AMIR MUHAMMED SA'AD

NEMO: a Neural Motion Estimator for Mooring Line Failure Detection of Offshore Platforms

São Paulo, Brazil
2023

AMIR MUHAMMED SA'AD

NEMO: a Neural Motion Estimator for Mooring Line Failure Detection of Offshore Platforms

Corrected Version

Doctoral thesis presented to the Escola Politécnica da Universidade de São Paulo to obtain the degree of Doctor of Science.

Concentration field:
Computer Engineering

Advisor:
Profa. Dra. Anna Helena Reali Costa

São Paulo, Brazil
2023

Catalogação-na-publicação

Sa'ad, Amir Muhammed
NEMO: A NEURAL MOTION ESTIMATOR FOR MOORING LINE FAILURE DETECTION OF OFFSHORE PLATFORMS / A. M. Sa'ad -- versão corr. -- São Paulo, 2023.
129 p.

Tese (Doutorado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1.Mooring line breakage 2.Classification 3.Machine Learning 4.Neural Networks 5.Offshore Platforms I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais II.t.

ACKNOWLEDGMENTS

I wish to express my profound gratitude and deep appreciation to Dra. Profa. Anna Helena Reali Costa, my esteemed supervisor, for her continuous support, invaluable guidance, unwavering encouragement, profound understanding, and boundless patience throughout the course of my research. Additionally, I extend my sincere thanks to Prof. Dr. Eduardo Aoun Tannuri, my esteemed co-supervisor, for his invaluable support and guidance. Special thanks to all the members of the Digital Twins research team.

I would also like to take this opportunity to express my gratitude to the Petroleum Technology Development Fund (PTDF), my esteemed sponsors, for their generous support.

Furthermore, I express my utmost gratitude to my family, whose dedication, and unwavering support have been a constant source of inspiration and motivation for me.

Lastly, I extend my sincere thanks to all those who have supported me during my study. Your support has been instrumental in helping me achieve my goals, and I am deeply appreciative of your efforts.

ABSTRACT

Floating offshore structures are maintained in the desired position by mooring lines attached to the seabed of the location. These systems are among the main components that guarantee not only the safety of the crew but also the various operations carried out on the platforms. In this thesis, the objective is to detect the rupture of the mooring lines of platforms with different levels of draft (load) based on the measurements of the platform motion provided by the Differential Global Positioning System (DGPS) and Inertial Measurement Unit (IMU) sensors. For this, a Neural Motion Estimator (NeMo) system was developed. NeMo consists of two modules: a motion prediction module comprising of a *feed forward* neural network (Multilayer Perceptron – MLP), which uses previous data from platform motions to predict future motion, and a multi-class classifier module, which uses the difference between predicted motion and measured actual motion as inputs to indicate whether or not there has been a failure, for various groups of mooring lines. The system was trained and tested using simulated data from a time- domain platform motion simulator. Results of the implemented NeMo system showed it is able to detect the occurrence of failure in the mooring lines, with errors between the forecast and the measured movements when there was a line breakage. These errors are such that the developed multi-class classifier had a 99% accuracy prediction rate when classifying the platform motions.

Keywords – Mooring lines breakage, Classification, Machine learning, Neural networks, Offshore Platforms.

RESUMO

Estruturas flutuantes offshore são fixadas no local desejado por meio de cabos de amarração ancorados no fundo do mar. Esses sistemas estão entre os principais componentes que garantem não só a segurança da tripulação, mas também das diversas operações realizadas nas plataformas. Nesta tese, o objetivo é detectar a ruptura dos cabos de amarração de plataformas, com diferentes níveis de calado (carga), com base nas medidas do movimento da plataforma fornecidas pelos sensores do Sistema de Posicionamento Global Diferencial (DGPS) e da Unidade de Medição Inercial (IMU). Para isso, foi desenvolvido o sistema *"Neural Motion Estimator"* (NeMo). O sistema é composto por dois módulos: um módulo de previsão de movimento composto por uma rede *feed forward* (Multilayer Perceptron – MLP), que usa dados prévios dos movimentos da plataforma para prever o movimento futuro, e um módulo classificador, que usa a diferença entre o movimento previsto e o movimento real medido como entradas para um classificador multiclasse que indica se houve ou não uma falha, para vários grupos de cabos de ancoragem. Os resultados do sistema NeMo mostram que ele é capaz de detectar a ocorrência de falhas nos cabos de ancoragem, mostrando erros entre os movimentos preditos e medidos quando houve um rompimento de cabo. Esses erros são tais que o classificador multiclasse desenvolvido teve uma acurácia de previsão de 99% ao classificar o movimento da plataforma.

Palavras-Chave – Rompimentos de Cabos de ancoragem, Classificação, Aprendizado de Máquina, Redes Neurais, Plataformas offshore.

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

3DoF Three Degree of Freedom – Surge, Sway and Yaw.

6DoF Six Degree of Freedom – Surge, Sway, Pitch, Heave, Roll and Yaw.

ANN Artificial Neural Network.

AutoML Automated Machine Learning.

BD Breakage Detection.

BO Bayesian Optimization.

BP Back-Propagation.

CNN Convolutional Neural Network.

DGPS Differential Global Positioning System.

DLP Damage Location Prediction.

DT Decision Tree.

EI Expected Improvement.

FN False Negative.

FP False Positive.

FPSO Floating Production Storage and Offloading.

GI Gini Index.

IMU Inertial Measurement Unit.

KNN K-nearest Neighbour.

LR Logistic Regression.

LSTM	Long Short Term Memory.
ME	Mean Error.
MedE	Median Error.
ML	Machine Learning.
MLP	Multilayer Perceptron.
MLR	Multinomial Logistic Regression.
MSE	Mean Squared Error.
NAS	Neural Architecture Search.
NeMo	Neural Motion Estimator.
NN	Neural Network.
RBF	Radial Basis Function.
ReLU	Rectified Linear Unit.
RL	Reinforcement Learning.
RMSE	Root Mean Square Error.
ROV	Remote Operated Vehicles.
RS	Random Search.
SA	Simualted Annealing.
SL	Supervised Learning.
StD	Standard Deviation.
SVC	Support Vector Classifier.
TN	True Negative.
TP	True Positive.
UL	Unsupervised Learning.
VRM	Vessel Response Monitoring.

CONTENTS

1 INTRODUCTION

Floating platforms have been used over the years to safely traverse to offshore locations with oil deposit for extraction and production of these deposits (MA et al., 2019). These platforms allow for storage of extracted crude oil on-board and are kept in the desired location using mooring lines that are attached to the platform and to the sea bed of the location.

Mooring systems are part of the key components that provide safety to personnel on-board the platform and they allow for the different operations conducted such as drilling, production and offloading of crude oil on the platform. These structures normally operate in deeper waters, where the need for continuous monitoring of the integrity of the platforms' mooring system is crucial, as system failure can result in the platform being deviated from the desired location, which can jeopardize the personnel on-board the platform and cause environmental pollution such as oil spillages. In the oceans, these platforms continuously experience stress and strain caused by waves, current and wind. As a consequence, the structural integrity of the platform and mooring lines are diminished during the life time of these components.

Studies by Ma et al. (2013) and Fontaine et al. (2014) have shown around 45% of mooring system failures are either single line failures or multiple line failures which can be attributed to corrosion and fatigue. In some cases of single-line mooring failure, additional mooring lines may be damaged due to the increased load, stress and strain experienced by the remaining lines, as single-line failure inadvertently increases the rate of degradation of the remaining mooring lines. Identifying mooring line damage is predominantly done through visual inspection of the mooring lines at a scheduled time within a scheduled time frame. However, it is difficult to identify damage because inspection operations by a Remote Operated Vehicles (ROV) are economically costly and also the resolution of the images is often relatively low (AYERS; O'HEAR, 2007).

To overcome these limitations, damage detection by mooring line monitoring methods based on sensors have been proposed in recent years. In those methods, damage is detected by analyzing the responses, line tension and inclination obtained through monitoring the mooring lines using inclinometers and load-cells (GAUTHIER; ELLETSON, 2014). However, when sensors are attached for mooring line monitoring, the results may include noises from the underwater environment and the frequency in which these sensors fail makes maintenance costly, thus, lowering the economic efficiency of the methods.

Damage detection by platform motion monitoring, instead of mooring lines, has also been proposed. Since these methods require sensors installed on the platform only, the monitoring results include less environmental noise and are more economical. However, as the platform response does not significantly change in proportion to the mooring lines damage and exhibits a very complicated trend due to the influence of the sea environment (such as waves or winds), local damage to the mooring lines may be difficult to detect by platform motion monitoring.

Recent improvements in technology and increase in processing capacity of computer systems with new techniques, algorithms, and theories feature solutions based on Machine Learning (ML) for real-time monitoring and failure detection of mooring system, bringing new possibilities for monitoring performed directly through the platform motion.

ML, a subset of Artificial Intelligence in which the models are built based on data by an algorithm with no explicit instructions being provided, is a good alternative for use in monitoring mooring systems. The availability of large amount of data measured from the platforms sensors, motion monitoring systems and environmental measurements in recent years with great quality can be used to train ML algorithms.

Nevertheless, Platform motions in the offshore with an intact mooring system and a single mooring line damage show subtle variation in their response to environmental measurements making it challenging to know when mooring line failure has occurred assertively. Therefore, ML models that are useful for identifying subtle differences in intricate patterns can be implemented to detect these changes.

1.1 Problem description

The platform we studied are located offshore and, therefore, are subject to different environmental conditions, which affect their motion. Another contributing factor to changes in platform motion is related to the draft level, which is the amount of crude oil and ballast water stored on the platform. Figure 1 illustrates how a platform with two draft levels displaces the surrounding water around the platform, causing it to sink deeper depending on the platform draft as shown in figure (1a), a platform with high load (fully loaded), and figure (1b), a platform with empty tanks (ballasted).

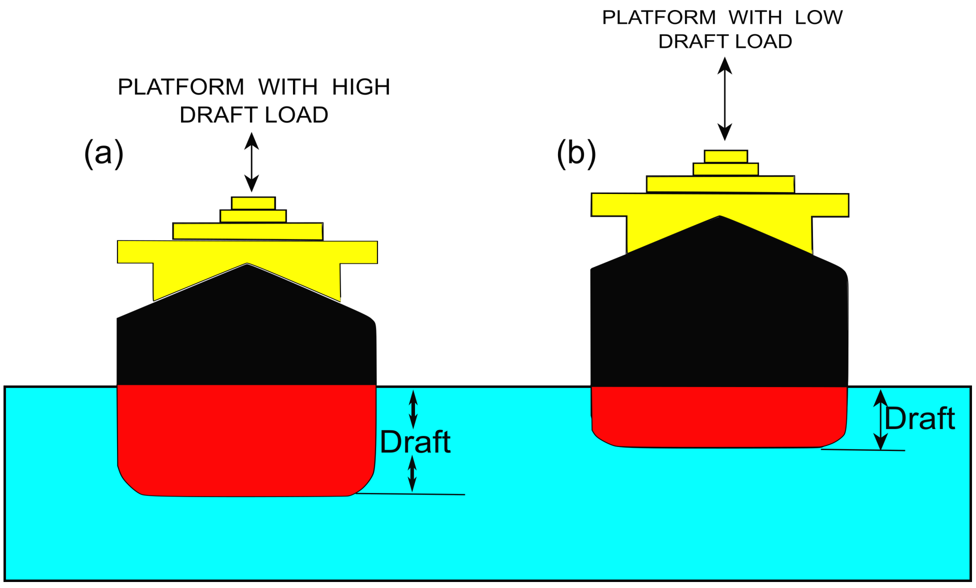


Figure 1: Platform with two drafts, (a) fully loaded and (b) Ballasted. Source:(MEYER et al., 2016)

Figure 2 shows how each extreme level of draft, 8 meters (ballasted) in blue and 21 meters (fully loaded) in orange, affects the platforms' response for the same environmental conditions by exhibiting different motions.

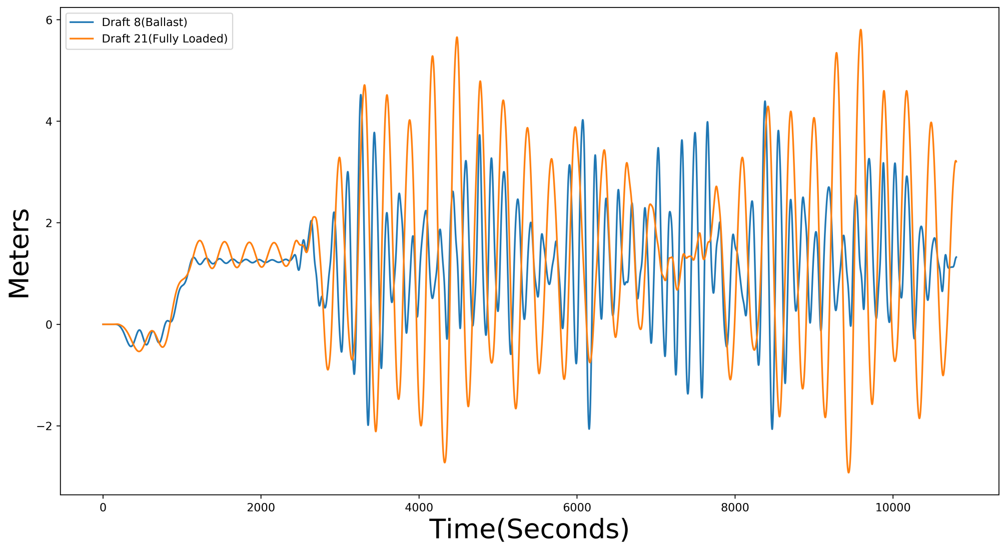
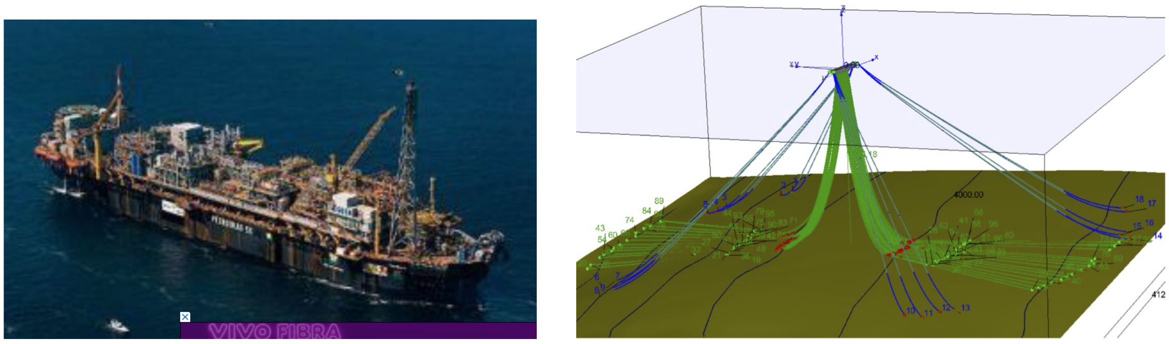


Figure 2: Simulated surge motion response of the platform for two draft levels. In blue is the ballasted surge platform motion response and in orange is the surge platform motion response with load (fully loaded).

Bearing in mind that the mooring lines used to secure the platform have to withstand high tension and stress loads, breakage can occur at any time. Therefore, the problem that we have to identify whether there was a line breakage or not also involves other factors that influence the motion of the platform, such as the draft level, weather conditions, and which of the lines was ruptured, for example.

In this thesis, a Floating Production Storage and Offloading (FPSO) platform named P50 (see figure (3a)), with 18 mooring lines attached for station-keeping and 78 risers attached for extraction of crude oil and natural gas illustrated in figure (3b), stationed at the Campos Basin (Bacia de Campos) of Rio de Janeiro, RJ, Brazil is used.



(a) FPSO P50 Platform. (b) P50 mooring lines & riser configuration.

Figure 3: Figure (3a) shows an FPSO P50 platform in the sea. Figure (3b) shows the P50 mooring lines and riser configuration. The green lines are the risers connected to the FPSO while the green lines with blue tips are the mooring lines anchored to the seabed.

The Floating Production Storage and Offloading (FPSO) platform has sensors that monitor its motion. In particular, Differential Global Positioning System (DGPS) and an Inertial Measurement Unit (IMU). A Global Positioning System (GPS) is a satellite-based navigation system which is used to give location information via a network of orbiting satellites. However, the GPS signals can be distorted by atmospheric conditions leading to erroneous location data. Thus, leading to the development of DGPS which improves GPS accuracy by using a network of stationary reference receivers with known positions to compare the GPS signals received by a mobile receiver, and it then compares the GPS signals from the reference receivers to those gotten from the mobile receiver and uses that to determine the errors caused by atmospheric conditions. The DGPS system then applies these corrections to the GPS signals received by the mobile receiver, resulting in much more accurate location information.

On the hand, Inertial Measurement Unit (IMU) are devices that provide information of a platform/ object's motions along the Six Degree of Freedom – Surge, Sway, Pitch, Heave, Roll and Yaw (6DoF) motions (shown in Figure 4), using accelerometers and gyroscopes in real-time. By combining these two system; DPGS and IMU, more precise and accurate platform location are gotten (CHIANG et al., 2004).

The 6DoF are defined as follows:

Surge: is the longitudinal movements,

Sway: is the lateral movements,

Yaw: is the rotation about the vertical axis,

Pitch: is the rotational motion about the transverse axis,

Roll: is the rotation about the longitudinal, and

Heave: is the vertical motion,

Which are measured at a fixed vessel reference frame, in this work, the 6DoF motions are generated using a simulator. The simulator takes into account the position of the platform and other configurations, such as draft setting, to simulate the platform's response to different environmental conditions.

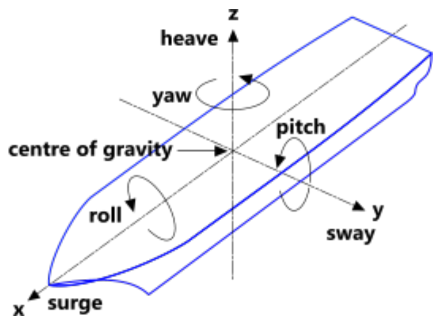


Figure 4: The six degrees of freedom motion (6DoF) of a platform in 3D plane.

A single mooring line breakage manifest itself in form of the platform experiencing a change in motions. Figure 5 illustrates how the 6DoF motions are affected after breakage. As it can be seen, there are sudden modifications in the 6DoF motions after breakage at time 5000 seconds demarcated by the red rectangle box, with transient oscillations and a new equilibrium position, due to the broken line.

For a single line break, as shown, the motions can change only slightly on some degree of freedom, which makes it difficult to detect a mooring line break. As these variations in the overall platform motions can also be attributed to other factors, such as environmental condition and level of platform draft, it is difficult to clearly define when there was a break in the mooring line, requiring a more sensitive and elaborate method for the process.

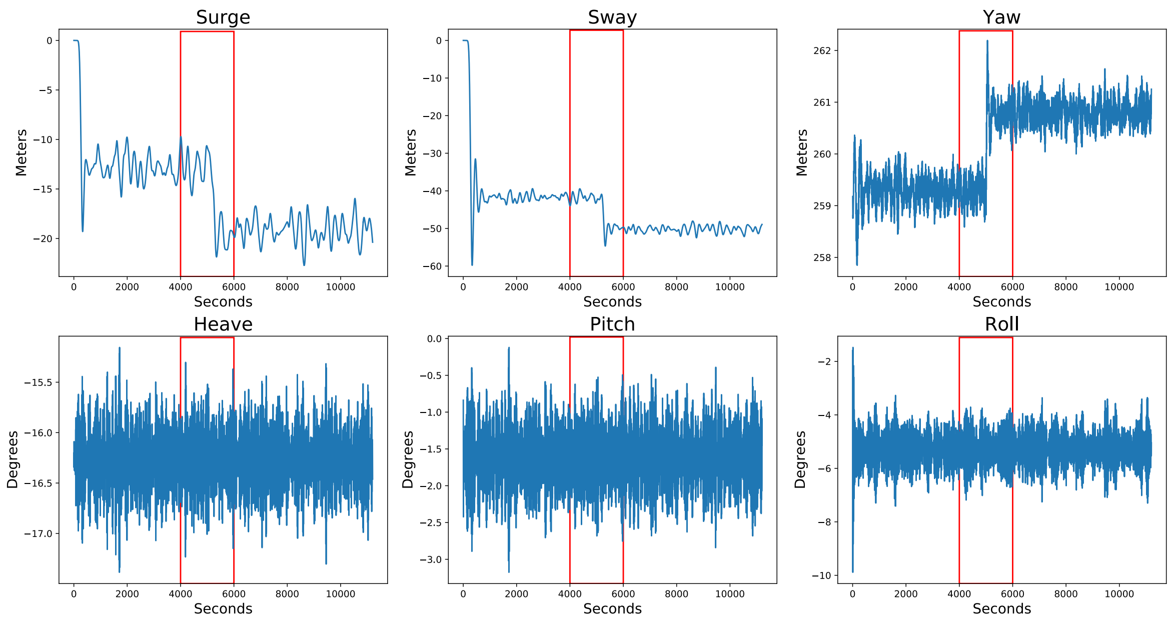


Figure 5: The six degrees of freedom (6DoF) of platform motion. The red rectangle highlights how the motion changes before time 4000 seconds and after line breakage at time 5000 seconds happens.

