, each number representing a layer in the network. The first and last number refers to the number of nodes in the input and output layer and the numbers in between refer to a hidden layer

with its corresponding number of nodes.

Table 8: Different MLP architecture compositions with 3600 input and 300 output nodes, and their respective error score.

Model Architecture	Activation Function	RMSE Error
3600, 1800, 300	Relu	2.26e -01
3600, 4800, 1200, 300	SGD	5.81e -01
3600, 3600, 1800, 300	Relu	2.61e -01
3600, 7200, 3600, 1800, 300	Relu	1.63e -01

The MLP model selected and implemented in this work can be seen in Table 8 in bold. The input of the MLP model is composed of the 600-second time series of the 6DoF platform motion features: sway, surge, heave, roll, pitch, and yaw. The output of the MLP model predicts 100 seconds of the platform horizontal motions, surge, sway, and yaw. Figure 18 shows input and output of the MLP model. The implemented model is

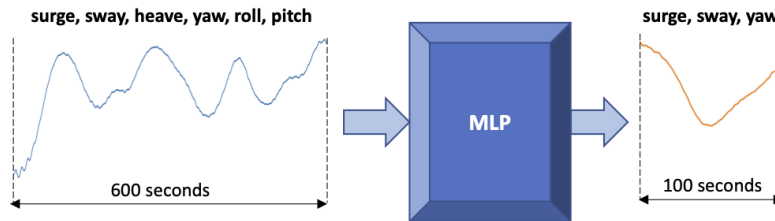


Figure 18: The MLP model has as input 600 seconds of time series of the features sway, surge, heave, roll, pitch, and yaw of the platform motion, and outputs 100 seconds of prediction of the three features sway, surge, and yaw.

composed of an input layer, 3 hidden layers and an output layer, which are fully connected to each node in each layer. An illustration of the MLP architecture can be seen in Figure 19. The input layer (input) receives 600 seconds of 6DoF motion features of the platform. Therefore, the MLP receives 3.600 different data points. The output layer predicts 100 seconds of the platform horizontal motion, surge, sway and yaw, which means the MLP outputs 300 data points in total. The 3 hidden layers (Figure 19) had 7200, 3600 and 1800 nodes respectively in each layer. The rectified linear activation function, or ReLU for short (see Appendix A.1), was used in all the layers with the exception of the output layer which used linear activation function to make the prediction. Adam optimizer algorithm with a learning rate of $1-e7$ was used to optimize the MLP network (see Appendix A.1.2).

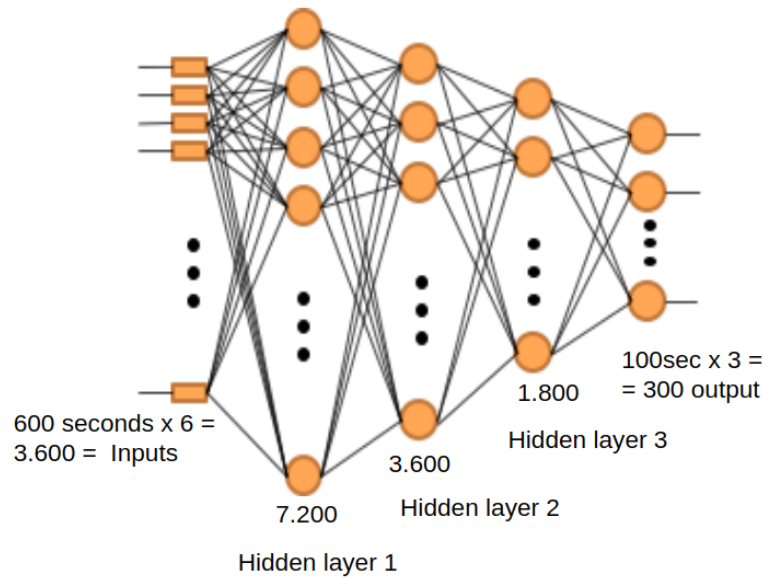


Figure 19: The fully connected Multi-Layer Perceptron (MLP) architecture used, with 3600 input nodes, 300 output nodes, and three hidden layers with 7200, 3600, and 1800 nodes, respectively.

4.2.4 MLP Prediction

After the MLP predictor model was trained, it was then used to predict motions of different environmental conditions. As explained previously, MLP makes predictions of 100s. The trained MLP model was then tested on 3 environmental conditions with calm, mild and rough conditions to gauge the prediction performance of the MLP model. These 3 environmental conditions in Table 9 were selected based on their wave height, swell height and wind speed of these environments. As it can be seen the wave height ($hs1$), swell height ($hs2$) and wind speed ($vento_vel$) of these conditions differ. The environmental condition selected to represent rough condition had a wave height of 2.84 meters, swell height of 0 meter and wind speed of 7.77 meters per second. The selected environmental condition representing mild condition had a wave height of 1.69 meters, swell height of 0.42 meters and wind speed of 4.72 meters per second and the selected rough environmental condition it had a wave height of 1.68 meters, swell height of 0 meter and wind speed of 7.77 meters per second.

Table 9: 3 environmental conditions selected.

Sea state	Index	hs1	tp1	dir1	hs2	tp2	dir2	hstotal	vento_vel	vento_dir	corr_vel	corr_dir
Calm	57	1.69	8.7	100.0	0.42	3.73	30.6	1.74	4.72	17.6	0.34	198.58
Mild	600	1.68	8.48	47.2	1.33	5.4	191.9	2.14	8.21	195.3	0.2	288.84
Rough	8818	2.84	17.04	192.2	0.0	0.0	0.0	2.84	7.77	208.6	0.45	216.93

Figure 20 shows the MLP prediction of the calm environmental condition. The result shows the MLP model was able to predict the frequency and amplitude of the platform motions. Figure 21 shows a zoomed version of the same environmental condition, where the blue vertical lines indicate the boundaries of a prediction of the MLP model.

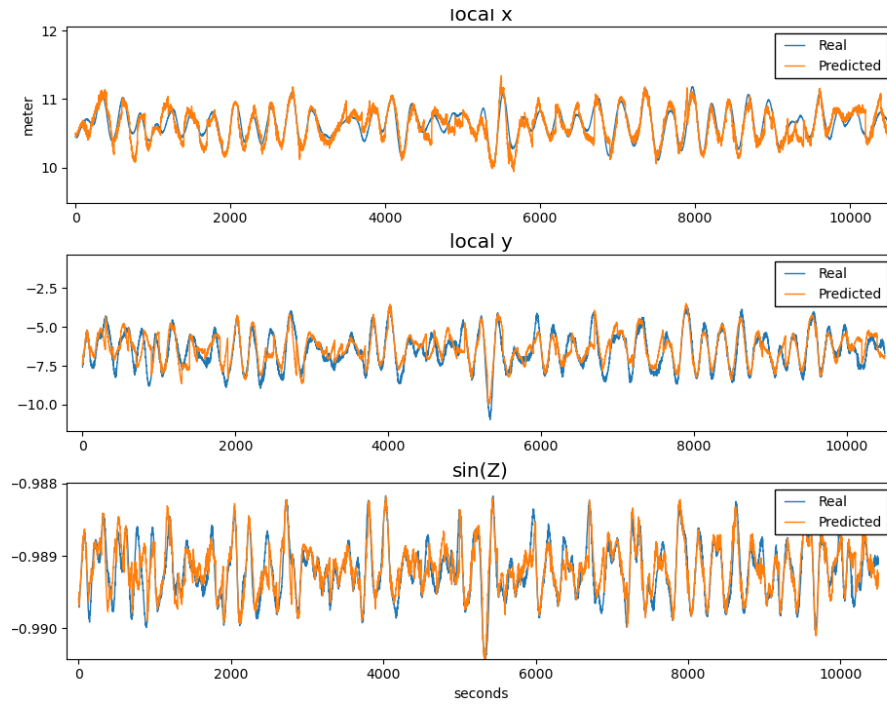


Figure 20: MLP prediction on a calm environmental condition with all mooring lines intact. Local x represents surge, local y represents sway, and $\sin(Z)$ represents yaw motion of the platform. The orange line is the MLP prediction and the blue line is the simulated platform motion.

Figure 22 shows the MLP prediction ability on mild environmental condition. A zoomed image of the MLP prediction on mild environmental condition is presented in Figure 23. The MLP in a mild environmental condition predicts the general oscillation of the platform motions adequately.

When the trained MLP model is given a stormy environmental condition (for example environment 8818 from Table 9) to predict, the MLP model is unable to fully predict the oscillation of the platform's 3DoF, i.e., surge, sway, and yaw. This indicates that the networks might need to be trained specifically to predict under these more dramatic environmental conditions. Figure 24 illustrates the difficulties the MLP model experiences on this environmental condition. Figure 25 provides a zoomed picture of the rapid motions the MLP model finds difficult to predict.

The trained MLP network is then used to predict situations showing a mooring line failure, under the same environmental conditions tested with all mooring lines intact.

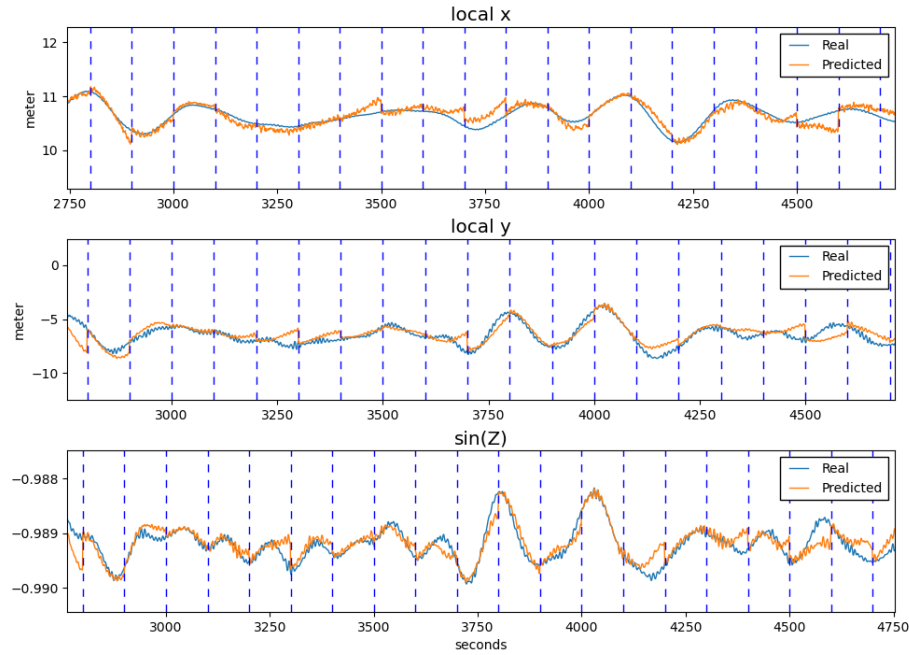


Figure 21: A zoomed version of the MLP prediction on a calm environmental condition. Local x represents surge, local y represents sway, and $\sin(Z)$ represents yaw motion of the platform. The orange line is the MLP prediction and the blue line is the simulated platform motion.

Figures 26, 29a and 29b show the MLP network predictions for cases where there is a mooring line failure in L1, L9, L12 and L18 respectively for a single mild environmental condition. It can be seen that after the mooring line failure at 4000 seconds an offset in the platform position occurs and the MLP model is unable to predict the motions of the platform thereafter. Figure 27 shows how the MLP prediction and the simulated platform motion deviates after a failure at 4000 seconds of simulation in line one, which is located at the left side of the platform's stern .

Figure 27 shows a zoomed image of Figure 26. It shows a change of -9 meters from initial 15 meter for the surge feature after the failure. Surge measurement oscillates for three predictions after which it stabilizes. The sway also shows change after a line failure of 3 meters. After the line failure the MLP model is no longer able to predict the motion.

It can be seen in Figures 28, 29a and 29b that there is an offset between the predicted and simulated platform motions regardless of the mooring line and environmental condition.

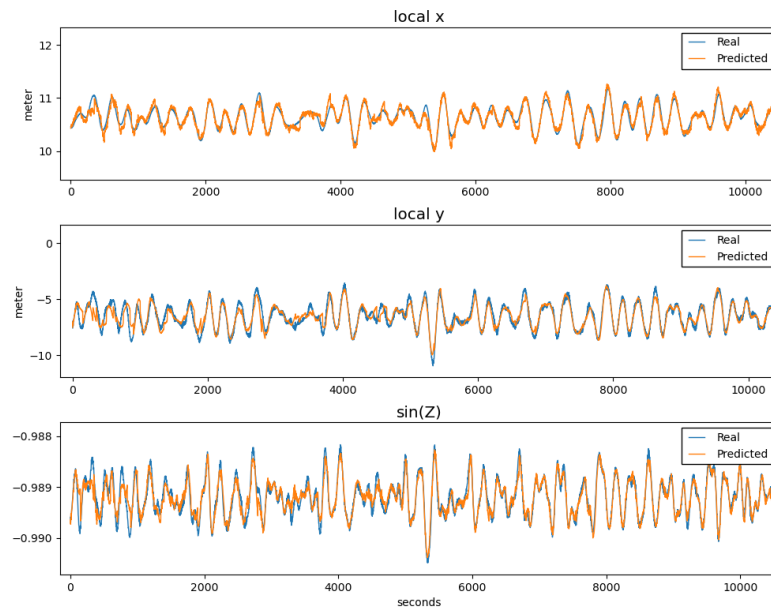


Figure 22: Illustration of MLP prediction on a mild environmental condition. Local x represents surge, local y represents sway, and $\sin(Z)$ represents yaw motion of the platform. The orange line is the MLP prediction and the blue line is the simulated platform motion.

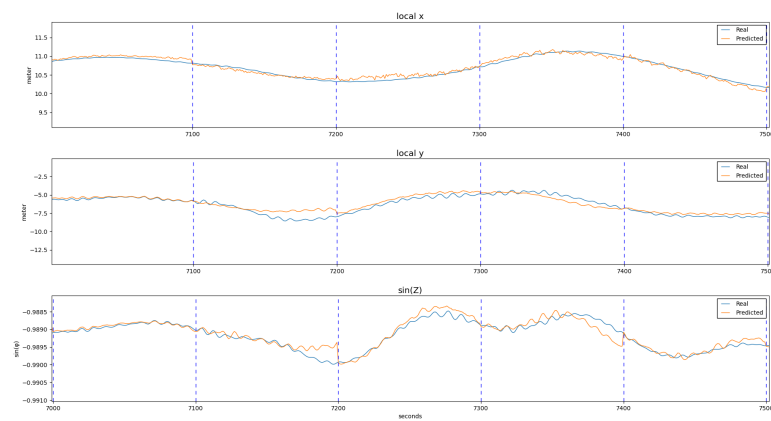


Figure 23: A zoomed illustration of MLP prediction with a mild environmental condition. Local x represents surge, local y represents sway, and $\sin(Z)$ represents yaw motion of the platform. The orange line is the MLP prediction and the blue line is the simulated platform motion.

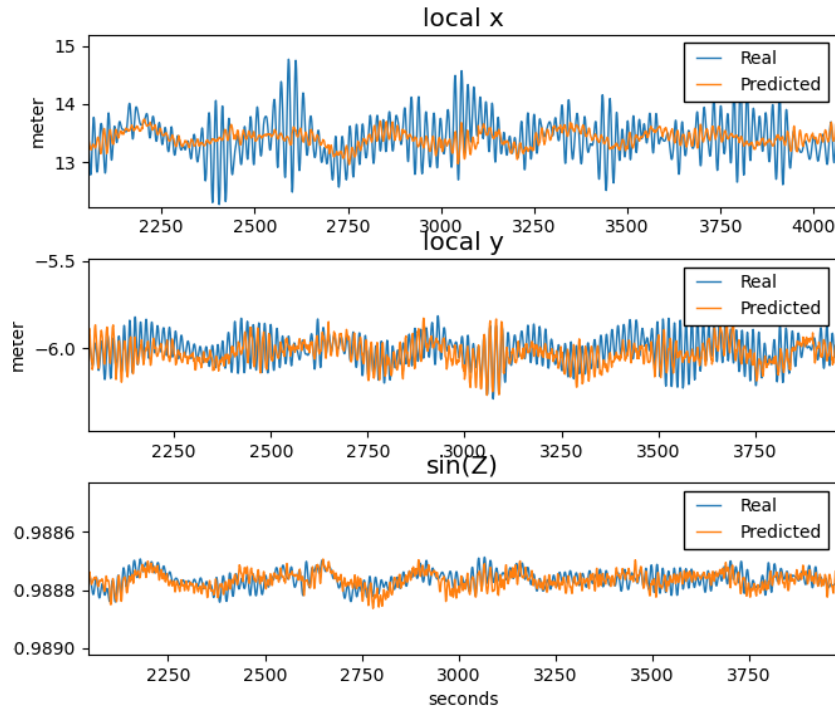


Figure 24: Illustration of MLP prediction on a stormy environmental condition with all mooring lines intact. Local x represents surge, local y represents sway, and $\sin(Z)$ represents yaw motion of the platform. The orange line is the MLP prediction and the blue line is the simulated platform motion.

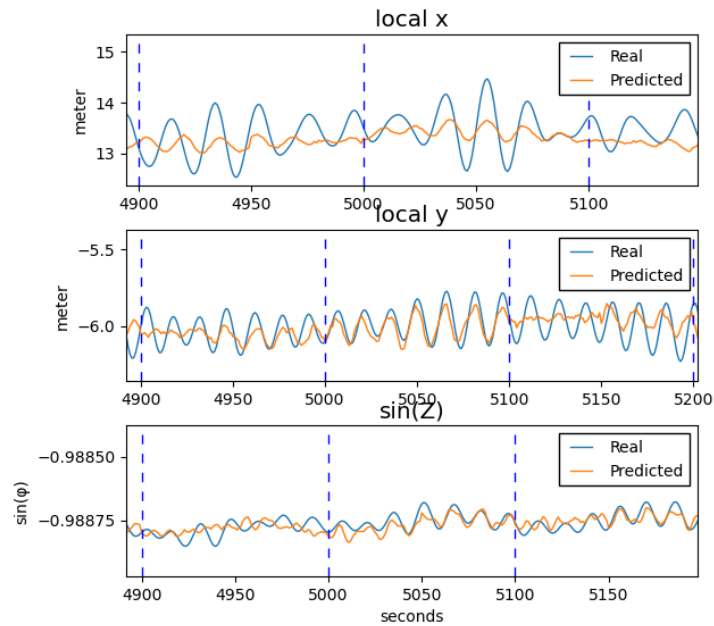


Figure 25: A zoomed illustration of MLP prediction on a stormy environmental condition with all mooring lines intact. Local x represents surge, local y represents sway, and $\sin(\phi)$ represents yaw motion of the platform. The orange line is the MLP prediction and the blue line is the simulated platform motion.

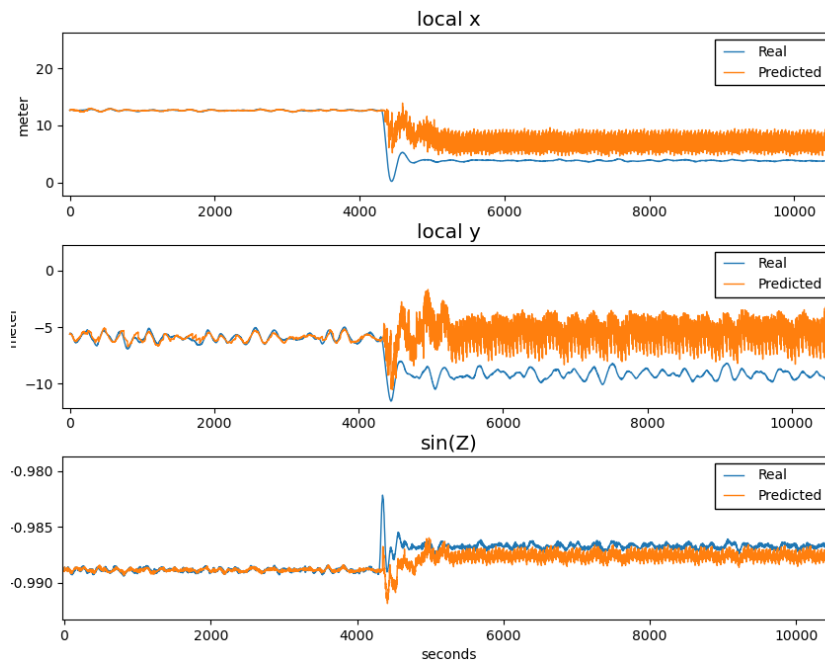


Figure 26: Illustration of mooring line failure of L1 at approximately time step 5000. Local x represents surge, local y represents sway, and $\sin(Z)$ represents yaw motion of the platform. The orange line is the MLP prediction and the blue line is the simulated platform motion.

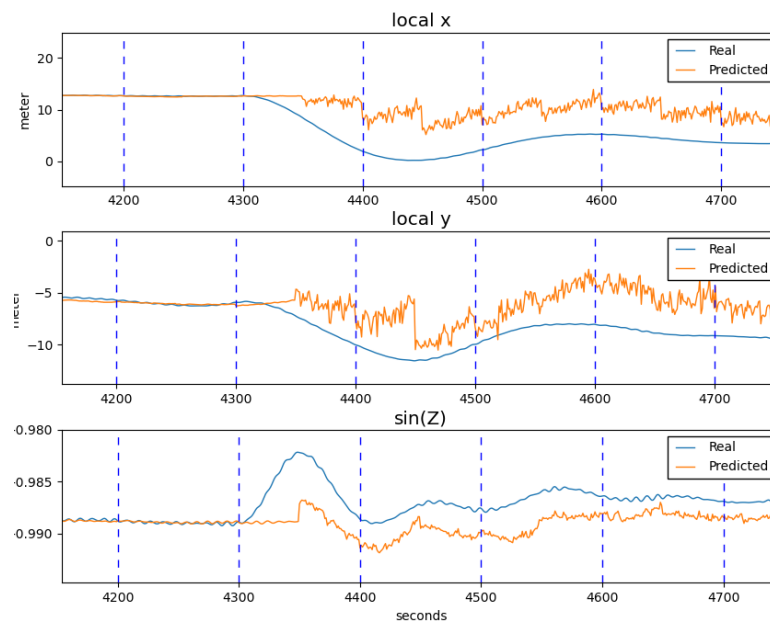


Figure 27: A zoomed illustration of Mooring line failure of line one. Local x represents surge, local y represents sway, and $\sin(Z)$ represents yaw motion of the platform. The orange line is the MLP prediction and the blue line is the simulated platform motion.

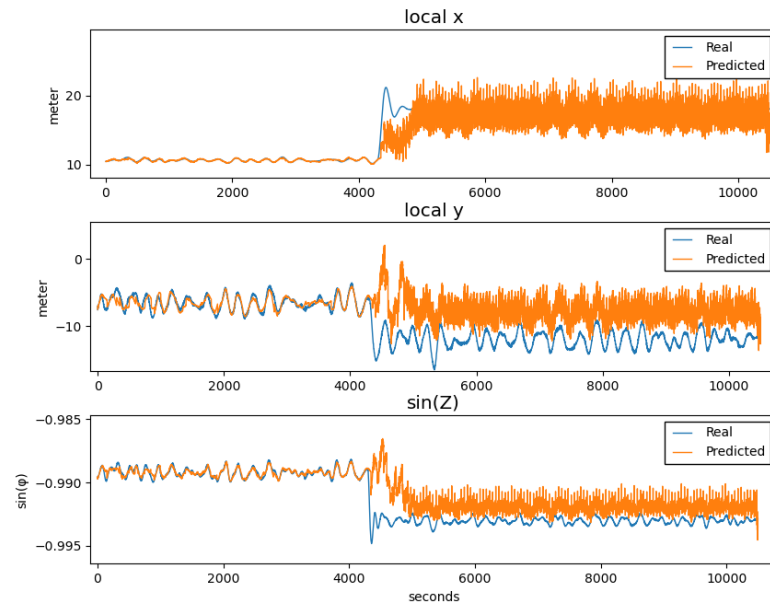
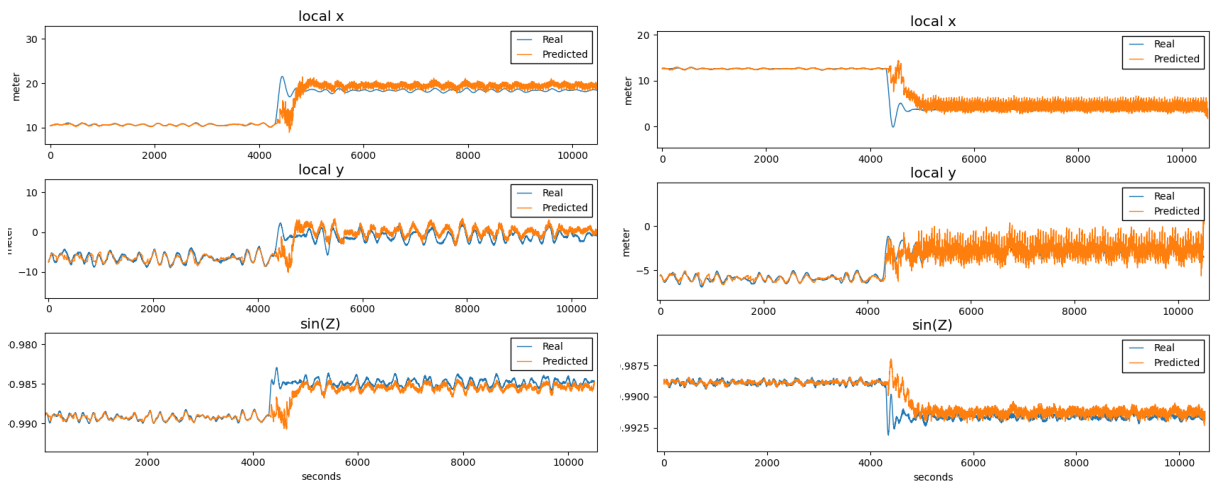


Figure 28: An illustration of mooring line failure of L9. Local x represents surge, local y represents sway, and $\sin(Z)$ represents yaw motion of the platform. The orange line is the MLP prediction and the blue line is the simulated platform motion.



(a) Mooring Line failure of L12.

(b) Mooring Line failure of L18.

Figure 29: Illustration of mooring line failure of L12 and L18 at approximately time step 5000. Local x represents surge, local y represents sway, and $\sin(Z)$ represents yaw motion of the platform. The orange line is the MLP prediction and the blue line is the simulated platform motion.

4.3 LSTM Predictor

This section presents the results found with a trained LSTM network. An overview of LSTM networks is given in Appendix A.1.4. The chapter is split into different parts representing the consecutive steps necessary for the implementation of the LSTM predictor as described in the section 4.1 of this chapter. First the environmental conditions used for training and validation of the LSTM model are presented. Afterwards the learning process based on the selected environmental conditions is shown in detail to then give an overview of the prediction results for different environmental conditions with all mooring lines intact and with mooring line failures.

4.3.1 Environmental condition selection

After experimentation with different configurations, the best results were achieved by designing a subset of 1000 environmental conditions for training and another 200 conditions for validation using the procedure explained in chapter 4.1.1. Figure 30 shows a histogram of the selected subset and the histogram of all measured conditions. It can be seen that the subset follows the same normal distribution and the same standard deviation as all recorded conditions.

4.3.2 Model architecture

Different LSTM models were trained based on this subset of cases, and then compared to each other. The different LSTM models consisted of models with a single LSTM layer and different models with multiple LSTM Layers. The model with the best performance is described in the following. The model with the most accurate prediction found was an encoder-decoder model employing two LSTM layers.