1.2 Objectives

The main objective of this work is to develop a system that can detect mooring line failure in real time for offshore platforms that utilizes mooring system for station-keeping. In particular, we aim to detect breakage of the mooring lines of FPSO P50 as early as possible using Machine Learning (ML) models to make predictions of the platform, monitoring the difference between predicted and measured motion to detect mooring line failure for multiple line groups with different drafts. The interest is to use only the data provided by the platform's DGPS and IMU sensors and its estimated draft. We also aim for the system to not be limited to only FPSO P50 but other types of offshore platforms.

For example, on 09/24/2019, a Petrobras P50 Platform had three moorings broken[1]. The production of the FPSO, of around 20,000 barrels of oil and 500,000 cubic meters of natural gas daily, was immediately interrupted by Petrobras, and a state of alert was issued. There were 178 people working on board the unit and there was the possibility of evacuating the vessel. Fortunately, the platform remained stable and secure, anchored by another 15 moorings and a temporary holding vessel. The repair could be carried out quickly, however, the stoppage of production caused a great loss. Petrobras has about 40

[1]https://mobile.clickpetroleoegas.com.br/amarras-se-rompem-e-petrobras-paralisa-a-producao-da-p50-na-bacia-de-campos/

proprietary platforms installed, which encompass around 590 mooring lines. The need for early mooring line breakage detection is critical.

Thus, the following research questions were formulated:

RQ1 Can a break in the mooring system of an offshore platform be detected using only the data regarding the motion and draft of this platform?

RQ2 Is a machine learning system based on Multilayer Perceptron (MLP) neural networks capable of predicting the future motions of a moored platform having as information only the previous motion and the platform draft?

RQ3 Would a classifier be able to detect which mooring group a broken line belongs to based solely on the discrepancy between the motion predicted by the MLP and that measured on the platform?

In this thesis we seek to answer these questions. This work aims to develop a system for identification of failures of mooring lines on platforms because it is not only of high impact in the offshore engineering industry, but also because it is complex enough to justify the use of advanced ML techniques.

1.3 Contribution

The concept of *Digital Twins* for stationary oil and gas production units offers many opportunities to increase oil production and reduce risks in the operation of offshore platforms. With digital twins, a virtual replica of a physical asset, such as an FPSO platform, can be created and modeled. The behavior of the platform in real-time can be simulated using data from sensors, equipment, and other sources. By using digital twins, valuable insights such as the platform's performance can be obtained, including identifying potential equipment failures or maintenance needs. By analyzing the data from the digital twin.

However, the availability of a large amount of historical data in real-time creates many challenges. Within this context, ML techniques can significantly contribute to solving problems of prediction, analysis, planning, operation, and maintenance of oil platforms. The advantage of ML methods over traditional techniques is the scalability for complex problems and large amounts of data because instead of programming the solutions, the idea is to create programs that “learn” the representation of the solutions from the data available. Thus, the benefits of this work for the oil, natural gas, and energy sector will be numerous, such as the reduction of operational bottlenecks and risks in decision making due to the effective use of available data, both historical and online. Other impacts include the reduction of subjectivity in decision-making .i.e., deciding if line breakage has occurred, better data trace-ability, and more effective planning for the various operations of the platforms.

Finally, this work provides a contribution to the field of computer engineering by demonstrating how techniques and methodologies from this field can be utilized for the purpose of detecting failures in mooring lines. Specifically, a neural network-based system was developed to detect the occurrence of mooring line failure, with the system being designed in sub-modules to enable the individual fine-tuning of each module with minimal downtime to the entire system.

During the development of this thesis, the following articles were published:

1. Saad, A. M., Schopp, F., Barreira, R. A., Santos, I. H., Tannuri, E. A., Gomi, E. S., & Costa, A. H. R. (2021). Using Neural Network Approaches to Detect Mooring Line Failure. IEEE Access, 9, 27678-27695. DOI: 10.1109/ACCESS.2021.3058592

2. Saad, A. M., Schopp, F., Queiroz Filho, A. N., Cunha, R. D. S., Santos, I. H., Barreira, R. A., Tannuri, E. A., Gomi, E. S., & Costa, A. H. R. (2021, June). FPSO Mooring Line Failure Detection Based on Predicted Motion. In International Conference on Offshore Mechanics and Arctic Engineering (Vol. 85116, p. V001T01A002). American Society of Mechanical Engineers. DIO:10.1115/OMAE2021-62413

3. Suller, T. M., Gomes, E. O., Oliveira, H. B., Cotrim, L. P., Sa'ad, A. M., Santos, I. H., Barreira, R. A., Tannuri, E. A., Gomi, E. S., & Costa, A. H. R (2021, November). Evaluation of Neural Architecture Search Approaches for Offshore Platform Offset Prediction. In Anais do XVIII Encontro Nacional de Inteligência Artificial e Computacional (pp. 326-337). SBC. DOI:10.5753/ENIAC.2021.18264

These published articles present the incremental steps taken in the refinement of our proposed system for detecting mooring line failures. In Saad et al. (2021a), the development and comparison of two Machine Learning (ML) networks: *a feed-forward network* (Multilayer Perceptron (MLP)) and *a recurrent network* (Long Short-Term Memory (LSTM)) were presented. These networks were configured to detect mooring line failure, based on the comparison between measured and predicted motion for platform motion with a single draft setting of 16m.

In Saad et al. (2021b), we focused on improving the MLP network developed in Saad et al. (2021a) by refining the dataset used for training and testing the network and introducing a binary decision tree classifier, to further confirm the occurrence of mooring line failure. The Multilayer Perceptron (MLP) network developed in Saad et al. (2021a) had difficulties in predicting platform motions under stormy environmental conditions due to the small amount of data in this situation. The methodology used in Saad et al. (2021b) was adopted in this article where in the MLP network implemented was then trained to estimate the platform's future motion based on its motion's temporal data without failure, the difference between the predicted and the measured motions was then used as inputs to the binary decision tree classifier to classify whether or not there is a failure in the mooring system.

Finally in Suller et al. (2021), a comparative analysis of three techniques for optimizing neural networks hyperparameters was performed, Bayesian optimization (Bayesian Optimization (BO)), Random Search (Random Search (RS)) and Simulated Annealing (Simualted Annealing (SA). This optimization process is called Neural Architecture Search (Neural Architecture Search (NAS)), a sub-field of AutoML, which focuses on automating the ML architecture design, reducing the need for manual architecture design by human expert. Lessons learnt from each article provided useful insights, which have been incorporated in the development of the final version of the proposed system.

1.4 Organization of this thesis

The remainder of this thesis is organized as follows. We present the fundamentals to methods and algorithms used in this thesis in chapter 2, *Background*, which covers the foundation for understanding our work. In chapter 3, *Related Work*, literature review and discussion about the articles found is presented. In chapter 4, *Neural Motion Estimator (NeMo)*, we introduce and describe our proposal. In chapter 5, *Experiments and Results*, we explained how data was generated for training and testing of our proposed neural network model. We also present the architecture of neural network model implemented and results of the implemented system in this chapter. In chapter 6, *Conclusion and Future Work*, we provide a conclusion of the developed system, and we close this thesis pointing to future works.

2 BACKGROUND

This chapter provides a summary of the basic concepts needed to understand the problem of interest and presents the methods that will be explored in the development of the proposal. We start by defining what time series predictions entails in section 2.1, given this is the type of data used in this work. We then describe how platform motions subject to different environmental conditions are generated and modelled using Dynasim, a time-domain platform motion simulator in section 2.2. Next, in section 2.3 we provide an introductory explanation of the concept of machine learning (ML), with a focus on neural networks, in particular, Multi-layer Perceptron (MLP). Then, we provide a brief introduction on hyper-parameter optimization techniques explored in this thesis, and we conclude this chapter with an introductory explanation of the classifiers used, as well as the metrics used to evaluate classifiers.

2.1 Time Series Prediction

Time series data is a sequence of numerical measurements $\langle x_1, ..., x_t \rangle$, recorded at equal space time interval over time t. This kind of data can be obtained from various source, such as stock exchange market, seismometer readings, and electrocardiogram (ECG) measurements. There are two types of time series data, discrete and continuous. Discrete time series data are obtained when data are recorded at fixed time intervals, while continuous time series are measurements recorded continuously for a time period. Furthermore, time series data can be either univariate or multivariate. A univariate time series is a series with a single time-dependent observation, while a multivariate time series has more than one time-dependent observations (BROCKWELL; DAVIS, 2002).

Time series prediction involves developing a model that uses previously recorded data points to predict future points. There are different approaches used for time series prediction such as one-step forecast and multi-step forecast. The number of previous data points used is known as *input window or sliding window*, and the number of data points predicted is known as the *horizon or forecast window* (w_f). In one-step forecast, a model M uses $k+1$ previous data points as the input window to make one data point prediction x_{t+1},

$$x_{t+1} = M(x_t, x_{t-1}, ..., x_{t-k}). \tag{2.1}$$

In multi-step forecasting, the forecast window w_f of the model M has f data points,

$$(x_{t+f}, ..., x_{t+1}) = M(x_t, ..., x_{t-k}). \tag{2.2}$$

An example of multi-step prediction is shown in figure 6, where the input window comprises of 600 previous seconds and the forecast window (w_f) is 100 seconds. In this work, multi-step forecasting of continuous time series is employed.

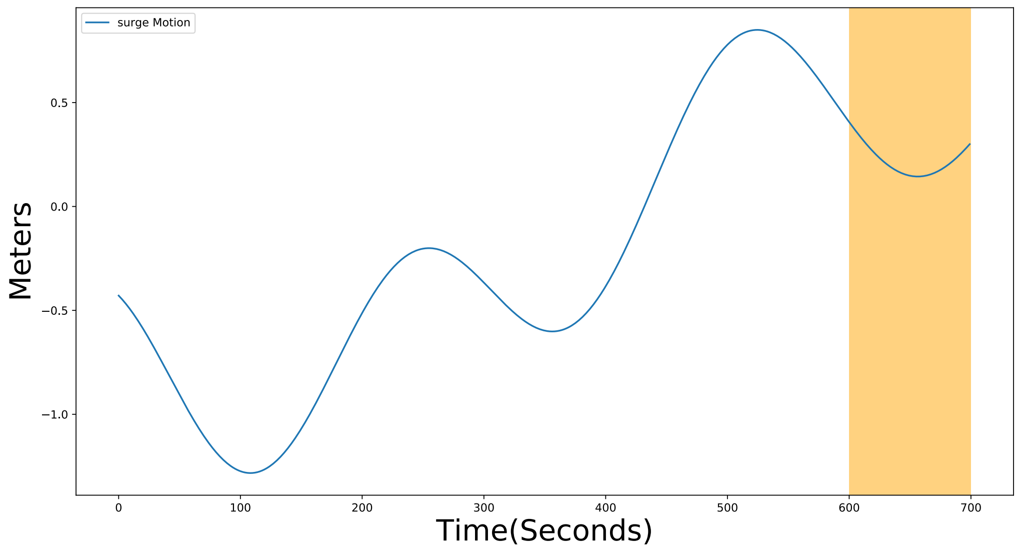


Figure 6: An illustration of multi-step prediction of a FPSO-P50 surge motion. The implemented model uses an input window of 600 seconds to predict 100 seconds, demarcated in the yellow background. Each data point refers to 1 second.

However, predicting large forecast window negatively impacts the accuracy of the model. Therefore, finding the optimal width of both the sliding window and forecast window is important and is investigated in this work.

Time series can be stationary or non-stationary. In the stationary case, the parameters of the distribution remain constant over time, i.e., all observations are sampled from the same distribution, which does not depend on the temporal dimension. On the other hand, non-stationary series have a more complex temporal dynamics, where the distribution from which we take the samples evolves with time (BISHOP, 2006). Dealing with non-stationary data is very difficult because one must have information about how the distribution changes over time. In this thesis we will consider that the motion series data are stationary.

The most popular ML models were not created specifically for time series forecasting. The great strength of ML models comes from having access to several samples of the same phenomenon. A large number of samples are needed so that the ML model can identify, by itself, characteristics intrinsic to the data. If the phenomenon is observed over a long period of time and there is good quality data derived from these observations, ML models become very attractive to be used in prediction.

2.2 Dynasim simulator

Dynasim is a time-domain numerical simulator created by the collaboration of Universidade de São Paulo[1] and Petrobras[2] and others (NISHIMOTO; FUCATU; MASETTI, 2002). We used the Dynasim simulator to simulate the motion of floating vessels and platforms subjected to forces provided by the environment, mooring lines and propellers.

Figure 7 shows the Dynasim interface. The simulator allows for analyzes of the dynamic behavior of moored platforms subjected to wind, wave, current and swell plus platform design specification to produce time-series data of the platforms' 6DoF motion – surge, sway, heave, pitch, roll and, yaw.

In this work, real environmental conditions were collected from a weather station located in Campos Basin (Bacia de Campos) of Rio de Janeiro, Brazil from 2003 to 2006. These environmental conditions were recorded at an interval of 3-hours, making $18,000$ distinct environmental conditions and it is used as input to the simulator. Table 1 shows the features present in the dataset of environmental conditions.

The Dynasim model of the P50 platform with specifications presented in Table 2 is used to generate motion data through the dynamic simulation of the vessel for the recorded environmental conditions (wave, swell, wind and current) and draft conditions. The simulations were carried out for the P50 platform subjected to different environmental conditions and different drafts, from 8m (ballast condition) to 21m (fully loaded condition) – with increments of 1m.

It is also possible to configure the simulator so that there is a rupture in a specific

[1]⟨http://tpn.usp.br/⟩
[2]⟨https://petrobras.com.br/en/⟩

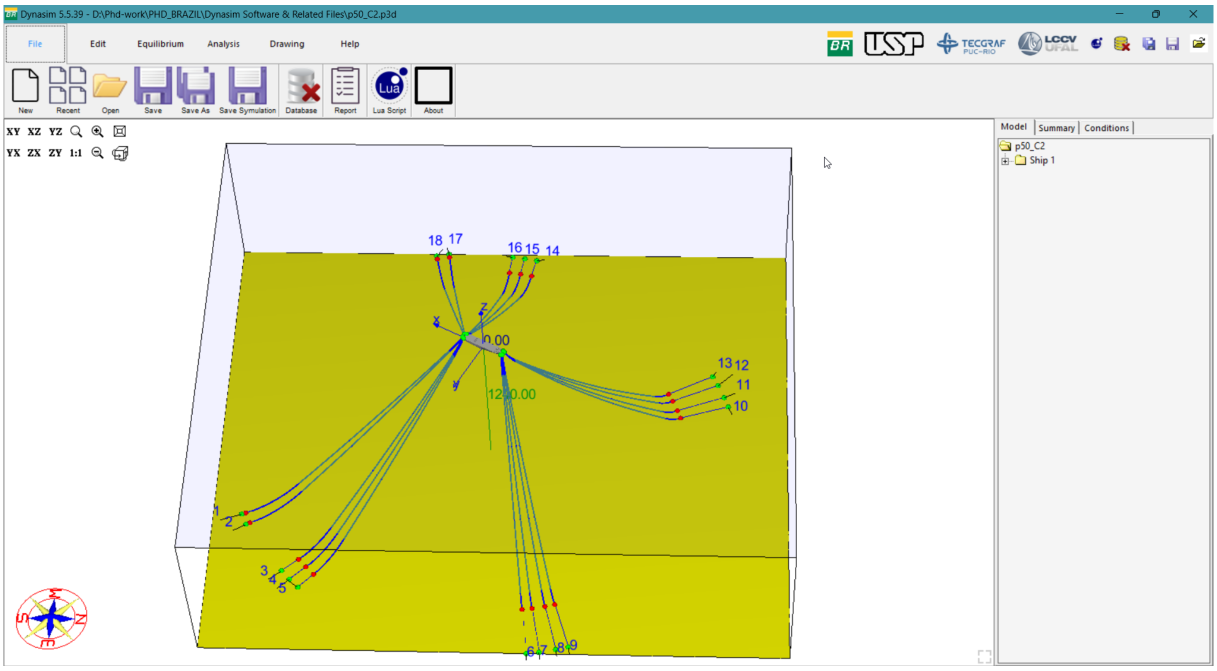


Figure 7: Dynasim Interface.

line of the mooring system, at a specified time. The output of the simulator consists of the 6DoF motion of the platform for each environmental condition simulated, with and without line failure, for all the 18 mooring lines. Figure 8 presents an example of the output of the surge motion time series generated by the simulator. The time series at the beginning, from 0 seconds to 3600 seconds in the figure, is composed of transient motion, which do not represent the actual motion of the platform. These occur because at time 0 seconds the simulator has not yet applied the effects of the environmental conditions, and after applying these effects, it takes some time to calibrate the platform's motion response. In this thesis, experiments were conducted with simulated platform motion time series excluding transient motions .

2.3 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 (DOMINGOS, 2012). ML has been applied in various domains such as for fraud detection (CHADEGANI et al., 2013), medical diagnostic (LUNDERVOLD; LUNDERVOLD, 2019) and in the automotive sector it is used for autonomous driving (AL-QIZWINI et al., 2017). ML can be divided into three categories: Supervised Learning (SL), Unsupervised Learning (UL) and, Reinforcement Learning (RL). Supervised Learning (SL) is

Table 1: Environmental measurements dataset features

Feature	Name	Unit
Date	data	DD/MM/YY
Hour	hora	HH:MM
Wave Height	hs1	Meters [m]
Wave Period	tp1	Seconds [s]
Wave Direction	dir1	Degrees [º]
Swell Height	hs2	Meters [m]
Swell Period	tp2	Seconds [s]
Swell Direction	dir2	Degrees [º]
Wave Total Height	hstotal	Meters [m]
Wind Speed	vento_vel	Meters per Second [m/s]
Wind Direction	vento_dir	Degrees [º]
Current Speed	corr_vel	Meters per Second [m/s]
Current Direction	corr_dir	Degrees [º]

Table 2: Platform dimension and configuration.

Variable	**Hull Dimensions**
Beam (m)	54.5
Depth (m)	27.8
Number of risers	78
Number of mooring lines	18
Length overall (m)	337.4
Length between prep. (m)	320.0
Draft(ballast condition (m))	8.0
Draft(fully loaded condition (m))	21.0

explained because we will employ it in this thesis while more on UL and RL can be found at (JAMES et al., 2013; DOMINGOS, 2012; SUTTON; BARTO, 2018).

Supervised Learning (SL) involves training an algorithm to be able to generate a mapping function that can predict output for a given input. The algorithm trains on labeled data until it achieves a reasonable level of accuracy in making accurate predictions. 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. Both categories will be used in this thesis. We will use an SL regressor model to predict the future motion of the platform, and an SL classifier model to classify whether there was a break and which group of lines the break occurred, based on the difference between the movement predicted by the regressor and the movement measured.

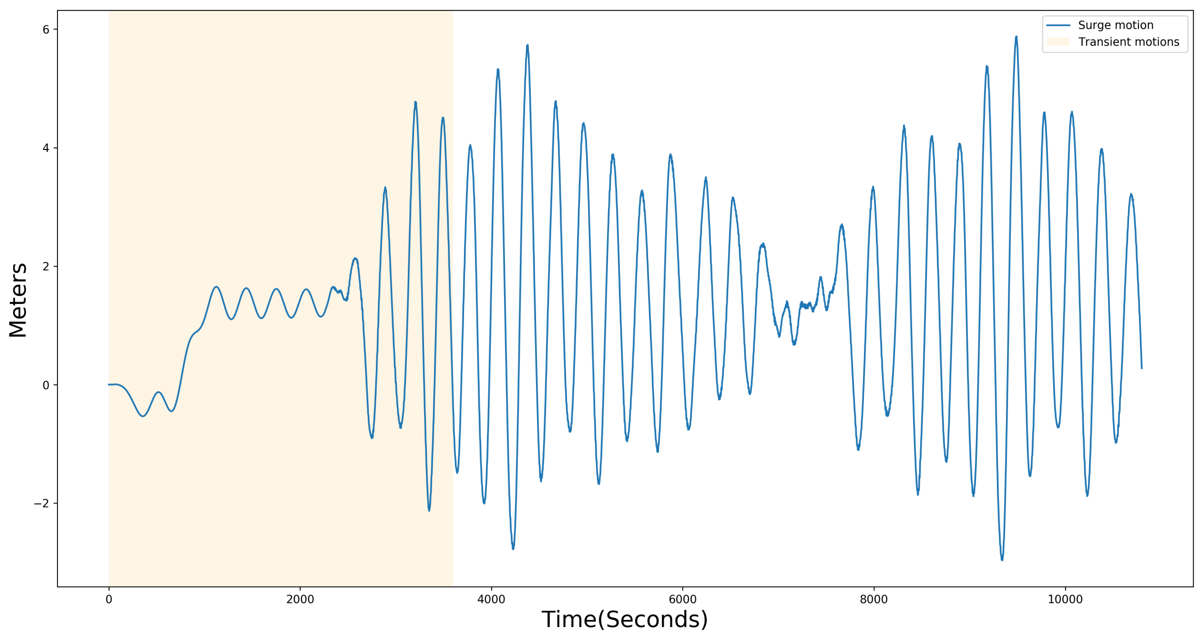


Figure 8: An example of the surge motion time series output generated by the Dynasim simulator. The first 3600 seconds (in yellow) represents the range dropped due to simulation transients.