The first LSTM layer is trained on understanding the input sequence. It encodes the input sequence into a vector with a fixed length which is then interpreted by the decoder. The decoder creates an output sequence based on the encoded information. The encoder and decoder share information using the internal vector to exchange encoded information that is understood by both layers. Therefore the conjunction is referenced in literature as Encoder-Decoder LSTM. Since it consists of two layers, whereas one is specialized in understanding the input and one in producing the output, it is commonly used for sequence to sequence problems [24]. Since the presented platform movement prediction problem is also a sequence to sequence problem, the given architecture is expected to

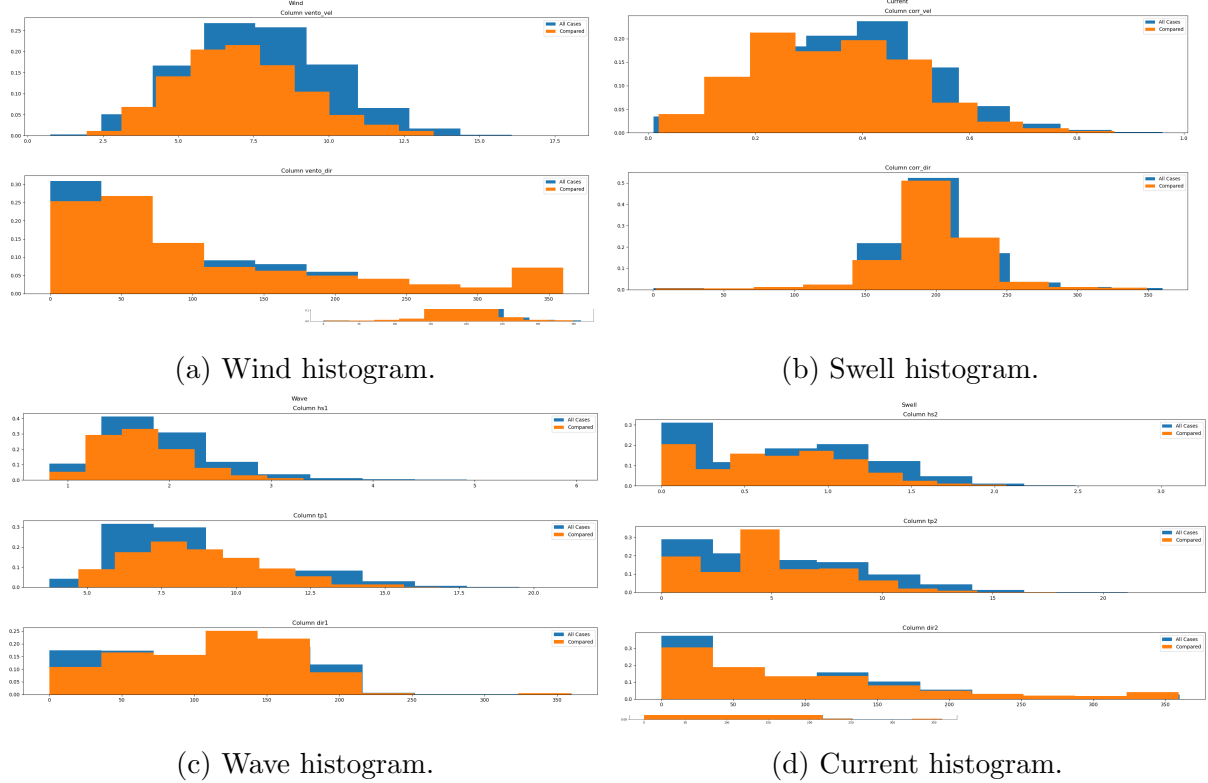


Figure 30: Illustration of the selected files histograms (orange) against the histogram of all 18000 files (blue) for the Long Short Term Memory (LSTM) network.

achieve the most precise predictions.

To employ this structure additional layers are needed that are explained in the following. The LSTM model structure implemented is illustrated in Figure 31 and is composed of an encoder and decoder LSTM layer, a repeat vector layer and two dense layer classes, that are wrapped in a time distributed class.

The first LSTM Layer consists of 200 units. Each unit of the layer gives a response to the seen input, which means that the output of this layer is a vector of 200 values. This layer is considered the encoder and it is trained to understand the input and translating it into a fixed length vector. In this project the Layer takes a two dimensional Array consisting of the last 1000 time steps of Surge, Sway and Yaw to decode it into a vector of 200 values, that carries all the important information for the decoder Layer.

Since the decoder LSTM layer needs a 2 dimensional input, a repeat vector layer is needed. The repeat vector layer repeats its input n times. In this case the input is repeated corresponding to the number of output steps. As the output of the predictor has a length of 400 steps, the encoded vector is repeated 400 times.

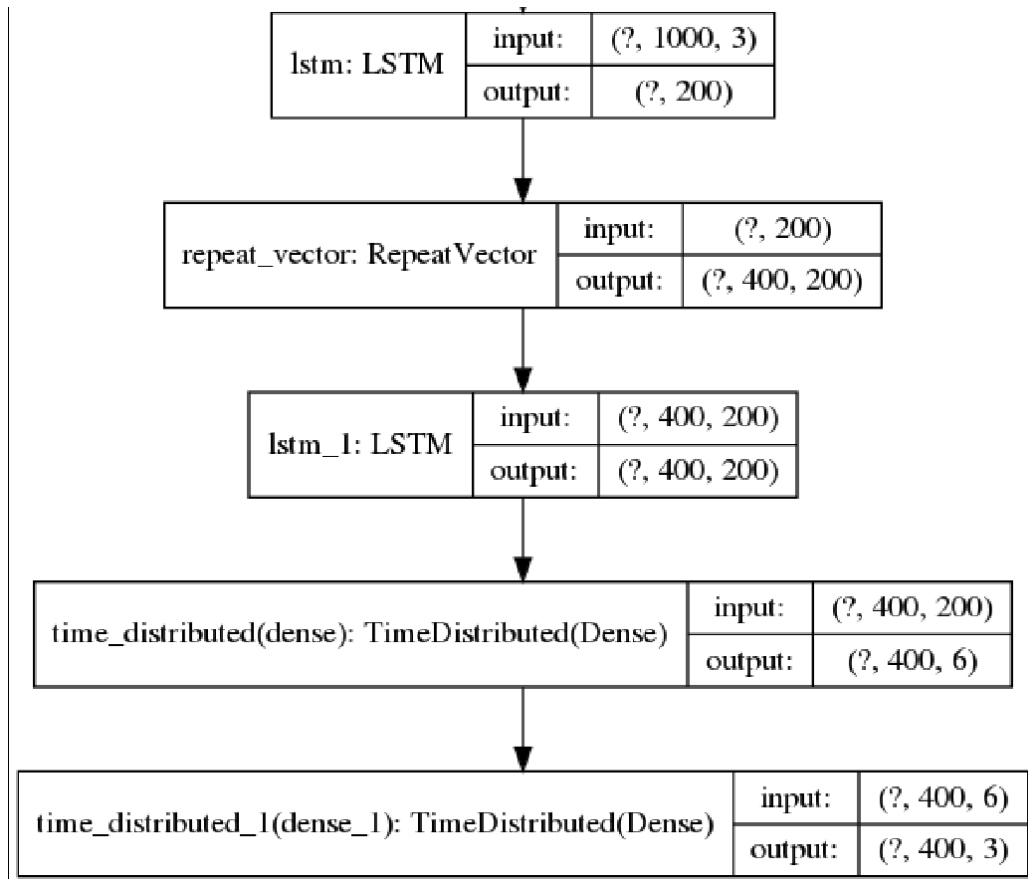


Figure 31: Illustration of the used LSTM architecture. In brackets are the number of training units, due to its variable size (depending on the number of selected environmental conditions) it is marked with "?", the second number represents the number of input steps and the last number the number of features

The second LSTM layer can be understood as the decoder of the structure. It consists also of 200 units. Each unit gives a response based on the encoded message, which in our case consists of the array of 200 values of the Encoder Layer repeated 400 times. This layer returns the hidden state for each input time step creating a two dimensional output. In our case the output as seen in figure 65 consists therefore of an two dimensional array consisting of the 400 hidden states of the 200 units.

After the internal vector is decoded, it needs to be translated in the needed output form. This is accomplished by 2 dense layers that reduce the 200 output values to the desired 3 DoF motion variables representing the horizontal platform motion (surge, sway, and yaw). Since dense layers work with one-dimensional input, they are wrapped in a time distributed layer, giving them the ability to understand the two dimensional input data.

The LSTM receives the last 1000 seconds of the platform horizontal motion – surge, sway, and yaw – to predicts 400 seconds of the horizontal platform motion.

All layers presented used ReLU as activation function and a stochastic gradient descent (SGD) optimizer algorithm as optimization function. Further information on the basic principles can be found in chapter A.1.

The training was based on the Keras Convolution Neural Network Layer (CNN), that is a Keras LSTM implementation specialized for improved GPU performance, by doing majority of the calculations in parallel.¹

4.3.3 LSTM Training

The LSTM network was trained for 1500 epochs. The first 700 epochs are shown in Figure 32. The blue curve shows the training accuracy and the orange curve shows the validation accuracy. The Validation was done using 200 environmental conditions. The Training was only stopped after the validation accuracy did not improve significantly within 200 epochs. In the project a significant improvement of the validation accuracy was considered to be an improvement of more than $1 * e^{-5}$. Usually the improvement became insignificant after 1500 epochs, depending on the Learning rate of the optimizer.

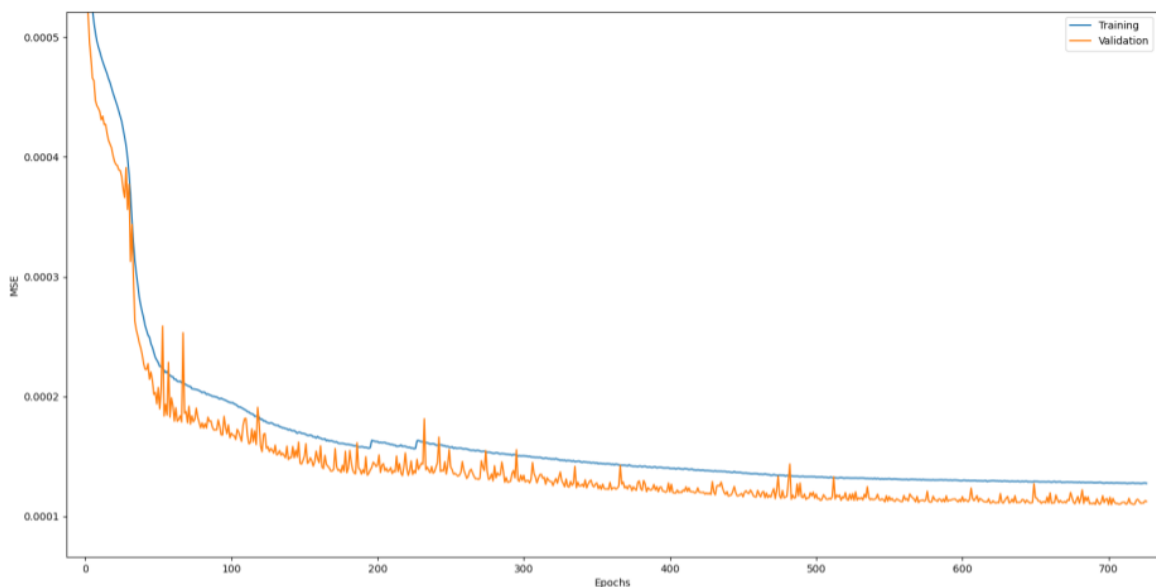


Figure 32: Training Curve of the first 700 epochs of the presented LSTM network using 1000 training environmental conditions (blue) and 200 validation environmental conditions (orange). The vertical axis represents the mean error and the horizontal axis represents the number of epochs. The error decreases steadily with number of epochs.

The training was done with 1000 separate environmental conditions, with 200 seconds

¹https://www.tensorflow.org/api_docs/python/tf/compat/v1/keras/layers/CuDNNLSTM

stride of the data window between consecutive training units (see Section 4.1.2 for details). The algorithm predicted 400 seconds based on 1000 seconds of horizontal platform motion, as shown in Figure 33.

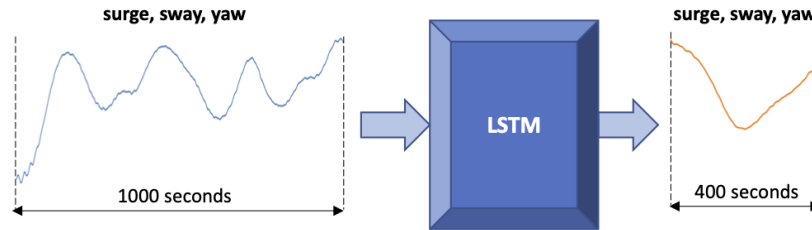


Figure 33: The LSTM model using the last 1000 seconds of the features surge, sway and yaw to predict 400 seconds of these three features.

4.3.4 LSTM Prediction

After the model was trained, different environmental conditions were then used for testing the models prediction accuracy. Figure 34 shows a prediction of platform motion without mooring line Failure. It can be seen that the prediction is reasonable close to the simulated platform motion. The blue lines indicate the borders of a single prediction. For better visualization over a longer period of time consecutive predictions were concatenated in the figures.

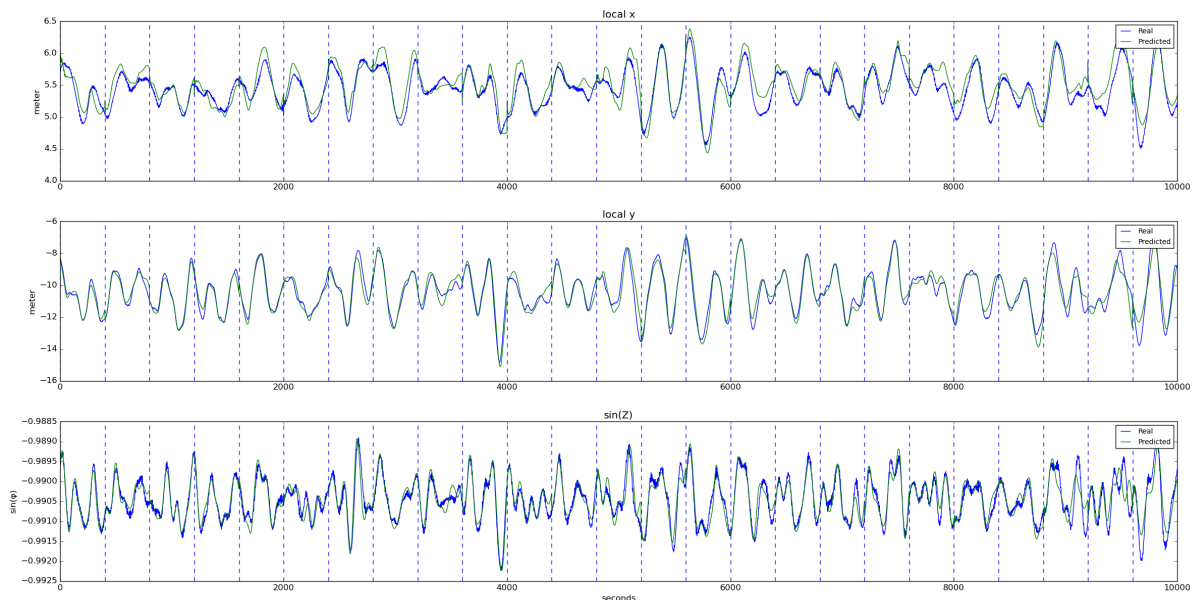


Figure 34: Illustration of LSTM prediction of platform motion under a single environmental condition with all mooring lines intact. The simulated data are in blue and the predicted in green. Local x means surge, local y sway, and $\sin(Z)$ yaw.

Different environmental conditions are selected to analyze the models capability to

predict the platform motion. The selected conditions can be seen in Table 10. The main difference between the selected conditions can be seen in the swell, wind and waves. It can be seen that the wave direction was varied between 123 and 190 degrees, with wave heights between 1,70 meter and 2,8 meter. The wind was varying in height and speed in correlation to the waves.

Table 10: Analyzed environmental conditions

case	hs1	tp1	dir1	hs2	tp2	dir2	hstotal	vento_vel	vento_dir	corr_vel	corr_dir
17432	2.24	6.74	123.9	0.00	0.00	0.0	2.24	9.34	112.9	0.27	238.37
25856	1.71	9.25	149.3	1.23	6.59	96.7	2.11	7.70	69.0	0.46	190.20
25874	2.84	17.04	192.2	0.00	0.00	0.0	2.84	7.77	208.6	0.45	216.93

The trained predictor is able to accurately predict the slow frequency motions of the platform, while it is not able to predict the rapid frequency motions. Index 25856 represents a rapid surge and yaw motion, as it can be seen in figure 35.

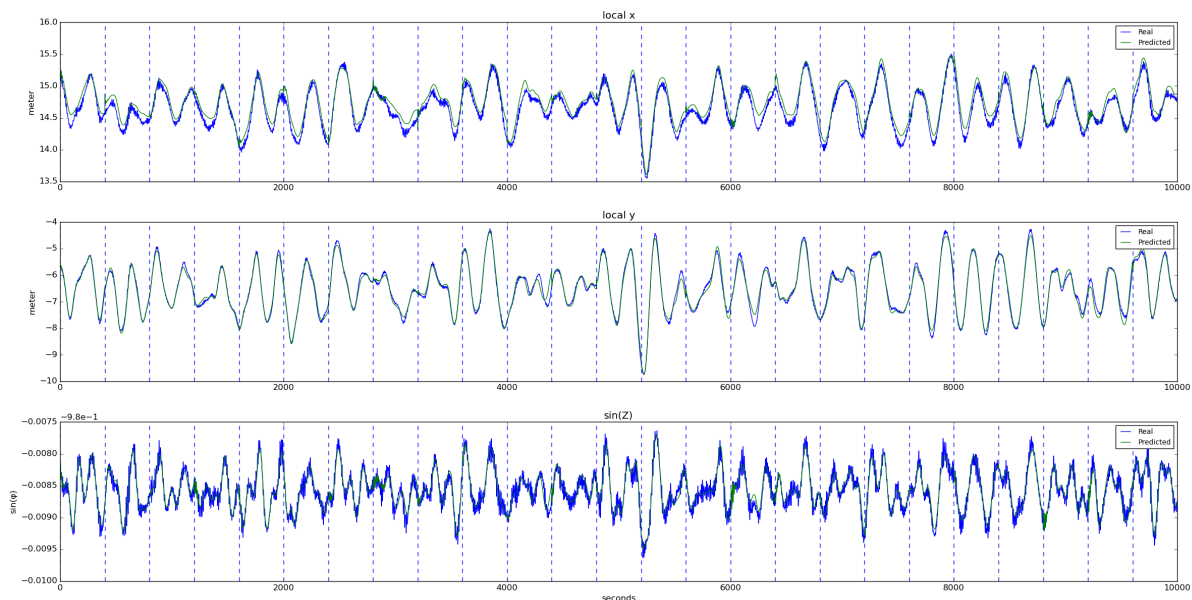


Figure 35: Illustration of LSTM prediction on a single environmental condition with all mooring lines intact. The simulated data are in blue and the predicted in green.

A zoomed version, presented in figure 36, shows that the LSTM predictor is predicting only the slow components of the platform motion accurately but is not able to follow the high frequency motions.

Figure 37 shows one of the most rapid platform motions found in all of the cases. It can be seen that the algorithm is predicting the slow most influential components but not the high frequency components of the platform motion. Since these cases occur rarely, there were only few selected cases with high frequency motion components in the training data.

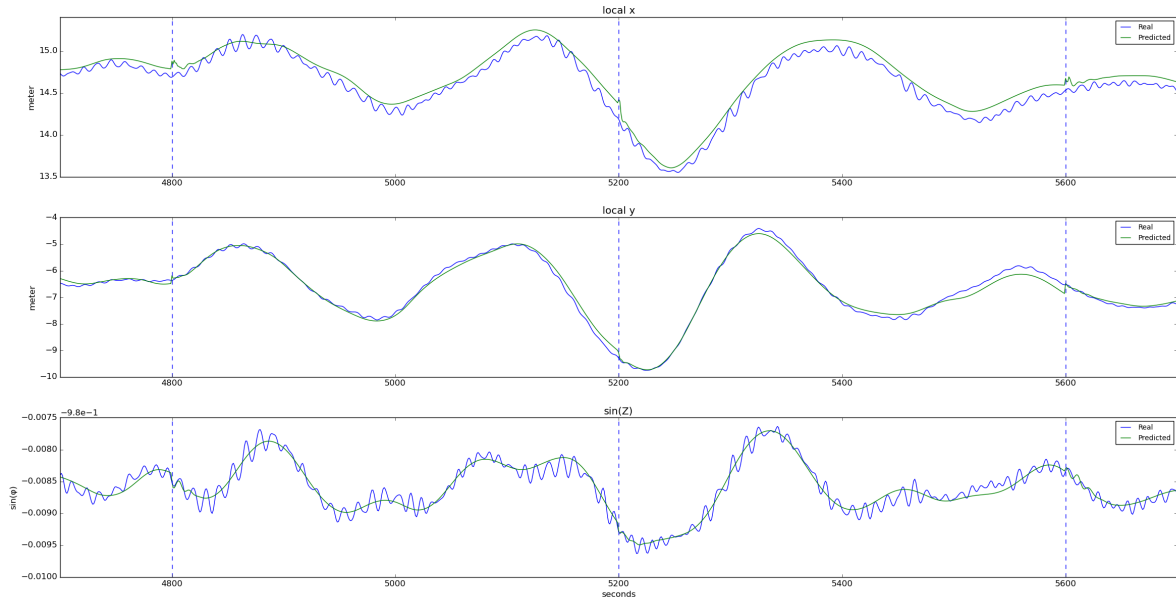


Figure 36: Zoomed illustration of the LSTM prediction on a single environmental condition with all mooring lines intact. The simulated data are in blue and the predicted in green, with the topmost graph representing the Local x position (surge), the middle graph local y position (sway), and the bottom graph the $\sin(Z)$ angle (yaw).

The same analysis was then done with Mooring line failure cases. Figure 38 shows the same environmental condition that also can be seen in figure 34, with a simulated Mooring Line failure of L1 after 3500 seconds. The failure is clearly visible as a change in positional offset. After the failure it can be observed that the predictor has difficulties predicting the platform motion. In a zoomed version there can be seen a clear difference between simulation and prediction when the Line failure happens, since the platform changes its local position after the failure of the Line and the predictor does not predict this change in position.

Figure 39 shows the two predictions done during the period of Line failure. The predictor predicts the motion staying close to the predicted position while the simulated platform position changes 3 meters in surge and sway from its old position. It can be observed that the predictor has problems predicting the platform motion in the new location.

The same behaviour can be observed for all different kind of environmental conditions and all mooring lines. The predictor is not able to follow the platform motion after a mooring line failure, leading to a permanent offset between predicted and simulated platform motion. Figure 40 gives an overview over different mooring Line failures in different environmental conditions. Depending on the Mooring Line failure the seen offset differs in sign and altitude. It can be seen that the offset is correlated to the orientation

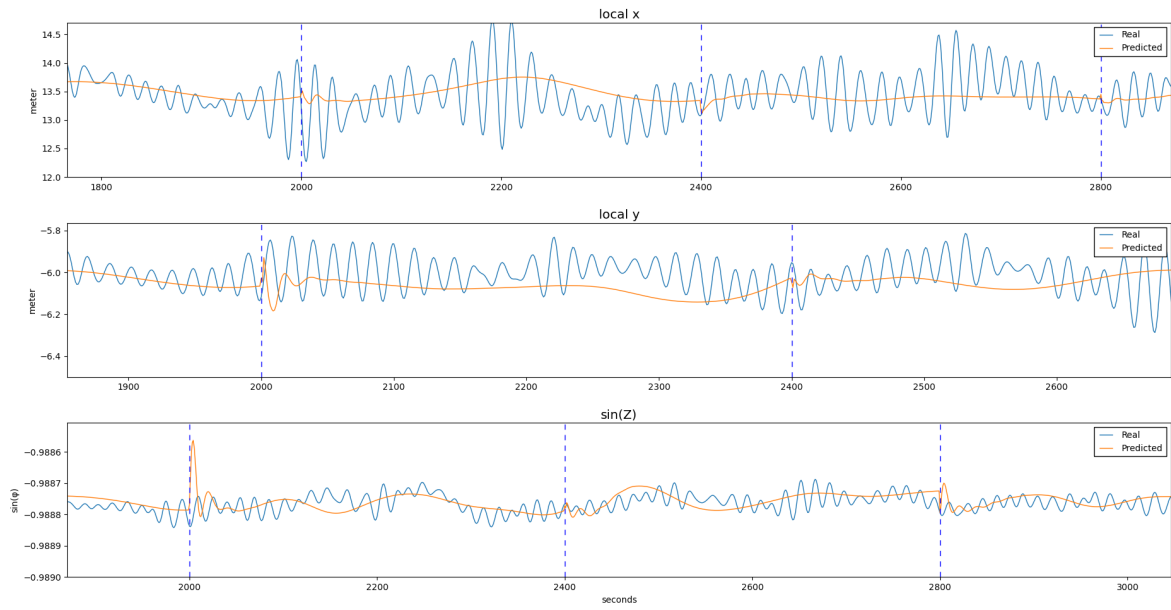


Figure 37: Zoomed illustration of LSTM prediction on a single environmental condition with all mooring lines intact and rapid motion. The simulated data are in blue and the predicted in green, with the topmost graph representing the Local x position (surge), the middle graph local y position (sway), and the bottom graph the $\sin(Z)$ angle (yaw).

of the line to the platform.

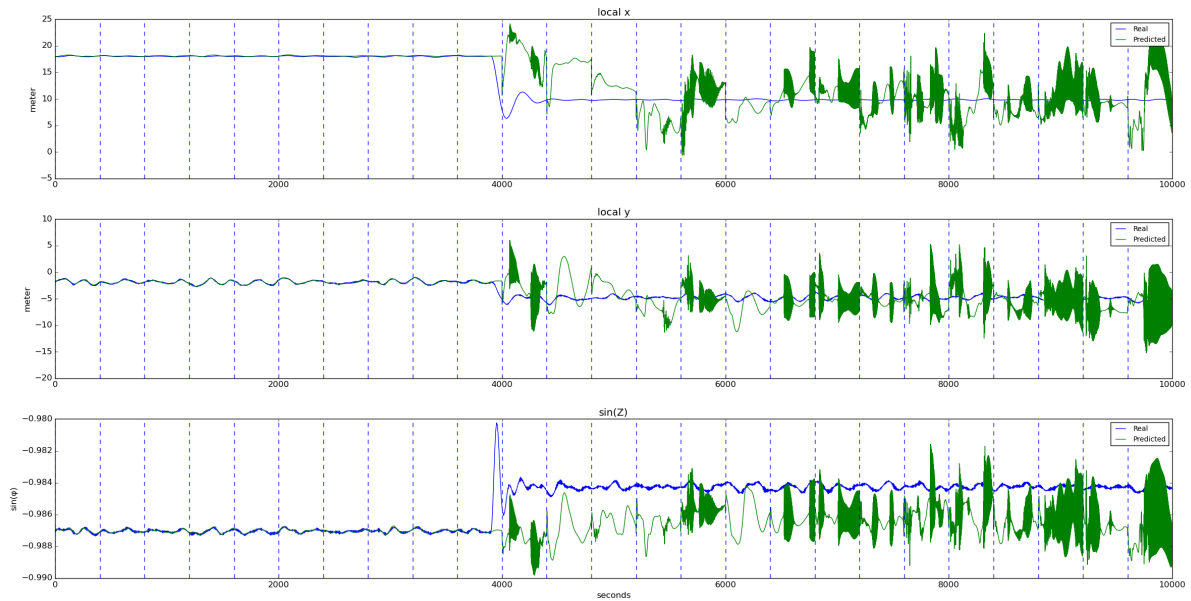


Figure 38: Illustration of LSTM prediction on a single environmental condition with a Mooring Line failure of Line 1 at 3500 seconds. The topmost graph representing the Local x (surge), the middle graph representing local y (sway), and the graph at the bottom representing $\sin(Z)$ (yaw).