2.3.1 Neural Network Principles

Many of the successful ML algorithms today are based on Neural Network (NN)s implemented in different architectures. NNs are able to change their structure based on incoming internal or external information, which allows them to learn patterns in data. NNs are composed of models of neurons called perceptrons.

A single unit of perceptron takes in inputs $\mathbf{x}$, given by $\langle x_1, x_2, x_3, .., x_n \rangle$ where each input (i) is connected to a node (j) with weights $\mathbf{w}$, $\langle w_1, w_2, w_3, .., w_n \rangle$ assigned to each input respectively. The node (j) generates a weighted sum of all the inputs values plus bias:

$$m_j = \left(\sum_{i=1}^{n} w_i.x_i + b \right), \tag{2.3}$$

where 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 h to generate an output $\hat{y}_j$,

$$\hat{y}_j = h(m_j). \tag{2.4}$$

The activation function h can be referred to as a gate which allows information to flow through or not. There are different types of activation functions available. The Rectified Linear Unit (ReLU) is a non-linear activation function commonly used to handle complex data due to its simplicity and efficiency. ReLU whose formula is $h(z) = max(0, z)$, allows values from the weighted sum (here $z = m_j$) greater than 0 to flow through and all values less than 0 are clipped at 0 and propagated forward as output $\hat{y}$ (GLOROT; BORDES; BENGIO, 2011). Figure 9 presents a unit of perceptron.

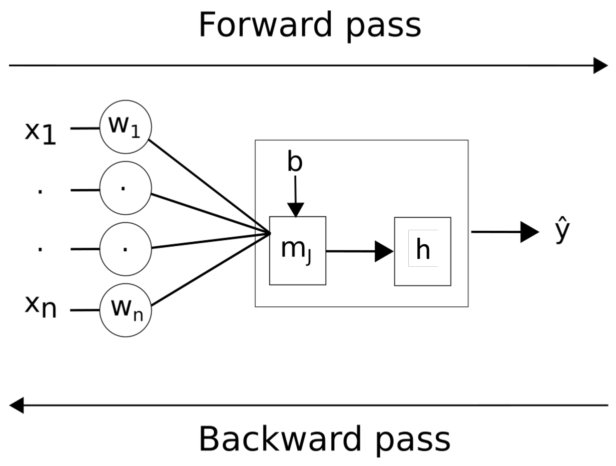


Figure 9: A single unit of a perceptron, where $\hat{y}$ is the output, h is the activation function, m_j is the weighted sum, b is the bias, x is the input, and w is the weight.

The perceptron learns by using the Back-Propagation (BP) algorithm which finds the weights $\mathbf{w}$ combination that minimizes a **loss function**, that is the error difference between the perceptron output $\hat{y}$ and the actual value y, i.e., the label for a given input in the labeled dataset. BP works by iterating between three steps. The first is a **forward pass** step which involves the flow of information from input to the activation function to produce an output (Figure 9). Then, there is a loss function calculation step, where the output of the perceptron $\hat{y}$ is compared against the actual value y, given this is a supervised learning were the actual output (or label) is known before hand in a training set consisting of pairs $\langle \mathbf{x}, y \rangle$. A common performance metric used to gauge the difference between the model prediction and the actual value is the **Mean Squared Error (MSE)**, $\mathsf{MSE} = (\hat{y} - y)^2$.

Finally, the **backward pass** step occurs and the error difference calculated is back propagated to adjust the weights of the perceptron starting from the output layer back to the input layer.

These steps are done iteratively until a stopping criteria is met, which can be either allowing the network to iterate n number of **epochs** or stopping the training 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.

2.3.2 Multi-Layer Perceptron

A **Multilayer Perceptron (MLP)** is a feed forward neural network (*FFNN*) in which information propagates uni-directionally. These type of networks do not posses the ability to use a feedback loop in which the outputs of the networks is cycled back into itself. An MLP is created when multiple perceptrons are structured in layers to solve complex problems. These layers are the input layer, one or more hidden layers, and an output layer. Following the same principle of a unit of perceptron, information flows from the input layer through the hidden layer(s) after which it passes to the output layer to make a prediction. The BP algorithm is used to train MLPs. Figure 10 presents a 2 x 2 x 2 MLP with one input layer, 1 hidden layer and an output layer.

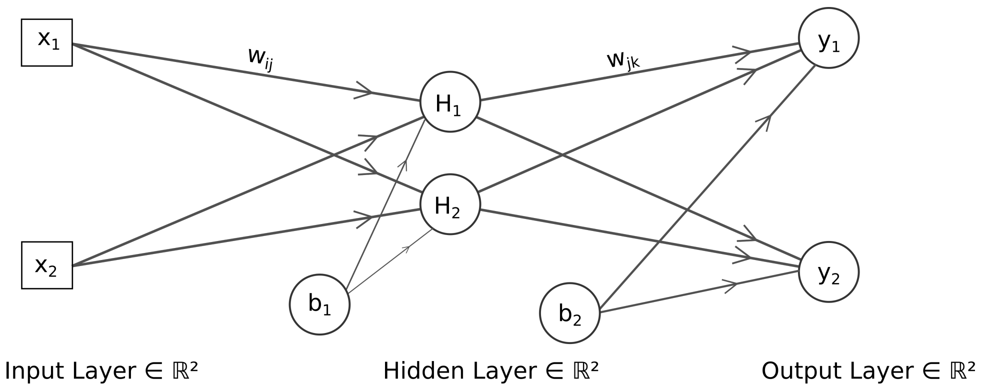


Figure 10: A Multi-layer Perceptron with two inputs, x_1 and x_2, one hidden layer with two perceptrons, H_1 and H_2, and bias b_1, and two outputs, y_1 and y_2 and bias b_2.

Formally, consider an input vector $\mathbf{x} \in \mathbb{R}^N$ and a single-hidden-layer MLP $f_{\mathbf{w}}(\mathbf{x})$, parameterized by weights $\mathbf{w} = \{\mathbf{w^{(1)}}, \mathbf{w^{(2)}}\}$, where $\mathbf{w^{(1)}} \in \mathbb{R}^{M \times N}$ are the weights associated with connections from input layer to hidden layer, and $\mathbf{w^{(2)}} \in \mathbb{R}^{K \times M}$ are the weights associated with connections from hidden layer to the output vector $\mathbf{y} \in \mathbb{R}^K$. In this thesis the MLP model is used to make time series predictions and for this a regression model is used. For a regression MLP with activation function h_1 in the hidden layer and h_2 in the output layer, the network's k-th output, $\hat{y}_k$, is given by:

$$\hat{y}_k = f_k(\mathbf{x}|\mathbf{w}) = h_2\Big(w_{k0}^{(2)} + \sum_{j=1}^{M} w_{kj}^{(2)} h_1\Big(w_{j0}^{(1)} + \sum_{i=1}^{N} w_{ji}^{(1)} x_i\Big)\Big), \tag{2.5}$$

where the weights $\mathbf{w^{(1)}} = \{w_{ji}^{(1)}\}$ and $\mathbf{w^{(2)}} = \{w_{kj}^{(2)}\}$ are shown in scalar form in eq. 2.5 in order to explicitly illustrate the calculation of output $\hat{y}_k$.

As said, given a dataset $\mathbb{D} = \{\mathbf{x^{(i)}}, \mathbf{y^{(i)}}\}_{i=1}^{N_D}$ comprised of N_D samples and with known inputs $\mathbf{x^{(i)}} \in \mathbb{R}^N$ and outputs $\mathbf{y^{(i)}} \in \mathbb{R}^K$, the process of training an MLP consists in finding the weights $\mathbf{w}$ that minimize a loss function L, typically the Mean Squared Error (MSE) given by:

$$L(\mathbf{w}) = \frac{1}{N_D} \sum_{i=1}^{N_D} \Big[\frac{1}{K} \sum_{k=1}^{K} \big(\hat{y}_k^{(i)} - f_k(\mathbf{x}^{(\mathbf{i})}|\mathbf{w}) \big)^2 \Big], \tag{2.6}$$

where K is the number of output neurons and f_k is given by Eq.(2.5). This can be done by performing gradient descent on the loss function and updating the weights in the direction that minimize it: $\mathbf{w} \leftarrow \mathbf{w} - \alpha \nabla L(\mathbf{w})$, where α is the learning rate. The gradient of the loss function with respect to the weights $\nabla L(\mathbf{w})$ is found by back-propagation of the error through the network.

2.3.3 Normalizing Inputs for an NN

It is important that the network inputs have the same ranges of values. To avoid using angles in the input quantities, it is important to project the angle of the quantities into their x and y components so that the values have no numerical discontinuity. The x-component is calculated by $x = v \cdot \cos(\phi)$, and the y-component by $y = v \cdot \sin(\phi)$, where v is the magnitude value and ϕ is its direction.

Since different inputs come with different value ranges it is important to normalize the value ranges. The data are scaled so that their values are in the range between 0 and 1. If no scaling is done different characteristics gains more influence than others. Therefore scaling distributes not only the values to a normalized range but also get rids of possible offsets. Here in this Thesis, for each variable, the minimum and maximum values are calculated and the data are then standardized by:

$$X_{scaled} = \frac{X - X_{min}}{X_{max} - X_{min}}, \tag{2.7}$$

in which X_{min} and X_{max} are the minimum and maximum values of the original variable X, and X_{scaled} is the scaled result.

Standardization consists of subtracting an amount relative to a measure from a measure on the scale, with z-score being one of the most used standardization score:

$$Z = \frac{X - \mu}{\sigma} \tag{2.8}$$

in which Z is the standardized result and X is the input, whose mean is μ and standard deviation is σ. The z-score transforms the original data to obtain a new distribution with mean 0 and standard deviation 1 from the mean.

2.3.4 Hyper-parameter Optimization

Building a neural network involves finding the right set of hyper-parameters that will lead to the best error loss for the given data. 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 an ML 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. These hyper-parameters are:

Finding the right number of neurons to use for the network. Having too many neurons in a layer leads the network to over-fitting and having a small number of neurons leads to under-fitting.

Finding the right number of layers to use. Layer composition can be either shallow or deep. Shallow network means a small number of layers while deep network means many hidden layers. A shallow network may or may not be adequate to get the best error loss and a deep network can lead the network to over-fitting and the gradient vanishing problem.

Choosing the right activation function and optimizer. Currently, the Adam optimizer is used in the ML community, but others may prove interesting for some problems (KINGMA; BA, 2017). Activation functions are used to determine how information should be propagated to the next layer, and different types may be more suitable in some situations.

Choosing the right learning rate (α). The learning rate is dependent on the type of problem being solved by the NN.

Choosing the right batch size and number of epochs. Batch size is the number of training samples to work through before the model's internal parameters are updated. Number of epochs is the number of times the entire training dataset is passed to the network during training.

Methods used to find the right set of hyper-parameter values are usually based on a trial and error approach where different models' performances on a validation dataset are compared in order to determine appropriate hyper-parameters, as they are highly dependent on the problem being addressed. Novel methods have been developed to automate model hyper-parameter optimization, thus greatly reducing reliance on the manual, painstaking human experimentation required to produce high quality models (FEUERER; HUTTER, 2019). These new approaches constitute the field of Automated Machine Learning (AutoML).

A subset of AutoML, known as **Neural Architecture Search (NAS)**, has gained more attention recently. In this field, automatic engineering of neural network architectures are studied. NNs with architectures obtained via NAS have achieved better results when compared to NNs whose network architecture was manually configured for problems such as regression or classification (MÄRTENS; IZZO, 2019; ZOPH et al., 2017).

Neural Architecture Search

NAS methods are defined by three dimension: a **Search space** $\mathcal{S}_{\mathcal{S}}$, which bounds the possible architectures evaluated during the NAS process; a **Search strategy**, which determines how the optimization algorithm explores the search space; and a **Performance estimation strategy**, which defines an *objective function* C used to evaluate the model performance during the search process (ELSKEN; METZEN; HUTTER, 2019).

The search space and performance estimation strategy are manually defined with the intent of obtaining the best optimization performance for the shortest execution times, and of ensuring models generated by this process are able to perform adequately on data which was neither used during architecture search nor during model training.

The search strategy is generally defined by iterative algorithms that follow the same general structure: they depend on a *search history* $\mathcal{H}$, which is an initially empty data structure that stores models architectures $\mathcal{M}$ and their performance, obtained by evaluating $C(\mathcal{M})$. The search process consists of a predetermined number n of iterations, also called *trials*, whose high-level overview is presented in Algorithm 1.

Various iterative search strategies can be employed in NAS, two of which were studied: Random Search and Bayesian Optimization. An overview of each algorithm follows.

Random Search. Random Search (RS) is implemented by sampling architectures (models $\mathcal{M}$) randomly from $\mathcal{S}_{\mathcal{S}}$, aiming to find the architecture which most closely reaches the goal of the optimization process, as described in Algorithm 2. RS is most commonly used as baseline for comparison with other methods, yet has been shown to achieve performance similar to that of state-of-the-art NAS algorithms on some specific problems (YU et al., 2019).

```
Algorithm 1: NAS Trial
  Input: search space $\mathcal{S_S}$, objective function $C$, initial architecture $\mathcal{M}_0$, number of
         iterations $n$
  Output: search history $\mathcal{H}$
  $C(\mathcal{M}_0) \leftarrow$ Evaluate($\mathcal{M}_0$);
  $\mathcal{H}_0 \leftarrow \{\langle \mathcal{M}_0, C(\mathcal{M}_0)\rangle\}$;
  for $i = 1, 2, \ldots, n$ do
      $\mathcal{M}_i \leftarrow$ Search on $\mathcal{S_S}$ from $\mathcal{M}_{i-1}$;
      Train $\mathcal{M}_i$;
      $C(\mathcal{M}_i) \leftarrow$ Evaluate($\mathcal{M}_i$);
      $\mathcal{H}_i \leftarrow \mathcal{H}_{i-1} \cup \{\langle \mathcal{M}_i, C(\mathcal{M}_i)\rangle\}$;
  end
```

```
Algorithm 2: Random Search
  Input: search space $\mathcal{S_S}$, objective function $C$, initial architecture $\mathcal{M}_0$, number of
         iterations $n$
  Output: search history $\mathcal{H}$
  $C(\mathcal{M}_0) \leftarrow$ Evaluate($\mathcal{M}_0$);
  $\mathcal{H}_0 \leftarrow \{\langle \mathcal{M}_0, C(\mathcal{M}_0)\rangle\}$;
  for $i = 1, 2, \ldots, n$ do
      $\mathcal{M}_i \leftarrow$ random($\mathcal{S_S}$) ;
      Train $\mathcal{M}_i$;
      $C(\mathcal{M}_i) \leftarrow$ Evaluate($\mathcal{M}_i$);
      $\mathcal{H}_i \leftarrow \mathcal{H}_{i-1} \cup \{\langle \mathcal{M}_i, C(\mathcal{M}_i)\rangle\}$;
  end
```

Bayesian Optimization. Bayesian Optimization (BO) consists of two key components: a probabilistic surrogate model S of the objective function C; and a policy P, denoted as *acquisition function*, for selecting new architectures based on the surrogate model. In each trial, an evaluation of C updates the surrogate model S, allowing P to select a new architecture $\mathcal{M}$ most likely to achieve the objective of the optimization for the next trial. The general procedure is shown in Algorithm 3.

The surrogate model is used as an estimate of the objective function C, and can be generated in a number of ways. In this work, Tree-structured Parzen Estimator (TPE) was employed, as it is able to scale to bigger search spaces with a smaller computational cost and has been shown to achieve better performance than other methods often employed such as Gaussian Process regression (FRAZIER, 2018). There are different choices available for the acquisition function, the most common being Expected Improvement (EI). EI is an iterative process were models with the best estimated performance are selected sequentially. EI is calculated using the surrogate model, and the architecture with the largest EI is selected for evaluation in the next trial. An in-depth of Bayesian

Algorithm 3: Bayesian Optimization

```
Input: search space $\mathcal{S_S}$, objective function $C$, initial architecture $\mathcal{M}_0$, surrogate
       model $S$, acquisition function $P$, number of iterations $n$
Output: search history $\mathcal{H}$
$C(\mathcal{M}_0) \leftarrow$ Evaluate($\mathcal{M}_0$);
$\mathcal{H}_0 \leftarrow \{\langle \mathcal{M}_0, C(\mathcal{M}_0) \rangle\}$;
Initialize the surrogate model $S$;
for $i = 1, 2, \ldots, n$ do
    $\mathcal{M}_i \leftarrow \arg\max_{\mathcal{M}' \in \mathcal{S_S}} P(\mathcal{M}', S)$;
    Train $\mathcal{M}_i$;
    $C(\mathcal{M}_i) \leftarrow$ Evaluate($\mathcal{M}_i$);
    $\mathcal{H}_i \leftarrow \mathcal{H}_{i-1} \cup \{\{\mathcal{M}_i, C(\mathcal{M}_i)\}\}$;
    $S \leftarrow$ updateSurrogate($S$, $\{\mathcal{M}_i, C(\mathcal{M}_i)\}\}$);
end
```

Optimization can be found in (FRAZIER, 2018; BERGSTRA et al., 2011).

2.3.5 Classifiers

In ML, classification is the problem of identifying the class label to which a new input observation belongs. A brief introduction to the different supervised classifiers used are presented.

K-nearest Neighbour (KNN) classifier

K-nearest Neighbour (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 all known samples. Based on the k nearest neighbours it then classifies the unknown label into the label that the majority of the neighbours have.

The KNN classifier is an easy-to-use classifier. There are only two KNN parameters: (1) the number of K nearest neighbours the classifier takes into account when classifying an unknown data sample; and (2) the distance metric to use. Here, Euclidean distance metric is used to calculate the distance,

$$Euclidean\ distance = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}. \tag{2.9}$$

Decision Tree (DT) classifier

Decision Tree (DT) is another type of algorithm used in supervised classification of categorical or continuous data. The DT algorithm has a tree like structure consisting of nodes, branches and leafs. An example of DT is illustrated in figure 11. The nodes of a decision tree represent features or attributes of a dataset, the branches represent decision rules made at the node and each 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 (MITCHELL, 1997).

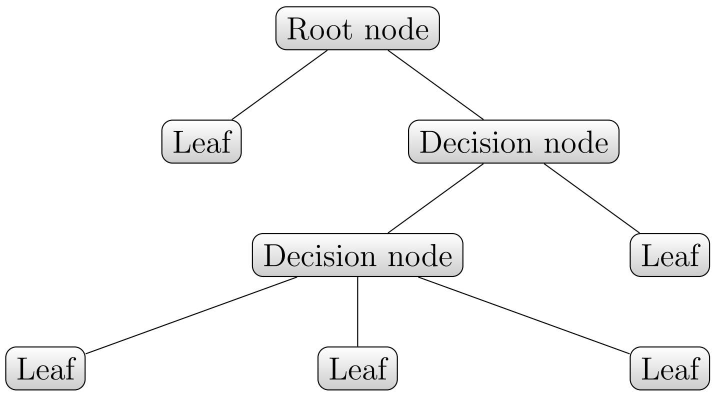


Figure 11: Decision Tree structure.

When building a tree the algorithm aims at splitting a 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 the data set is split into two groups.

This process of binary questioning is then recursively carried out until a subset where all data points present have the same values or labels is reached – this subset then becomes a leaf node. To build a tree, splitting decisions are made in the decision node. In order to find the best attribute or feature on which a splitting decision for dividing the data set is made, different approaches can be used. Different selection measures, such as entropy-based information gain, are used to find the best attribute (MITCHELL, 1997).

The 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, given by:

$$GI(D) = 1 - \sum_{1=i}^{c} P(k_i|D)^2 \tag{2.10}$$

and it measures the degree of probability of a data point being wrongly assigned to a group when it is randomly 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, for equally represented classes with probability $P(k_i|D) = \frac{1}{c}$, the GI then has the value $\frac{c-1}{c}$. The goal is to have a low GI value, which means that the data points are correctly classified (ZAKI; JR, 2019).

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 data set and it is the core algorithm most decision tree are built using. The Classification and Regression Tree algorithm (CART) uses GI as attribute selection metric for building a tree(LOH, 2011).

Support Vector Classifier (SVC)

A 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 the two-dimensional space, this hyper-plane is a line that separates each class, where each class can be on one side.

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

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

and 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, SVC algorithm finds the support vector for each hyper-plane. Support vectors are the closest observations to a hyper-plane from both classes. SVC 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 12 presents a binary linear SVC , 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 (JAMES et al., 2013).

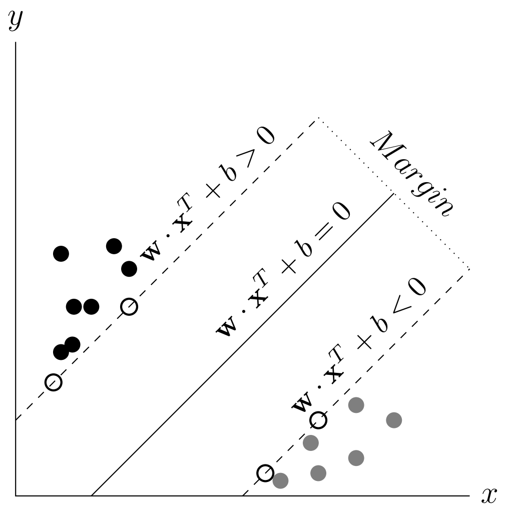

y
x
$\mathbf{w} \cdot \mathbf{x}^T + b > 0$
$\mathbf{w} \cdot \mathbf{x}^T + b = 0$
$\mathbf{w} \cdot \mathbf{x}^T + b < 0$
Margin

Figure 12: Binary linear Support Vector Classifier. 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.

Multinomial Logistic Regression (MLR)

An MLR classifier is an extension that generalizes Logistic Regression (LR) to multi-class problems – more than three or more class. LR are models that have a certain fixed number of parameters that depend on the number of input features, and they output categorical prediction, for example if an email is a spam or not, this means that our data has two kinds of observations (class 1 and class 2 observations). The LR model fits an S shape curve, called Sigmoid function,

$$\sigma(x) = \frac{1}{1 - e^{-x}} \tag{2.11}$$

to our observations. The sigmoid function (σ) squashes the weighted sum of values of the input features x into the range between 0 and 1, which in turn gives the probability of an observation falling into one of the two classes. Observations with values closer to 1 are predicted to belong to class 1 and observations with values close to 0 belong to class 2 (MOUNT; ZUMEL, 2019).

The MLR classifier works by creating multiple LR models on K number of classes using the one-vs-all approach. In this approach, the classifier trains a single classifier per class, with the samples of that class as positive sample and all other samples as negatives. The assumption is that there are K independent classification problems, meaning K classes, and for each class we learn a logistic (probability) model. The key assumption is that each of these problems is independent of the other $K-1$ logistic regression problems. Therefore, for each sample we either classify this as class Y_i or not. This is repeated for all classes.

2.3.6 Evaluating classifier performance

A **confusion matrix** is a table used to access the performance of a classification model on a set of test dataset in which the true value (i.e., the actual class or group or label) is 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 represent the instances of the predicted class. Figure 3 presents a binary classification confusion matrix. The confusion matrix is not limited to binary class classification – it is also used for multi-class classification. Here binary class classification is shown (ZAKI; JR, 2019).

		Actual Group		
		True (t_1)	False (t_2)	Total
Predicted Group	True (t_1)	$True\ Positive\ (TP)$	$False\ Positive\ (FP)$	$TP + FP$
	False (t_2)	$False\ Negative\ (FN)$	$True\ Negative\ (TN)$	$FN + TN$
	Total	$TP + FN$	$FP + TN$	N

Table 3: Confusion Matrix for two classes.

The confusion matrix puts data into 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,

$$TP = |\{\mathbf{x}_i|\hat{y}_i = y_i = t_1\}|.$$

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

$$TN = |\{\mathbf{x}_i|\hat{y}_i = y_i = t_2\}|.$$

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

$$FP = |\{\mathbf{x}_i|\hat{y}_i = t_1 \ and \ y_i = t_2\}.$$

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

$$FN = |\{\mathbf{x}_i|\hat{y}_i = t_2 \ and \ y_i = t_1\}|.$$

By combining these groups, accuracy, precision or recall of a classifier can be measured. The accuracy metric in ML can be misleading when used alone. 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 issue, 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:

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

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:

$$Precision = \frac{TP}{TP \ + FP}.$$

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:

$$Recall = \frac{TP}{TP \ + FN}.$$

F1 Score: is a weighted combination of Recall and Precision:

$$F1\ score = 2 \times \frac{Precision \times Recall}{Precision + Recall} = \frac{2TP}{2TP + FP + FN}.$$

3 RELATED WORK

Researchers in diverse domains are increasingly adopting ML techniques, and this is also true for the offshore industry. We present different approaches adopted in the literature toward detection of mooring line breakage using use neural networks as their primary ML technique.

A summary of the articles published in the last 12 years retrieved focusing on FPSO with query

> “(machine AND learning OR neural AND network OR artificial) AND (vessel OR platform OR fpso OR floating production storage and offloading) AND (mooring OR line AND breakage OR Failure OR damage AND detection)”

on the title, abstract and keywords fields of Scopus[1], Google Scholar[2], and Onepetro[3] database are presented in Table 4. The table details the paper year, inputs used, type of ML network and application target, which is the method of mooring line monitoring carried out, for each paper, respectively.

Mooring system monitoring using NNs in the offshore industry are implemented for various application target within the context of failure detection such as monitoring and predicting mooring line tension (GUMLEY; HENRY; POTTS, 2016), mooring line fatigue prediction (SIDARTA et al., 2017), complete mooring line breakage detection (BD) (SIDARTA; O’SULLIVAN; LIM, 2018), mooring line damage location prediction (DLP) where segments of a mooring line, i.e., top, middle or bottom location, are predicted (LEE et al., 2021) and vessel response movements (VRM) monitoring, where differences between the 6DoF movements of a platform readings before and after breakage are monitored.

From the list of articles cited, it can be seen there is no standard set of ML network model, input variables and application target used for detecting mooring line failure. This

[1]⟨https://www.scopus.com/⟩
[2]⟨https://scholar.google.com/⟩
[3]⟨https://onepetro.org/⟩

Table 4: Summary of ML models

Paper (year)	**ML Model**	**Model Input**	**Model Output**	**Target**	Multi Draft Detection	No. of Mooring Lines
(MAZAHERI et al., 2003)	MLP	Wave height and direction, wind velocity and direction, and current velocity and direction	The surge, sway and total FPSO's offset	VRM	No	Not provided
(GUMLEY; HENRY; POTTS, 2016)	MLP	Wave height, period, direction, wind velocity, current velocity and vessel draft	The mean offset of a vessel, maximum offset and, significant offset	VRM	No	1
(JAMALKIA; ETTEFAGH; MOJTAHEDI, 2016)	MLP	Global Positioning System (GPS) & FPSO draft	Classify mooring line status of 3 mooring lines, i.e., damaged or intact	BD	No	3
(SIDARTA et al., 2017)	MLP	6DoF movements of the FPSO	Predict mooring tension of four mooring lines of the FPSO	BD	No	4
(PRISLIN; MAROJU, 2017)	MLP	Meta-ocean measurements & 6DoF movements	Classification-based output, indicating whether or not there was a line break, and regression-based output, estimating which mooring line is broken from nine mooring line	BD & DLP	No	12
(SIDARTA; O'SULLIVAN; LIM, 2018)	MLP	Global Positioning System (GPS) & FPSO draft	Classify mooring line status of 3 mooring lines, i.e., damaged or intact	BD	No	3
(SIRÉTA; ZHANG, 2018)	MLP & LSTM	Surge, yaw, pitch, roll, heave movements of the FPSO	Sway movement	BD	No	21
(JAISWAL; RUSKIN, 2019)	CNN	Statistics of the horizontal position parameters of the vessel and the root mean square values of the 6DoF accelerations	Classify mooring line status, i.e., damaged or intact	BD	No	21
(WANG et al., 2020)	RBF	Offset of the surge and heave movements, pitch movements of the FPSO & mooring line tension readings at the fairleads	Damage severity of mooring lines	DLP	No	12
(CHUNG et al., 2020)	MLP	Mean and standard deviation of meta-ocean measurements & 6DoF of the FPSO	Predict location where mooring line damage occurred, i.e., top, middle or bottom location	DLP	No	8
(QIAO et al., 2021)	LSTM	6DoF movements of the FPSO	Predict top tension and dip of mooring lines	DLP	No	9
(LEE et al., 2021)	MLP	6DoF movements & wind, wave, and current measurements	Predict location where mooring line damage occurred for 12 mooring lines, i.e., top, middle or bottom location	DLP	No	12
Our proposal	MLP	6DoF movements of the P50-FPSO	Classify mooring line status, i.e., damaged or intact	**BD**	**Yes**	**18**

Note: Where meta-ocean measurements are mentioned we refer to wave height, peak wave period, and direction, swell height, swell peak period, and direction, wind velocity, direction, current velocity and, direction. 6Dof refers to surge, sway, yaw, pitch, heave, and roll movement of the platform obtained from sensors. Vessel Response Monitoring (VRM), Breakage Detection (BD) and Damage Location Prediction (DLP).

is because ML models require different inputs variables and architecture design depending on the target application and they are often obtained via trial and error or hyperparameter optimization (see section 2.3.4). This is illustrated by different input variables and ML models used by the different authors. However, common attributes can be seen depending on the type of detection considered, and one can also observe the prevalence of the MLP model used in many solutions.

For models where **vessel response monitoring** is the target, a combination of meta-ocean measurements of the wave, wind and current variables plus vessel specifications, i.e., draft or vessel dimension, are used as input to their respective model. Here in an abstract level, the aim is to train a neural network to learn the response of a platform to a set of incident meta-ocean measurements, to predict vessel response such as offset and

6DoF movements, where the difference between the predicted variables and the true variable measurements are used to indicate the possibility of mooring system failure. Works by Mazaheri et al. (2003) and Gumley, Henry and Potts (2016) showed the successful application of ML models for vessel response monitoring. In the former, a 1 hidden layer MLP network was used to predict the surge, sway and total offset of an FPSO, using 6 meta-ocean measurements, wave height and direction, wind velocity and direction, and current velocity and direction as input to the MLP network. The authors highlighted the MLP developed could calculate the responses of an FPSO for 1000 sets of meta-ocean measurements in less than 1 minute, whereas when using hydrodynamic modelling simulator such as SAMRES, an in-house simulator, it takes up to 80 days to model on the same computer. This is beneficial to mooring system design as engineers will be able to model different mooring line setups for different meta-ocean conditions. In the latter (GUMLEY; HENRY; POTTS, 2016), wave height, period and direction, wind velocity, current velocity, and vessel draft were used as input to a 1 hidden layer MLP network to output the mean offset, maximum offset, and significant offset of a vessel.

For models where breakage detection or mooring line damage location prediction is the target application, neural networks are trained to learn the difference between movements without and with breakage. Most networks trained use a combination of 6DoF readings, vessel specification and meta-ocean measurements as input to their models, unlike ship monitoring where meta-ocean measurements and ship specification are the main data used as input.

In the mooring line **breakage detection** approach, the FPSO's 6DoF movements subject to meta-ocean conditions such as calm, mild, and stormy are monitored. (SIDARTA et al., 2017; JAISWAL; RUSKIN, 2019; KWON et al., 2020; SIDARTA et al., 2021; LEE et al., 2021) all used a combination of the 6Dof movements and meta-ocean measurements as input to their system. For example, Sidarta, O'Sullivan and Lim (2018) trained two types of ML models: a feed forward network (MLP) and a recurrent network (Long Short Term-Memory – LSTM), to predict the sway movements of an FPSO using previous movements of surge, heave, roll, pitch, and yaw movement of a simulated FPSO. Both models were able to forecast the sway movements of the FPSO and also detect mooring line breakage by showing a strong change in the direction of sway movements after breakage. Prislin and Maroju (2017) proposed a novel concept for mooring system integrity monitoring by training an MLP, which they referred to as Position Response Learning System (PRLS). Inputs to the system were the 6DoF movements of a simulated vessel, and the system provided two forms of outputs: classification-based output, indicating whether or not there was a line break, and regression-based output, estimating which mooring line is broken from nine mooring line groups. Sidarta et al. (2017) showed that using just Global Positioning System (GPS) readings from an FPSO and the draft of the FPSO as input to an MLP with two hidden layers was enough for the network to predict the status of 20 mooring lines, and the network accurately detected that 3 of the 20 mooring lines were broken, in different sea states.

Another approach used in mooring system integrity monitoring is **damage location detection**, that is early identification of segments of individual mooring lines where damage is suspected to have occurred i.e. top, middle or bottom. In this approach, the focus is on monitoring the mooring line stiffness, not the complete mooring line breakage, as the reduction in stiffness can be attributed to early signs of failure and, based on which line and section of the mooring line reduction occurs, the vessel's response changes accordingly. Successful implementation of this approach can be seen in the works of (CHUNG et al., 2020; JAMALKIA; ETTEFAGH; MOJTAHEDI, 2016; QIAO et al., 2021; LEE et al., 2021; KWON et al., 2020; UDDIN et al., 2012; WANG et al., 2020). Chung et al. (2020) in their paper used an MLP model with 5 hidden layers to detect mooring line damage and indicate which part of the mooring line is damaged, i.e., top, middle or bottom. Inputs to the MLP were the mean and standard deviation of wind and wave environmental measurements and the floater's 6DoF responses. The output of the network consisted of binary classification of the mooring line status, that is, compromised or not compromised and it also indicated which part of the mooring line was damaged. The developed MLP was trained and tested on simulated data generated by CHARM3D software from Texas A&M University. During training the MLP was trained on data without a compromised mooring system and tested with data containing damaged mooring lines. Random noise was added when data was being generated in order to replicate real-life weather conditions. The MLP model developed was able to detect when a mooring line damage occurred.

In conclusion, mooring system monitoring using Artificial Neural Network (ANN)'s as shown can be conducted using different approaches, however, given we are interested in detecting complete mooring line breakage and not detecting location where breakage along the mooring line has occurred and, we only have FPSO-P50 6DoF movements data, we are thus limited to adopting the breakage detection approach.

The work by Siréta and Zhang (2018) in which they developed two ML models; an MLP and a LSTM model was found to closely resemble the problem we intend to solve, therefore, we drew inspiration from their work, by combining the auto correlation and cross correlation approach they implemented of the 6DoF movements of their platform where only RMS error was used to identify the occurrence of mooring line breakage. We on the other hand, use two more errors, making three errors; RMSE, mean and median to complement and affirm the occurrence of mooring line failure.

In their implementation for both the cross correlation and auto correlation approaches the LSTM model used a time lag of 1 second of input to the network to predict the subsequent second while the time lag window for the MLP model for both approaches had two windows 10 and 100 respectively, it was found using a longer window for the MLP decreases it performance overall.

Our proposed network in comparison has a long prediction window. Their works also focused on identifying mooring line breakage of a platform with a single draft and also in all the articles cited none attempted mooring failure identification for platforms with multiple drafts, while our developed network identifies mooring line breakage of a platform with multiple platform drafts.

In the literature thus far creation of training and test data of the ML models in all the papers mentioned were done using hydrodynamic simulators such CHARM3D software from Texas A&M University (CHUNG et al., 2020), AQWA and OrcaFlex4 numerical simulator software (QIAO et al., 2021) among others, and we also used Dynasim simulator to generate the 6DoF motion of our platform. Note so far in the literature space, no real life platform motion data have been used to develop ML models.

Our main contribution is that our developed ML network uses only the 6DoF movements of the FPSO-P50 platform as input, to indicate the occurrence or not of failure in the mooring line as output, for different draft cases in the FPSO. Our proposal is detailed in the next chapter.

4 NEURAL MOTION ESTIMATOR (NEMO)

This chapter details our proposal for implementing a solution towards mooring failure detection using Machine learning algorithms. We split this chapter into sections representing the different sub-modules of the proposed system in detail, describing the incremental steps carried out for developing each sub-module.

Section 4.1, provides the general overview of our proposed system named Neural Motion Estimator (NeMo). Section 4.2 presents the evolution of the first sub-module named predictor module of the NEMO system, starting from the implementation and evaluation of two ML models and, ending in section 4.2.1, where we chose the final ML model to represent the predictor module. Next, in section 4.2.2, we implemented Neural Architecture Search (NAS) methods to investigate and optimize the network architecture composition of developed predictor ML network, followed by an ablation study evaluating the minimum number of variables of the 6DoF motions required to develop a network capable of detecting mooring line failure in section 4.2.3. Section 4.3, describes the second module of the NEMO system, named Error Calculator module, which is responsible for producing different error metrics used to determine the condition of a platform motion. Section 4.4 details the evolution of the final submodule of the NEMO system, named Classifier Module, from the implementation and evaluation of different binary classifier algorithms and ends with the steps necessary to extend the classifier module to handle multiclass classification.

4.1 General Architecture of our Proposed System

Our proposed ML based system, named **Neural Motion Estimator (NeMo)**, has as a specified prerequisite that its inputs use only information derived from the time series measurements of the Differential Global Positioning System (DGPS) and Inertial Measurement Unit (IMU) sensors, and from the estimation of the platform draft level to detect mooring line failure. The platform model used is FPSO P50 with 14 different draft levels (draft level $\in \{8, 9, \ldots, 21\}$ meters), whose dimensions are described in Table 2 (see Chapter 2). The P50 platform has a mooring system with 18 lines, arranged in 4 groups, as illustrated in Figure 13 and described in Table 5.

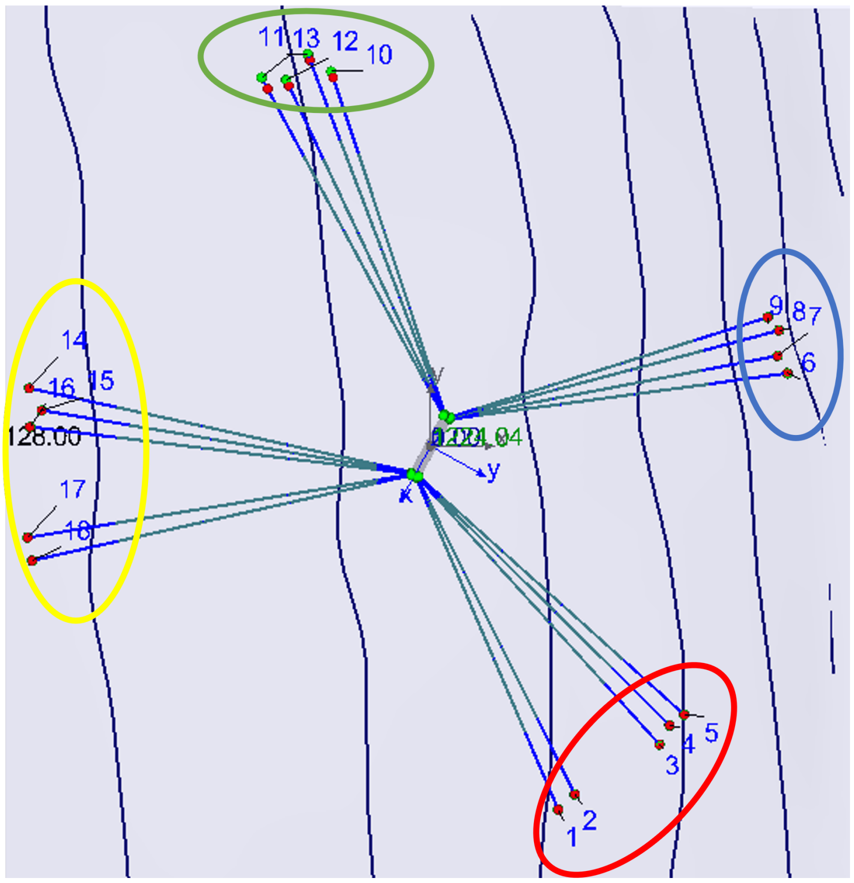


Figure 13: P50 Line groups illustrated in Dynasim interface. Group 1 (red), group 2 (blue), group 3 (green) and group 4 (yellow). The green lines with blue tips are the mooring lines anchoring the P50 platform and the platform is positioned in the middle.

Table 5: FPSO-P50 Mooring line groups.

Mooring Line Groups	
Group 1	L1, L2, L3, L4, L5
Group 2	L6, L7, L8, L9
Group 3	L10, L11, L12, L13
Group 4	L14, L15, L16, L17, L18

NeMo's general architecture is composed of three modules that work together: a predictor module, an error calculator, and a classifier model, as illustrated in Figure 14.

A combination of Differential Global Positioning System (DGPS) and IMU measurements (surge, sway, heave, roll, pitch, and yaw), making the 6DoF of the platform motion, is used as inputs to the **predictor module** whose outputs are used as input to the **error**

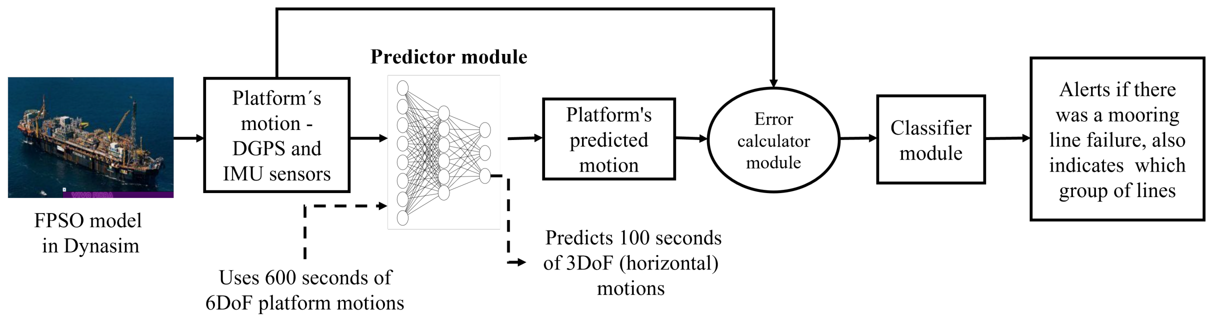

Predictor module
FPSO model in Dynasim
Platform´s motion - DGPS and IMU sensors
Uses 600 seconds of 6DoF platform motions
Predicts 100 seconds of 3DoF (horizontal) motions
Platform's predicted motion
Error calculator module
Classifier module
Alerts if there was a mooring line failure, also indicates which group of lines

Figure 14: Proposed general architecture: a predictor module predicts the motion of a platform without failure based on its previous motion; the difference between the predicted value and the one measured by the sensors is then classified by a classifier module, indicating whether there was a failure or not.

calculator module which compares the error difference between the predictor modules' prediction and the motions of the platform and stores the scores which are then used as inputs to the **classifier module** of NeMo system to provide status of the mooring lines of the platform.