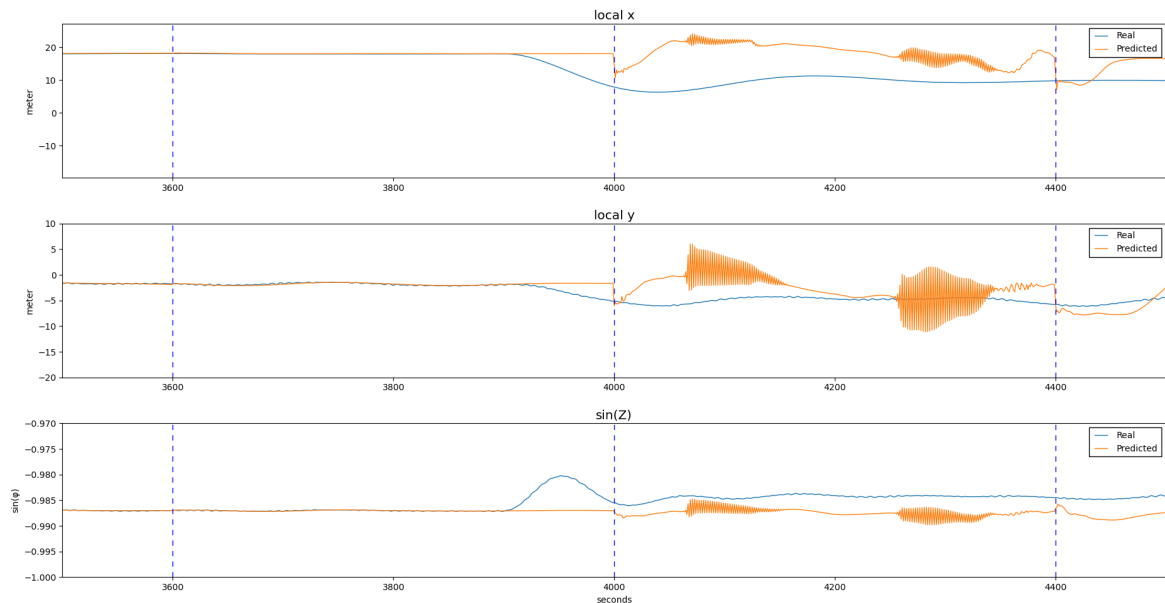
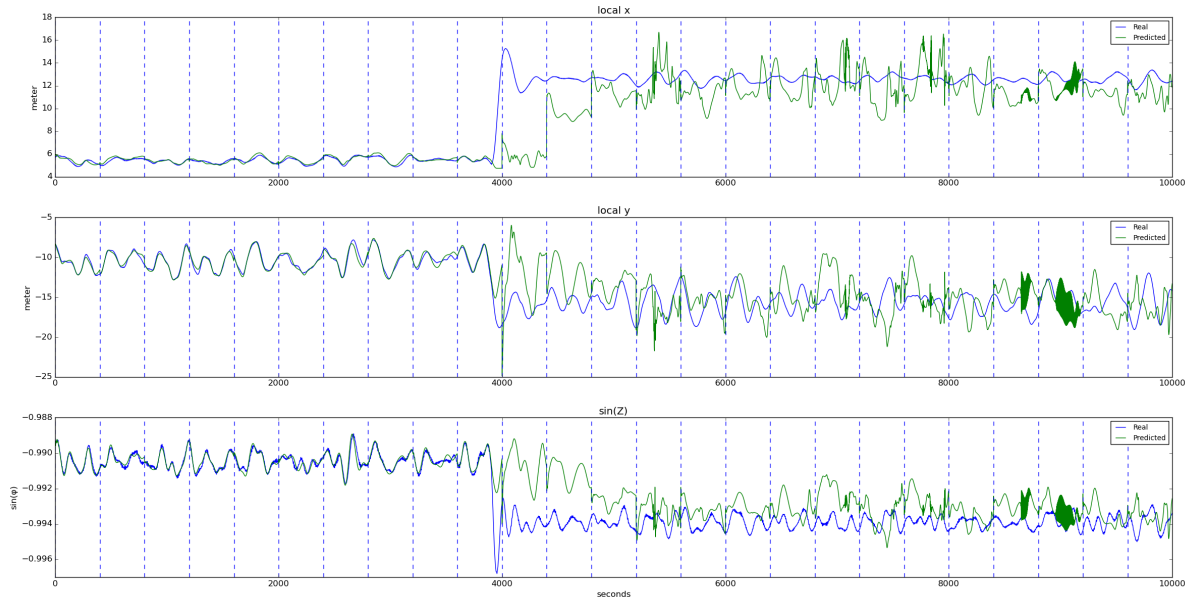
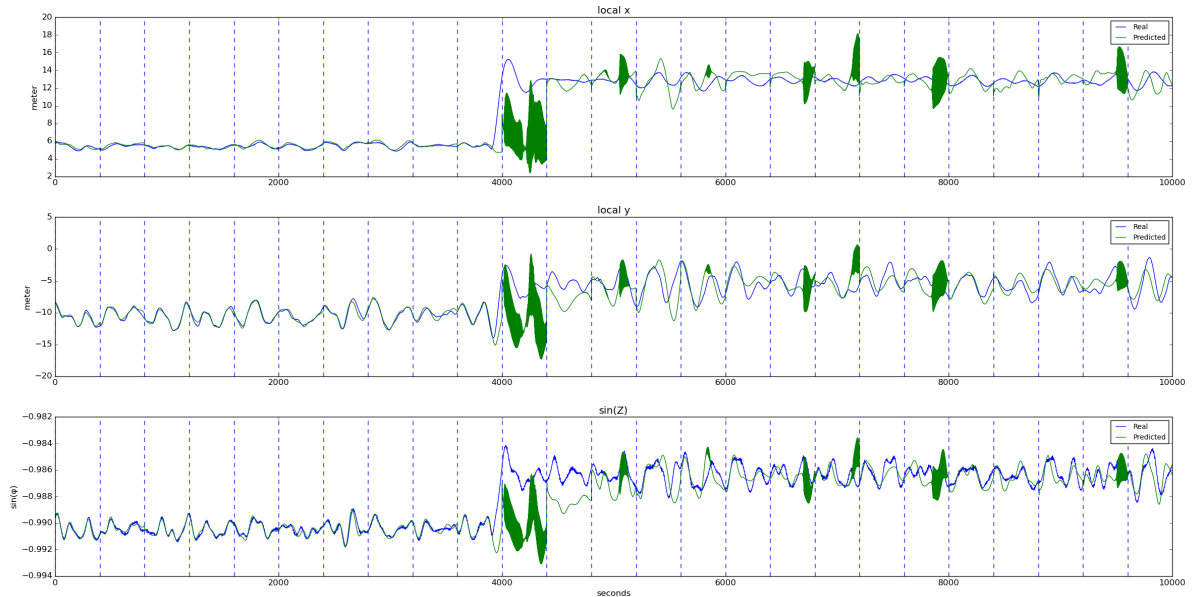


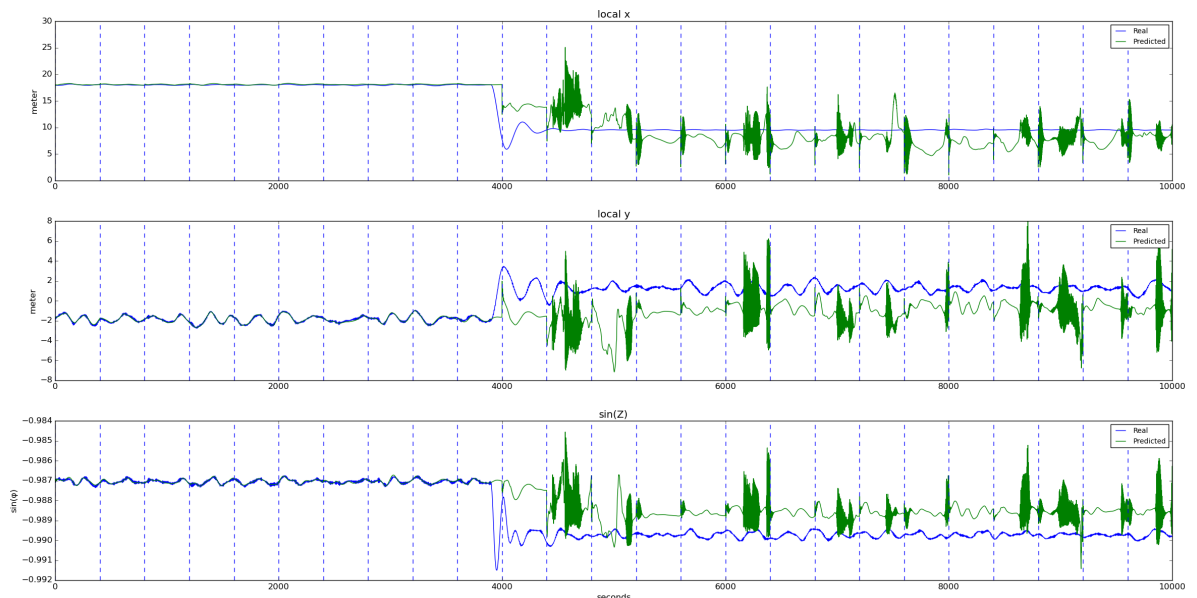
Figure 39: Zoomed illustration of LSTM prediction on a single environmental condition with a Line failure of Mooring Line 1 at 3500 seconds. The topmost graph representing the Local x (surge), the middle graph representing local y (sway), and the graph at the bottom representing $\sin(Z)$ (yaw).



(a) Mooring Line failure Line 09



(b) Mooring Line failure Line 12



(c) Mooring Line failure Line 18

4.4 Discussion

In this section the results of the MLP and LSTM models implemented are discussed. In Section 3, the proposal to build two predictor models, MLP and LSTM, was presented hypothesizing that an irregularity in platform motions can be a good indicator for mooring line failure. Results of the two predictors presented in Section 4.2 for the MLP model and Section 4.3 for the LSTM model support the made hypothesis.

It is assumed that the platform motion changes in its intensity as well as in its frequency after a line failure occurs. Since both predictor models were only trained to respond well to platform motions with all mooring lines intact, the irregular motion after a line failure can be seen as a difference between the simulated and predicted motions.

As it can be seen for the MLP in Figure 20 and LSTM in Figure 34, both models were capable of predicting the motions of the platforms. The results showed that the LSTM model performed better at predicting the platform motions than the MLP predictor, by being capable to predict more accurately for a longer period of time. This can be seen in Figures 26, 28, 29 for the MLP network and in Figures 40a, 40b, 40c, 39 for the LSTM network. An explanation for this may be due to the nature of the LSTM model which is a recurrent neural network and thus can remember past information, which the MLP model as a feed-forward network cannot do.

After mooring line breakage the MLP model was unable to predict the platform motions accurately. The LSTM model also demonstrated difficulty in predicting the platform motion after mooring line failure but to a lesser extent when compared to the MLP network on the same platform motion. This behaviour can be attributed to the fact that both models were trained on platform motions with intact mooring lines. So after mooring line failure, the motions of the platform with a compromised mooring system are unknown to these trained models.

Both models were also tested on different environmental conditions that had different sea states. The MLP model predicted the oscillation of the simulated platform in calm and mild environmental conditions but for a stormy environmental condition, the MLP could not predict the rapid motions of the platform. The LSTM performed better in all cases when compared against the MLP model. However, LSTM also found it difficult to predict the motion of the platform in environmental conditions with rapid oscillations, predicting only the slow, most influential components of the motion in these conditions. A possible reason why both models found it difficult to predict the platform motion of

environmental condition with rapid oscillation could be because the models were trained with only few environmental conditions of these types in their training dataset. In the entire environmental condition measurements gotten from the weather station located in Campos Basin (Bacia de Campos) of Rio de Janeiro (RJ), Brazil, since 2003 (see Section 4.1.1) there were only 100 environmental conditions with wave heights greater than 4 meters, which characteristically lead to motions with faster frequency components .

The LSTM network used fewer platform movement variables as inputs (3 horizontal platform motion variables) when compared to the MLP network, that used surge, sway, yaw, roll, pitch and heave as input. However, they both predicted the same 3 platform motion features (surge, sway and yaw).

The LSTM model was able to handle a larger input (i.e., more time steps at the input) than the MLP network, and it was also able to forecast a longer output. The LSTM network used an input of 1000 seconds for surge, sway and yaw, and the MLP used a smaller input of 600 seconds of all 6 DoF. The forecast time for the LSTM network (400 seconds) was also four times larger than the 100 second forecast time for the MLP network. Therefore, the LSTM model can be used for longer platform motion forecasts than the MLP model.

A comparison of the number of platform motions data used for training and validation in the training phase shows that MLP used five times the number than the LSTM model. On the other hand, LSTM takes much longer to be trained. While MLP was trained in approximately 7 hours, LSTM required approximately 20 hours to be trained.

5 COMPARATOR

After the predictor is trained, as explained in Section 4.2 and 4.3, it is used to predict platform motion based on the previous motion. The test units must have the same dimensions as those defined for the pair $\langle \text{input} - \text{output} \rangle$ of the training units (see Section 4.1.2). The input of a data-set is fed to the trained predictor, which then predicts the output of the set.

The idea is to provide input data to the predictors and compare the predicted output of the predictors with the simulated data, measuring the disparity between them. Assuming that a mooring line failure leads to a significantly greater disparity when compared to platform motions without mooring line failure, the disparity can then be used to identify mooring line failures.

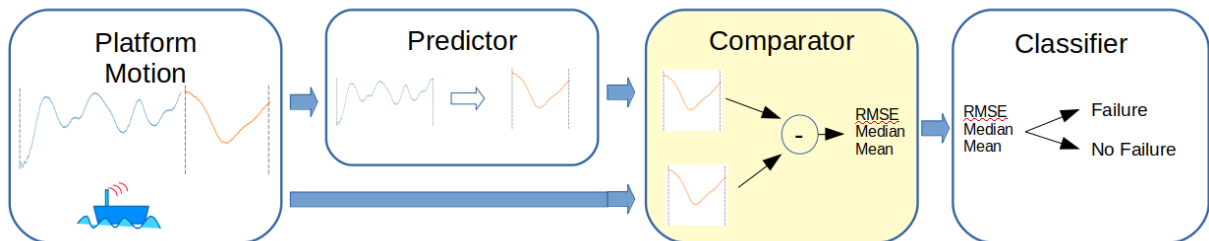


Figure 41: Figure showing the comparator module as part of the modular system structure

The comparator is used to calculate error scores reflecting the disparity between simulation and prediction. To improve the reliability of these error scores, the calculation is based on multiple predictions. Additionally different error scores are calculated to lower the influence of outliers and inaccurate measurements. To perform the error index calculation, error windows are created based on the predictors output. Section 5.1 explains the error window creation and the error score calculations. In section 5.2 shows the calculated error indexed for the MLP predictor and section 5.3 shows the error scores of the LSTM predictor. In the last section of this chapter the found error scores are then discussed and the two predictors are compared against each other. The comparator is the second

module of the data pipeline as it can be seen in figure 41.

The next step is then to identify mooring line failure using the calculated error scores. Since the network was only trained using scenarios without line failures, the difference between simulated and predicted platform motion is expected to be greater for the cases with line failure.

5.1 Error Score Calculation

As explained in section 4 the trained predictors are used to predict different testing units. The tested units are selected so that for each simulated time step at least one predicted value exists. If multiple values exist for the same time step the mean value is calculated. After concatenating all predicted platform motions the prediction can be plotted against the simulated platform motion. The difference between the simulated and predicted motion is given by

$$\Delta = Y_{pred} - Y_{sim}. \quad (5.1)$$

Where Y_{sim} is the simulated motion and Y_{pred} is the predicted motion. The result can be plotted as seen in Figure 42, where the red line is the difference between the prediction (green) and simulation (blue).

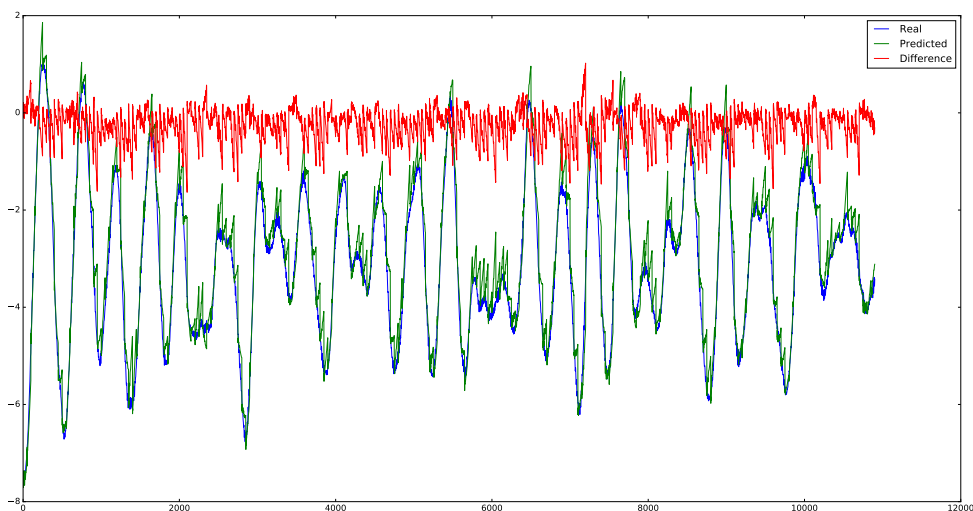


Figure 42: In this plot are the actual motion (simulated data here, in blue), the predicted values (in green) and the difference between them (in red).

To calculate the errors, the difference graph is then split into windows, hereafter referred to as error windows, with the size of two prediction time intervals (which, as

already mentioned, is a parameter of the system and depends on the predictor used and the training done). The stride of the error windows is one prediction window. Figure 15 uses a prediction window of 100 seconds resulting in the error window size of 200 seconds with stride 100 seconds.

Since the error window size is two times longer than the prediction window, the windows overlap. For example, for a predictor predicting 50 seconds, the error window would have a size of 100 seconds and its stride would be 50 seconds, as it can be seen in Figure 43. For visualization purposes the mean value is shown when there are multiple predicted values for the same time step.

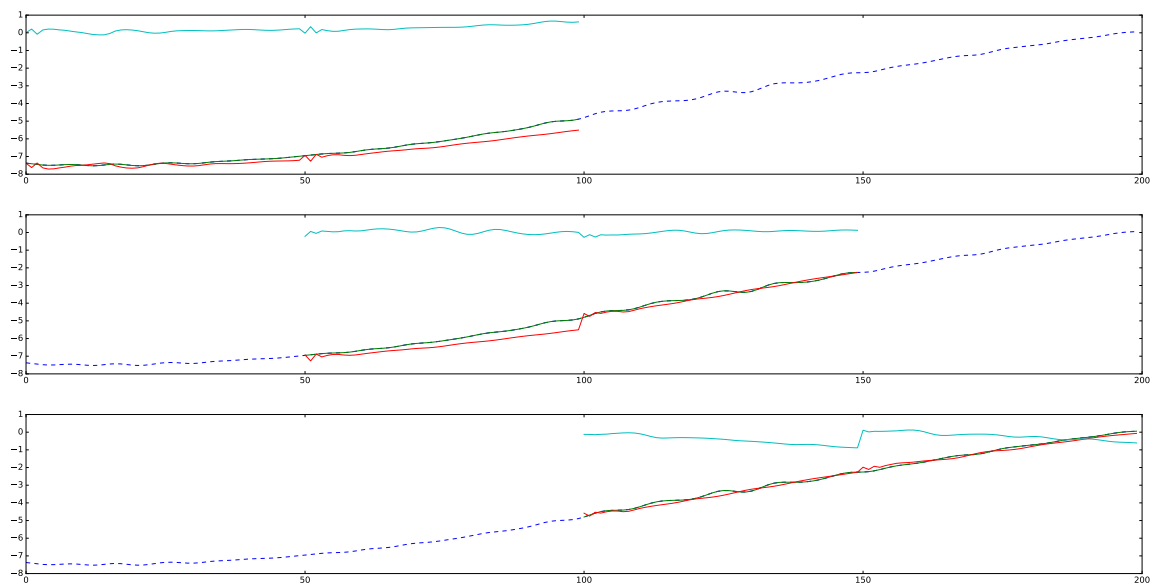


Figure 43: Prediction step by step. Each graph shows two 50s predictions concatenated in a 100s error window, and the respective difference (in green) between the predictions (in red) and the actual data (dashed in blue). The 50 second stride between one graph and another can be seen as time difference between the graphs.

For each error window the root mean square error, mean and median error were calculated. The calculation was done using

$$ME = \frac{\sum_{i=0}^n \Delta_i}{n}, \quad (5.2)$$

for the mean error (ME),

$$MedE = V_{sorted}[(N - 1)/2] \quad (5.3)$$

for the median error (MedE) where V_{sorted} represents a sorted array of the Δ values and N represents the number of array elements. The RMSE is calculated by

$$RMSE = \sqrt{\frac{\sum_{i=0}^n \Delta_i^2}{n}}. \quad (5.4)$$

with n being the number of time steps in the error window and Δ the point-to-point difference between the predicted and the simulated data.

The errors were then plotted against the prediction and simulated platform motion as it can be seen in Figure 44. The three errors were calculated for each error window.



Figure 44: Error Calculation. Top: predicted (orange) and actual (blue) motion values. Bottom: RMSE (green) median (orange) and mean (blue) error scores for each error window.

5.1.1 Scatter plot visualization

Since both predictors predict the horizontal motion, which consists of 3 degrees of freedom, each error score can be interpreted three dimensional with every feature as a dimension and can therefore be plotted as a 3D Scatter plot. The error score in surge is presented on the x axis, the error score in sway on the y axis and the error score in yaw on the z axis.

The error calculation is then repeated for all the environmental conditions available and the results are all visualized in a 3D scatter plot, each calculated error score represented as a point in the 3D scatter plot. The RMSE, Mean and Median errors are then plotted in three different scatter plots.

5.2 MLP Error Scores

Section 5.1 gave a detailed description of the steps required for calculating the errors between our implemented model prediction and the simulated platform motion. This section presents the error scores for different environmental conditions the MLP model was tested on, showing the correlation of error score, environmental condition and mooring line status. Using the error scores of six different environmental conditions with different mooring line setup, the RMSE, mean and median errors of these conditions are compared.

Table 11 compares the RMSE errors of six different environmental condition. It can be seen that compared to the surge RMSE of case 57 – which has all its mooring lines intact – with the same case 57 but with mooring L18 Failure the error becomes 10 times bigger. The same difference is seen for sway and yaw. There is always a difference in error score between situations with intact and compromised mooring lines under a certain environmental condition.

Table 11: RMSE MLP error scores

RMSE	surge	sway	yaw
Case 57	0.257	1.656	5.483 e-04
Case 600	0.300	0.619	2.337 e-04
Case 8818	0.748	0.579	2.066 e-04
Failure L1 Case 57	6.061	6.289	3.594 e-03
Failure L9 Case 57	5.761	10.038	4.594 e-03
Failure L12 Case 57	7.756	5.802	4.523 e-03
Failure L18 Case 57	10.425	1.780	3.241 e-03

Table 12 compares median errors of these six environmental conditions and Table 13 compares the mean errors. The same observations are true for these comparisons. The error score after a line failure is 10 times higher for cases with a compromised mooring line when compared to cases with intact mooring systems. A clear difference is noticeable between the scenario with and without Mooring Line Failure as it was expected.

After calculating all error scores for all simulated environmental conditions the results are plotted in a 3D scatter plot as described in section 5.1.1.

Figure 45 shows the scatter plot of the RMSE scores of all available training sets . It can be seen that there is a clear separation between the cases with all Lines intact and the cases that show a Mooring Line failure. As aforementioned the error score gives no further information about the location of the Line since the error calculates only positive error scores.

Table 12: Median MLP error scores

Median Error	surge	sway	yaw
Case 57	0.232	-0.878	-2.058 e-04
Case 600	-0.264	-0.509	-1.007 e-04
Case 8818	-0.472	-0.546	-1.288 e-04
Failure L1 Case 57	-5.415	-6.225	3.143 e-03
Failure L9 Case 57	5.766	-9.913	-4.827 e-03
Failure L12 Case 57	7.430	5.609	4.512 e-03
Failure L18 Case 57	-10.019	-08.416	-3.102 e-03

Table 13: Mean MLP error scores

Mean Error	surge	sway	yaw
Case 57	0.240	-0.982	-2.445 e-04
Case 600	-0.237	-0.540	-1.311 e-04
Case 8818	-0.438	-0.555	-1.255 e-04
Failure L1 Case 57	-5.296	-6.086	3.097 e-03
Failure L9 Case 57	5.478	-9.869	-4.486 e-03
Failure L12 Case 57	7.430	5.398	4.265 e-03
Failure L18 Case 57	-10.026	-1.010	-3.007 e-03

Figure 46 shows the scatter plot of all calculated Mean error scores. It can be seen that there is a clear separation between the cases with all Lines intact and the cases that show a Mooring Line failure. Furthermore the cases can be divided into four groups that represent the four different cardinal directions.

Figure 47 shows the scatter plot of all calculated Median error scores. It can be seen that there is a clear separation between the cases with all Lines intact and the cases that show a Mooring Line failure. Furthermore the cases can be divided into four groups that represent the four different cardinal directions. The error plot is similar to the mean error scatter plot as expected.