A hypothesis is made that, depending on the status of the mooring line, a significant difference between the predictor modules' prediction and the value measured by the DGPS and IMU platform sensor will exits, signifying a change in the platform line status.

Our proposed solution breaks the mooring line detection problem into smaller components, allowing for independent solutions and fine tuning of the sub-modules to be possible. In the following sections the three modules of the NeMo system are described.

4.2 Predictor Module

In this section, we explain the incremental steps conducted in developing the predictor module of the NeMo system.

We initially developed and compared the efficacy of two machine learning models: a *feed-forward network*–(Multilayer Perceptron (MLP)) and a *recurrent neural network* specifically a Long Short-Term Memory (LSTM) network, to access whether machine learning model were capable of mooring line failure detection (SAAD et al., 2021a). These networks were configured to use the previous 6DoF motions of FPSO-P50 platform to forecast the motion of the platform with a single draft.

The difference between the networks' prediction and the motions of the platform are then used to determine the status of the mooring system, hypothesising the difference between the predictions of the network and the actual motions of the platform can be used as an indicator of the mooring system being compromised.

Figure 15 presents the first iteration of the general architecture of the mooring line failure detection system. These two ML networks used function differently, as the MLP

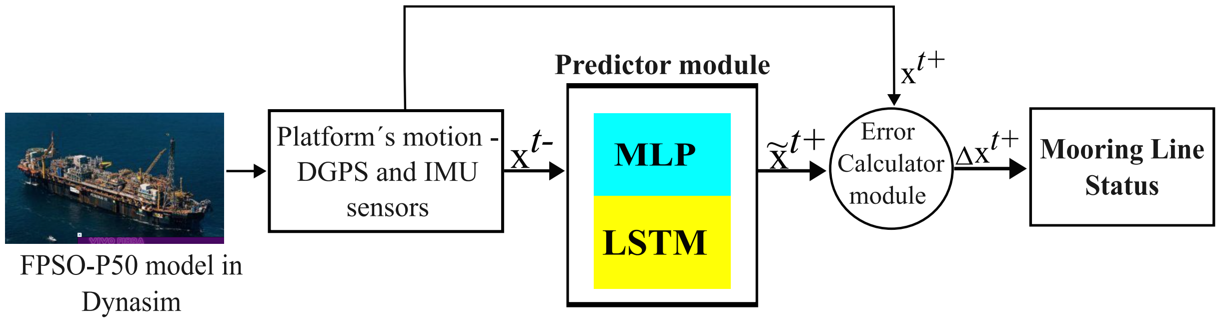


Figure 15: In a first step, the MLP and LSTM models were evaluated and compared.

network is a feed forward model that processes information unidirectional and previously seen information is not retained within the network; on the other hand, the LSTM network has the ability to retain previously seen information and uses it to adjust the weights and biases of the network thus making it a suitable network for time series data prediction.

Different architectures composition, input window length, forecast window length, number of the 6DoF platform motion variables to use as input and forecast for both the MLP and LSTM network were evaluated and the best architecture composition found is presented here.

For the MLP network, the architecture consisting of an input layer which receives 600 seconds of the 6DoF motions of the platform as input, making 3600 data points, with 3 fully connected hidden layers consisting of 7200, 3600 and 1800 neurons respectively in each layer and an output layer that predicts 100 seconds of 3DoF– surge, sway and yaw motions was the best architecture found and it is illustrated in Figure 16.

For the LSTM network, an encoder-decoder network with 200 LSTM units at the encoder and decoder layers with a repeat vector layer between the encoder and decoder layers and two fully connected feed forward layers was the best LSTM architecture found. The LSTM network used as input 1000 previous seconds of the 3DoF motions and it predicted 400 seconds of the same 3DoF motions. Figure 17 illustrates the architecture of the LSTM network .

Data Generation: To training and test the two ML networks, 18,000 real environmental conditions measured in the Campos Basin, in Rio de Janeiro, Brazil, were collected plus

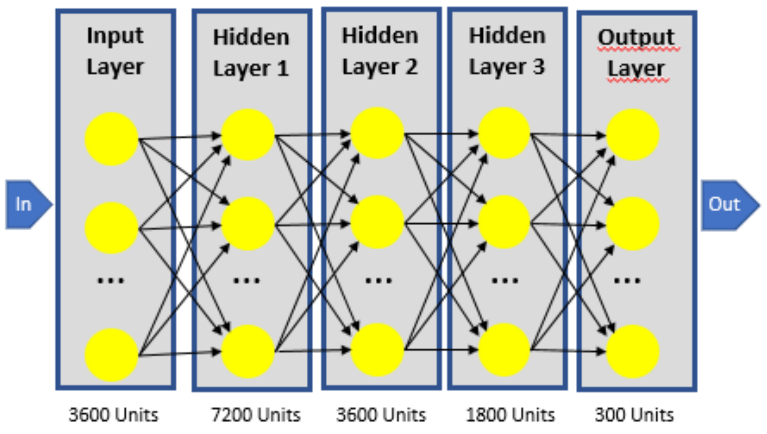


Figure 16: MLP Architecture

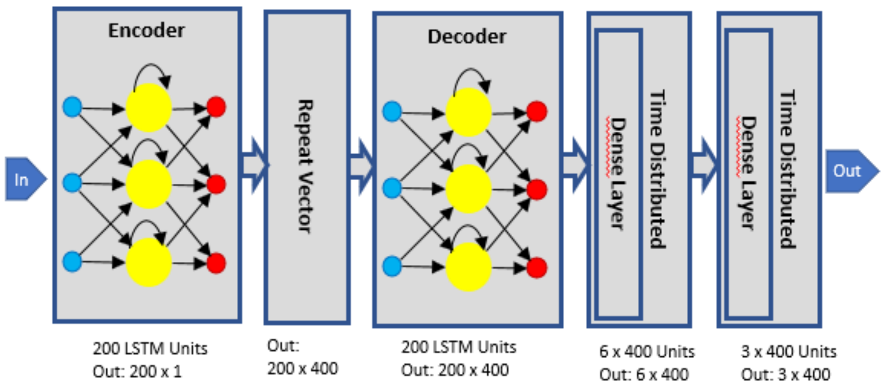


Figure 17: LSTM encoder-decoder architecture

FPSO-P50 specifications with a single draft load of 16 meters was used as input to Dynasim simulator to generate the 6DoF motions of FPSO-P50, for motions comprising of all the 18 mooring lines anchored to the platform intact and motions with a single mooring line broken. Using the simulator, each environmental condition measured was simulated for 3 hours, for all the combination i.e., motions with broken and intact mooring lines.

The general data creation pipeline is described in the next chapter.

From these simulated movements, three distinct subsets were used for training, validating and testing both networks. Table 6 details parameters used. The two networks were trained on platform motion without mooring line breakage and they are used to predict the motions of platform without and with mooring line breakage.

Results: To compare and access the prediction accuracy of these networks, they were evaluated on several platform motions with diverse environmental conditions. For illustration purpose, two simulated platform motions with different environmental conditions, which

Table 6: Training Data Composition for the MLP and LSTM Network

Network	Training data No.	Validation data No.	Test data No.	Training epochs No.
MLP	5000 intact motions	1000 intact motions	Remaining platform motions	3000
LSTM	1000 intact motions	200 intact motions	Remaining platform motions	1500

we classified to be mild and stormy based on the wave height, are presented in Table 7. These results were used to evaluate the prediction accuracy of both networks under the same environmental conditions.

Prediction under intact mooring line conditions: The prediction performance of the MLP and LSTM networks were evaluated under two different conditions whose respective conditions are shown in Table 7.

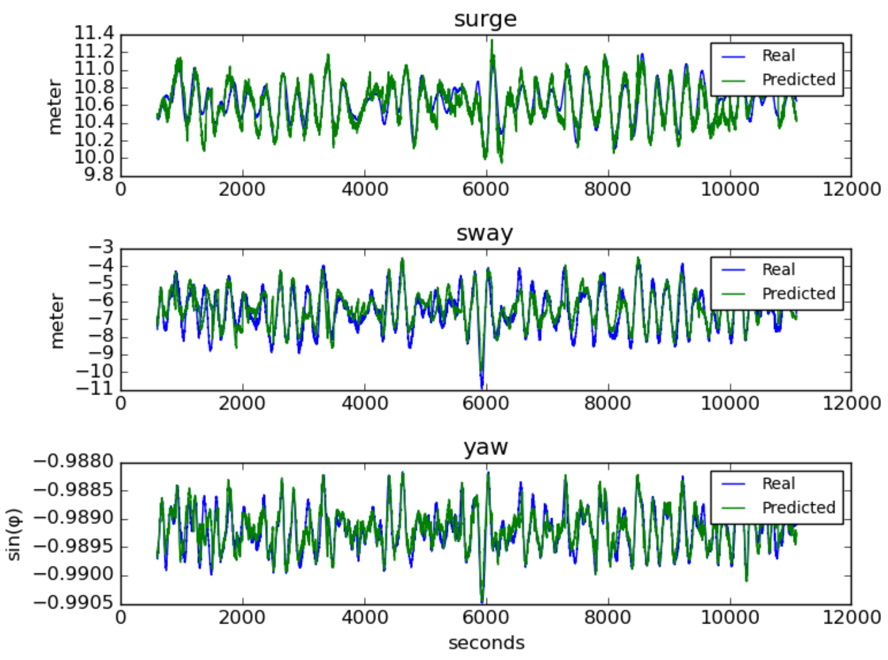
Table 7: Two environmental conditions selected

Sea Condition	Index	hs1 (meters)	tp1 (seconds)	dir1 (deg)	hs2 (meters)	tp2 (seconds)	dir2 (deg)	wind_vel (m/s)	wind_dir (deg)	current_vel (m/s)	current_dir (deg)
Mild	600	1.68	8.48	47.2	1.33	5.4	191.9	8.21	195.3	0.2	288.84
Stormy	8818	2.84	17.04	192.2	0.0	0.0	0.0	7.77	208.6	0.45	216.93

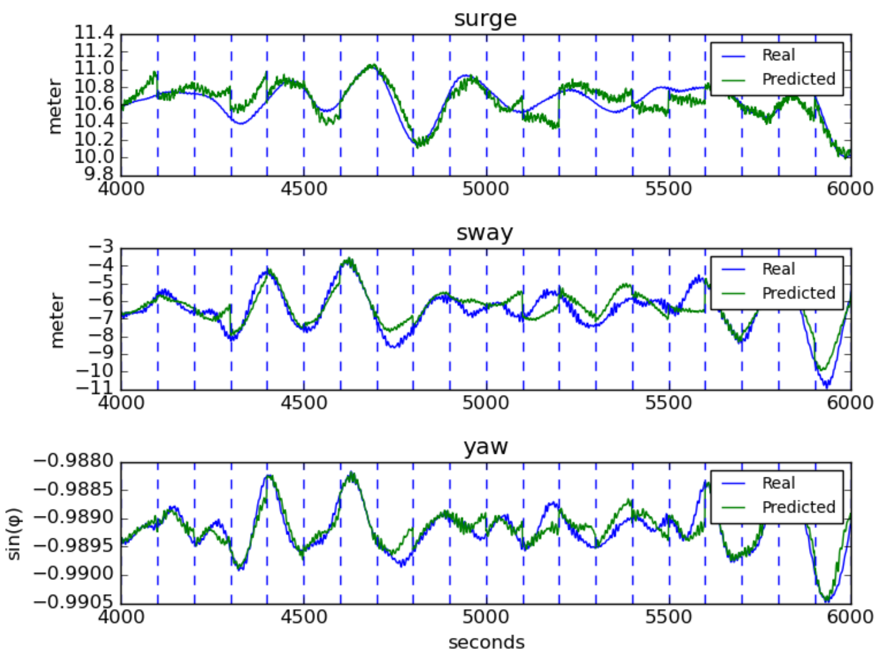
We started by accessing the performance of these networks on mild and stormy environmental conditions where the mooring system of the platform is intact. Figure 18 shows the MLPs' prediction for platform motions without mooring line failure for scenario with mild environmental conditions, and Figure 19 shows the prediction for stormy environmental conditions. Figures 20 and 21 show the LSTMs' prediction for platform motions without mooring line failure for scenarios with mild and stormy environmental conditions, respectively.

Prediction under compromised mooring line conditions: The prediction performance of the two ML networks for platform motion with mild environmental conditions with a broken mooring line are presented in Figures 22 and 23 for the MLP and LSTM network, respectively.

Discussion: As can be seen in Figure 18 and 19, the MLPs' prediction for the two scenarios, mild and stormy are presented, showing the network could predict the oscillation of the simulated platform in mild environmental conditions (see Figure 18a), but for stormy environmental condition, it found it difficult to predict the rapid motions of the platform as shown in Figure 19a while in Figure 20 and 21, the LSTMs' prediction for the same two scenarios, mild and stormy are presented, showing the network could also predict the oscillation of the simulated platform in mild environmental conditions (see Figure 20a)

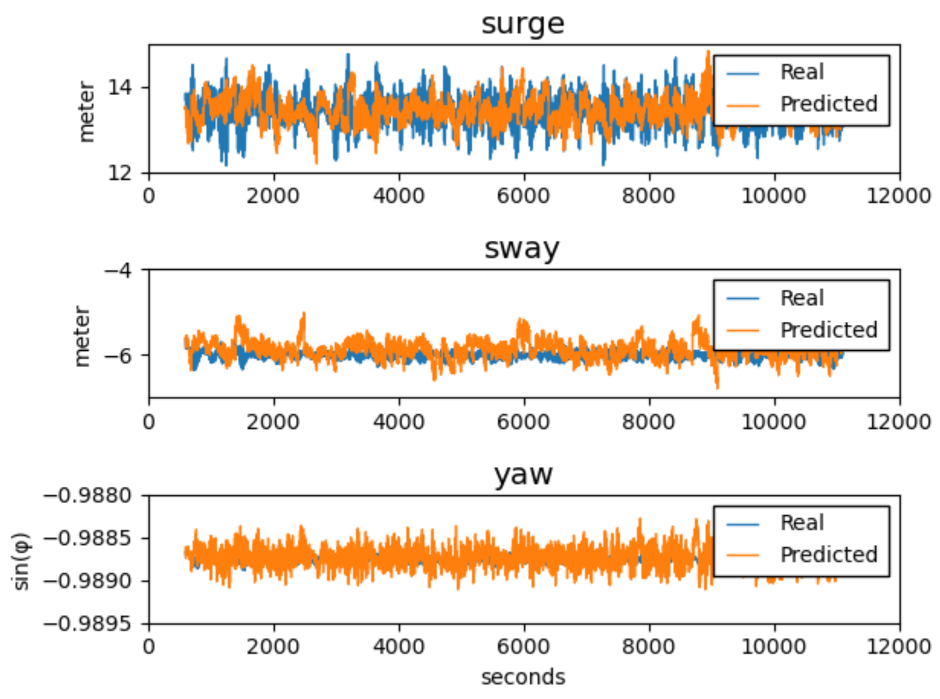


(a) Illustration of the MLP predictor of a platform motion under a mild scenario with all mooring lines intact.

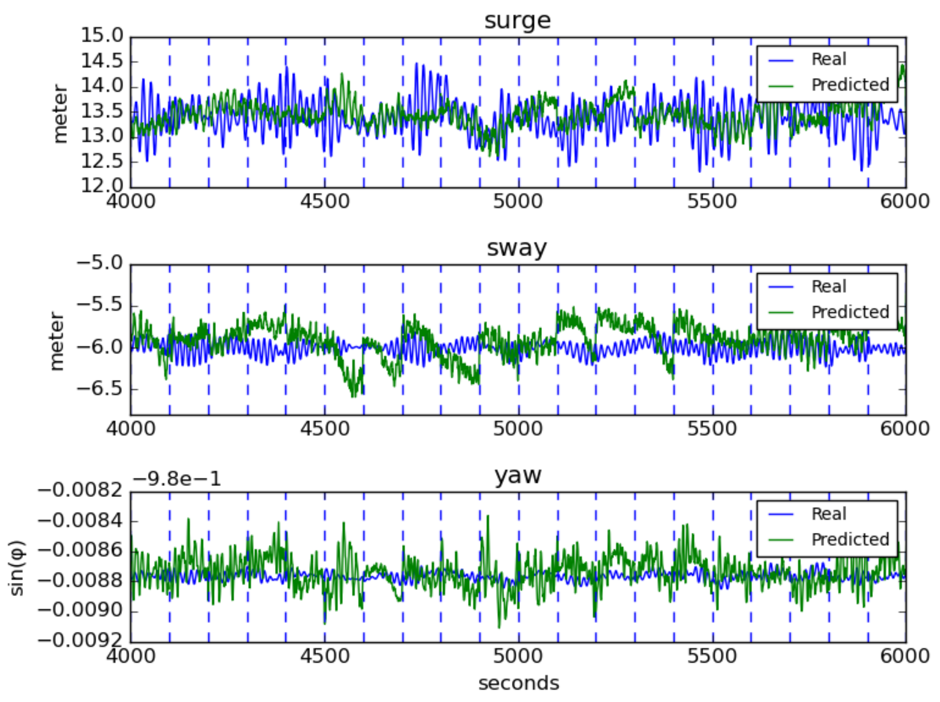


(b) A zoomed illustration of the MLP predictor of a platform motion under a mild scenario with all mooring lines intact.

Figure 18: MLP prediction for mild environmental condition with all mooring lines intact. The simulated data is in blue and the predicted in orange. Top: surge, middle: sway, and bottom: yaw motions.

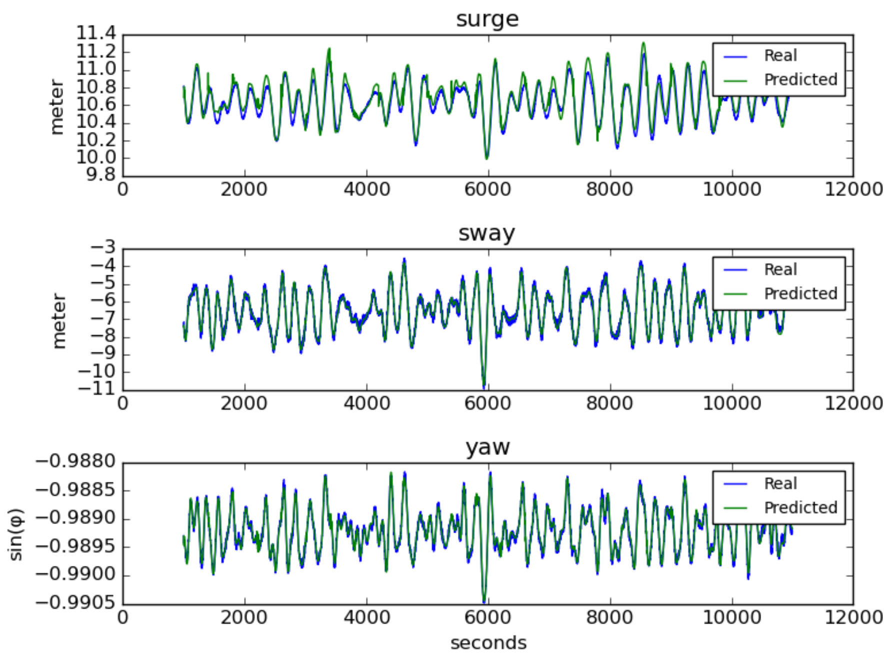


(a) Illustration of the MLP predictor of a platform motion under a stormy scenario with all mooring lines intact.