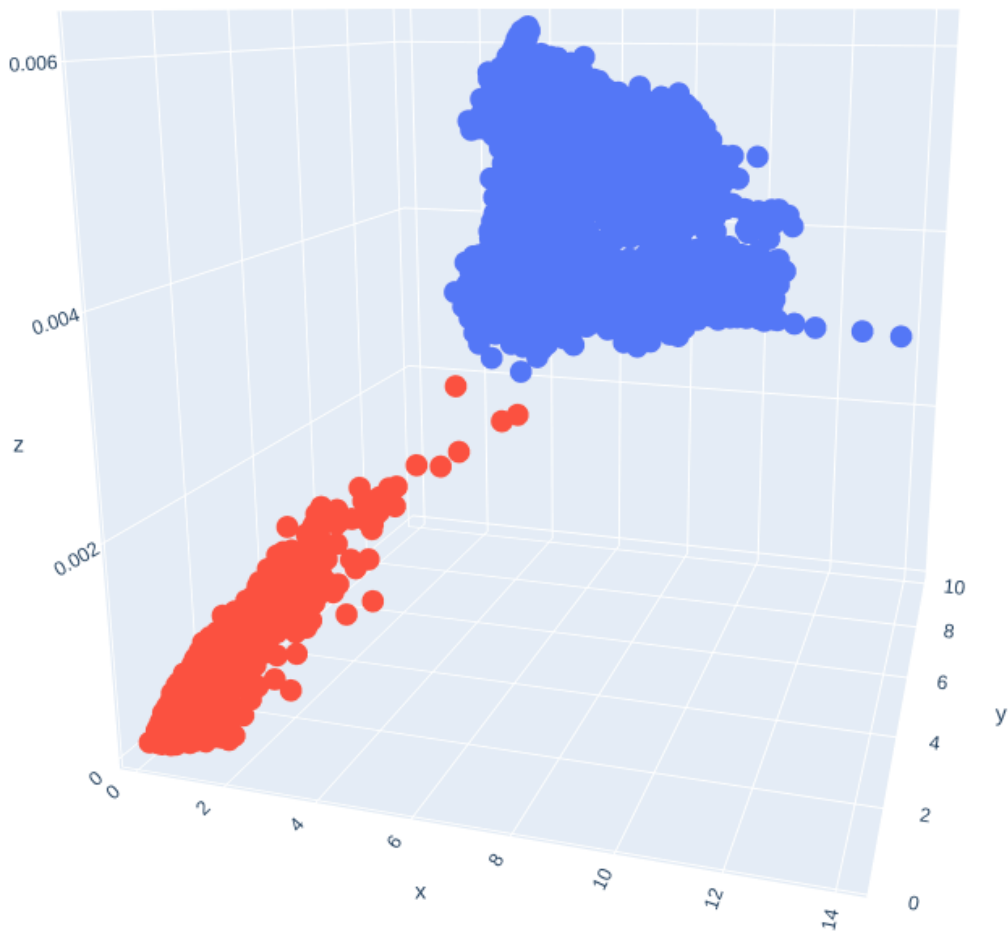


Figure 45: RMSE error score of features; surge, sway and yaw indexes for all environmental conditions in the test set. The red circles represent cases without mooring line failure and the blue circles represent cases with a mooring line failures. The x axis represents the error score for surge, the y axis represents sway and the z axis represents the error score for the yaw feature

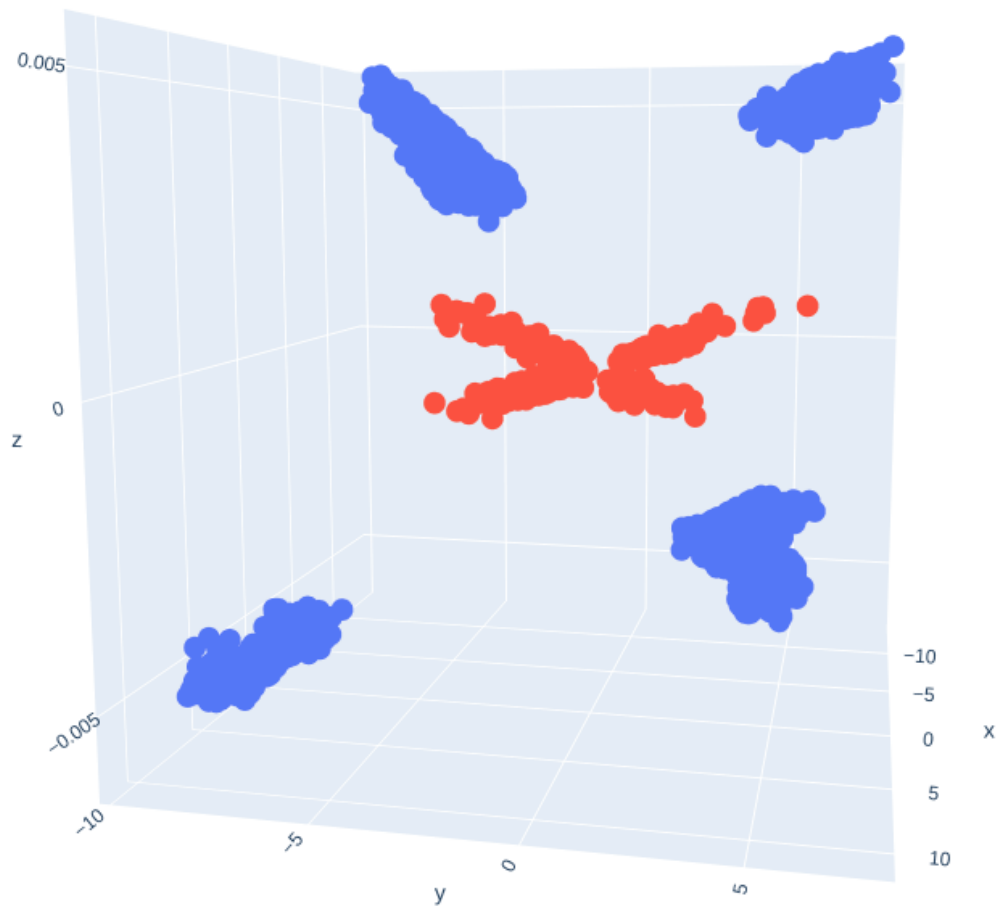


Figure 46: Mean error score of features; surge, sway and yaw of indexes for all environmental conditions. The red circles represent cases without mooring line failure, and the blue circles represent cases with mooring line failure. The x axis represents the error score for surge, the y axis represents sway and the z axis represents the error score for the yaw feature

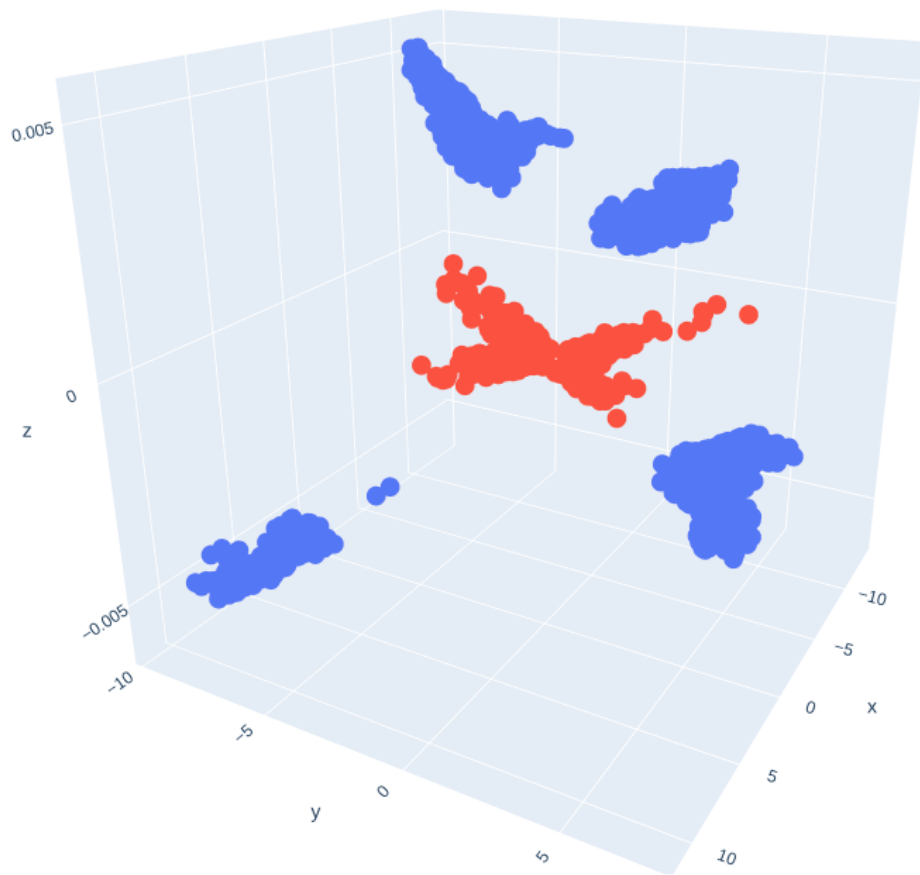


Figure 47: Median error score of features; surge, sway and yaw of indexes for all environmental conditions. The red circles represent cases without mooring line failure, and the blue circles represent cases with mooring line failure. The x axis represents the error score for surge, the y axis represents sway and the z axis represents the error score for the yaw feature

5.3 LSTM error Scores

As mentioned in the theoretical chapter 4.1.2 different error scores can be calculated using the difference between the simulated and predicted platform motion. The following chapter is showing the calculated error scores for the different error types.

Table 14: LSTM: RMSE Error scores

RMSE Error	surge	sway	yaw
Case 17432	0.196	0.583	3.907 e-04
Case 25856	0.125	0.191	1.088 e-04
Case 25874	0.450	0.163	5.385 e-05
Failure L1	8.306	3.928	2.995 e-03
Failure L9	5.511	4.469	3.102 e-03
Failure L12	5.116	4.706	3.353 e-03
Failure L18	5.206	4.333	2.104 e-03

The RMSE error scores for different cases can be seen in Table 14. The shown cases are the selected environmental conditions from the chapter before, where case 8818 represents a rough environmental condition and 376 a calmer situation. Since RMSE calculates always the square error there can be no statement made regarding the direction of the offset. It can be clearly seen that there is a big gap between the scenarios with and without line failure.

The Mean error scores for different cases can be seen in Table 15. It can be observed that the error score after a line failure is 10 times higher than in cases without a line failure. It can also be observed that line failures from different groups lead to different signs of the error score.

Table 15: LSTM: Mean error scores

Mean Error	surge	sway	yaw
Case 17432	-0.149	-0.182	-5.675 e-05
Case 25856	-0.105	0.047	-2.254 e-05
Case 25874	-0.129	0.049	-2.165 e-05
Failure L1	-7.241	-2.175	2.956 e-03
Failure L9	5.041	-3.650	-2.963 e-03
Failure L12	3.569	3.630	2.997 e-03
Failure L18	-4.612	3.821	-1.954 e-03

The Median error scores for different cases can be seen in Table 16. It can be observed that here the error score after a line failure is also 10 times higher than in cases without a line failure. It can also be observed that line failures from different groups lead to different signs of the error score.

Table 16: LSTM: Median error scores

Median Error	surge	sway	yaw
Case 17432	-0.151	-0.120	-5.499 e-05
Case 25856	-0.110	0.041	-2.440 e-05
Case 25874	-0.151	0.056	-2.166 e-05
Failure L1	-7.666	-2.177	2.817 e-03
Failure L9	5.750	-4.204	-3.134 e-03
Failure L12	2.898	3.375	2.7081 e-03
Failure L18	-4.401	3.294	-2.086 e-03

After calculating all error scores for all simulated environmental conditions the results are plotted in a 3D scatter plot as described in section 5.1.1.

Figure 48 shows the scatter plot of all calculated RMSE scores. It can be seen that there is a clear separation between the cases with all Lines intact and the cases that show a Mooring Line failure. As aforementioned the error score gives no further information about the location of the Line since the error calculates only positive error scores.

Figure 49 shows the scatter plot of all calculated Mean error scores. It can be seen that there is a clear separation between the cases with all Lines intact and the cases that show a Mooring Line failure. Furthermore the cases can be divided into four groups that represent the four different cardinal directions.

Figure 50 shows the scatter plot of all calculated Median error scores. It can be seen that there is a clear separation between the cases with all Lines intact and the cases that show a Mooring Line failure. Furthermore the cases can be divided into four groups that represent the four different cardinal directions. The error plot is similar to the mean error scatter plot as expected.

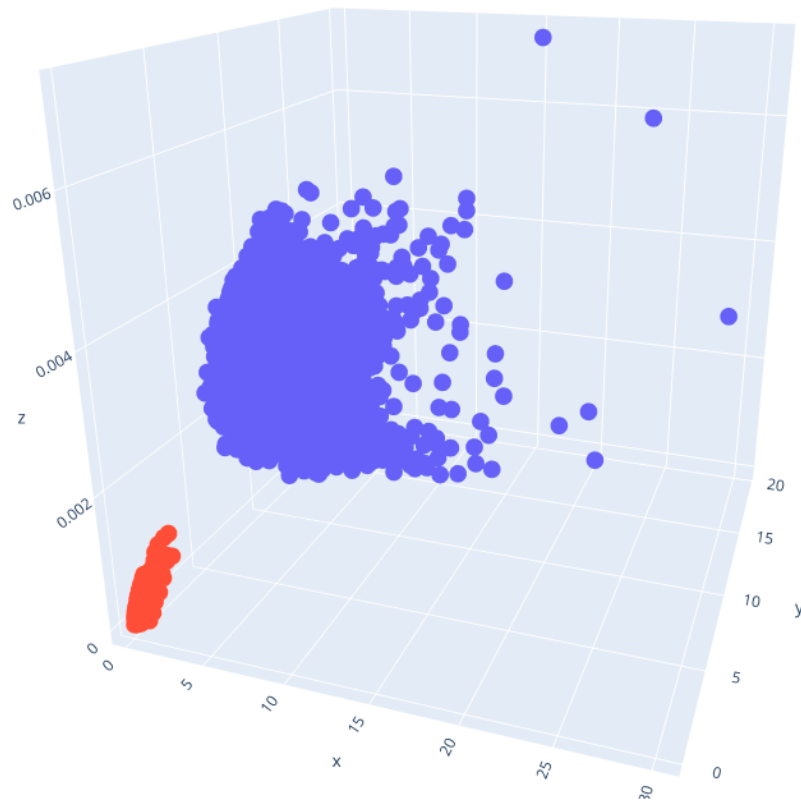


Figure 48: Scatter plot of RMSE error score of features; The red circles represent cases without mooring line failures, and the blue circles represent cases with a mooring line failure. The x axis represents the error score for surge, the y axis represents sway and the z axis represents the error score for the yaw feature.

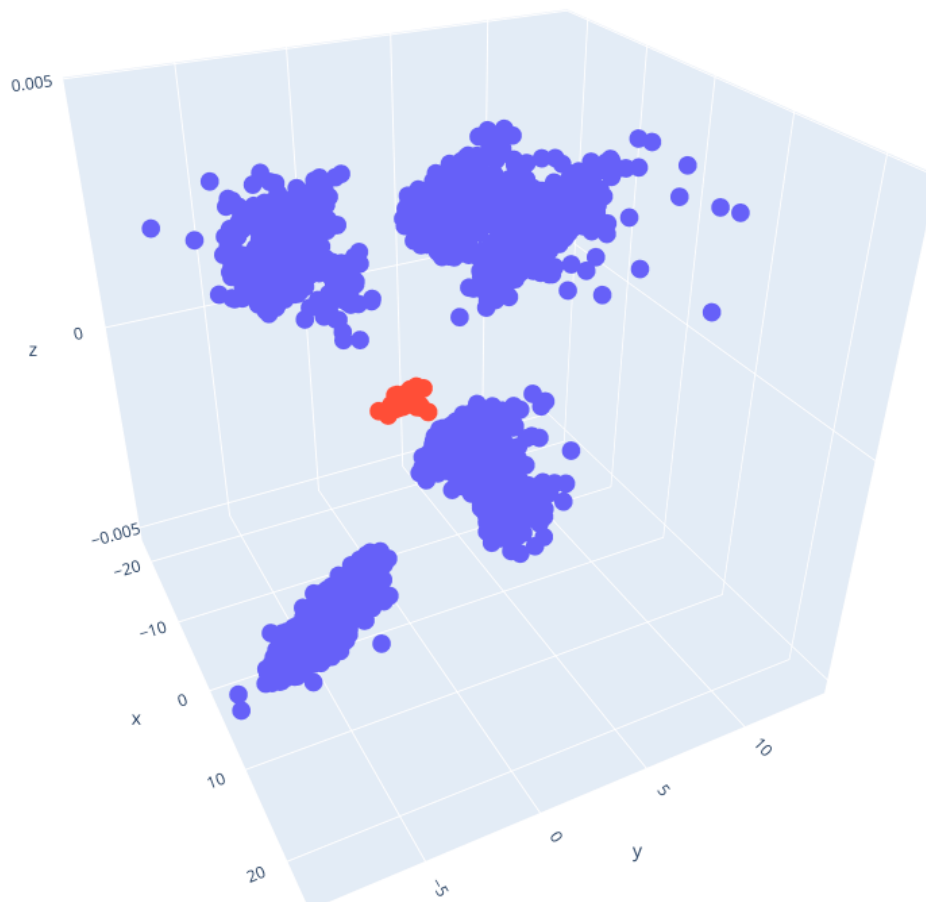


Figure 49: Scatter plot of Mean error score of features. The red circles represent cases without mooring line failure, and the blue circles represent cases with a mooring line failure. The x axis represents the error score for surge, the y axis represents sway and the z axis represents the error score for the yaw feature.

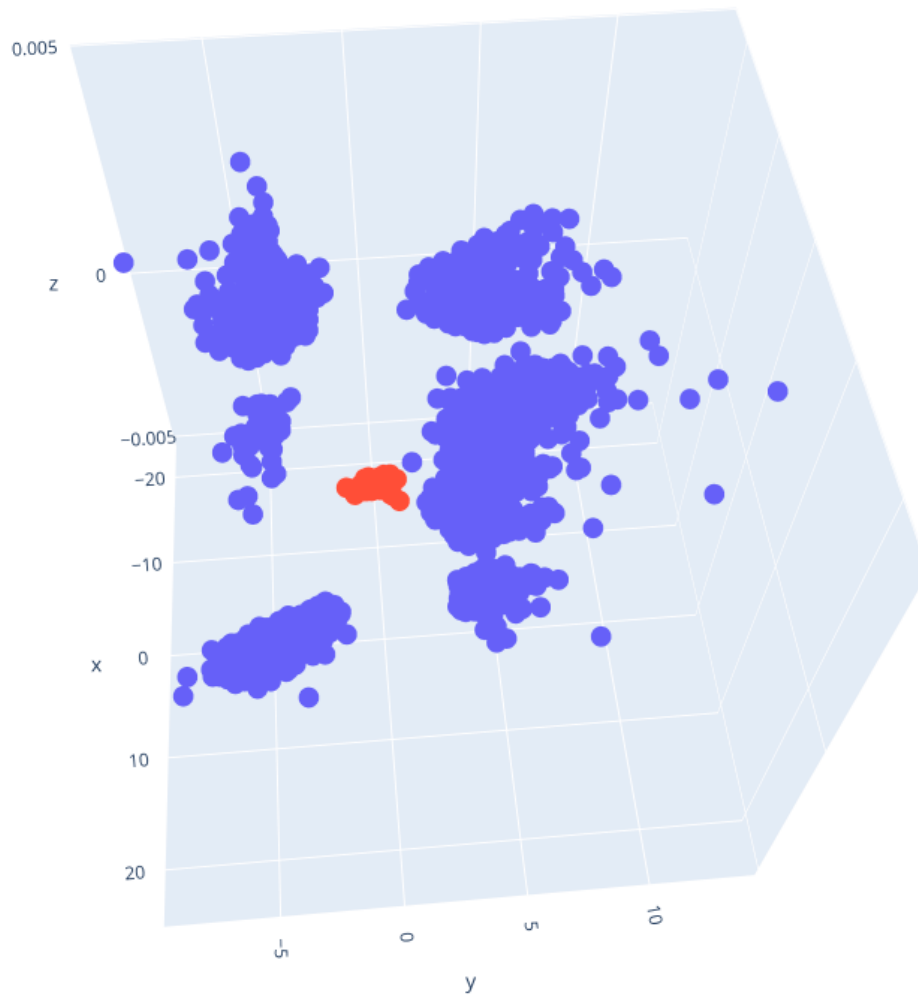


Figure 50: Scatter plot of Median error score of features; surge, sway and yaw of indexes for all environmental conditions. The red circles represent cases without mooring line failure, and the blue circles represent cases with a mooring line failure. The x axis represents the error score for surge, the y axis represents sway and the z axis represents the error score for the yaw feature.

5.4 Discussion

As it was already confirmed in the discussion section of chapter 4, both models were capable of predicting the motions of the platforms. This was also confirmed by comparing the error scores of both models for the same test data. The MLP error scores in Tables 11, 12 and 13, and the LSTM error scores in Tables 14, 15 and 16 show the LSTM errors are always smaller than the MLP error score in all test scenarios.

The in chapter 4 discussed difficulty to predict the platform motion after a line breakage can also be observed, resulting in a high error score after a mooring line failure, due to a high disparity between the simulation and prediction.

In this chapter the predictor error scores of the test units were used to generate 3D scatter plots of the 3 motion variables, sway, surge and yaw. The scatter plots generated based on the RMSE error scores show a clear separation between the cases with all mooring lines intact and the cases that had mooring line failures for both models, showing the applicability for binary classification. As this error depends on the quadratic difference between the measurements, it is not possible to assess the signs of changes when a line fails, meaning that the information of the direction of offset change is lost. The mean and median errors still keep this information.

The scatter plot generated with the mean and median error scores for both models also show a clear separation between the cases with all of its mooring lines intact and the cases with mooring line failures. In Figures 46, 47 for MLP, and Figures 49, 50 for LSTM, five different data groupings can be seen. One group of data in red for cases with intact mooring lines for both models, and the remaining four data groupings in blue indicating the cases with mooring line failure.

It can be further seen that these blue data points were separated into the four different cardinal directions. The result of the scatter plot indicate the mean and median error scores can be used to detect which side of the platform shows a mooring line failure, i.e., which group of mooring lines shows a failure. It can be concluded that using the mean and median error a multi-group mooring line failure detection can be done, and that the RMSE error can be used for binary mooring line identification.

6 CLASSIFIER

Classification is the problem of identifying the group or subset to which a new observation belongs. In Appendix A.1 different classifiers were introduced and their fundamental functioning were explained. In this report, classifiers are used to define whether a simulated platform motion contains a line failure or not, having as inputs the error scores of the NN predictors, as shown in the Figure 51.

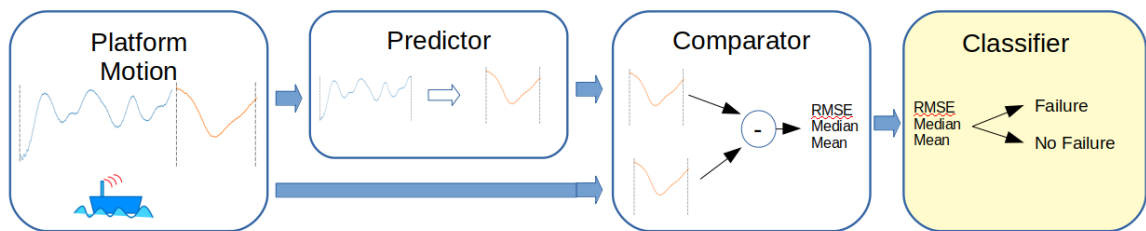


Figure 51: Figure showing the classifier module as part of the modular system structure

Different classifiers were tested against each other to assess which classifier more accurately predicts line failures based on the error scores calculated by the comparator. These classifiers are illustrated in Figure 52: the K-nearest neighbour (KNN) classifier, decision tree (DT) classifier, and support vector classifier (SVC).

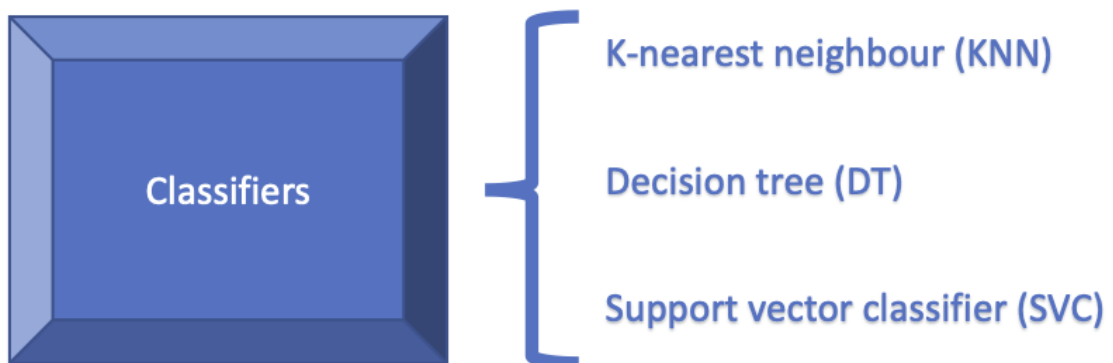


Figure 52: Types of classifiers used.

Classifiers are trained based on training sets. Each training set consists of an input and a desired output. In this work, the input consists of the calculated error scores (RMSE, mean and median errors) from the difference between predicted and simulated platform motion for different environmental conditions. The desired output consists of the class the platform motion belongs to, either line failure or no line failure. These labels are defined by designers for each input.

To train and test the classifiers, a subset of all the available simulated platform motions listed in Table 17 was used. It can be seen in this table that the amount of simulated platform motions is imbalanced. Platform motions labeled no-failure, failure L9 and failure L12 have a greater number of simulated motion data from the platform than those with other line failures.

Table 17: All simulated platform motions from Dynasim. Dates indicate which periods of actual environmental conditions were used to generate the simulated data.

Line	From	To	Number of platform motions
No Failure	03.11.2003	31.12.2009	18000
Failure L1	01.01.2006	31.12.2006	2920
Failure L5	01.01.2006	31.12.2006	2920
Failure L6	01.01.2006	31.12.2006	2920
Failure L9	03.11.2003	31.12.2009	18000
Failure L10	01.01.2006	31.12.2006	2920
Failure L12	03.11.2003	31.12.2009	18000
Failure L15	01.01.2006	31.12.2006	2920
Failure L18	01.01.2006	31.12.2006	2920
Total number of platform motions			71520

6.1 Training set balancing

To balance the number of platform motions, methods such as over-sampling or under-sampling of data can be employed. Over-sampling of data is an approach that aims to balance class distribution by replication of minority class samples until an equal number of the minority class samples and the majority class samples is obtained. On the other hand, making the data balance while under-sampling aims to balance class distribution by random deletion from the majority class samples [25]. In this project, the under-sampling approach is adopted to balance the number of platform motions. The platform motions – No Failure, Failure L9 and Failure L12 – with 180000 conditions are under-sampled to 2920 platform motions to make it the same as the minority class platform motions which are the platform motions with line failures (See Table 18).

An analysis of data from all simulated platform motions was made and the platform motions of the year 2006 were selected for presenting fast platform movements and also for showing more stormy conditions when compared to the rest of the years measured. Since both models were good at predicting the motion of the platform under mild and calm environmental condition with intact mooring lines (See section 4.2.4 for the MLP prediction results and section 4.3.4 for the LSTM prediction results) and both models found it difficult to predict the motion of the platform with stormy environmental condition, we chose to test the classifier on platform motion of this year (2006), to gauge the performance of the classifier since our predictor models found it difficult to predict the motions of the platform. The error scores for this data, provided by the predictors, are then used as input to the classifiers.

The whole year was decomposed into 3 hour sections, starting from January, 1st of 2006 to December, 30th of 2006. The data with line failure had one of its specified mooring line failure at 4000 seconds of simulation (the first 100 seconds of each simulation are neglected to avoid initial simulator instability). From this pool of data with different platform motions a subset with 2000 elements (see Table 19) is selected for training the classifier and the rest is used to test the classifiers. Based on the results of the classification made in the test set, a confusion matrix is created and the data for which the classifiers erroneously classified the data is investigated in more detail.

Table 18: Balanced simulated platform motions

Line	From	To	Number of simulated platform motions
No Failure	01.01.2006	31.12.2006	2920
Failure L1	01.01.2006	31.12.2006	2920
Failure L5	01.01.2006	31.12.2006	2920
Failure L6	01.01.2006	31.12.2006	2920
Failure L9	01.01.2006	31.12.2006	2920
Failure L10	01.01.2006	31.12.2006	2920
Failure L12	01.01.2006	31.12.2006	2920
Failure L15	01.01.2006	31.12.2006	2920
Failure L18	01.01.2006	31.12.2006	2920
Total number of simulated platform motions			26280

The classifier classifies platform motions into two groups, failure and no-failure. The ratio of training data between the failure group and the no-failure group is kept equal, which means 1 : 1 and the classifier is trained on all the different line failures seen in Table 18. There are 8 different line groups with simulated line failure, that are lines 1, 5, 6, 9, 10, 12, 15, and 18. To keep the data ratio equal, the no-failure group needs 8 times more cases than that of individual line failure groups. Also, to keep the line failure cases balanced, the same number of platform motions is selected for each mooring line

failure. An example of how the training data is selected can be seen in Table 19. Here 125 platform motions with line failure are used and to keep it balanced 1000 platform motions without line failure are selected.

Table 19: Example of Training data for the binary classification

Line	platform motions by index	group
No Failure	1-1000	no failure
Failure L1	1-125	failure
Failure L5	1-125	failure
Failure L6	1-125	failure
Failure L9	1-125	failure
Failure L10	1-125	failure
Failure L12	1-125	failure
Failure L15	1-125	failure
Failure L18	1-125	failure
Total number of platform motions	2000	

For testing, all the remaining platform motions of the year 2006 were then classified using the trained classifiers. Table 20 shows the platform motions used for testing.

Table 20: Example of Test data for the binary classification

Line	platform motions by index	group
No Failure	1000 - 2920	no failure
Failure L1	1000 - 1240	failure
Failure L5	1000 - 1240	failure
Failure L6	1000 - 1240	failure
Failure L9	1000 - 1240	failure
Failure L10	1000 - 1240	failure
Failure L12	1000 - 1240	failure
Failure L15	1000 - 1240	failure
Failure L18	1000 - 1240	failure
Total number of platform motions	3840	

Three classifier algorithms were trained and tested. The results of the classifiers for binary classification are shown. For each model, MLP and LSTM, the result of the three classifiers are shown together with their accuracy, precision and recall. In the following section the result of the MLP error classifiers are shown, after which the LSTM error classifiers are also shown.

6.2 MLP classifier results

In this section, the results of three classifiers which used the error scores gotten by the MLP predictor shown in section 4.2 are presented.

The classifiers implemented were all trained on 2000 platform motion errors as mentioned before (see Table 19). For testing, 1920 platform motions without line failure and 1920 platform motions with line failure were used. In total, the classifiers implemented make prediction of 3840 platform motions (see Table 20). The different classifier results are presented in the following sections.

6.2.1 K-nearest neighbour (KNN) classifier result

A K-nearest neighbour (KNN) classifier is implemented (See Appendix A.1.5 for details) which uses $K = 100$ nearest neighbours to classify platform motions into two groups: “failure” and “no-failure” groups. The distance metric the KNN classifier used is the euclidean distance metric.

Result of the KNN classifier prediction on 3840 platform motions are presented in a confusion matrix in Figure 53. It can be seen the KNN classifier classified the 1920 platform motions with mooring line failure accurately into the group “Failure” while for the “No-Failure” group, out of 1920 conditions it misclassified 10 platform motions into the “Failure” group. The overall prediction accuracy of the KNN classifier was 99% (See Table 21).

The error scores are then calculated with the formulas introduced in Section A.1.5. If we consider no failure as positive and failure as negative it can be seen that the algorithm shows a precision of 99.5%, and a recall score of 100% (See Table 21).