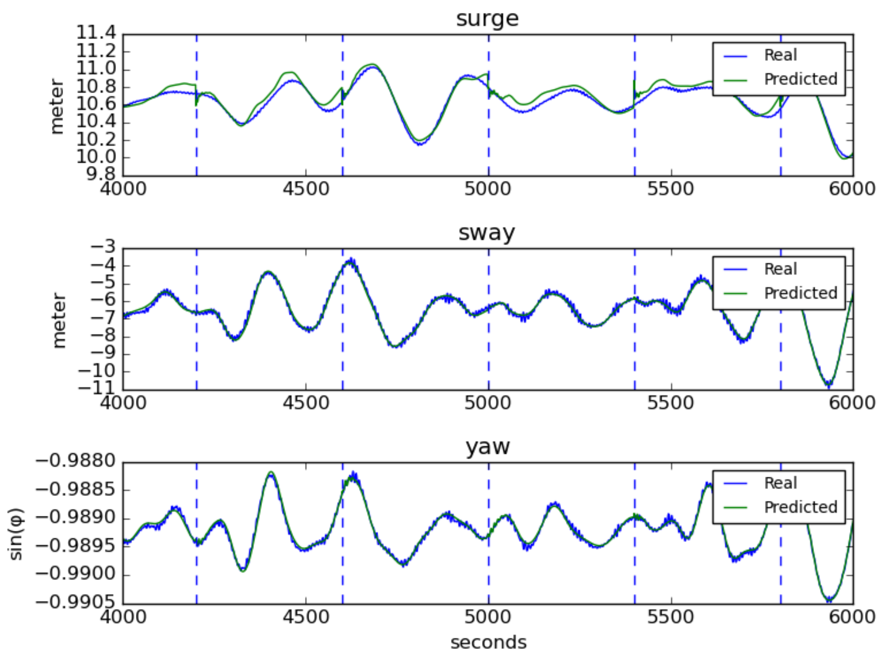


(b) A zoomed illustration of the MLP predictor of a platform motion under a stormy scenario with all mooring lines intact.

Figure 19: MLP prediction for stormy environmental condition with all mooring lines intact. The simulated data is in blue and the predicted in orange. Top: surge, middle: sway, and bottom: yaw motions.

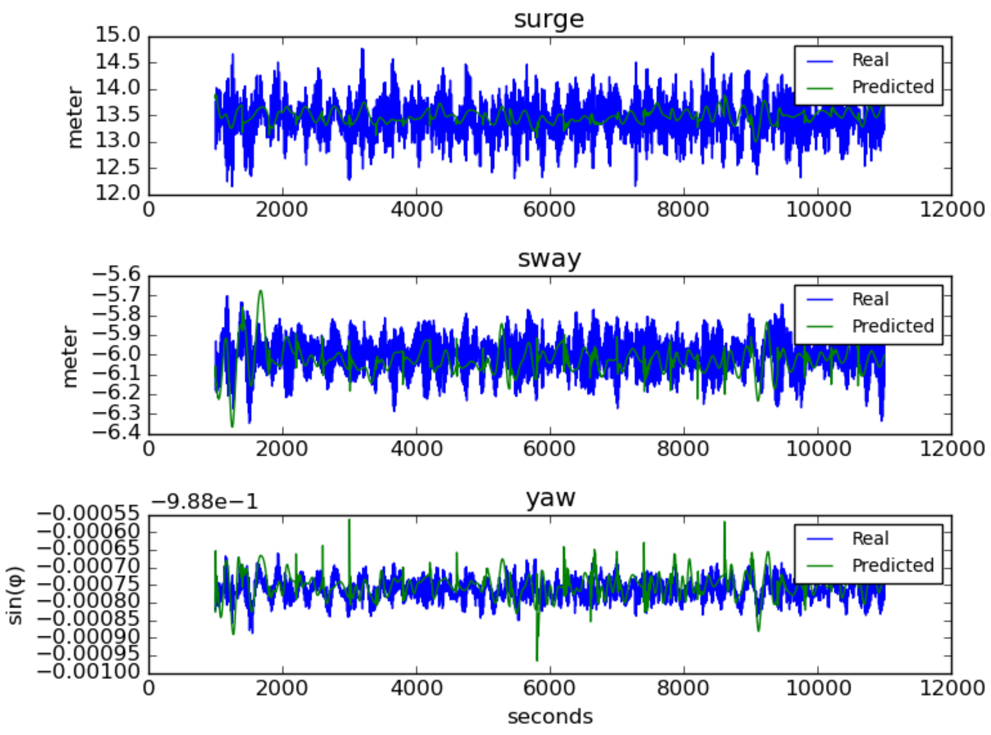


(a) Illustration of the LSTM predictor of a platform motion under a mild scenario with all mooring lines intact.

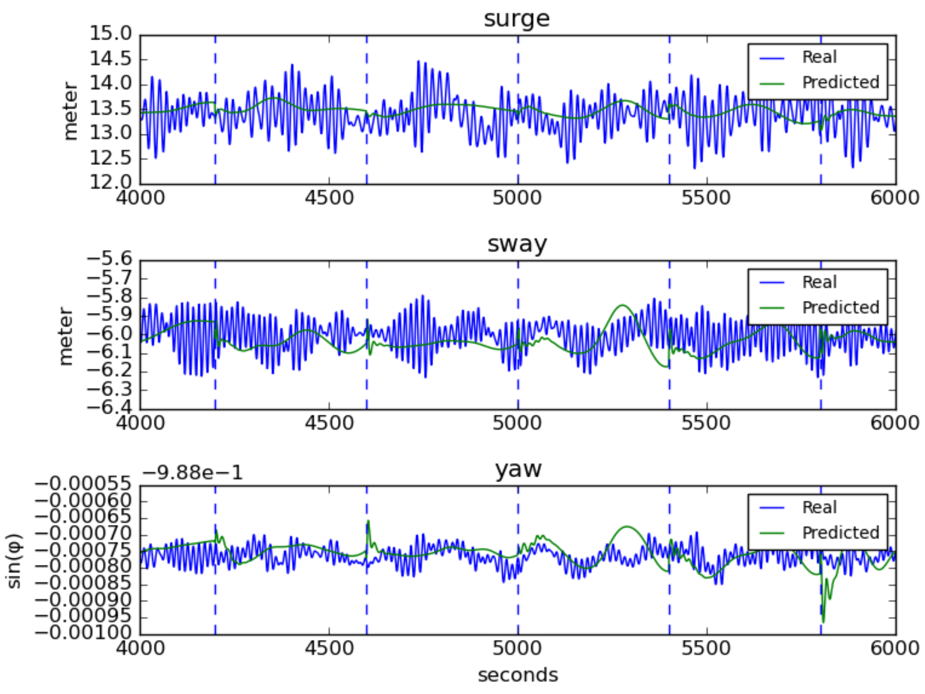


(b) A zoomed illustration of the LSTM predictor of a platform motion under a mild scenario with all mooring lines intact.

Figure 20: Long Short Term Memory (LSTM) prediction for mild environmental condition with all mooring lines intact. The simulated data is in blue and the predicted in green. Top: surge, middle: sway, and bottom: yaw motions.



(a) Illustration of the LSTM predictor of a platform motion under a stormy scenario with all mooring lines intact.



(b) A zoomed illustration of the LSTM predictor of a platform motion under a stormy scenario with all mooring lines intact.

Figure 21: LSTM prediction for stormy environmental condition with all mooring lines intact. The simulated data is in blue and the predicted in green. Top: surge, middle: sway, and bottom: yaw motions.

but for a stormy environmental condition, it also found it difficult to predict the rapid motions of the platform as shown in Figure 21a.

Furthermore, comparing the prediction performance between these two networks shows the LSTM network performed a little better than the MLP network. This is illustrated by how close the LSTM network was able to predict the platforms' 3DoF oscillations for mild environmental conditions as shown in Figure 20b while for the MLP network there are prediction windows where the network prediction does not closely match the simulation motions as shown in Figure 18b.

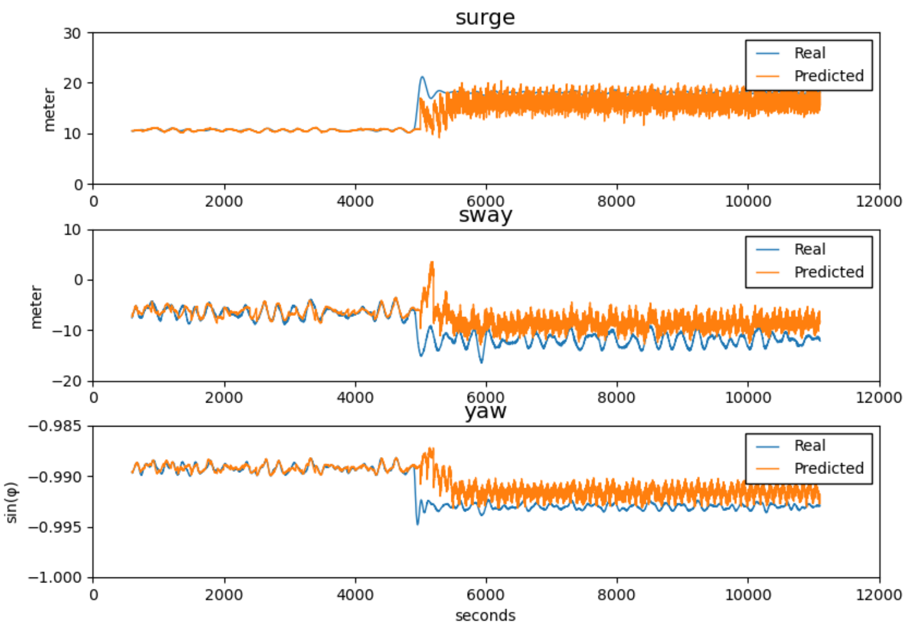


Figure 22: Illustration of breakage in mooring line L1 at time step 5000 seconds. The orange line is the MLPs' prediction and the blue line is the simulated platform motion. Top: surge, middle: sway, and bottom: yaw motions of the platform.

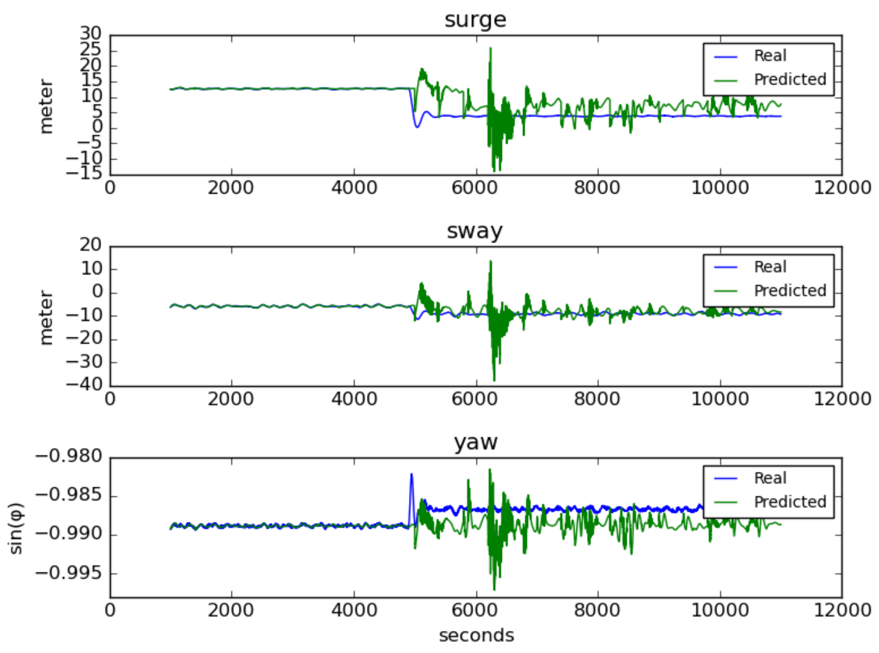


Figure 23: Illustration of breakage in mooring line L1 at time step 5000 seconds. The green line is the LSTM prediction and the blue line is the simulated platform motion. Top: surge, middle: sway, and bottom: yaw motions of the platform.

These networks were also evaluated on scenarios where the simulated motions had a single broken mooring line. For illustration purpose, motions with mild environmental condition were used. Here the goal was to access whether the implemented networks were able to detect platform motions with compromised mooring system.

Figure 22 for the MLP model and Figure 23 for the LSTM model show the networks prediction performance on platform motion with mooring line failure in line 1 (L1) of the mooring system. It can be seen that after mooring line breakage happens at time step 5000 seconds, an offset in the platform position occurs, where both models were unable to predict the motions of the platform henceforth, indicating the occurrence of mooring line breakage.

Comparison between MLP and LSTM Predictor Modules: The results of the developed and trained MLP and LSTM networks showed that both networks were able to detect the occurrence of mooring line failure for all test scenarios presented to the models. Despite demonstrating a somewhat low accuracy in the prediction of movement when under a stormy sea state, it is still observed that they were able to detect cases with failure in these conditions. Table 8 shows a comparison between the two trained predictors.

Table 8: Comparison between the two predictor models. 6DoF refers to sway, surge, heave, roll, pitch, and yaw, and 3 DoF stands for sway, surge, and yaw.

Characteristic	MLP	LSTM
Input variables	6DoF	3 DoF
Input	600 seconds	1000 seconds
Output	100 seconds	400 seconds
Output variables	3 DoF	3 DoF
Training units	55,000 units	10,000 units
Validation units	11,000 units	2,000 units
Trainable parameters	58,872,900	485,227
Training time	short ($\approx 5h$)	long ($\approx 24h$)
Execution time (one prediction)	negligible (real time)	

It can be seen that the LSTM model is 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 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. The LSTM network also used fewer platform motion variables as inputs (3 horizontal platform motion variables) than the MLP network, which used surge, sway, yaw, roll, pitch and heave as input. However, they both predicted the same three platform motion features (surge, sway and yaw). As the LSTM training process is slower than the MLP, the number of training units was significantly lower than the MLP training units without affecting the performance of the LSTM in the execution phase. The LSTM model performed better in all cases when compared to the MLP model. Both models are equally quick to predict once they are trained. Although the LSTM network was better than the MLP network in all test cases, one of the disadvantage of using this network was the long training time in contrast to MLP network. This can greatly impact the real application, as several changes can occur, such as changes in sensors or in the platform mooring system.

Furthermore, as shown in Figure 23 the LSTM network has a tendency to associate the mooring line breakage as an artifact after which it begins to learn to predict movement sometime after the breakage, absorbing the change. On the other hand, the MLP network cannot recover after the failure, being unable to predict the platform movements unless the failure is repaired. Thus, the MLP shows a clearer difference between platform movement with and without a compromised mooring system.

Hence, the MLP network was the preferred ML model to be use and to be further refined. The MLP model used all the 6DoF motion of the platform as input, it had a shorter training window which allows for faster evaluation and tuning of the network and finally the difference between the MLP prediction accuracy and that of the LSTM network was negligible.

4.2.1 Improved Predictor Module

Due to the fact that both investigated networks (MLP and LSTM) found it difficult to accurately predict the simulated platform movements under sea conditions that were classified as stormy, an investigation was carried out into the number of sea conditions with stormy sea conditions. It was noted that of the $18,000$ environmental conditions measured, only 100 were in the stormy sea category, prompting a review of the data to be used to develop the predictor module.

The revision conducted comprised of grouping the entire $18,000$ environmental conditions into clusters and sampling equal number of environmental conditions, according to their severity embracing calm to stormy conditions. In this way, the database was balanced between the possible categories, both in training and test data. More on the revision of the data is discussed in chapter 5. Nevertheless, using the new dataset the predictor module was retrained.

In the project phase described in this section, the MLP network was retrained with the now balanced data and new tests were performed. A binary classifier was also added to the architecture that indicates whether or not there was a break in a line based on the difference between the predicted and measured motions, as shown in Figure 24.

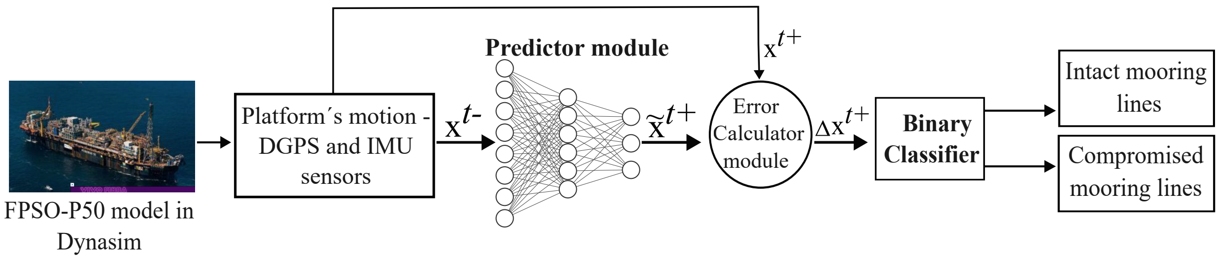


Figure 24: Second architecture proposed: mooring line failure identification is based on the prediction of FPSO-P50 platform motions

The MLP network hyper-parameters such as number of layers, number of neurons, activation functions used, were all kept the same like that used in the first proposed MLP architecture configured in section 4.2.

Results of the revision implemented showed the data refinement done improved the prediction accuracy of the MLP network and the binary classifiers were able to classify platform motions into the group it belong to accurately on test data (SAAD et al., 2021b). More on the classification performance is provided in section 4.4.

In the next section, network architecture optimization is investigated.

4.2.2 Architecture Optimization

In order to determine if the MLP architecture used in the first version of NeMo (SAAD et al., 2021b) was optimal, Neural Architecture Search (NAS) methods were employed in order to optimize the number of neurons in each of the three hidden layers of the proposed network.

Given the dependence on a good quality surrogate model to act as an objective function estimate, the model evaluation hyper-parameters can be especially relevant in the execution of Bayesian Optimization (BO). Due to the lack of literature on the optimal configuration for such hyper-parameters, three configurations were evaluated regarding the amount of data available and the number of training epochs during the optimization, as detailed in Table 9.

The following hyper-parameters were kept constant: inputs to the MLP were 600s of the 6DoF movements of the FPSO-P50 platform and outputs were 100s of the 3DoF

Table 9: Parameters used for the Bayesian Optimization (BO) experiments.

Experiment	**Data used**	**Training epochs**
BO1	40%	1600
BO2	80%	1600
BO3	40%	5000

motions; for each of the 3 hidden layer, a maximum of 9000 neurons and a minimum of 100 neurons, were set with a step size of 100 neurons for each trial (e.g 100, 200,....,9000); batch size of 64, and Mean Squared Error (MSE) was used as the loss function.

Data preparation for NAS: Simulated platform motions of FPSO-P50 with draft 16m, sampled from the clusterization step (described before and detailed in section 5.1.1) was used to conducted the NAS experiments. The simulated motions is comprised of 7000 P50s 6DoF movements, the first 3600 seconds of each simulated motion were removed (transients) and a third order Butterworth low-pass filter (BUTTERWORTH, 1930) with a cutoff frequency of 0.02Hz was applied to the remaining 7400s time-series motion, which were then split into windows of 700 seconds for each environmental condition simulated (details in section 5.1.2). The windows generated were randomly shuffled to form the dataset used for training and testing of the ML network.

NAS Experiments: We started the NAS optimization using the Random Search (RS) algorithm. RS has a policy of random choice of models and does not guarantee that it will arrive at an optimal model. Despite this, the literature reports good results, often comparable to those of state-of-the-art algorithms. Therefore, RS was used as a baseline to evaluate the model efficiency. However, this approach was considered time consuming to converge, managing to evaluate only a small number of architectures.

BO was then implemented to explore newer architectures during trials. The use of a surrogate model by BO accelerates the search process in the hyperparameter space, thus converging to optimal regions of the search space faster than RS. In BO, neural network architectures are iteratively chosen in trials to maximize the Expected Improvement (EI) of the objective function which, in our case, is the Average Cross-Validation MSE. The implementation and default parameters of both algorithms provided by the Optuna library were used (AKIBA et al., 2019).

NAS Optimization Result:

The models that presented the smallest error (i.e., smallest difference between the expected outputs and those predicted by the model) were trained using 80% of the data and 5000 epochs and evaluated on the holdout test set. The optimized, as well as the current, architectures are detailed in Table 10.

Table 10: Current architecture and those resulting from optimization. HL stands for Hidden Layer in the MLP architecture.

Model	**Neurons per HL**		
	1	2	3
Current	7200	3600	1800
RS	6100	2300	1800
BO1	9000	0300	8500
BO2	8700	1000	5500
BO3	8500	0500	6800

The performances of the architectures were compared and a Wilcoxon signed-rank test (REY; NEUHÄUSER, 2011) – a nonparametric statistical test which compares two paired groups – was used to verify the statistical significance of the differences between architecture error distributions. Test results are presented in Figure 25 via a Critical Difference (CD) diagram (FAWAZ et al., 2019a) of the holdout test MSE of each model. The Current model is represented here as Base. In the CD diagram, models with statistically similar MSE distributions are connected by a horizontal line (which is not present in Figure 25), and those with better performance (i.e. lower MSE) present a higher statistic.

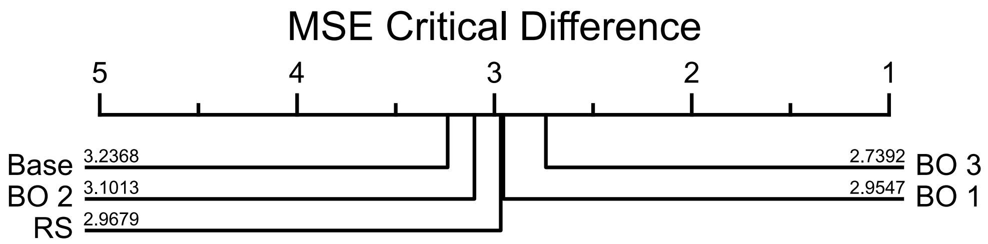


Figure 25: Comparison of different architecture performance via a Critical Difference diagram.