Table 21: K-nearest neighbour (KNN) classifier metrics

Accuracy	0.997
Precision	0.995
Recall	1

Figure 53 shows the KNN classifier is classifying 10 platform motions wrongly. Analysis of environmental conditions used to simulate these platform motions in Table 22 shows the wave heights ($hs1$) measured was greater than 4 meters. The peak to peak wave height ($tp1$) measured was greater than 8 meters and the wind speed ($vento_vel$) measured was greater than 7 meters per second. The 10 platform motions wrongly predicted occurred

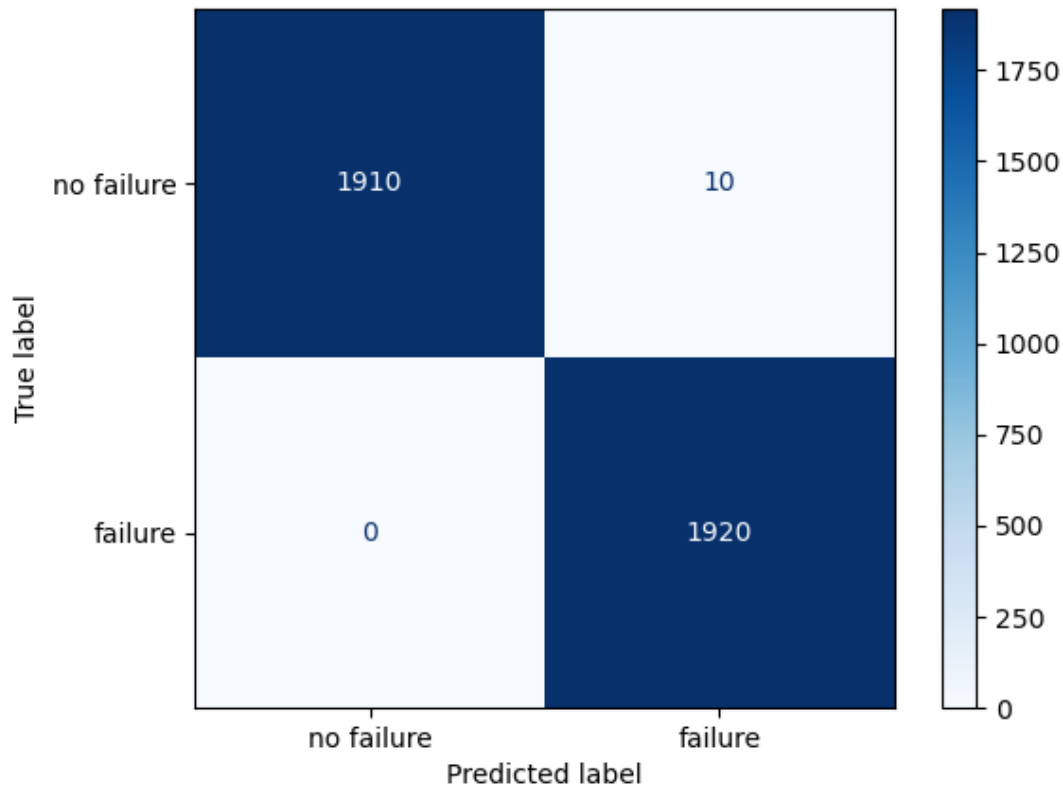


Figure 53: K-nearest neighbour (KNN) classifier prediction on mooring line status using MLP predictor.

within a 2 day span, starting from the 11th of November to 13th of November. The environmental conditions of these 10 platform motions are presented in Table 22 and the index of these environmental conditions are 25907, 25909, 25910, 25911, 25912, 25913, 25914, 25915, 25916 and 25917.

As shown the KNN classifier was predicting 10 platform motion erroneously. This could be an artifact of the MLP predictor, because as it was mentioned in section 3.4.3.4, the MLP model was not able to predict the motions of a platform in environmental condition with rapid oscillations adequately and since the error scores of the MLP predictor for each platform motion are the inputs used by the classifiers, this inadequacy manifests here.

6.2.2 Decision Tree classifier result

A Decision Tree (DT) classifier was implemented (See Appendix A.1.5 for details), and it uses binary decision rules to classify platform motion into two groups, “failure”

Table 22: Environmental condition of the 10 platform motions, the K-nearest neighbour (KNN) classifier classified wrongly.

Index	data	hora	hs1	tp1	dir1	hs2	tp2	dir2	vento_vel	vento_dir	corr_vel	corr_dir
25907	13/11/2006	09:00:00	4.24	9.26	136.4	0.00	0.00	0.0	4.24	140.3	0.49	229.69
25909	13/11/2006	15:00:00	4.03	9.47	124.7	2.31	14.20	174.4	4.64	119.1	0.50	233.14
25910	13/11/2006	18:00:00	4.70	9.77	120.9	0.00	0.00	0.0	4.70	116.6	0.50	234.81
25911	13/11/2006	21:00:00	4.70	9.77	120.9	0.00	0.00	0.0	4.70	116.6	0.51	236.44
25912	14/11/2006	00:00:00	4.68	9.77	118.3	0.00	0.00	0.0	4.68	114.1	0.51	238.02
25913	14/11/2006	03:00:00	4.66	9.10	112.6	0.00	0.00	0.0	4.66	104.9	0.50	234.01
25914	14/11/2006	06:00:00	4.63	8.88	103.4	0.00	0.00	0.0	4.63	95.2	0.48	229.55
25915	14/11/2006	09:00:00	4.02	9.71	105.4	2.19	11.21	154.8	4.57	100.5	0.47	224.61
25916	14/11/2006	12:00:00	4.18	10.72	121.1	0.00	0.00	0.0	4.18	68.3	0.45	219.19
25917	14/11/2006	15:00:00	3.15	10.83	135.4	2.19	7.39	64.4	3.84	35.7	0.44	213.35

and “no-failure” groups. The attribute selection metric the DT classifier is using is the *gini* index measure and the DT approach uses the Classification and Regression Tree algorithm (CART). The DT classifier was trained on the proposed platform motions in the introductory part of this chapter. Hence, a total of 2000 platform motions were used for training the classifier and all available platform motions were then used for testing, resembling 1920 platform motion without line failure and 1920 platform motion with line failure.

Result of the DT classifier on 3840 platform motions are presented in Figure 54. It can be seen, the DT classifier was accurately classifying 1920 platform motions without mooring line failure accurately into the group “No-Failure” while for the “Failure” group, out of 1920 conditions it misclassified 8 condition into the “No-Failure” group. The overall prediction accuracy of the DT classifier was 100%.

The error scores are then calculated with the formulas introduced in section A.1.5. If we consider no failure as positive and failure as negative, it can be seen that the algorithm shows a precision of 99%, and a recall score of 100% (See Table 23).

Table 23: Decision Tree (DT) classifier metrics

Accuracy	0.997
Precision	0.996
Recall	1

Figure 54 shows the DT classifier is classifying 8 platform motions wrongly. The environmental conditions used to simulate these platform motions are provided in Table 24. Analysis of the environmental conditions in Table 24 shows the wave height measured (*hs1*) was greater than 3 meters. The peak to peak wave height (*tp1*) measured was greater than 8 meters and the wind speed (*vento_vel*) measured was greater than 7 meters per second. These 8 platform motions wrongly predicted all occurred within a 2 day span,

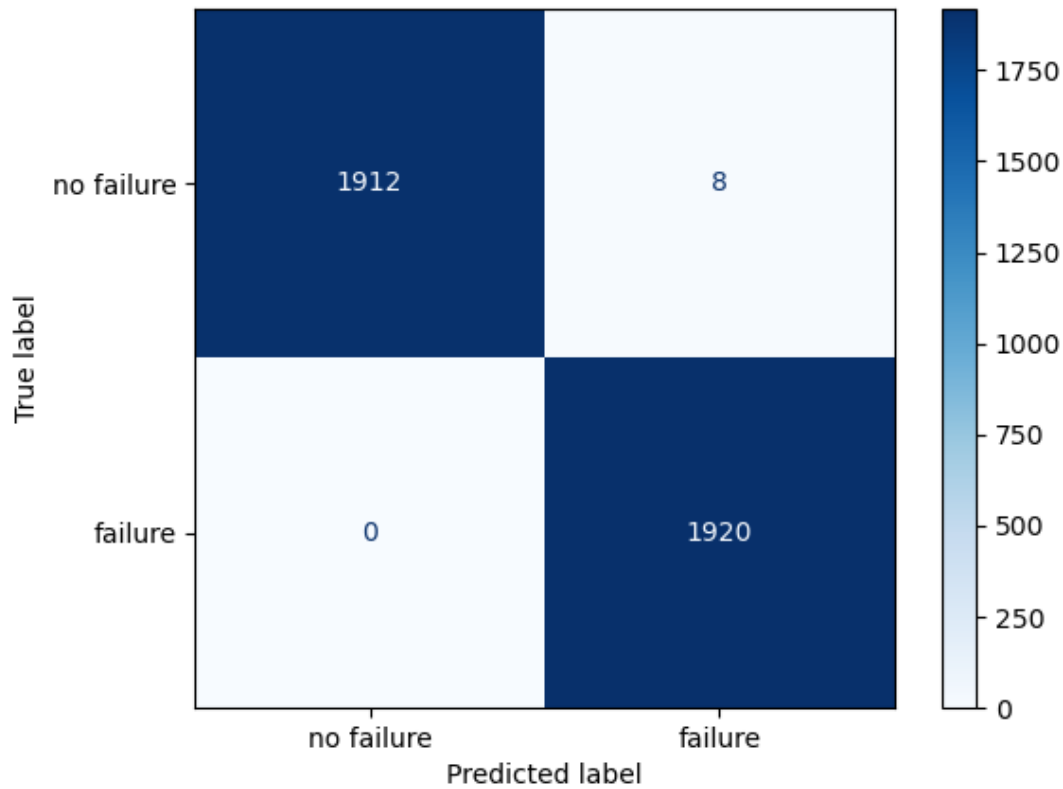


Figure 54: Decision Tree (DT) classifier prediction on mooring line status using MLP predictor.

starting on the 13th of November to the 14th of November. The environmental conditions of these 8 platform motions are presented in Table 24 and the index of these environmental conditions are 25909, 25911, 25912, 25913, 25914, 25915, 25916 and 25917.

Table 24: Environmental condition of the 8 platform motions the decision tree (DT) classifier classified wrongly.

Index	data	hora	hs1	tp1	dir1	hs2	tp2	dir2	vento_vel	vento_dir	corr_vel	corr_dir
25909	13/11/2006	15:00:00	4.03	9.47	124.7	2.31	14.20	174.4	4.64	119.1	0.50	233.14
25911	13/11/2006	21:00:00	4.70	9.77	120.9	0.00	0.00	0.0	4.70	116.6	0.51	236.44
25912	14/11/2006	00:00:00	4.68	9.77	118.3	0.00	0.00	0.0	4.68	114.1	0.51	238.02
25913	14/11/2006	03:00:00	4.66	9.10	112.6	0.00	0.00	0.0	4.66	104.9	0.50	234.01
25914	14/11/2006	06:00:00	4.63	8.88	103.4	0.00	0.00	0.0	4.63	95.2	0.48	229.55
25915	14/11/2006	09:00:00	4.02	9.71	105.4	2.19	11.21	154.8	4.57	100.5	0.47	224.61
25916	14/11/2006	12:00:00	4.18	10.72	121.1	0.00	0.00	0.0	4.18	68.3	0.45	219.19
25917	14/11/2006	15:00:00	3.15	10.83	135.4	2.19	7.39	64.4	3.84	35.7	0.44	213.35

As shown the DT classifier was predicting 8 platform motion erroneously. As it was mentioned in section 3.4.3.4, the MLP model was not able to predict the motions of a platform in environmental condition with rapid oscillations adequately and since the error scores of the MLP predictor for each platform motion are the inputs used by the classifiers,

this issue manifests here.

6.2.3 Support Vector Classifier (SVC) result

A Support Vector Classifier (SVC) is implemented (See Appendix A.1.5 for details) which classifies platform motions into two groups, “failure” and “no-failure” groups. The SVC classifier was trained on a total of 2000 platform motions and all available platform motions were then used for testing, resembling 1920 platform motion without line failure and 1920 platform motion with line failure.

Result of the SVC classifier on 3840 platform motions are presented in Figure 55. It can be observed that the SVC classifier was accurately classifying 1920 platform motions with mooring line failure accurately into the group “failure” while for the “no-failure” group, out of 1920 conditions it misclassified 28 conditions into the “failure” group. The overall prediction accuracy of the SVC classifier was 99.3% (See Table 25).

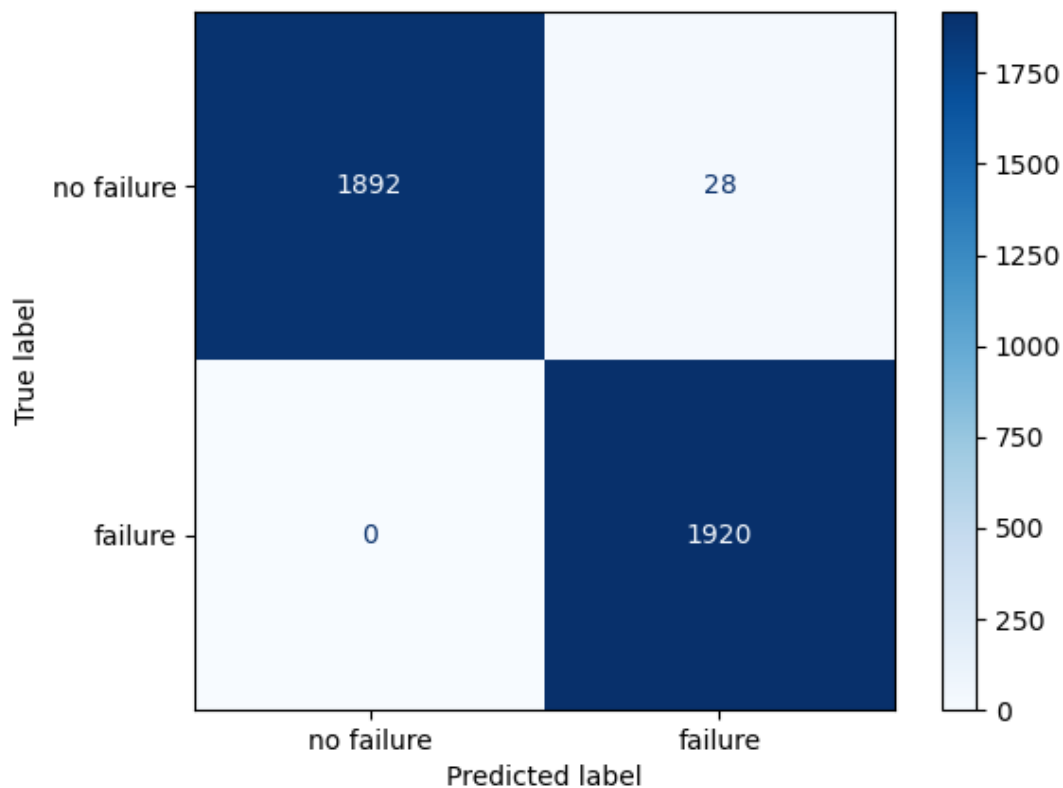


Figure 55: Support Vector Classifier (SVC) prediction on mooring line status using MLP predictor.

The error scores are then calculated with the formulas introduced in section A.1.5. If

we consider no failure as positive and failure as negative, it can be seen that the algorithm shows a precision of 99%, and a recall score of 100% (See Table 25).

Table 25: Support vector classifier (SVC) metrics

Accuracy	0.993
Precision	0.985
Recall	1.0

Figure 55 shows the SVC classifier is classifying 28 platform motions wrongly. The environmental conditions of these platform motions are provided in Table 26. Analysis of the environmental conditions used for simulating platform motions in Table 26 shows the wave heights (*hs1*) measurements ranged from 2 meters to 5.22 meters. The peak to peak (*tp1*) measurements were all greater than 7 meters and the wind speed (*vento_vel*) was greater than 7 meters per second.

Out of the 28 platform motions classified wrongly, 16 platform motions were all from the month of November and it occurred within a 3 days span starting on the 13th of November to the 15th of the November. The environmental conditions of these 16 platform motions are presented in Table 26 and the index of these environmental conditions are 25907, 25908, 25909, 25910, 25911, 25912, 25913, 25914, 25915, 25916, 25917, 25919, 25921, 25925, 25926 and 25927 (See Table 26)

Out of the 12 remaining platform motions wrongly predicted, 2 platform motions whose environmental conditions indexes are 24428, 24429 were from the 12th of May and another 2 platform motions were from the month of December and their environmental conditions indexes in Table 26 are 26069,26149. These occurred on separate days, 3rd of December and 13th of December.

In the remaining 8 platform motions, 3 platform were from the 28th of June and the environmental conditions indexes of the platform motions are 24802, 24803 and 24804. Another 3 platform motions were from the 25th of September and their indexes are 25357, 25358 and 25359 while the remaining 2 platform motions occurred on the 1st of July with environmental condition index 24827 (See Table 26) and on the 28th of October with environmental conditions index 25778.

All the platform motions the SVC classifier classified incorrectly occurred between May to December. As it was described in Section 3.4.3.4, the MLP model was not able to predict the motions of a platform with rapid oscillations adequately and since the error scores of the MLP predictor for each environmental condition are the inputs used by the

classifiers, this may have caused these misclassifications.

Table 26: Environmental condition of the 28 platform motions the Support Vector Classifier (SVC) classified wrongly.

Index	data	hora	hs1	tp1	dir1	hs2	tp2	dir2	vento_vel	vento_dir	corr_vel	corr_dir
24428	12/05/2006	12:00:00	2.72	7.53	245.1	1.07	8.59	102.3	2.92	250.8	0.15	159.28
24429	12/05/2006	15:00:00	2.66	7.39	240.1	0.94	8.62	94.3	2.82	240.7	0.16	161.26
24802	28/06/2006	06:00:00	5.21	14.78	208.9	0.00	0.00	0.0	5.21	195.2	0.13	285.27
24803	28/06/2006	09:00:00	5.22	14.81	208.4	0.00	0.00	0.0	5.22	193.2	0.12	284.05
24804	28/06/2006	12:00:00	5.20	14.98	206.9	0.00	0.00	0.0	5.20	189.0	0.11	282.68
24827	01/07/2006	09:00:00	2.37	14.47	157.4	1.97	6.60	97.0	3.08	95.1	0.37	219.12
25357	05/09/2006	15:00:00	5.08	9.16	202.0	0.00	0.00	0.0	5.08	194.6	0.19	250.46
25358	05/09/2006	18:00:00	5.08	9.16	202.0	0.00	0.00	0.0	5.08	194.6	0.18	263.89
25359	05/09/2006	21:00:00	4.96	8.94	198.8	0.00	0.00	0.0	4.96	192.4	0.17	276.45
25778	28/10/2006	06:00:00	4.11	13.89	203.2	0.00	0.00	0.0	4.11	238.6	0.28	214.90
25907	13/11/2006	09:00:00	4.24	9.26	136.4	0.00	0.00	0.0	4.24	140.3	0.49	229.69
25908	13/11/2006	12:00:00	4.50	9.04	131.5	0.00	0.00	0.0	4.50	129.3	0.49	231.43
25909	13/11/2006	15:00:00	4.03	9.47	124.7	2.31	14.20	174.4	4.64	119.1	0.50	233.14
25910	13/11/2006	18:00:00	4.70	9.77	120.9	0.00	0.00	0.0	4.70	116.6	0.50	234.81
25911	13/11/2006	21:00:00	4.70	9.77	120.9	0.00	0.00	0.0	4.70	116.6	0.51	236.44
25912	14/11/2006	00:00:00	4.68	9.77	118.3	0.00	0.00	0.0	4.68	114.1	0.51	238.02
25913	14/11/2006	03:00:00	4.66	9.10	112.6	0.00	0.00	0.0	4.66	104.9	0.50	234.01
25914	14/11/2006	06:00:00	4.63	8.88	103.4	0.00	0.00	0.0	4.63	95.2	0.48	229.55
25915	14/11/2006	09:00:00	4.02	9.71	105.4	2.19	11.21	154.8	4.57	100.5	0.47	224.61
25916	14/11/2006	12:00:00	4.18	10.72	121.1	0.00	0.00	0.0	4.18	68.3	0.45	219.19
25917	14/11/2006	15:00:00	3.15	10.83	135.4	2.19	7.39	64.4	3.84	35.7	0.44	213.35
25919	14/11/2006	21:00:00	2.61	10.86	138.1	2.43	6.91	51.1	3.56	42.3	0.41	200.73
25921	15/11/2006	03:00:00	2.91	7.67	66.1	1.96	10.40	140.7	3.51	54.7	0.43	191.79
25925	15/11/2006	15:00:00	2.39	9.61	109.9	2.04	6.80	64.1	3.14	59.6	0.56	184.93
25926	15/11/2006	18:00:00	3.04	9.61	108.0	0.00	0.00	0.0	3.04	59.1	0.60	183.71
25927	15/11/2006	21:00:00	2.48	8.15	65.3	1.80	9.80	130.8	3.06	50.8	0.63	182.62
26069	03/12/2006	15:00:00	2.15	9.38	134.3	0.00	0.00	0.0	2.15	93.0	0.33	185.83
26149	13/12/2006	15:00:00	2.35	7.43	74.3	0.55	7.51	147.5	2.41	74.7	0.49	193.51

6.3 LSTM classifier results

The section is comparing the results of the different classifiers using the error scores obtained from the LSTM predictor shown in section 4.3. The different classifiers are presented in the following sections.

6.3.1 K-nearest neighbour classifier results

A K-nearest neighbour (KNN) classifier was implemented (See Appendix 7.1.5.1 for details), and it uses $K = 200$ nearest neighbours and the euclidean distance as distance metric to classify environmental conditions into two groups: “failure” and “no-failure” groups.

The training was done using the proposed files in the introductory part of this chapter. A total of 2000 platform motions was used for training the classifier. All available platform motions were then used for testing, resembling 1920 platform motions without line failure and 1920 with line failure. Out of these 3840 platform motions, the KNN classifier scored an accuracy of 99.77%. The confusion matrix can be seen in Figure 56. It can be observed that the classifier never predicted wrongly the platform motions without a line failure, and predicted only 9 times that there was no-failure when there was one.

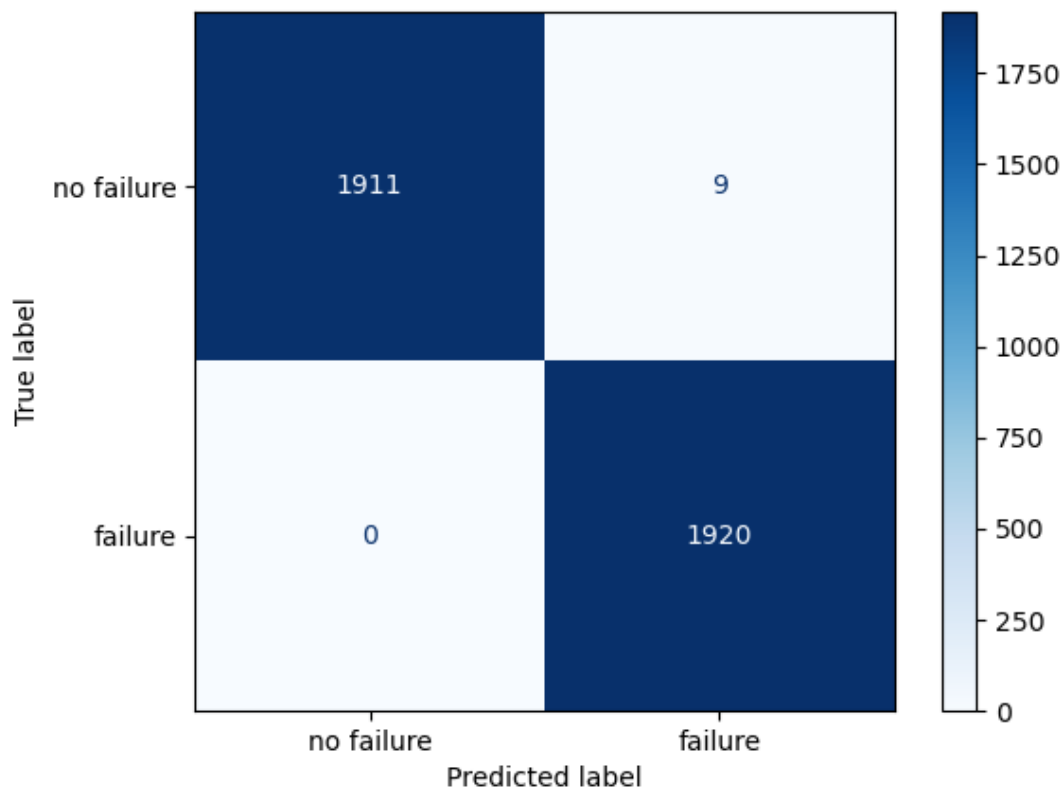


Figure 56: K-nearest neighbour (KNN) prediction on mooring line status using LSTM predictor.

The error scores are then calculated with the formulas introduced in Section A.1.5. If we consider no-failure as positive and failure as negative, it can be seen that the classifier

shows a high recall score with 100%, while the precision score is a little lower with 99.53%.

Table 27: Error rating of K-nearest neighbour (KNN) classifier based on LSTM predictor

Accuracy	0.9977
Precision	0.9953
Recall	1

It can be seen that a high overall accuracy was scored using the predictor and the classifier configuration. The 9 error platform motions come all from the same period of time and can be seen in Table 28.

This could be an artifact of the LSTM predictor, because as it was mentioned in Section 4.3, the LSTM model was not able to predict the motions of a platform in environmental condition with rapid oscillations adequately and since the error scores of the LSTM predictor for each environmental condition are the inputs used by the classifiers this inadequacy manifested here. The platform motions the KNN classifier predicted wrongly belong to these environmental condition with rapid oscillations.

Table 28: Environmental condition of the 9 platform motions the K-nearest neighbour (KNN) classifier classified wrongly

Index	data	hora	hs1	tp1	dir1	hs2	tp2	dir2	vento_vel	vento_dir	corr_vel	corr_dir
25909	13/11/2006	15:00:00	4.03	9.47	124.7	2.31	14.20	174.4	12.87	119.1	0.50	233.14
25911	13/11/2006	21:00:00	4.70	9.77	120.9	0	0	0.0	12.86	116.6	0.51	236.44
25912	14/11/2006	00:00:00	4.68	9.77	118.3	0	0	0.0	12.84	114.1	0.51	238.02
25913	14/11/2006	03:00:00	4.66	9.10	112.6	0	0	0.0	12.44	104.9	0.50	234.01
25914	14/11/2006	06:00:00	4.63	8.88	103.4	0	0	0.0	12.07	95.2	0.48	229.55
25915	14/11/2006	09:00:00	4.02	9.71	105.4	2.19	11.21	154.8	13.17	100.5	0.47	224.61
25916	14/11/2006	12:00:00	4.18	10.72	121.1	0	0	0.0	7.55	68.3	0.45	219.19
25357	05/09/2006	15:00:00	5.08	9.16	202.0	0.00	0.00	0.0	14.18	194.6	0.19	250.46
25358	05/09/2006	18:00:00	5.08	9.16	202.0	0.00	0.00	0.0	14.18	194.6	0.18	263.89

6.3.2 Decision Tree classifier results

A decision tree (DT) classifier was implemented (see Appendix 7.1.5.1 for details), and it uses binary decision rules to classify environmental conditions into two groups: “failure” and “no-failure” groups. The training was done using the proposed files in the introductory part of this chapter. A total of 2000 platform motions were used for training the classifier. All available platform motions were then used for testing, resembling 1920 platform motions without mooring line failure and 1920 with mooring line failure. Out of these 3840 platform motions, the DT classifier scored an accuracy of 99.77%. The confusion matrix is shown in Figure 57 and it can be observed that the classifier never

classified the platform motions without a line failure wrong and only 9 times it predicted that there was no-failure when there was one.

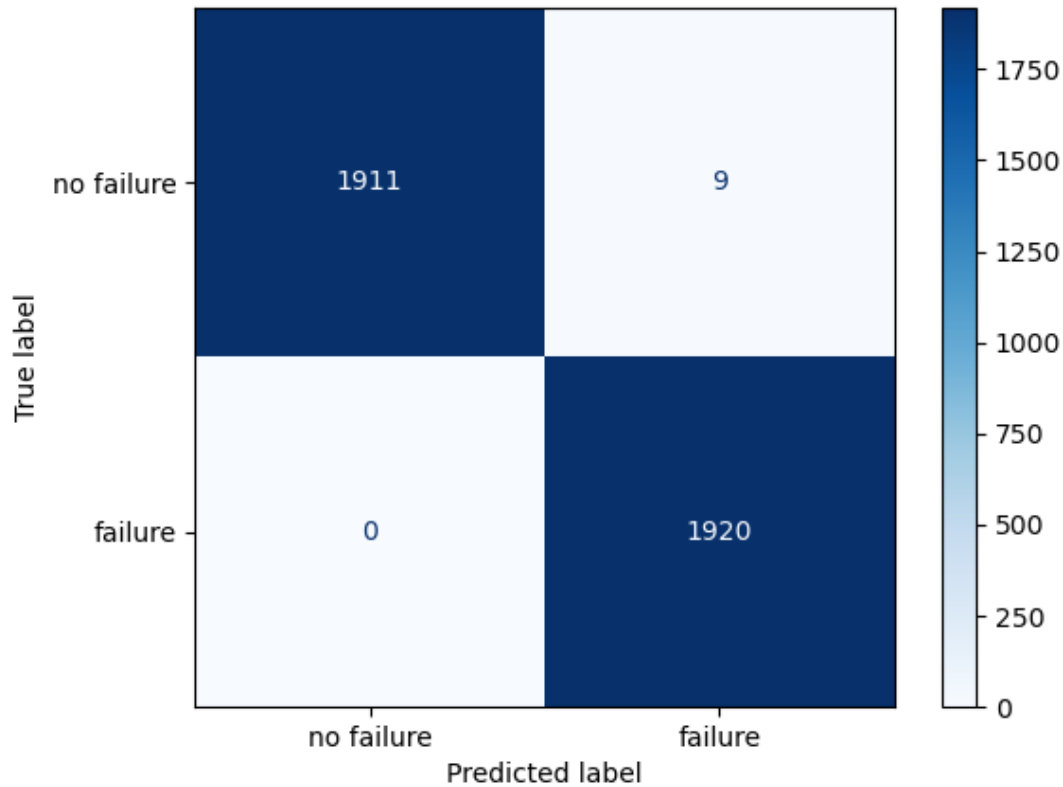


Figure 57: Decision Tree (DT) classifier prediction on mooring line status using LSTM predictor

The error scores are then calculated with the formulas introduced in Section A.1.5. If we consider no-failure as positive and failure as negative, it can be seen that the classifier shows an excellent recall with 100%, while the precision score is a little lower with 99.53%.

Table 29: Error rating of Decision Tree (DT) classifier based on LSTM predictor.

Accuracy	0.9977
Precision	0.9953
Recall	1

It can be seen that a high overall accuracy was scored using the LSTM predictor and the DT classifier configuration. The 9 error platform motions come all from the same period of time and can be seen in Table 30.

As described in Section 4.3, the LSTM model has difficulties in making good predictions in conditions of rapid motions and, therefore, these difficulties are reflected in the

results of the classifier. The platform motions the DT classifier predicted wrongly belong to the environmental condition with rapid oscillations.

Table 30: Environmental conditions of the 9 platform motions the Decision Tree (DT) classified wrongly

Index	data	hora	hs1	tp1	dir1	hs2	tp2	dir2	vento_vel	vento_dir	corr_vel	corr_dir
25909	13/11/2006	15:00:00	4.03	9.47	124.7	2.31	14.20	174.4	12.87	119.1	0.50	233.14
25911	13/11/2006	21:00:00	4.7	9.77	120.9	0	0	0	12.86	116.6	0.51	236.44
25912	14/11/2006	00:00:00	4.68	9.77	118.3	0	0	0	12.84	114.1	0.51	238.02
25913	14/11/2006	03:00:00	4.66	9.10	112.6	0	0	0	12.44	104.9	0.50	234.01
25914	14/11/2006	06:00:00	4.63	8.88	103.4	0	0	0	12.07	95.2	0.48	229.55
25915	14/11/2006	09:00:00	4.02	9.71	105.4	2.19	11.21	154.8	13.17	100.5	0.47	224.61
25916	14/11/2006	12:00:00	4.18	10.72	121.1	0	0	0	7.55	68.3	0.45	219.19
25357	05/09/2006	15:00:00	5.08	9.16	202.0	0.00	0.00	0.0	14.18	194.6	0.19	250.46
25358	05/09/2006	18:00:00	5.08	9.16	202.0	0.00	0.00	0.0	14.18	194.6	0.18	263.89

6.3.3 Support Vector Classifier (SVC) results

A Support Vector classifier (SVC) is implemented (See Appendix 7.1.5.1 for details) and classifies environmental conditions into two groups: “failure” and “no-failure” groups. The training was done using the proposed files in the introductory part of this chapter. A total of 2000 platform motions were used for training the classifier. All available platform motions were then used for testing, resembling 1920 platform motions without mooring line failure and 1920 with mooring line failure. Out of these 3840 platform motions, the SVC classifier scored an accuracy of 99.69%. The confusion matrix can be seen in Figure 58. It can be observed that the classifier never classified the platform motions without a line failure wrong and only 12 times did it predicted that there was no-failure when there was one.

The error scores are then calculated with the formulas introduced in Section A.1.5. If we consider no-failure as positive and failure as negative, it can be seen that the classifier shows a high precision of 99.37%, while the recall score was 100%.

Accuracy	0.9969
Precision	0.9937
Recall	1

Table 31: Error rating of support vector classifier (SVC) based on LSTM predictor

It can be seen that a high overall accuracy was scored using the predictor and the classifier configuration. The 12 wrongly classified platform motions come all from the same period of time and can be seen in Table 32.

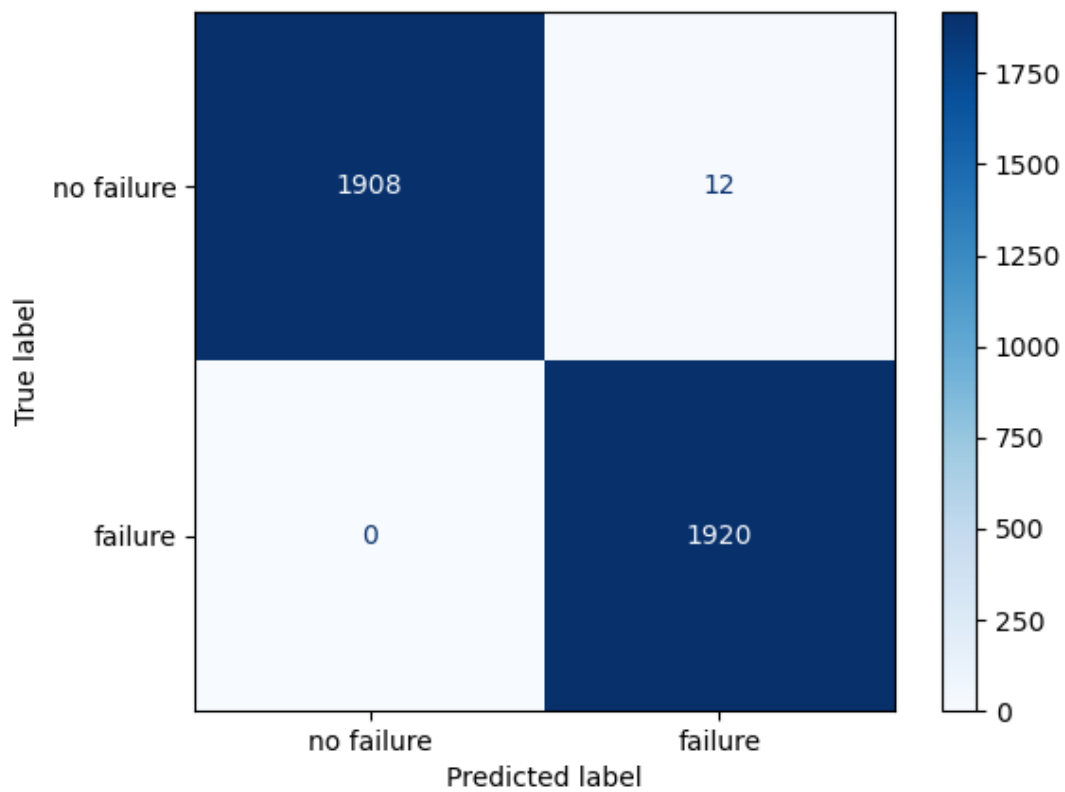


Figure 58: Support vector classifier (SVC) prediction on mooring line status using LSTM predictor.

As it was mentioned in Section 4.3, the LSTM model was not able to predict the motions of a platform in environmental condition with rapid oscillations adequately, and since the error scores of the LSTM predictor for each environmental condition are the inputs used by the classifiers, this issues are manifested here. The platform motions the SVC classifier predicted wrongly belong to the environmental condition with rapid oscillations.

Table 32: Environmental condition of the 12 platform motions the support vector classifier (SVC) classified wrongly.

Index	data	hora	hs1	tp1	dir1	hs2	tp2	dir2	vento_vel	vento_dir	corr_vel	corr_dir
24802	28/06/2006	06:00:00	5.21	14.78	208.9	0.00	0.00	0.0	11.82	195.2	0.13	285.27
24803	28/06/2006	09:00:00	5.22	14.81	208.4	0.00	0.00	0.0	11.40	193.2	0.12	284.05
24804	28/06/2006	12:00:00	5.20	14.98	206.9	0.00	0.00	0.0	10.56	189.0	0.11	282.68
25909	13/11/2006	15:00:00	4.03	9.47	124.7	2.31	14.20	174.4	12.87	119.1	0.50	233.14
25911	13/11/2006	21:00:00	4.7	9.77	120.9	0	0	0	12.86	116.6	0.51	236.44
25912	14/11/2006	00:00:00	4.68	9.77	118.3	0	0	0	12.84	114.1	0.51	238.02
25913	14/11/2006	03:00:00	4.66	9.10	112.6	0	0	0	12.44	104.9	0.50	234.01
25914	14/11/2006	06:00:00	4.63	8.88	103.4	0	0	0	12.07	95.2	0.48	229.55
25915	14/11/2006	09:00:00	4.02	9.71	105.4	2.19	11.21	154.8	13.17	100.5	0.47	224.61
25916	14/11/2006	12:00:00	4.18	10.72	121.1	0	0	0	7.55	68.3	0.45	219.19
25357	05/09/2006	15:00:00	5.08	9.16	202.0	0.00	0.00	0.0	14.18	194.6	0.19	250.46
25358	05/09/2006	18:00:00	5.08	9.16	202.0	0.00	0.00	0.0	14.18	194.6	0.18	263.89

6.4 Discussion

In this section the results of the classifiers based on MLP and LSTM predictor error scores are discussed.

Results of the classifiers based on the MLP predictor error scores showed that the three classifiers implemented had a combined accuracy of 99.6%. Out of the three classifiers, the Decision Tree (DT) algorithm had the lowest misclassification rate, followed by the K-nearest neighbour (KNN) classifier and the Support Vector Classifier (SVC) classifier. The DT classifier miscategorized 8 platform motions out of 1920 platform motions classified with “no-failure” into the “failure” group.

Analysis of the environmental conditions of the platform motions each classifier predicted wrongly was conducted. For the classifier with the best results, the DT classifier, the wave height measured (*hs1*) was greater than 3 meters for all of the environmental conditions of the platform motions, while the peak to peak wave height (*tp1*) measured was greater than 8 meters and the wind speed (*vento_vel*) measurement was greater than 7 meters per second.

Analysis of the platform motions each classifier categorized wrongly revealed there was an overlap of 8 platform motions which all the classifiers predicted wrongly. The index of the environmental conditions of these platform motions are 25909, 25911, 25912, 25913, 25914, 25915, 25916 and 27917. All the classifiers were classifying these platform motion with intact mooring lines into the “failure” group. The environmental conditions

of these 8 platform motions revealed they occurred in November of 2006 and they all had wave heights (*h_{s1}*) that were greater than 4 meters and wind speed (*vento_vel*) that was greater than 7 meters per second.

The classifiers misclassified these data because the data they received at their input already showed inconsistencies, since the MLP had difficulties to correctly predict motions in these environmental conditions, as explained in Section 4.2.4.

Results of the classifiers based on the LSTM predictor error scores showed that the three classifiers implemented had an average accuracy of 99.9%. Out of the three classifier's, the Decision Tree (DT) and the K-nearest neighbour (KNN) classifiers had the lowest misclassification rate, while the Support Vector Classifier (SVC) had the highest misclassification rate.

The DT and the KNN classifiers misclassified 9 out of 1920 platform motions without mooring line failure into the group "failure". Analysis of the environmental condition of the platform motions each classifier predicted wrongly was conducted, and both, DT and KNN, misclassified the same platform motions.

The joint analysis of the motions incorrectly classified by the three classifiers revealed that there was an overlap of 9 platform motions that all were wrong. The index of the environmental conditions of these platform motions are 25909, 25911, 25912, 25913, 25914, 25915, 25916, 25357 and 25358 (see in Table 30). The environmental conditions of these 9 platform motions showed they occurred in November of 2006 and they all had wave heights (*h_{s1}*) that were greater than 4 meters and wind speed (*vento_vel*) that was greater than 7 meters per second.

A joint comparison of the MLP and LSTM classifiers showed the DT classifier had the best accuracy for both. In addition, the comparison revealed that the LSTM-based classifier was better as expected, since the LSTM predictor obtained better results.

For both, MLP- and LSTM-based classifiers, 7 misclassified platform motions were present for all classifiers implemented in this project. The index of the environmental condition of these platform motion are 25909, 25911, 25913, 25914, 25915, 25916 and 25917 (see in Table 30). It can be observed that the environmental conditions of these misclassified platform motions showed similar environmental conditions and they indicate a rough sea state.

In this project, having a low number of false-negative classification is very important, because in the event there is mooring line failure and the classifier predicts there isn't,

it could be disastrous for the safety of the personnel on the platform and the structural integrity of the platform. Both the MLP-based and LSTM-based classifiers had no false-negative prediction.

7 CONCLUSION

Mooring systems give stability to floating platforms against environmental conditions by anchoring the platform with mooring lines attached to the seabed. These systems are among the main components that guarantee the safety of staffs and the various operations carried out on the platforms.

Petrobras has many platforms installed, which encompass a large number of mooring lines. Thus, the rapid detection of failure of a mooring system is very important, as this failure can result in damage or loss of property, environmental pollution, personnel endangerment and depending on the severity of failure, in some cases oil production shutdown.

Two different Neural networks were proposed to detect mooring line failure:

MLP, a feed-forward neural network, described in Section 4.2.

LSTM, a recurrent neural network, described in Section 4.3.

A predictor module based on neural networks performs the prediction of future platform motion considering that there are no breakage in the mooring lines. Sensors measure the motion effectively performed by the platform. The difference between the two signals, predicted and measured, indicates whether or not there was a failure, which is detected by a classifier.

It can be seen that both MLP and LSTM were able to predict the platform's motion in a wide variety of environmental conditions. The LSTM network was able to predict a longer period than the MLP network, while also showing a higher accuracy. Different classifier networks were then trained to identify line breakage based on the difference between prediction and measured data. It could be seen that the classification was able to identify line breakages with an accuracy around 99 percent in balanced data. While it classified all cases of line failure correctly, it showed false positives under stormy conditions.

It is important to note that, in this first phase of the project, an architecture was

proposed to tackle the problem, following two distinct approaches that have the potential to be complementary. The architecture was implemented and preliminary tests with simulated data were performed, achieving very positive results. However, it should be noted that more tests with simulated data should be performed, in order to better identify the positive and negative points of the two approaches and the situations at which they can complement each other. In addition, it cannot be overlooked that the use of real data will bring many more challenges, which will need to be treated with care.

Appendices

A APPENDIX

This appendix explains the fundamental machine learning techniques used in this work and gives an overview over the Python environment and additional tools used in the implementation. Furthermore it emphasizes the development process using Docker and continuous integration. It describes the creation of the code documentation and the implementation of web services.

A.1 Machine Learning

Machine learning (ML) is a subset of artificial intelligence, in which a machine learns from data without being issued explicit instruction on how to solve a given task [26]. ML over the decade has expanded rapidly into many subject domains: in the business sector it is used for fraud detection [27], in the health sector it is used for medical diagnostic [28] and in the automotive sector it is used for autonomous driving (self-driving cars) [29].

ML can be divided into three categories: supervised learning (SL), unsupervised learning (UL) and reinforcement learning (RL) [26, 30].

Supervised Learning involves training an algorithm to be able to generate a mapping function that can predict output for a given input. The algorithm trains on labelled data until it achieves a reasonable level of accuracy in making accurate prediction. SL can be sub-divided into two categories namely: classification and regression. Supervised classification algorithms attempt to generate a mapping function that classifies input into categorical output based on input features, while regression algorithms attempts to generate a mapping function that predicts numeric or continuous output from input variables.

Unsupervised learning (UL) searches for patterns previously not explicit in the data and allows the modeling of probability densities over inputs. Unlike what happens in SL, UL dispenses with data labels and human supervision.

Reinforcement learning is learning from a dynamic environment in which it must perform a certain goal. The feedback is done in form of rewards which the algorithm tries to maximize.

In this work machine Learning will be used in a supervised manner to make a Time series prediction of an offshore platform motion. Further details on the ML area can be found in [26, 30, 31].

A.1.1 Neural Network Principles

Most ML algorithms are based on neural networks, implementable in different forms. Although the basic function of them is always similar. Neural networks are able to change some parts of their structure based on received internal or external information. The basic ability to change their structure enables neural networks to learn patterns in data structures. Most commonly a unit known as perceptron is used as basic unit. Figure 59 presents a unit of perceptron.

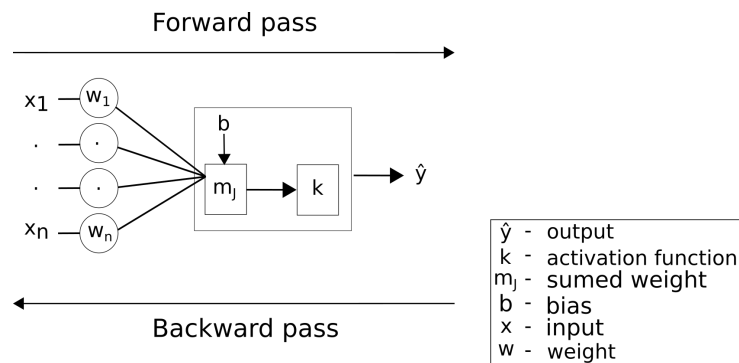


Figure 59: A single unit of a perceptron.

A single unit of perceptron takes in inputs \mathbf{x} , given by $x_1, x_2, x_3, \dots, x_n$, to generate an output \hat{y} . Each input (i) is connected to a node (j) with weights \mathbf{w} , with $w_1, w_2, w_3, \dots, w_n$ assigned to each input respectively. These weights determine the strength of a connection to node (j). The node (j) generates a weighted sum of all the inputs values plus bias:

$$m_j = \left(\sum_{i=1}^n w_i \cdot x_i + b \right), \quad (\text{A.1})$$

where the n is the number of nodes in the input, x_i is the input value in node i , w_i is the weight in the connection from i to j , b is the bias. The result of the weighted sum (m_j) is passed through an **activation function** k to generate an output l_j

$$l_j = k(m_j), \quad (\text{A.2})$$

where if l_j is an input to another perception it becomes x or \hat{y} if it is the final output.

The activation function k can be referred to as a gate which allows information to flow through or not. There are different types of activation functions available. However for brevity, threshold activation function also known as step function whose output is binary 0 or 1, sigmoid activation function whose output is between 0, 1 or $-1, 1$ and Rectified Linear Unit function (ReLU) whose output is between $0, \infty$ are explained.

The threshold activation function works as follows; if the output from the weighted sum plus bias (m_j) is above a specified threshold, the threshold activation function activates and 1 is given as output or 0 is given as output if it is less than the specified threshold. With this, a supervised binary classification problem can be done where the target output is either 1 or 0.

For neural networks to handle complex data, non-linear activation function such as sigmoid activation function and ReLU are commonly used among other types of activation function. Sigmoid activation function whose formula equals:

$$k(z) = \frac{1}{1 + e^{-z}} \quad (\text{A.3})$$

is governed by the following, the value coming from weighted sum (here $z = m_j$) is squashed to be in the range of 0, 1 and it is propagated forward. Figure 60 illustrates a sigmoid function.

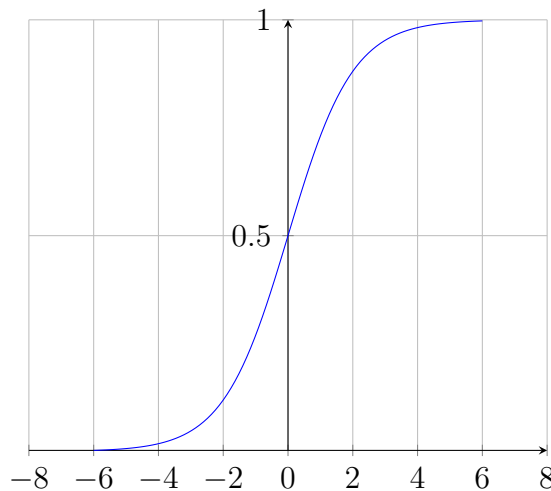


Figure 60: Sigmoid function plot.

ReLU whose formula equals $k(z) = \max_z(0, z)$ is governed by the following, if the weighted sum ($z = m_j$) is greater than zero the information from the input is allowed to pass while if $z = m_j$ is 0 or negative it is clipped at 0 and 0 is passed as output [32].

Figure 61 shows the ReLU function.

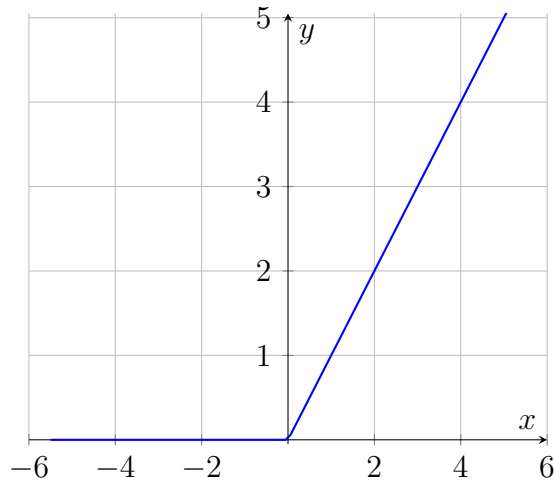


Figure 61: ReLU function plot.

Depending on the decision taken by the activation function k the output \hat{y} is gotten. The perceptron learns by using the error difference between the perceptron output and the actual value to improve. The learning process is done in three steps:

Forward pass: involves the flow of information from input to the activation function to produce an output (Figure 59).

Loss function calculation: the output of the perceptron \hat{y} is compared against the actual value y , given this is a supervised learning where the actual output (also called label) is known before hand in a training set consisting of pairs $\langle \mathbf{x}, y \rangle$.

Backward pass: the error difference is then back propagated to adjust the weights of the perceptron. Starting from the output layer back to the hidden(s) layer(s) and back to the input layer.

These three steps are done recursively until a stopping criteria is met. The stopping criteria can either be allowing the network to iterate n number of **epochs** or when the error difference between the network predictions and the true label is minimal. Epoch refers to the number of times the network is instructed to cycle through the whole data set. These learning steps are what is referred to as backpropagation.

During training of a neural network the objective function is to minimize the error between a networks prediction and the true label. The goal is to have a model that is able to predict with good accuracy on unseen data. To achieve this goal the model must not over-fit or under-fit on the training data as both of these situations affect the models

accuracy when predicting on unseen data. A model is said to be over-fitted when the model is trained on too much data. That is, the model predicts too good on data included in the training set but predicts very bad on a test set. While a model under-fit is the inverse of an over-fitted network. Here the model is unable to learn the underlying features of a training set thus when it is given a test data it predicts very bad. In order to have a good model capable of predicting good on unseen data a balance between these situations is needed. The model needs to learn just enough underlying representation of the training data features. Some ways of addressing this is by using techniques such as early stopping technique, where the model is instructed to stop training when it starts to over-fit. In the case of an under-fitting network more training data should be provided, or more layers to the network should be added.

A.1.2 Multi- Layer perceptron

A **multi-layer perceptron (MLP)** is a feed forward neural network that is created when multiple perceptrons are structured in layers to solve complex problems. In feed forward networks information propagates in one direction. Information moves from the input layer to the hidden layer to the output layer. These networks do not possess the ability to use a feedback loop in which the outputs of the networks is cycled back into itself.

The MLP follows the single perceptron principle but on a larger scale. To construct an MLP network, perceptrons are structured in layers. These layers are the input layer (I), hidden layer (H) and an output layer (O) and in these layers perceptron units are found. Following the same principle of a unit of perceptron, information flows from the input layer through the hidden layer after which it passes to the output layer to make a prediction. Figure 62 presents a 2 x 2 x 2 MLP with one input layer, 1 hidden layer and an output layer.

To improve the accuracy of a perceptron or MLP or NN, training is done using **back propagation algorithm**. The back propagation algorithm functions by using the error difference between the MLP output and the expected output to make adjustments to the weights in each layer of the network, starting from the output layer backwards to the input layer with the objective of reducing the error difference of the MLP output and the actual output. A common performance metric used to gauge the difference between the

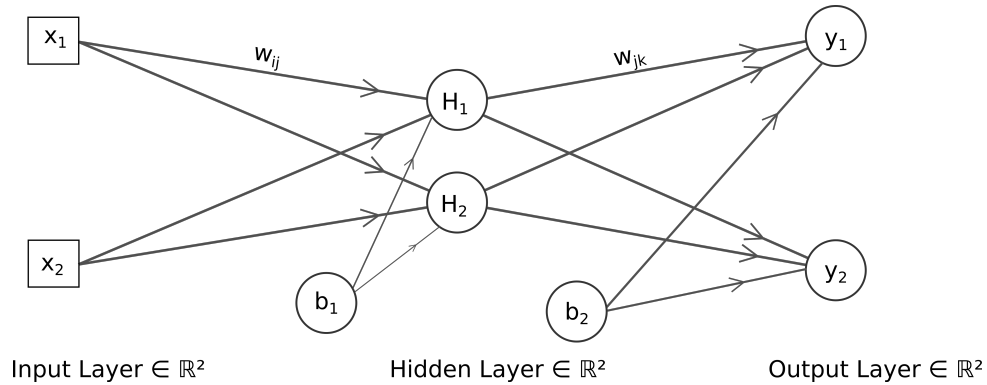


Figure 62: Multi-layer Perceptron

models prediction and the actual value is the **Mean Square Error (MSE)**

$$\text{Loss function} = \text{Error} = \frac{1}{2}(\hat{y} - y)^2$$

which can be regarded as the loss function.

Using the loss function, the weights and bias of the network can be adjusted with the aim of reducing the loss function. To reduce the loss function, gradient descent is employed to find the slope of the loss function by using partial derivative with respect to each weight in a layer, after which the weights and bias are updated until the loss function is lower. To further improve a neural network model, optimizer algorithm's are used. Since a neural network learns by adjusting its internal weights and biases. Optimizer algorithm are used to adjust the internal weights of the network with aim of minimizing the loss function.

There are different optimizer available, three will be explain in here. Gradient descent (GD), Stochastic Gradient Descent (SGD) optimizer and Adaptive Moment Estimation Algorithm (Adam) optimizer. Gradient descent optimizer is the basic optimizer found in neural networks. It functions by using a whole set of training data set of n data samples to calculate the gradient of the slope that leads to the local minima where the loss function is reduced.

Stochastic Gradient Descent (SGD) optimizer is a variant of gradient descent optimizer. In SGD instead of using the entire training data set to compute the gradients. Each data point in a training set of n data samples is used to compute the gradient.

Adaptive Moment Estimation Algorithm (Adam) optimizer , a method for efficient stochastic optimization that only requires first-order gradients with little memory requirement. The method computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients [33] By using the available

optimizer a neural network performance and accuracy can be improved.