Overall, no models presented statistically equivalent performances, and, contrary to expectations, the optimization process did not find an architecture with smaller error than the current one. Furthermore, even though the execution of BO2, using 80% of the data, reached the second smallest error, the remaining executions BO1 and BO3 resulted in statistically worse models than the one found by RS, and no architecture had performance comparable to that of the current one (Base in figure). Thus, the manually determined architecture developed was validated.

4.2.3 Ablation study of input to the Network

In this section, we conducted an ablation study to examine the impact of input to the network. Ablation studies in ML involve disabling or removing individual components of a model to evaluate their impact on the model's overall performance. The goal of ablation study is to better understand how individual components of the model affect its performance, which can inform improvements by removing unnecessary components or adding new ones (MEYES et al., 2019).

Therefore, in this study the goal was to determine the minimum number of platform motion variables required and the time required to train the network to detect line breakage. Additionally, we evaluated whether a network trained with fixed yaw motions at 210° while allowing normal motion of the remaining platform degrees of freedom can identify mooring line rupture.

In ML, a trade off between model accuracy and training time is always present. This requires a compromise to be made so that a model is trained just enough to solve the problem at the desired accuracy.

In this study, draft of 16m without mooring line breakage was selected based on feedback from experts (Petrobras personnel) that indicated, the FPSO-P50 platform normally stays with draft between 14m to 16m. Thus, draft of 16m was used for this study.

Three MLP networks were trained and tested on the same number of data. For training, 3000 random environmental conditions and for validation 1500 different environmental conditions were used. 1000 environmental conditions were used as test data. The network hyper-parameters such as number of layers, batch size, number of epochs for training were kept the same (see Table 11), with the only difference being the number of platform motion variables fed to the input layer of the networks.

Table 11: Fixed Hyper-parameters

Setting	Number
Training Dataset	3000
Validation Dataset	1500
Test Dataset	1000
Number of Layers	5
Batch size	32
Epochs	2000

The three models trained are:

- **Model A** which uses all 6DoF motions as inputs to the model to predict the horizontal motions – surge, sway and yaw of the platform,
- **Model B** had only the horizontal motions of the platform as inputs and it predicted the same motions and,
- **Model C** uses surge, sway and yaw, the latter being fixed to 210°, as inputs to predict horizontal movements.

The average Root Mean Square Error (RMSE) error of the models and training time are presented in Table 12.

Table 12: Average RMSE error of the three models tested on 1000 platform motions.

Models	**Average RMSE Error**	**Input Neurons**	**Training Time**
Model A	1,21E+16	3,600	1 day 18 hours and 53 minutes
Model B	1,06E+16	1,800	13 hours and 11 minutes
Model C	1,18E+16	1,800	5 hours and 47 minutes

Note: Experiments were perform on a desktop station with 32GB RAM and GTX 1080 GPU card running Ubuntu 20.04.

The result shows **Model B** which used only the Three Degree of Freedom – Surge, Sway and Yaw (3DoF) had a lower average error score. This is counter intuitive to result expected. We expected **Model A** which uses 6DoF motions to have a lower error, as the network has the complete platform motion to utilize. It can be hypothesised that the heave, pitch and roll motions were negatively impacting the learning ability of the model. As they are related to the vertical motions of the platform, they can be considered noises at the input of the model, negatively impacting not only the interpretation of the model, but also unnecessarily increasing its complexity (by having more inputs, the model has many more weights to learn).

In Figure 26, the 6DoF movements of a sample test case without any mooring line breakage are depicted. The heave, pitch, and roll motions can be observed to oscillate rapidly, which makes it challenging for the network to learn effectively. On the other hand, Figure 27 demonstrates that even in the presence of mooring line breakage, it is difficult to precisely identify the moment when the breakage occurs in the heave, pitch, and roll motions. However, for the surge, sway, and yaw motions, it is apparent that the breakage happens at approximately 5000 seconds.

Model C also showed that, even with yaw motion constant, it was able to generalize to unseen data but the average error was higher than **Model B** with a small margin,

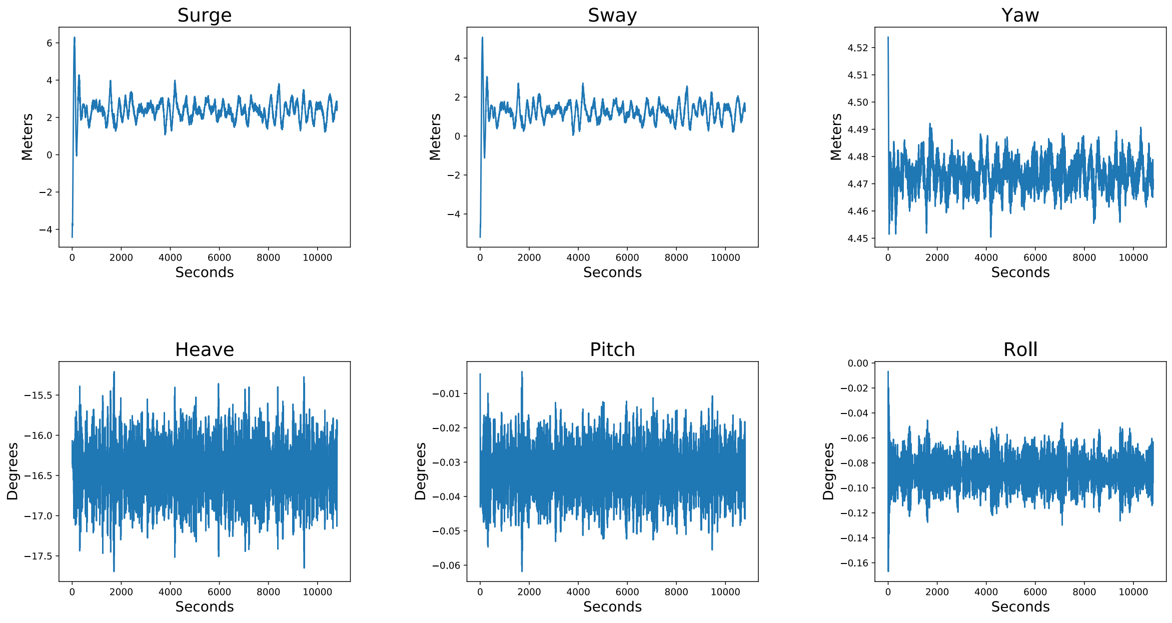


Figure 26: An illustration of an six degrees of freedom (6DoF) motion of a platform without line breakage.

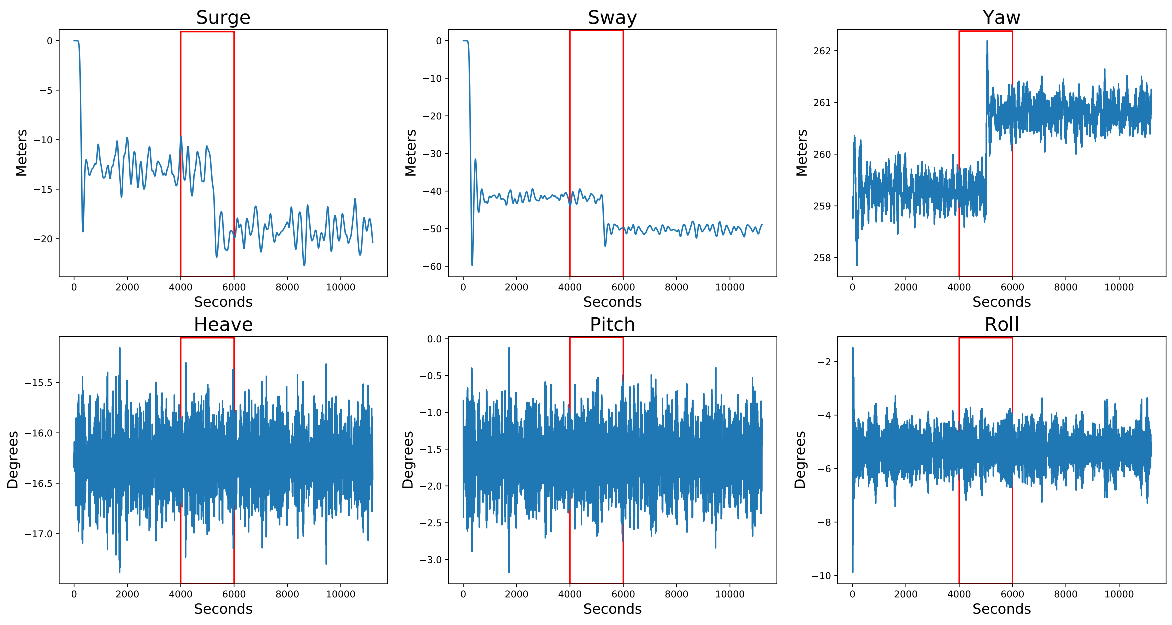


Figure 27: The six degrees of freedom (6DoF) of platform motion. The red rectangle highlights how the motion changes before time 4000 seconds and after line breakage at time 5000 seconds happens.

which is understandable because in reality it was only using to 2Dof motion – surge and sway.

For training time, **Model A** had the longest training time. This can be attributed to the model having more parameters to learn than the other models with less inputs to the network.

Thus it can be concluded that using only the 3DoF horizontal motion as input is sufficient for detecting line breakage and the training time is not too expensive when compared to model A.

Figure 28 shows the mooring failure detection ability of the models, in figure 28a, Model B– which used only the 3DoF for input and predicts the same motion is shown and in figure 28b, Model C–which uses the surge, sway and a fixed yaw at 210° as inputs to predict the horizontal movements is also shown. Since the 3 models were all predicting the same horizontal motion, model A prediction was not included.

As it can be seen, both models were able to detect when mooring line failure occurs by showing an offset between the models' prediction and the actual simulated platform motion for the same environmental condition. Although they were able to detect mooring line rupture, the error score for the same platform motion presented in Table 13 shows Model C had a higher error score values when compare to Model B.

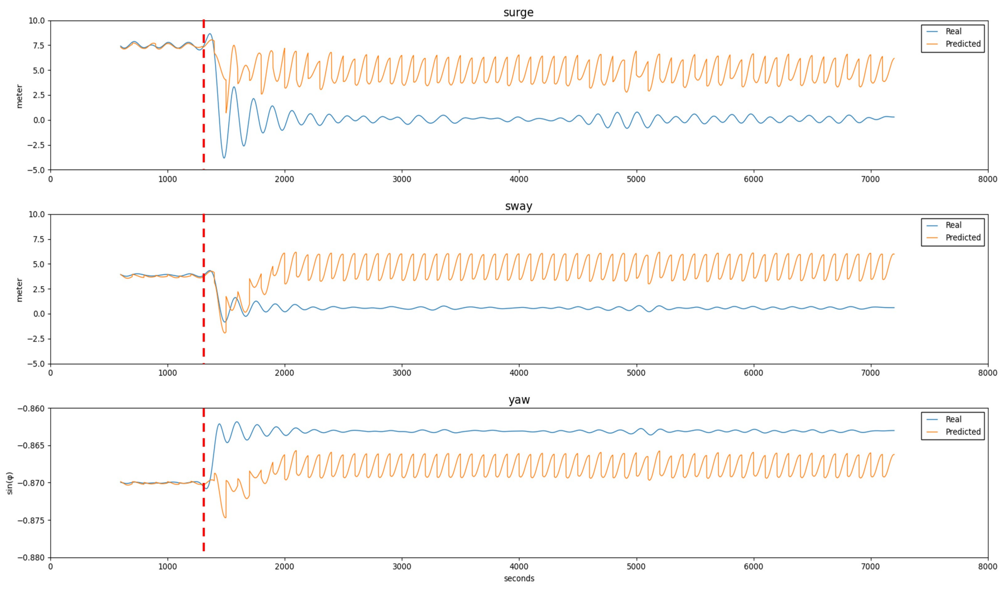
Table 13: Error score comparison between model B with a normal yaw and model C with a fixed yaw

Error score	Model B (Yaw Normal)			Model C (Yaw Constant)		
	RMSE	Mean	Median	RMSE	Mean	Median
Surge	5.618	-5.289	-5.495	7.177	-6.537	-6.335
Sway	4.285	-4.163	4.158	3.064	-2.870	-2.852
Yaw	0.00821	0.00795	0.0085	0.0034	0.0034	0.0033

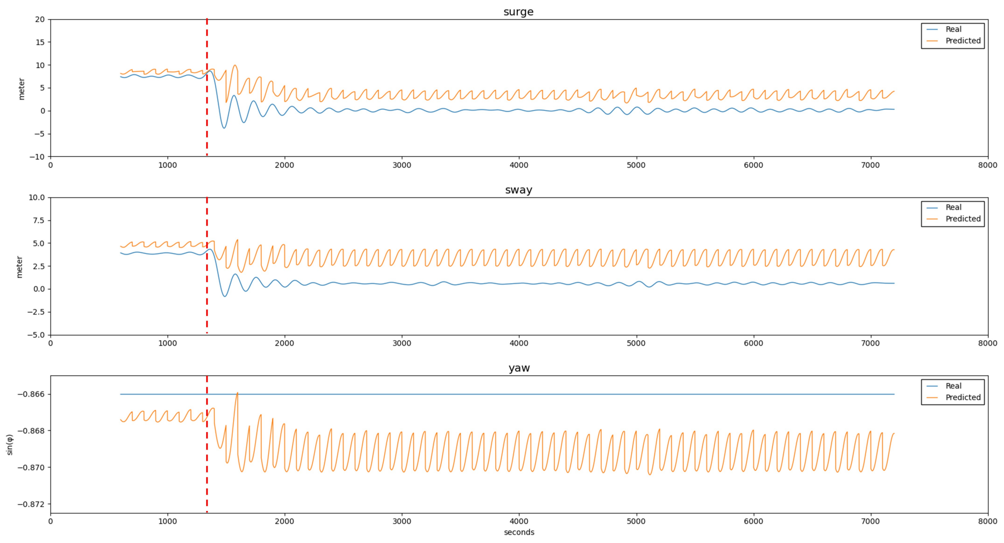
The next step will be to extend the functionality of the prediction module to handle mooring line fault identification for platform motions with various draft levels. The interest here is to train a single MLP model that is capable of detecting the mooring line break for all possible draft levels. Thus, the need to train individual networks for each draft level would be eliminated, reducing the computational cost and time associated with training each network.

4.3 Error Calculator Module

After the ML model is trained, it is used to predict platform motion based on the previous motion. Here the idea is to compare the error score generated between the predicted motions with the simulated ones, measuring the disparity between them. Figure



(a) Model B breakage detection. The orange line is the model B prediction and the blue line is the simulated platform motion.



(b) Model C breakage detection. The orange line is the model C prediction and the blue line is the simulated platform motion.

Figure 28: Illustration of breakage in mooring line L1 at time step 5000 seconds, demarcated with the red vertical line. The orange line is the models' prediction and the blue line is the simulated platform motion. Top: surge, middle: sway, and bottom: yaw motions of the platform

29 shows where the Error Calculator module fits in the architecture of the NeMo system.

Assuming that a mooring line failure leads to a irregular motion, the disparity between prediction and simulation can then be used to identify mooring line failures. 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 conditions.

Using the error calculator module, error scores RMSE, Mean Error (ME) and Median Error (MedE) are calculated for each simulated platform motion and are then used to discerning the status of the mooring lines of the platform.

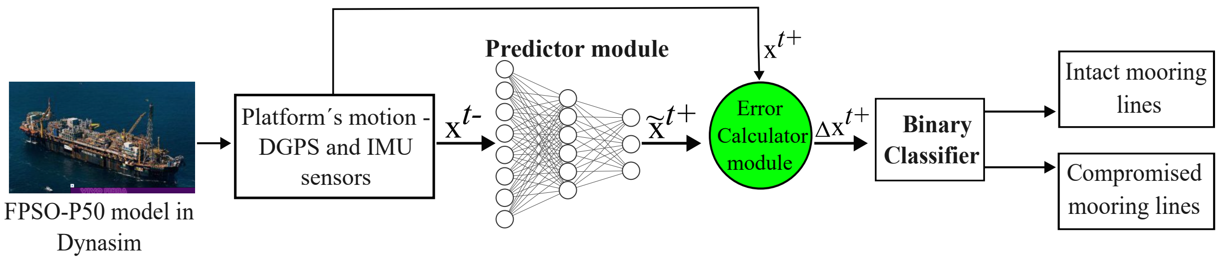


Figure 29: Error calculator module location in the NeMo system highlighted in green.

The error scores presented here are based on the result gotten from the second version of the MLP module discussed in section 4.2.1. The trained network was validated on different platform motions with different mooring line configurations based on a single draft setting.

Table 14 presents the RMSE error scores for test sample with six different configurations, listed below:

1. Test sample with mild environmental condition with intact mooring lines.
2. Test sample with stormy environmental condition with intact mooring lines.
3. Test sample with mild environmental condition with line 1 broken.
4. Test sample with mild environmental condition with line 9 broken.
5. Test sample with mild environmental condition with line 12 broken.
6. Test sample with mild environmental condition with line 18 broken.

Table 15 details the ME and MedE error scores of the same aforementioned test sample cases.

Table 14: RMSE MLP Error Scores

RMSE	Surge	Sway	Yaw
Mild Sea State	0.257	1.656	5.483 e-04
Stormy Sea State	0.748	0.579	2.066 e-04
Failure L1, Mild Sea State	6.061	6.289	3.594 e-03
Failure L9, Mild Sea State	5.761	10.038	4.594 e-03
Failure L12, Mild Sea State	7.756	5.802	4.523 e-03
Failure L18, Mild Sea State	10.425	1.780	3.241 e-03

Table 15: Median and Mean MLP Error Scores

Case	Median Error			Mean Error		
	Surge	Sway	Yaw	Surge	Sway	Yaw
Mild Sea State	0.240	-0.982	-2.445 e-04	0.232	-0.878	-2.058 e-04
Stormy Sea State	-0.438	-0.555	-1.255 e-04	-0.472	-0.546	-1.288 e-04
Failure L1, Mild Sea State	-5.296	-6.086	3.097 e-03	-5.415	-6.225	3.143 e-03
Failure L9, Mild Sea State	5.478	-9.869	-4.486 e-03	5.766	-9.913	-4.827 e-03
Failure L12, Mild Sea State	7.430	5.398	4.265 e-0	7.430	5.609	4.512 e-03
Failure L18, Mild Sea State.	-10.026	-1.010	-3.007 e-03	-10.019	-08.416	-3.102 e-03

As can be seen in Table 14 and Table 15, the error scores for motions with intact mooring line setup, that is mild and stormy case, are one order of magnitude smaller than motions with broken mooring line. This disparity between error scores confirms the breakage has occurs and can be detected.

Also, the mean and median error scores, Table 15 reveals the error score signs, i.e., positive or negative, changes depending on which mooring line is broken, highlighting they could be used to determine which line of the platform mooring system was compromised.

Further tests were carried out to investigate the spread of error scores between platform motions with intact mooring line configuration and compromised mooring system which can used as a threshold for considering the occurrence of mooring line failure. For this test 500 different platform motions were sampled and the test samples used to represent cases with broken mooring line were based on platform motions with breakage in line 1. Table 16 shows the Standard Deviation (StD) and Mean (ME) of the surge, sway and yaw motions of these platform with intact mooring lines.

Table 17 details the StD and ME of the surge, sway and yaw motion of 500 platform motions with one mooring line breakage and it reveals the error score spread is small, indicating a good chance of detection of mooring line breakage.

Comparing the two Tables 16 and 17, we can see that the error scores for motions

Table 16: Error score of 500 test data with intact mooring line cases.

Error score	RMSE		Mean		Median	
	Mean	StD	Mean	StD	Mean	StD
Surge	0.715	0.769	0.119	0.972	0.119	0.972
Sway	0.454	0.316	-0.107	0.413	-0.107	0.414
Yaw	6.2183e-4	4.506e-4	1.023e-4	3.291e-4	1.023e-4	3.291e-4

Table 17: Error score of 500 test data with compromised mooring lines cases.

Error score	RMSE		Mean		Median	
	Mean	StD	Mean	StD	Mean	StD
Surge	8,935	1,604	-8,848	1,656	-8,848	1,656
Sway	5,958	0,649	-5,599	0,667	-5,599	0,667
Yaw	1.050e-2	3.704e-4	3.704e-4	6.290e-4	9.596e-3	6.289e-4

with mooring line breakages were 10 times higher than error scores for motions with intact mooring lines. Hence, a stark contrast is evident amongst platform motions with and without a compromised mooring system.

4.3.1 Scatter Plot Visualization

Following the computation of all error scores for all simulated environmental condition with all the different mooring line configurations, the results are presented in a 3D scatter plot. Each axis of the graph belongs to one of the platform motions: surge, sway and yaw.

Using the RMSE error scores of 500 test environmental conditions with and without a compromised mooring system, a 3D scatter plot is presented in Figure 30, illustrating how distinct the error scores of these motions without mooring failure and with mooring line are. Given RMSE depends on the quadratic difference between the conditions, it is impossible to assess the signs of changes when a line fails. However, clear division is observed between platform motion with and without a compromised mooring system using the error metric in the plot.