Learning rate (LR) is a parameter of the optimizer which is used to indicate by how much should a gradient descent make. During training the objective function is to minimise the error the neural network makes and this is done by calculating the gradient of loss w.r.t each of weights of the network. The weights of the model are then multiplied by Lr. Lr is normally a small positive number between 0.0 and 1.0 doing this ensures the values of the gradient are small. The values of the gradient with Lr is then used to update the weights over several small step.

Batch size is a hyper-parameter of gradient descent that controls the number of training samples to work through before the model's internal parameters are updated.

Building a neural network involves finding the right sets of hyper-parameter which will lead to the best error loss. Hyper-parameters are user specified parameters whose values are used to control the learning process and are not derived via training. The process of fine-tuning a machine learning model consists in the determination of appropriate hyper-parameters for the desired application, resulting in faster convergence time, higher prediction accuracy, model generalization capability and efficient use of computational resources. Usually a trial and error approach is followed and different models' performances on a validation dataset are compared in order to determine appropriate hyperparameters, as they are highly dependant on the desired application. These hyper-parameters are

1. Finding the number of nodes to use for the model. Having too many neurons in a layer leads to the network to over-fitting and having a small number of neurons leads to under-fitting.
2. Finding the number of layers to use. Layer composition can be either shallow or deep. Shallow network means a small number of layer while deep network means layers greater than 3 hidden layer. A shallow network may or may not be adequate to get the best error loss and a deep network can lead to the network to over-fit.
3. Choosing the right activation function.
4. Choosing the right optimizer. Currently, the Adam optimizer is used in the ML community.
5. Choosing the right learning rate for the optimizer
6. Choosing the right number of training epochs to use.

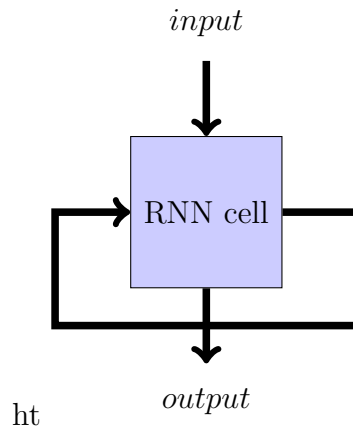


Figure 63: Basic RNN unit

7. Choosing the batch size to use.

Methods used to find the right sets of hyper-parameter values includes the use of grid search, random search and Bayesian search method. More on these method can be found in [34, 35]

In the next section A.1.3 a variant of neural network different from feed foreword network is introduced.

A.1.3 Recurrent Neural Networks

The human way of learning is based a lot on memory and prior experiences. Our decision taking is based on what we experienced before. Recurrent Neural Networks (RNN) try to copy this approach by giving the network persistent information's that the network can use for classification. Traditional neural networks do not show this kind of persistence and perform therefor relatively poor in situations that they never encountered in the exact same manner before.

Recurrent Neural Networks build upon the principles of feed forward networks, introducing a further loop back mechanism where information from previous time step t_{-1} is reused in the current time step t to produce an output. This in turn gives RNN the ability to retain information it has seen. This ability can be considered as memory, therefore RNN networks are known to have a better performance with sequence to sequence data.

Examples of applications of RNN include language translators such as Google translator or personal assistant on mobile devices such as Siri from apple and Google assistant or in time series forecasting such as stock market prediction.

In figure 63 the network receives a input and based on this input it will give an output. The loop adds a time component to this output.

For better understand the structure could be replicated by a traditional neural network by just unrolling the structure into multiple nodes like seen in figure 64. Figure 64 presents the basic structure of an RNN, where sequential inputs x_1, x_2, x_3, x_4 are fed to the RNN reproducing outputs y_1, y_2, y_3, y_4 . o_1, o_2, o_3, o_4 refer to the output of node in which information after its has passed through an activation function is shared between the nodes. By sharing information of the previous output between each node in each recurrent hidden layer, the RNN is able to learn the order of temporal data.

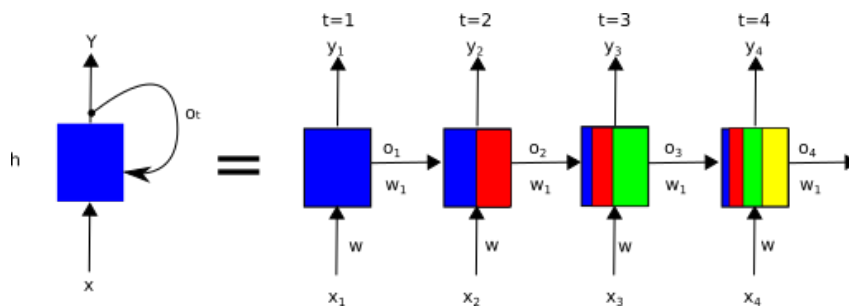


Figure 64: A recurrent neural network (RNN)

The left image shows an RNN with one input layer x , one hidden layer h , one output y and o_t represents the loop back of output into the hidden layer which becomes the new hidden state of the layer. While the image in the right shows the hidden layer unfolded, where the boxes represents nodes in the hidden layer and the colors refers to how much information is remembered by each node at time $t = 1, t = 2, t = 3, t = 4$ after training.

The vanishing gradient problem occurs during training, when the weights of the RNN network are being adjusted. This occurs due to the back propagation algorithm whose objective function is to reduce the error between what the network gives as output and the true output the network is supposed to give. The back propagation algorithm uses gradient descent to adjust the weights with respect to each layer. Each node in a layer calculates it's gradient with respect to the effect of the gradient in the layer prior to it. Meaning when the adjustments to prior layer is small, the current layers nodes weights will be even smaller when it is adjusted. As the changes to the weights in each layer are being conducted backwards, this causes the gradients to continuously become smaller. Thus the vanishing gradient problem is encountered.

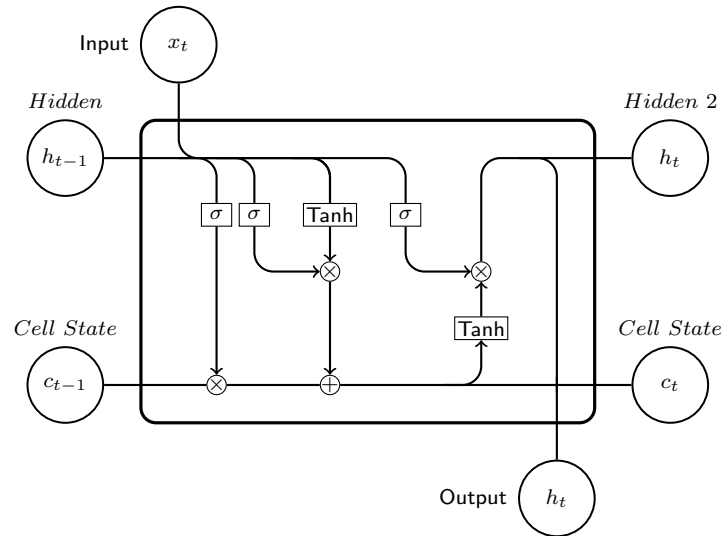


Figure 65: long short term memory (LSTM) diagram

A.1.4 Long Short Term Memory

Long Short Term Memory (LSTM) is a variant of RNN, which was developed to solve the vanishing gradient problem. The RNN structure is improved upon by having 3 gates and a cell memory in each node. These gates are: the input gate; forget gate; output gate and memory cell which all together make up an LSTM unit. Figure 65 presents an LSTM diagram which may aid better understanding of how the LSTM network functions. For LSTM the chain like structure is more advanced and consists of four layers. Every layer is designed to interact with the other layers and is explained in the following paragraphs individually.

Cell state

LSTM is keeping the cell state by a belt like structure. This cell state is shared by a LSTM cells equally and each cell has the option to manipulate its cell state in 4 different manners, also known as gates. Outer events have no influence on the cell state. A visualization of this state can be seen in figure 66.

Gates are used for manipulating the cell state. They are activated by a sigmoid layer and consist of a pointwise multiplication operation. Sigmoids layer can take a value between 0 and 1, where 0 means that the gate is closed and with 1 the influence of this gate is maximal. A LSTM consists of 3 gates.

1. The entrance gate of a LSTM unit is the forget gate. Its activation function defines the amount of information that should be forgotten or kept in the cell state. The

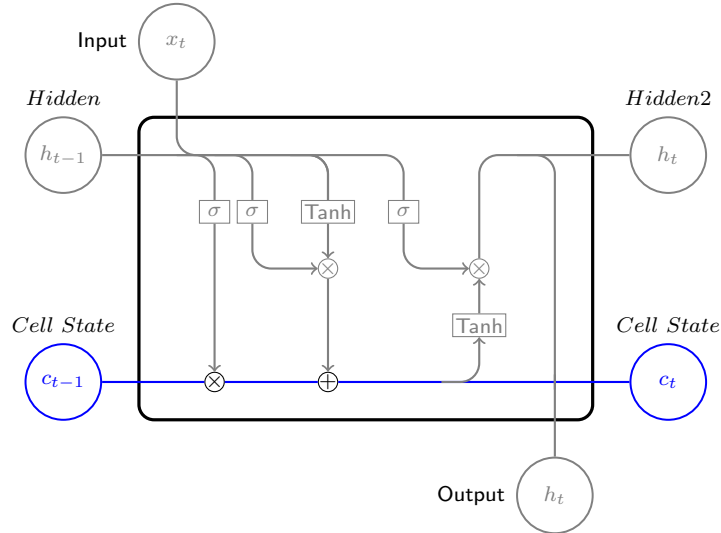


Figure 66: Cell state of a LSTM Cell

gate decides based on the previous output h_{t-1} and the input x_t , the activation function creates a vector for each feature of the cell state c_{t-1} .

The forget gate (f_t) decides what amount of information to forget from the memory of the LSTM cell. By using a **sigmoid activation function** (σ) where by it combines the prior state (h_{t-1}) .i.e the information from the previous output of a LSTM cell, and the information of the current input (x_t) .i.e. information coming from the input layer, and squashes these two information to outputs 0 or 1. Where 1 denotes retain the information and 0 denotes forget the information stored.

$$f_t = \sigma(W_{f_x} \cdot x_t + W_{f_h} \cdot h_{t-1} + b_f) \quad (\text{A.4})$$

A visualization of the the forget gate is shown in figure 67

2. The input gate (i_t) of the LSTM cell decides the information to be allowed into the memory of the LSTM cell. Two copies of the current input (x_t) and prior state (h_{t-1}) information's are created where one copy goes to a sigmoid activation function (σ) whose values are 0, 1 where 1 means allowing information into the memory and 0 denies information flow into the LSTM cell. And the second copy made goes through a tanh activation function whose output (\tilde{c}_t) are -1, 1 which is used to help regulate the LSTM cell. The product of the two activation functions are found and stored as q in equation 9 A.5.

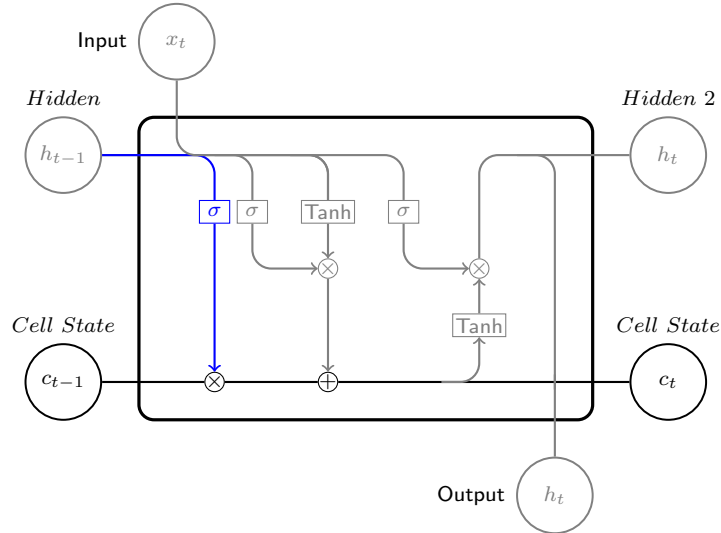


Figure 67: Forget gate of a LSTM Cell

$$\begin{aligned}
 i_t &= \sigma(W_{i_x} \cdot x_t + [W_{i_h} \cdot h_{t-1}] + b_i) \\
 \tilde{c}_t &= \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \\
 q &= i_t \cdot \tilde{c}_t
 \end{aligned} \tag{A.5}$$

3. For example one value of the cell state vector could be remembering the country of the scene for giving country specific predictions. Everytime one cell would receive another country information, the old information could be forgotten and if the information is unrelated to the location the gate would be kept close, meaning a value of the sinuide of 0.

The second gate is responsible for the updating of the cell state It consists of two parts a layer that creates possible candidates, which is implemented a tanh layer and a sigmoid layer which decides on the values of the vector state that should be updated. The two outputs are combined and added to the cell state. In the previous example the new country information would be updated. The process consists of two steps, first the state is multiplied by the forgetting state to get rid of the invalid information's and then the updated values are added to the sector.

4. The last step is necessary to decide what information to give to the next cell. Since we just found out about the country or region we would be expecting the end of the phrase or a new subject. So the cell state would encode this information in the output. The output gate (o_t) therefore, decides what information is stored in the memory of the LSTM cell. By either choosing to preserve the information already in the LSTM memory or adding more information to the memory. The output is

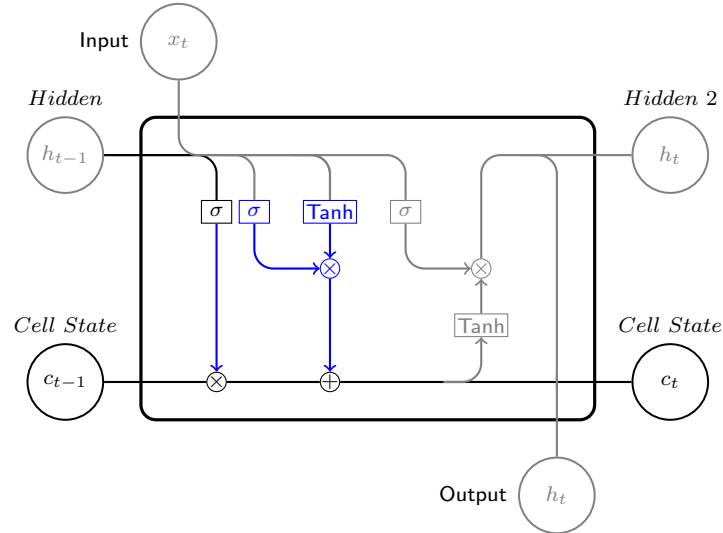


Figure 68: Update gate of a LSTM Cell

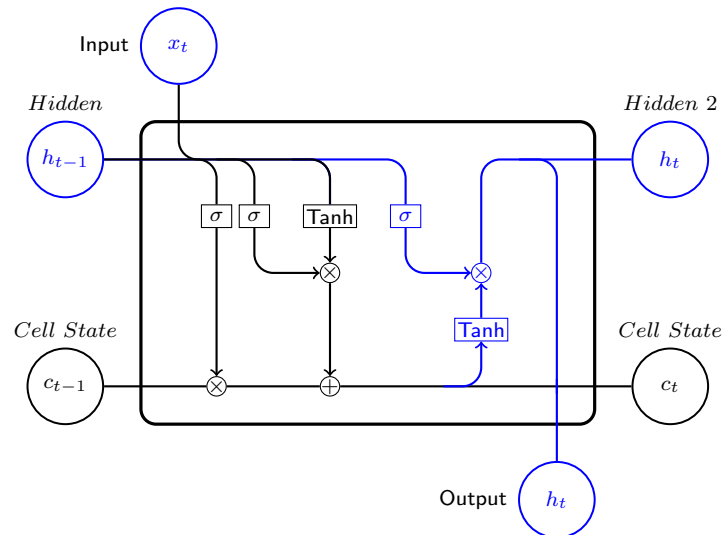


Figure 69: Output gate of a LSTM Cell

the new hidden state (h_t) of the LSTM unit which is sent to the next LSTM in the hidden layer as (h_{t-1}). The output gate take the new memory cell (C_t) information and passes it through a tanh activation function and also takes the prior hidden state (h_{t-1}) .i.e the information from the previous output of a LSTM cell, and the information of the current input (X_t) and passes it through a sigmoid activation. The outputs of these two activation functions; sigmoid and tanh are multiplied together to produce the new hidden state (h_t .)

$$o_t = \sigma_o(W_{o_x} \cdot x_t + W_{o_h} \cdot h_{t-1} + b_o) \quad (\text{A.6})$$

Figure 69 visualizes the paths that are responsible for creating a cell output.

A.1.5 Classifiers

This section focuses on different classifier implementations and their usage.

K-nearest neighbor (KNN) classifier

K-nearest neighbor (*KNN*) is a supervised learning algorithm used for classification of categorical or continuous data. It functions by classifying an unknown data sample based on known data samples group, meaning the *KNN* calculates the distance of an unknown sample to a known samples closest to it and classifies the unknown data sample to the closest known data sample. To calculate the distance, Euclidean distance metric

$$Euclidean\ distance = \sqrt{\sum_{i=1}^n (x_i - y_i)^2} \quad (A.7)$$

or Manhattan distance metrics defined by:

$$Manhattan\ distance = d(X, Y) \equiv |X_x - Y_x| + |X_y - Y_y| \quad (A.8)$$

are commonly used.

The *k*-NN classifier is an easy to use classifier. There are only two parameters a user has to make: (1) decide the number of nearest neighbour *K* the classifier takes in to account when classifying an unknown data sample. (2) decide the distance metric to use.

Decision tree (DT) classifier

Decision tree (DT) algorithms is another type of algorithm used in supervised classification of categorical or continuous data. The decision tree algorithm has a tree like structure consisting of nodes, branches and leafs. An example of DT is illustrated in figure 70. The node/s of a decision tree represent a feature or attribute of a dataset, the branches represents the decision rules made at the node and the leaf represents the outcome of the nodes where there is no further decision that can be made. The leaf is also known as the terminal node which means the data points present in this leaf node belong to a group of similar values [23].

When building a tree the algorithm aims at splitting a set of data set based on some attribute or feature. This splitting decision is made in every node. Starting from the top which is called the root node, where all the data points of a data set provided is found. An attribute or feature is chosen and a binary question is asked, based on the response

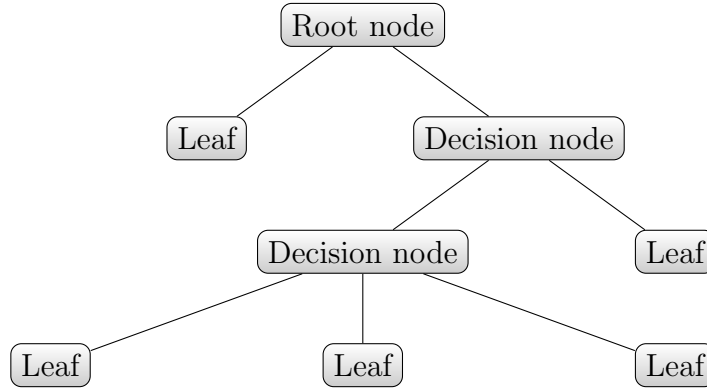


Figure 70: Decision Tree structure

the data set are split into two groups of similar attribute or features forming a node.

This process of binary questioning is then recursively carried out until a sub set of group where all the data points present are of similar values is reached. The sub set of group where all the values are of similar attribute becomes the leaf node. To build a tree, splitting decision are made in the decision node and in order to find the best attribute or feature on which a splitting decision for dividing the dataset is made. A attribute selection measures such entropy, information gain, and gini index are used [23].

Entropy whose formula :

$$Entropy(D) = \sum_{i=1}^c -P_i * \log_2 p_i \quad (A.9)$$

where D is the number of classes in a node and P is the distribution of the class i in the node.

The entropy metric is used to measure how pure or impure a set of class with data points are. That is, when entropy is one or zero this indicates the data points in leaf or in that group contains similar values or also known are homogeneous while if it is between zero and one .e.g. 0.75, this normally means the group can be further split.

Information gain whose formula:

$$Gain(D, l) = Entropy(D) - \sum_{k \in Values(l)} \frac{|D_k|}{|D|} Entropy(D_k) \quad (A.10)$$

where $Values(l)$ is the set of all the possible values for feature or attribute l and D_v is subset of D for which feature l has value k (i.e., $D_k = \{d \in D \mid l(d) = k\}$). Entropy (D) in the first part of the Equation A.10 refers to entropy of the training data set (D) and

second part

$$\sum_{k \in \text{Values}(l)} \frac{|D_k|}{|D|} \text{Entropy}(D_k)$$

refers to sum of the entropies of every subset D_k , weighted by the fraction of data points $\frac{|D_k|}{|D|}$ belonging to D_k . By using the information gain for each attribute of training data D a measure of how much reduction in entropy for using the selected attribute for making a splitting decision is gotten. which is then used to select the best attribute that will lead to a subset of group with similar values [23].

Gini Index (GI) is another way used to decide on which attribute to use for getting the best splitting of the data set that leads to the correct classification of a data point. It is used to determine the homogeneity of a splitting decision. Gini Index (GI) whose formula:

$$\text{Gini index}(D) = 1 - \sum_{i=1}^c P(k_i|D)^2 \quad (\text{A.11})$$

measures the degree of probability of a data point being wrongly assigned to a group when it is random selected. If a split is pure or of similar values, then the probability of the majority class becomes one and the probability of remaining classes is zero. and thus the GI is zero. However, if each class is represented equally, with probability $P(k_i|D) = \frac{1}{c}$, the GI then has value $\frac{c-1}{c}$.

In GI the goal is to have a low GI value which means the data points is correctly classified [36].

To build a tree, DT algorithm such as Iterative Dichotomiser 3 algorithm (ID3), C4.5 (successor of ID3) and Classification and Regression Tree algorithm (CART) are employed. ID3 uses a top-down, greedy search method to split and build trees. The ID3 employs entropy and information gain when deciding on which attribute or feature to split a dataset and it is the core algorithm most decision tree are built using. And Classification and Regression Tree algorithm (CART) uses Gini Index (GI) as attribute selection metric for building a tree [37].

Support Vector Classifier (SVC)

An Support vector classifier is a discriminative classifier defined by a separation hyper-plane. Given the labeled input data, i.e., the correct label for each observation, the training algorithm, in a SL approach, generates an optimal hyper-plane that separates observations (inputs) in categories based on its features. In two-dimensional space, this hyper-plane is a line that separates each class, where each class lay in either side.

Given a known training set T of n observations, each one represented by p features and y labels. In a binary linear SVC classifier each observation belongs to two classes either -1 or 1 ,

$$T = \{(\mathbf{x}_i, y_i), \dots, (\mathbf{x}_n, y_n)\}, \text{ with } \mathbf{x}_i \in \mathbb{R}^p, \text{ and } y_i \in \{-1, 1\},$$

the goal of the algorithm is to find a hyper-plane that separates the observation correctly. In this case the hyper-plane can be described as: $\mathbf{w} \cdot \mathbf{x}_i^T + b = 0$, with \mathbf{w} meaning weight and b meaning bias. The observation that falls to the right side of the hyper-plane has label 1 :

$$\mathbf{w} \cdot \mathbf{x}_i^T + b > 0, \text{ if } y_i = 1,$$

and the observation that falls to the left side of the hyper-plane has label -1 :

$$\mathbf{w} \cdot \mathbf{x}_i^T + b < 0, \text{ if } y_i = -1.$$

However, since there can be more than one hyper-plane that separates the data, SVM algorithm finds the support vector for each hyper-plane. Support vectors are the closest observations to a hyper-plane from both classes. SVM computes the distance between each hyper-plane and its respective support vector, known as margin. The hyper-plane with the maximum margin is chosen as the optimal hyper-plane.

Figure 71 presents a binary linear SVM, illustrating training observation separated into classes. However, in cases where the problem being solved is non-linear, a kernel function can be implemented to transform the problem into a linear one by implicitly mapping the inputs into high-dimensional feature spaces [30].

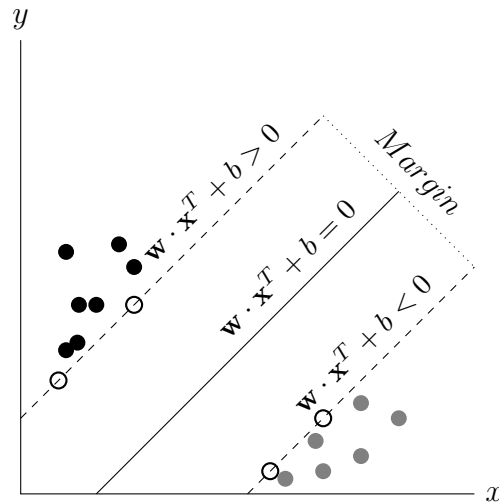


Figure 71: Binary linear Support Vector Machine. There are two classes of training observations, presented in black and gray. The hyper-plane is presented as a solid line separating the classes. Observations on the dotted lines (unfilled circles) are the support vectors.

A.1.6 Confusion matrix

A Confusion matrix is a table used to access the performance of a classification model on a set of test data-set in which the true value (actual class/ group) are known, this is generally used in supervised learning problem. Each column of the matrix table represents the instances in an actual group while the rows represents the instances of the predicted class. Table 33 present a binary classification confusion. The confusion matrix table is not limited to binary class classification it is also used for multi-class classification. Here binary class classification is shown [36].

Table 33: Confusion Matrix for two classes

		Actual Group		Total
		True (t_1)	False (t_2)	
Predicted Group	True (t_1)	<i>True Positive (TP)</i>	<i>False Positive (FP)</i>	$TP + FP$
	False (t_2)	<i>False Negative (FN)</i>	<i>True Negative (TN)</i>	$FN + TN$
Total		$TP + FN$	$FP + TN$	N

The confusion matrix puts data in to classes, there are four groups in the table which together are used to measure the accuracy of a classifier. These four groups are:

1. True positive (TP) : number of data correctly predicted by the classifier as positive.

$$TP = \{X_i | \hat{y}_i = y_i = t_1\}$$

2. True Negative (TN): number of data correctly predicted by the classifier as Negative.

$$TN = \{X_i | \hat{y}_i = y_i = t_2\}$$

3. False Positive (FP): number of data predicted by the classifier to be positive, which in reality belongs to the negative group.

$$FP = \{X_i | \hat{y}_i = t_1 \text{ and } y_i = t_2\}$$

4. False Negative (FN): number of data predicted by the classifier to be negative, which in reality belongs to the positive group.

$$FN = |\{X_i | \hat{y}_i = t_2 \text{ and } y_i = t_1\}|$$

By combining these groups; accuracy, precision, recall of a classifier can be measured. The accuracy metric in machine learning can be misleading when used only. For example in cases where there is miss-balance of data, where one class has more instances of data than other class. To prevent this metrics such as recall and precision are used to measure the model performance. Accuracy: is the portion of correctly predicted test results of the total dataset. Its formula is:

$$\frac{(TP + TN)}{(TP + FP + FN + TN)} * 100\%$$

Precision : is the fraction of relevant instances among the reclaimed instances. It is used to gauge how accurate the classification is .i.e., out of those predicted positive, how many of them are actual positive. Its formula is given as:

$$\frac{TP}{TP + FP}$$

Recall : is the fraction of relevant instances that have been reclaimed over the total number of instances. It calculates how many of the Actual Positives instances the classification result classified, it belongs to the True positive class.

Its formula is:

$$\frac{TP}{TP + FN}$$

A.2 Dynasim simulator

Dynasim is a numerical simulator application which was developed and is maintained by University of São Paulo (USP), Federal University of Alagoas (UFAL), Petrobras and Pontical Catholic University of Campinas (PUC). The application was developed for the purposes of studying the dynamic behavior of moored platforms and analysis of offloading with dynamic position system.

Dynasim uses sea environment conditions .i.e. wind, current and waves to produce times series motion of a platform position, acceleration and speed (in relation to the 6 degree of freedom of floating platform). Furthermore, Dynasim allows for coupled analysis of floats and mooring lines attached to the floaters where in the dynamics of how the floater affects the line dynamic can be conducted. The mooring lines are modeled using the *lumped mass* method. Figure 72 shows the Dynasim interface.

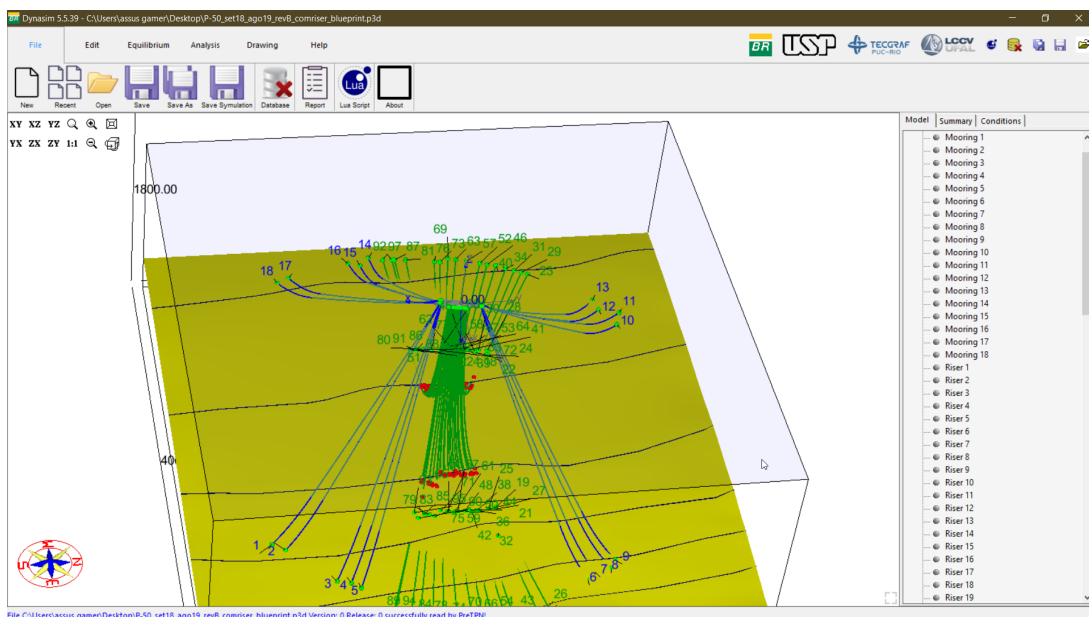


Figure 72: Dynasim Interface source: User manual- Dynasim

A.2.1 Coordinate systems

Dynasim uses two co-ordinates system (figure 73) with six degree of freedom to model the dynamics of a platform motion. The co-ordinate systems are as follows;

- Global or inertia co-ordinate system, **OXYZ** which is fixed to the ground/earth. In which the trajectory of the platform's motion over time is described in relation to this coordinate system.

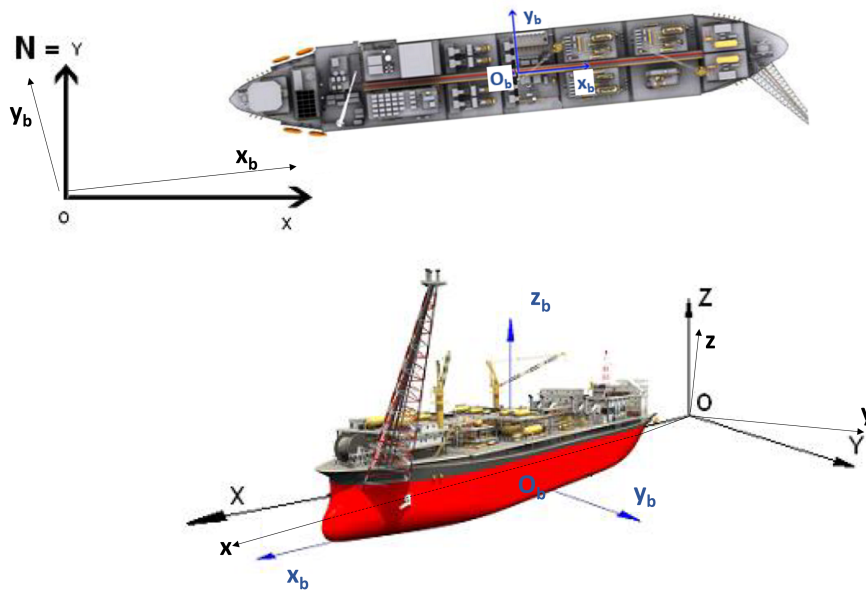


Figure 73: platform Co-ordinate System Source: User manual- Dynasim

- Local co-ordinate system, \mathbf{Oxyz} , which is centered at the same point as the global co-ordinate system, but rotated according to the platform's heading so that the local x-axis is parallel to the platform's main length (stern to bow), the local y-axis points towards the platform's port (left side when facing forward) and the local z-axis remains vertical (parallel to the global Z-axis). This reference frame is used to describe the platform's surge, sway, heave, roll, pitch and yaw positions respectively, as well as the corresponding velocities and accelerations.
- Body-Fixed co-ordinate system, $O_b x_b y_b z_b$, which is centered at the platform's center of gravity or another point of interest and is fixed to the platform. The body-fixed axes are chosen to coincide with the principal axes of inertia: x_b is the longitudinal axis (directed from stern to bow), y_b is the transversal axis (directed to starboard) and z_b is the normal axis (directed from top to bottom). This reference frame is mainly used to describe the positions of relevant points within the platform, such as fairlead connections.

The six degree of freedom of platform (figure 73) are the surge which is the linear motion along x, sway; linear motion along y, heave; linear motion along z, pitch; rotational motion around y, roll; rotational motion around x and yaw; rotational motion around z.

Unit	Hull dimension
Length between prep. (m)	320
Length overall (m)	337.359
Lateral area (m ²)	16800
Frontal area (m ²)	2041
Draft (m)	16
Beam (m)	54.5
Depth (m)	27.8
Number of riser	79
Number of mooring lines	18

Table 34: platform dimension and setup

A.2.2 Platform model

In Dynasim the platform model with all Mooring Lines and risers can be represented in a P3d file. Which describes the area and positional characteristics of each platform. Figure 73 shows a rigid body's 6DOF. The model used in this work was a P50 FPSO in front. The platform characteristics as mass, dimensions, number of risers and mooring lines are taken into consideration during simulation and can be found in 34.

A.2.3 Environmental Conditions

For the platform motion simulation it is necessary to define the environmental conditions of the sea. This chapter will give an overview over all necessary environmental conditions and their simulation in Dynasim.

Waves

Water waves are propagating on the surface of the ocean, they are a direct result of the interaction between wind and the fluid surface of the ocean. During the onset of a storm, intense winds provide energy to the waves short (high-frequency, short wavelength) that absorb the most of the energy supplied by the wind. So at the beginning of the storm there is a concentration of energy at high frequencies, and the sea is said to be developing. Next, these waves grow too big and break, dissipating energy to longer waves (lower frequencies). At this moment, with the developed sea, the energy is distributed more evenly by frequencies. To model the alteration in wave height and peak to peak times spectrum's were developed. One of the most commonly used wave spectrum in this context is the JONSWAP (Joint North Sea Wave Project), introduced in the 17th International Towing

Tank Conference (ITTC, 1984). The spectrum was developed by extensive measurements of the northern sea in the years 1968 and 1969. The model can be described with the formula;

$$S(\omega) = \frac{\alpha_0 g^2}{\omega^5} \exp\left(-\frac{5}{4} * \frac{\omega_0^4}{\omega}\right) \gamma^{\exp(-(\omega-\omega_0)^2/(2\sigma^2\omega_0^2))} \quad (\text{A.12})$$

Since the simulator was based on the environment in Rio de Janeiro the basic JON-SWAP spectrum was adapted to use specific parameters that were measured by the company Petrobras. Dynasim simulation uses local waves and global waves (swell) to simulate the platform motion. Swell usually have a long wavelength and are faster. Swells and local waves can be defined by the Peak to Peak times, by the direction and the average weight height.

Wind

Wind is the flow of air on a large scale. It consists of the bulk motion of air. The wind is defined with the parameters velocity and direction. Wind speed, however, is not constant over time. It can be considered to have a slowly varying portion in time, responsible for quasi-static efforts on the system, and a high frequency oscillating portion. This portion, known as gust, is described statistically by means of wind spectra. One of the most widely used spectral formulations for wind gusts is the so-called Harris spectrum (Harris, 1971), given by:

$$S_V(\omega) = 1146 * C * V * \left(2 + \frac{286\omega^2}{V}\right)^{-\frac{5}{6}} \quad (\text{A.13})$$

Current

An ocean current is a continuous, directed motion of sea water generated by a number of forces acting upon the water, including wind, the Coriolis effect, breaking waves, cabbeling, and temperature and salinity differences. The current is defined by the parameters velocity and direction. Conventional current force models were initially proposed by Abkowitz (1964). It is based on a Taylor series with the most important parameters velocity and direction of the swell. Since the floating platforms can be considered static other components can be neglected.

A.2.4 File Structure

Dynasim stores the Data in a binary file with the structure seen in fig ???. Where x represents the surge which y the sway, z the heave, xx the roll, yy the the pitch and zz the yaw.

Dynasim is generating position values for a predefined step size, which defaults to 0.5s. In our project a simulation step size of 1 second is used.

A.3 Standardization of Measurements

Since different features come with different value ranges it is important to standardize the value ranges before working with them. In statistics, standardization is the process of putting different variables on the same scale. This process allows you to compare scores between different types of variables. The standardization process we use is very simple, here called scaling. If no scaling is done different characteristics gains more influence than others.

To equalize differences in the dimensions of the characteristics, the values are standardized to a value range between zero and one. This is done for each column (i.e., each feature) individually. First the Peak to Peak value is calculated by finding the minimum and maximum value of each individual column and calculating their difference, using

$$PP = Max - Min. \quad (A.14)$$

Then the values are scaled by

$$X_{out} = X_{in} - Min/PP \quad (A.15)$$

By doing so the minimum value becomes 0 and the maximum value becomes 1. Therefore scaling distributes not only the values to a standardized range but also get rids of possible offsets.

Table 35 shows example Data in different value ranges. It can be observed that the value ranges are different and each feature can have other offsets. Table 3 shows the resulting values after scaling. After each Column is scaled individually it can be observed that each individual column is now standardized. Additionally the scaling got rid of the

Table 35: Unscaled Dataset

x	y	z
30	0	-20
40	0.5	-25
35	1	-30

Table 36: Scaled Dataset

x	y	z
0	0	1
1	0.5	0.5
0.5	1	0

offsets.

A.4 Specifications

This section focuses on the tools used in this project for the code implementation and the mechanics used for easier deployment. Some of those best practices have already been implemented, as we now describe.

A.4.1 Git

The tool Git is part of any serious code development. Git enables team work around the same code base, keeps track of coding development and allows developers to have a flexible workflow. There are many different ways to interact with Git as a team using *Git Workflows* or *Branching Models*. There is a myriad of Git Workflows available to use, each of them with different goals and application. Although there is not a set of rules to define a Git Workflow, the following considerations should be taken while choosing one, according to Atlassian's article on Git Workflows [38]:

- Does the workflow scales with team-size?
- Is it easy to undo mistakes and errors with this workflow?
- Does this workflow impose any new unnecessary cognitive overhead to the team?

Following those principles, the most suitable workflow for our project is the *Feature Branch Workflow* [2].

A.4.1.1 Branch Workflow

The branch workflow assumes there is a central repository, usually named **main** or **master**. The **main** repository represents the official project history, but instead of committing code directly to **main**, developers create a new branch every time they start to work on a new feature.

After finishing their work, developers issue a **pull request**; in other words, they request their code to be merged into **main**. The request is then analyzed by other developers of the project that decide on merging the code or not.

The reasons that support the choice of this workflow are:

- Its simplicity and ease of implementation, avoiding an unnecessary overhead;

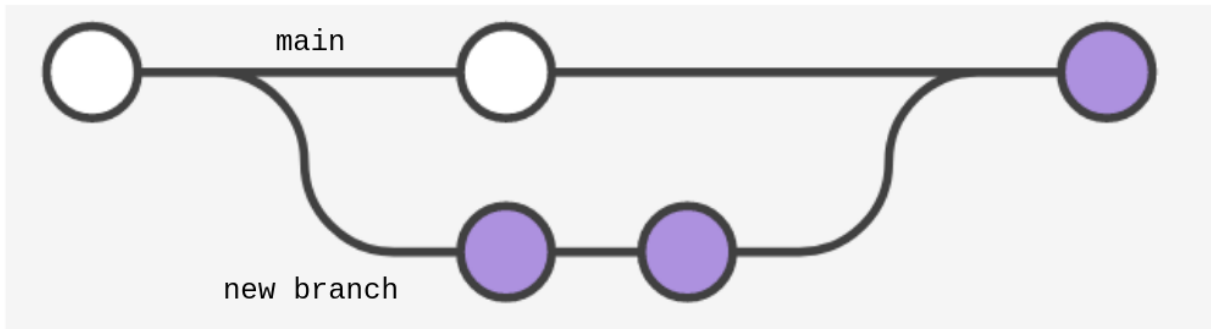


Figure 74: The main repository and a new branch [2].

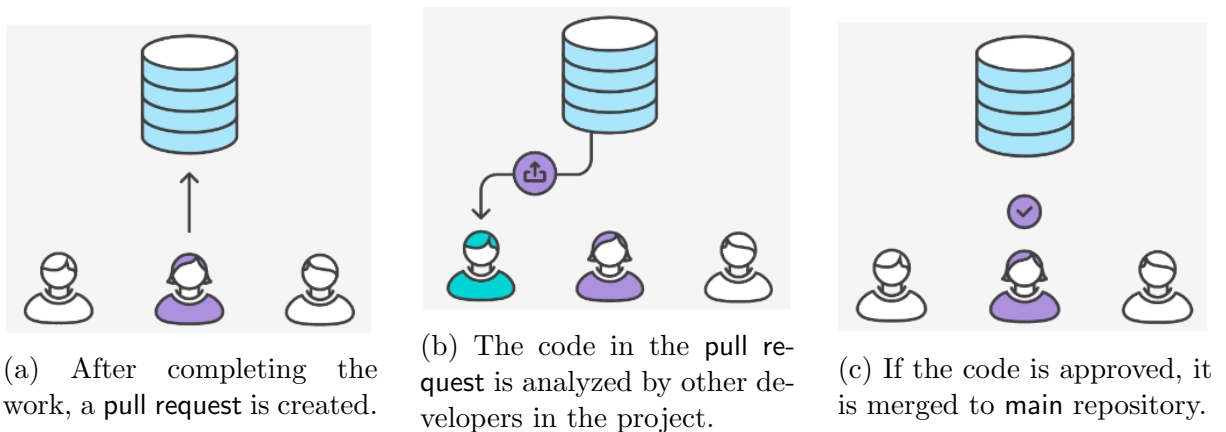


Figure 75: The pull request concept on the *feature branch* git workflow [2].

- It enhances the interaction between the team members, because of the **pull request** concept;
- It makes easier to track code evolution and to undo errors and mistakes.

In any case, the adoption of a Git Workflow is not definitive, and it can be modified as the project evolves in order to adapt to new scenarios.

To enforce this workflow, pushes to the main branch are prohibited in the project settings.

A.4.1.2 Continuous integration

To ensure high code quality and full code functionality, continuous integration scripts are implemented in the git workflow. Continuous integration runs different scripts after each individual push request online. The scripts in this project control code structure using the python script flake8 and run Unit tests using the script pytest. Flake8 verifies that the code does not have unnecessary imports and that the structural form of the code like line length and indent are equal for all the different files. Unit tests is testing 95 %

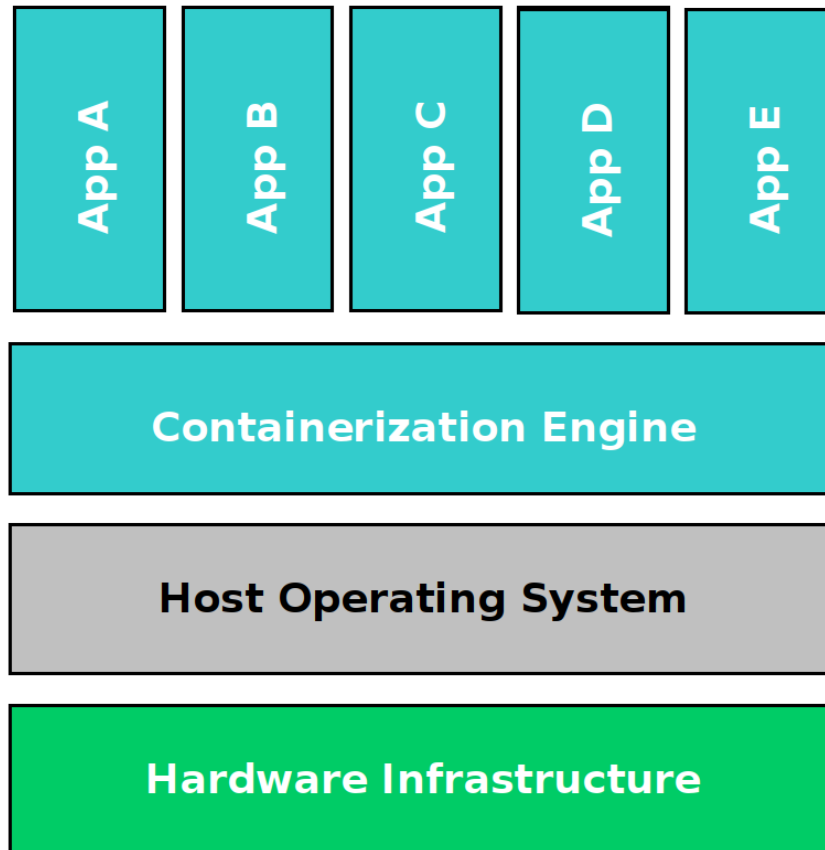


Figure 76: Containerized applications.

of all the code in this project, testing the functionality of the single code blocks.

To further reduce the risk of push requests of nonfunctional code, pre-commit scripts are executed locally before new code can be committed. If the unit tests or the layout tests fail the new code is rejected and can not be committed.

Merge requests can only be completed when the continuous integration completed without errors.

A.4.2 Containerization

Containerization refers to the process of building software that package up code and all its dependencies, allowing the application to run quickly and reliably in different computing environments [39]. There are many containerization engines; the most popular are Docker and Kubernetes. The main difference between them is that Kubernetes is meant to run in a computer cluster, while Docker runs in a single computer. That said, it is possible to use them together for scaling and delivery containerized applications. The Figure 76 illustrates the layers involved in containerization.

At the current stage, our project is still on its experimentation phase. Hence the Docker engine is the better choice for containerization (as scaling is not a concern at the moment). Nonetheless, it will be possible to integrate with Kubernetes when scaling becomes relevant issue.

All ML applications developed in our project run inside Docker containers. Dependencies needed to run the ML applications are defined through Docker Images, which are fully described and configured with text files named `Dockerfiles` [40]. This practice improves control over the computing environments used on development.

A.4.3 Software specifications

The chapter addresses the used software components as well as the implemented mechanics.

A.4.3.1 Python and libraries

Python is the most commonly used high level programming language for machine Learning problems, since it comes with a lot of libraries for machine Learning. It is object oriented and relatively simple to learn due to its unique syntax and dynamic binding options for variables. The most commonly Machine Learning library is Keras which is presented in the following chapter.

Keras is a python library for neural network development. It is open-source and focused on easy usability, due to its modular extension structure. Due to its open source structure and huge community Keras nowadays is one of the most commonly used machine Learning library. Frameworks like TensorFlow build on Keras and facilitate the complex data flow implementations, by integration further mathematical functionality to the Keras library. Moreover, given the great adherence of the community to Keras, there is a great abundance of support and examples available on the Internet, which reduces the learning curve necessary for the practical application of neural networks.

Pandas is a data processing library written in Python. Panda allows easy data Loading and manipulation from different sources. Data is loaded in so called Dataframes which represented the data in form of an 2 Dimensional array, from which it is possible to perform several functions such as re-sampling, grouping and several other functions necessary for data processing. The library is widely used by the community and there is a large volume of support material available on the internet, making it extremely simple to use.

NumPy is a library written in Python, which adds multi-dimensional matrices support. It incorporates a lot of high-performance and high-level mathematical functions to operate on the multi-dimensional arrays. NumPy is open-source software and has many contributors. This library is widely used in any data manipulation driven programs with a large community and a lot of usage examples online, which facilitates the usage.

A.4.3.2 Website

The website was created using react with material-ui components for the front end.

React is an open source JavaScript library focused on creating user interfaces on web pages. It is maintained by Facebook, Instagram, other companies and a community of individual developers.

The back end was for project integration purposes developed using flask. Running on a simple rest api for generating simulations and requesting information.

The API back end contains interactive graphics the code documentation and the testing overview.

A.4.4 System Specification

The Training and Simulation was executed on four Desktop working stations with Intel Icore-7 CPU@ 3.40GHz located in the University of São Paulo. The four workstations were equipped with a GTX 1080 GPU and the Training process was done using the Keras GPU library and the GPU capabilities of the system. The access to the systems was established via SSH and Jupyter Notebooks.

REFERENCES

- [1] Mark A. Cohen and Alan Krupnick. Deepwater drilling: Recommendations for a safer future.
- [2] Atlassian Blog. Feature-Branch-Workflow. <https://www.atlassian.com/git/tutorials/comparing-workflows/feature-branch-workflow>, 2020. Accessed at 2020-10-06.
- [3] Kai-Tung Ma, Yong Luo, Chi-Tat Thomas Kwan, and Yongyan Wu. *Mooring System Engineering for Offshore Structures*. Gulf Professional Publishing, 2019.
- [4] Darrell Leong, Ying Min Low, and Youngkook Kim. Long-term extreme response prediction of mooring lines using subset simulation. In *ASME 2018 37th International Conference on Ocean, Offshore and Arctic Engineering*. American Society of Mechanical Engineers Digital Collection, 2018.
- [5] Ma Kai-Tung, Yong Luo, Kwan Thoma, and Wu Yongyan. *Mooring System Engineering for Offshore Structures*. Elsevier, Oxford, UK, 1 edition, 2019.
- [6] Kai-tung Ma, Hongbo Shu, Philip Smedley, Didier L’Hostis, and Arun Duggal. A historical review on integrity issues of permanent mooring systems. In *Offshore Technology Conference*, page 14, Houston, Texas, USA, 2013. Offshore Technology Conference.
- [7] Dae Hyuk Kim and Nakwan Kim. An auto weather-vaning system for a DP vessel that uses a nonlinear controller and a disturbance observer. *International Journal of Naval Architecture and Ocean Engineering*, 6(1):98–118, mar 2014.
- [8] Dmitry Sadovnikov and Piotr Sujkowski. Maintaining Integrity of FPSO Mooring System. <https://www.semanticscholar.org/paper/Maintaining-Integrity-of-FPSO-Mooring-System-Sadovnikov-Sujkowski/b862a48f6ada4f64a09999707de59925ca7226b4>, 2012.
- [9] Kai-tung Ma, Hongbo Shu, Philip Smedley, Didier L’Hostis, Arun Duggal, et al. A historical review on integrity issues of permanent mooring systems. In *Offshore technology conference*. Offshore Technology Conference, 2013.
- [10] Emmanuel Fontaine, Andrew Kilner, Christopher Carra, Daniel Washington, KT Ma, Amal Phadke, Derrick Laskowski, Greg Kusinski, et al. Industry survey of past failures, pre-emptive replacements and reported degradations for mooring systems of floating production units. In *Offshore Technology Conference*. Offshore Technology Conference, 2014.
- [11] Martin G Brown, TD Hall, DG Marr, M English, RO Snell, et al. Floating production mooring integrity jip-key findings. In *Offshore technology conference*. Offshore Technology Conference, 2005.

- [12] Sandip Ukani, Walter Maurel, Renaud Daran, et al. Mooring lines-integrity management. In *Offshore Technology Conference*. Offshore Technology Conference, 2012.
- [13] Robert B Gordon, Martin G Brown, Eric M Allen, et al. Mooring integrity management: a state-of-the-art review. In *Offshore Technology Conference*. Offshore Technology Conference, 2014.
- [14] Det Norske Veritas Group. Machine learning can make mooring safer and more cost effective - dnv gl. <https://www.dnvgl.com/oilgas/perspectives/machine-learning-can-make-mooring-safer-and-more-cost-effective.html>, 2019. Accessed: 2019-10-03.
- [15] Da Tang, Xianpeng Zeng, Deyu Wang, Wenhua Wu, Yanlin Wang, Qianjin Yue, Bingsen Wang, Bin Xie, Shisheng Wang, and Jiaguo Feng. The research of soft yoke single point mooring tower system damage identification based on long-term monitoring data. *Applied Ocean Research*, 76:139–147, 2018.
- [16] Igor Prislín, Soma Maroju, et al. Mooring integrity and machine learning. In *Offshore Technology Conference*. Offshore Technology Conference, 2017.
- [17] JM Gumley, MJ Henry, and AE Potts. A novel method for predicting the motion of moored floating bodies. In *ASME 2016 35th International Conference on Ocean, Offshore and Arctic Engineering*. American Society of Mechanical Engineers Digital Collection, 2016.
- [18] Djoni E Sidarta, Jim O’Sullivan, and Ho-Joon Lim. Damage detection of offshore platform mooring line using artificial neural network. In *ASME 2018 37th International Conference on Ocean, Offshore and Arctic Engineering*. American Society of Mechanical Engineers Digital Collection, 2018.
- [19] Djoni E Sidarta, Johyun Kyoung, Jim O’Sullivan, and Kostas F Lambrakos. Prediction of offshore platform mooring line tensions using artificial neural network. In *ASME 2017 36th International Conference on Ocean, Offshore and Arctic Engineering*. American Society of Mechanical Engineers Digital Collection, 2017.
- [20] Vivek Jaiswal, Alex Ruskin, et al. Mooring line failure detection using machine learning. In *Offshore Technology Conference*. Offshore Technology Conference, 2019.
- [21] François-Xavier Siréta, Ding Zhang, et al. Smart mooring monitoring system for line break detection from motion sensors. In *The Thirteenth ISOPE Pacific/Asia Offshore Mechanics Symposium*. International Society of Offshore and Polar Engineers, 2018.
- [22] Shifeng Li and Zhanzhi Qiu. Prediction and simulation of mooring ship motion based on intelligent algorithm. In *2016 Chinese Control and Decision Conference (CCDC)*, pages 1556–1560. IEEE, 2016.
- [23] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.
- [24] Kyunghyun Cho, Bart van Merriënboer, Çağlar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014.
- [25] Aleksander Kołcz, Abdur Chowdhury, and Joshua Alspector. Data duplication: An imbalance problem? 2003.

- [26] Pedro Domingos. A few useful things to know about machine learning. *Communications of the ACM*, 55(10):78–87, 2012.
- [27] Arezoo Aghaei Chadegani, Hadi Salehi, Melor Yunus, Hadi Farhadi, Masood Fooladi, Maryam Farhadi, and Nader Ale Ebrahim. A comparison between two main academic literature collections: Web of science and scopus databases. *Asian Social Science*, 9(5):18–26, 2013.
- [28] Alexander Selvikvåg Lundervold and Arvid Lundervold. An overview of deep learning in medical imaging focusing on MRI. *CoRR*, abs/1811.10052, 2018.
- [29] Mohammed Al-Qizwini, Iman Barjasteh, Hothaifa Al-Qassab, and Hayder Radha. Deep learning algorithm for autonomous driving using googlenet. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 89–96. IEEE, 2017.
- [30] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013.
- [31] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.
- [32] A. Gensler, J. Henze, B. Sick, and N. Raabe. Deep learning for solar power forecasting — an approach using autoencoder and lstm neural networks. In *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 002858–002865, 2016.
- [33] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [34] Pushparaaja Murugan. Hyperparameters optimization in deep convolutional neural network / bayesian approach with gaussian process prior. *CoRR*, abs/1712.07233, 2017.
- [35] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.*, 13(null):281–305, February 2012.
- [36] Mohammed J Zaki and Wagner Meira Jr. *Data Mining and Machine Learning: Fundamental Concepts and Algorithms*. Cambridge University Press, 2019.
- [37] Wei-Yin Loh. Classification and regression trees. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(1):14–23, 2011.
- [38] Atlassian Blog. Comparing-Git-Workflows. <https://www.atlassian.com/git/tutorials/comparing-workflows/>, 2020. Accessed at 2020-10-06.
- [39] Docker. What-is-a-Container? <https://www.docker.com/resources/what-container>, 2020. Accessed at 2020-10-07.
- [40] Rodrigo Cunha. Dockerfile-Reference. <https://git.tpn.usp.br/grupo-ai-amarracao/sgs/>, 2020. Accessed at 2020-10-07.