Figure 31 presents the mean (ME) error score scatter plot of 500 test environmental conditions with and without a compromised mooring system. Here it can be seen there are five groups in the plot which corresponds to which mooring line is broken. The plot reveals that the use of multiclass classification in the error score ME can allow the identification of which mooring line is compromised, since ME provides the error signals that solve the

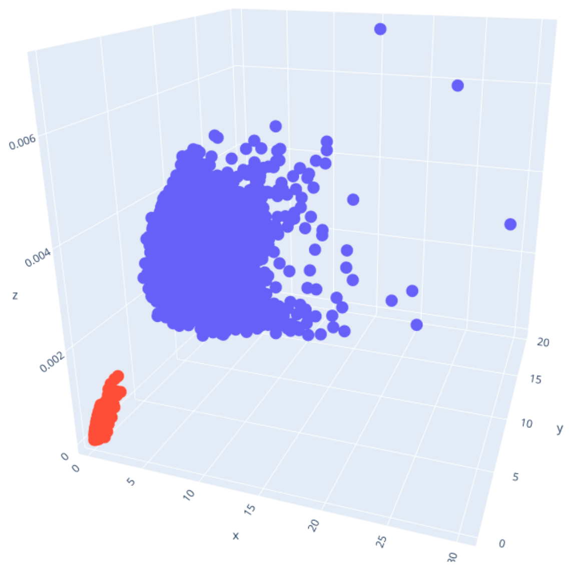


Figure 30: RMSE error score of platform motions; surge, sway and yaw for 500 test environmental conditions. Red dots correspond to intact lines and blue dots to broken lines.

problem encountered using only the RMSE error score.

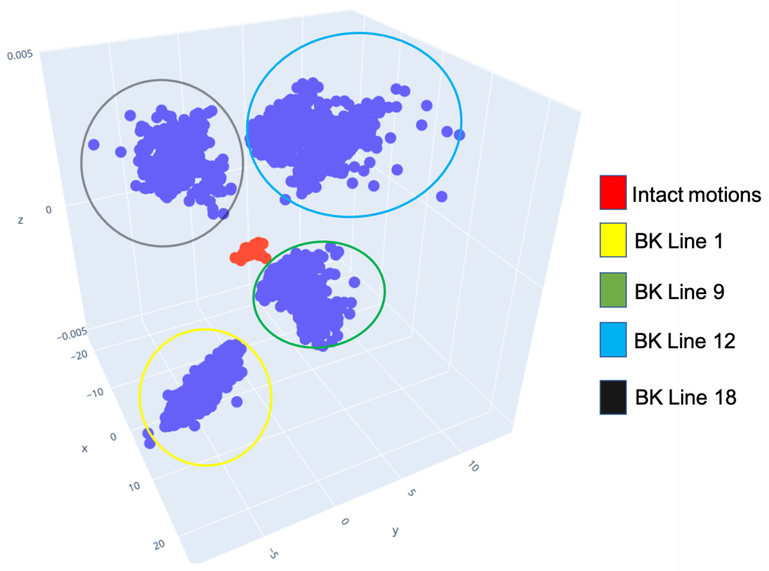


Figure 31: Mean (ME) error score of platform motions surge (x), sway (y) and yaw (z) for 500 test environmental conditions. Red dots represent non-breaking lines, dots circled in yellow are motions with a break on line 1, circled in green are motions with a break on line 9, circled in blue are motions with a break on line 12, and circled in black are motions with a break on line 18.

Figure 32 presents the median (MedE) error score scatter plot of 500 test environmental conditions with and without a compromised mooring system. Here it can also be seen that there are five groups in the plot which corresponds to which mooring line is broken. Red dots represent no breaks and blue dots are breakages. The plot also reveals using the MedE error score, multi-class classification can be done as the error scores are distinct enough, allowing for identification of which mooring line is compromised.

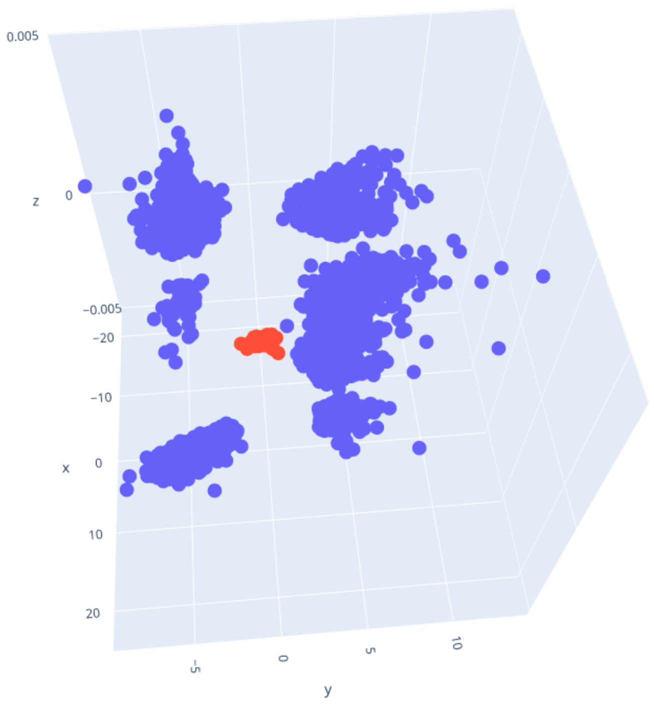


Figure 32: Median (MedE) error score of platform motions; surge, sway and yaw for 500 test environmental conditions.

In conclusion, the scatter plots for RMSE error in Figure 30, and the Mean and Median errors in Figures 31 and 32, respectively illustrate the clear separation that exists between the platform motions with all the mooring lines intact and the platform motions that have a failure on the mooring line. As the RMSE error depends on the quadratic difference between the conditions, it is impossible to assess the signs of changes when a line fails. This issue no longer occurs with mean and median errors. There is also a clear separation between the four groups of mooring lines of the platform used, indicating that there is a possibility to identify which group the failure belongs to.

However, as these errors are based on the platform with a single draft configuration, further investigation should be done for the case where multiple drafts exist.

4.4 Classifier Module

Classification is the problem of identifying the class label to which a new input observation belongs. In this thesis, the final component of the NeMo system is a supervised classifier module. This module uses the three error scores, RMSE, mean and median computed by the error score calculator module as input to classify the status of the platform mooring system. The initial implemented version of this module was a binary classifier that classifies platform motions into two groups, as "failure" and "no-failure". Different classifiers were implemented and compared to find the most capable binary classifier. Figure 33 presents the general architecture of the first implemented classifier module.

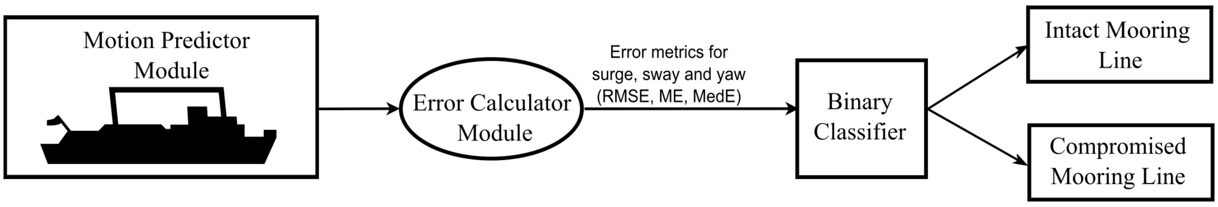


Figure 33: The input and output format of the classifier module used to classify mooring line status using the different error scores calculated by the error score calculator module

The three ML classifier implemented were a K-nearest Neighbour (KNN) classifier, Decision Tree (DT) classifier and, a Support Vector Classifier (SVC) (see Chapter 2 for details). These classifiers were trained and tested on error scores of platform motions without and with mooring line failure, under different simulated environmental conditions and mooring line configurations for platform FPSO-P50 with a single draft setting (16 meters). The total number of training and test data used are detailed in Table 18.

Table 18: Data for classification

Line	Number of training data	Number of testing data	Status
No Failure	2000	5000	no failure
Failure L1	2000	5000	failure
Failure L5	2000	5000	failure
Failure L6	2000	5000	failure
Failure L9	2000	5000	failure
Failure L10	2000	5000	failure
Failure L12	2000	5000	failure
Failure L15	2000	5000	failure
Failure L18	2000	5000	failure
Total number of platform motions	18000	45000	

These classifiers were trained on $2,000$ platform motions without mooring line failure and on another $16,000$ platform motions with mooring line failure, making $18,000$ different platform motions. For testing the classifiers were test on 45, 000 platform motions. Results of the trained classifiers are as follows:

K-nearest Neighbour (KNN) classifier: The first binary classifier implemented was a K-nearest Neighbour (KNN) classifier, which used $K = 100$ nearest neighbours to determine the group in which a platform motion belongs to: "failure " or " no failure " group. The euclidean distance metric was the distance metric for the KNN. Figure 34 presents the result of the trained KNN classifier on 40, 000 platform motions with mooring line failure into the group "Failure" while for the "No-Failure" group, out of 5, 000 motions, it misclassified 106 motions into the "Failure" group. The prediction accuracy of the KNN classifier was 0.9976, however since the accuracy metric is not always a good indicator of the performance of a classifier, the recall, F1-score and precision score were also calculated. The recall score of the KNN was 1, precision score was 0.9788 and F1-score was 0.9893.

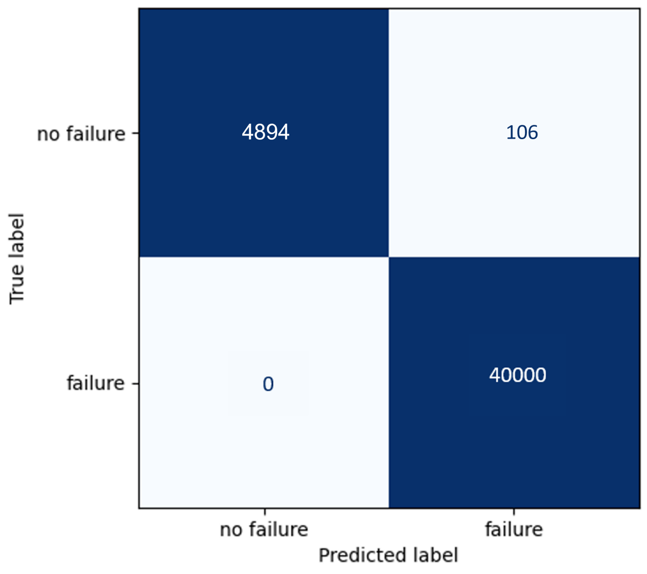


Figure 34: K-nearest Neighbour (KNN) classifier prediction on mooring line status.

Decision Tree (DT) classifier: The second classifier implemented was a DT classifier. The classifier was configured to use the *gini* index measure for attribute selection metric and the classifier used the Classification and Regression Tree algorithm (CART) approach for classifying the motions of the platform. The DT classifier was trained on 18, 000 platform motion and tested on 40, 000 platform motions with mooring line failure where it correctly classified all the motions into the group "Failure" while for the "No-Failure" group, out of 5, 000 platform motions, it misclassified 6 motions into the "Failure" group as shown in Figure 35. The prediction accuracy of the DT classifier was 0.9999, the recall score was 1, precision score was 0.9988 and F1-score was 0.9994.

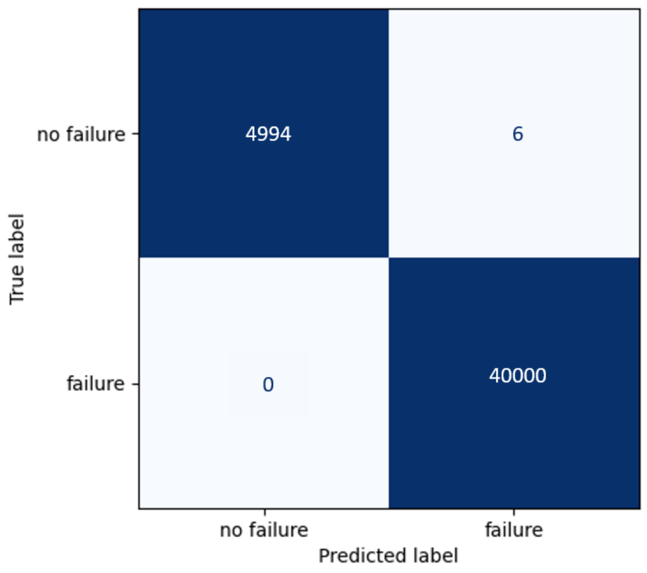


Figure 35: Decision Tree (DT) classifier prediction on mooring line status.

Support Vector Classifier (SVC): The third classifier implemented was a SVC classifier. The SVC classifier implemented was also trained on 18, 000 platform motions to assign platform motions into two groups: Failure or No-Failure group. Result of the trained SVC is presented in Figure 36 and it shows that the classifier correctly inferred the group to which the 40, 000 platform motions with anchor line failure belonged, whereas for the 5, 000 motions of the "No-Failure" group, it incorrectly classified 131 motions in the group "Failure". The prediction accuracy of the SVC classifier was 0.9971, the recall score was 1, precision score was 0.9738 and F1-score was 0.9867.

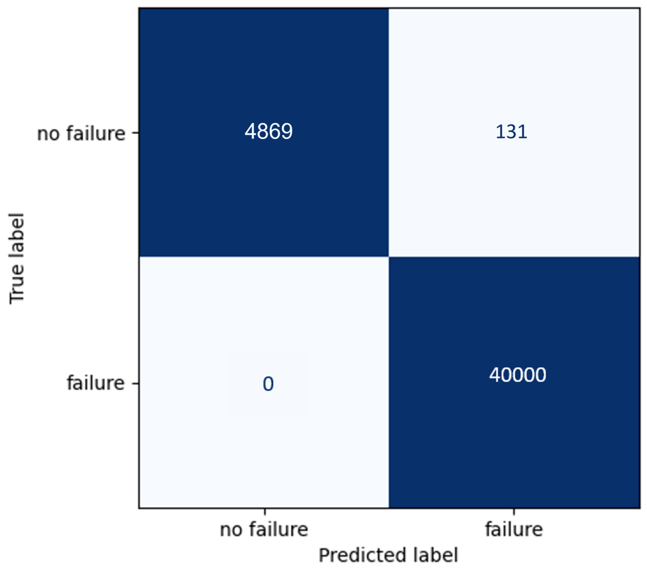


Figure 36: Support Vector Classifier (SVC) classification on mooring line status.

The three binary classifiers implemented in the initial phase of the classifier module were able to assign the group to which platform motions belong to accurately with low misclassification rate. Platform motions without failure were discovered to be misclassified in all cases; nevertheless, it is important to note that false-negative classification did not occur in all of the classifiers. This is very important because, if an anchor line collapse does occur, despite the classifier's prediction that nothing has happened, the results can be disastrous both for the structural integrity of the platform and for the safety of the user.

A comparison between the three binary classifier designed and implemented showed the DT classifier had the lowest misclassification rate, followed by the KNN classifier and, lastly the SVC classifier.

The next step is extending the classifier module functionality to handle multi-class classification. In this phase the objectives are :

- To implement a classifier capable of classifying the status of platform motions into five groups: No_break, Failure Group 1, Failure Group 2, Failure Group 3 and Failure Group 4 as illustrated in Figure 37.
- To implement a classifier capable of providing the probabilities for each possible line status.
- Implement a classifier capable of handling different levels of drafts of a platform without the need to train several classifiers for each of the possible drafts.

In this way, more information will be made available to assist in decision making and scheduling the physical inspection of the mooring cables, giving more safety to the floating platforms. In the next section, experiments and results of our proposed system NeMo are presented.

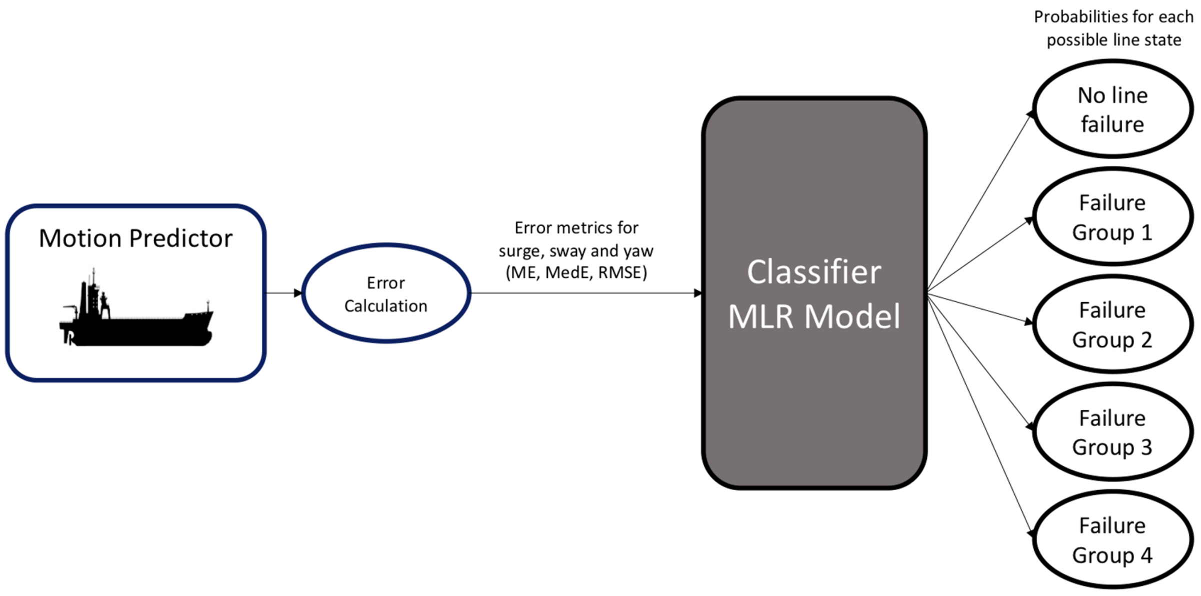


Figure 37: The input and output format of the multi-class classifier module used to classify mooring line status using the different error scores calculated by the error score calculator module.

5 EXPERIMENTS AND RESULTS

This chapter is split into three sections representing the different parts necessary for implementing the proposed system (NeMo): firstly, in Section 5.1, we describe the data generation and engineering process for NeMo. Secondly, in Section 5.2, the predictor module of the NeMo system is discussed – we present the preparation steps done for the simulated motions for training and evaluating of the predictor module; and then the training and test dataset used in developing the predictor module are provided; and finally, we detail experiments and results obtained. Figure 38 presents the phases executed for the predictor modules.

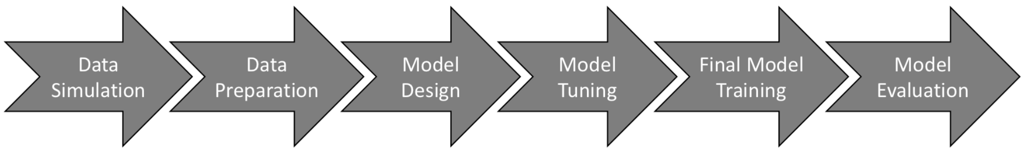


Figure 38: Pipeline of the predictor module of NeMo.

Finally, in Section 5.3, the classifier module of NeMo is discussed — we start with a prediction error analysis (Section 5.3.1), followed by detailing the training and testing dataset composition in Section 5.3.2, the result of the implemented classifier is provided in Section 5.3.3 and finally, Section 5.3.4 concludes indicating topics for future research for the module.

5.1 Data Engineering

The data used in this work are simulated motion data which are described in the next sections, starting with the generation of simulated data, then describing how the data is prepared and divided making it adequate to be understood by the ML model implemented.

5.1.1 Data Simulation

In this work, the Dynasim simulator was given environmental conditions, specifications of P50-FPSO and other parameters such as draft, breaking time and line to be broken to generate times-series motions of the 6DoF motions of platform under the various environmental conditions. The general pipeline for this phase of data preparation is shown in Figure 39.

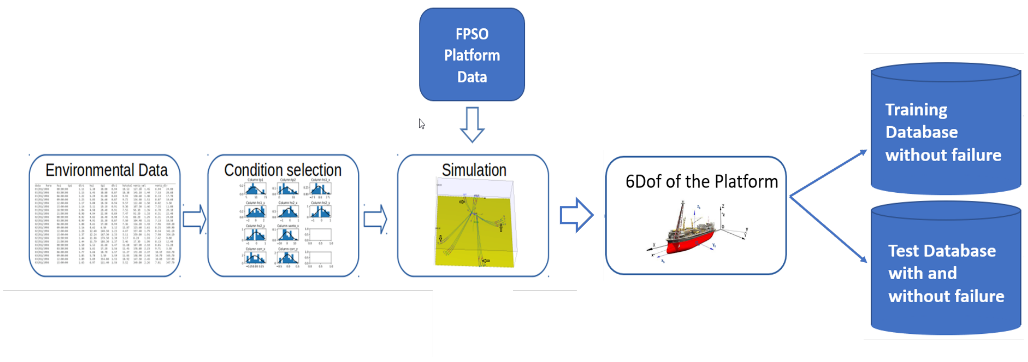


Figure 39: Generation of training and test data: Initially real data of environmental conditions are analyzed and sampled, feeding the simulator that then generates motion data for the specified platform.

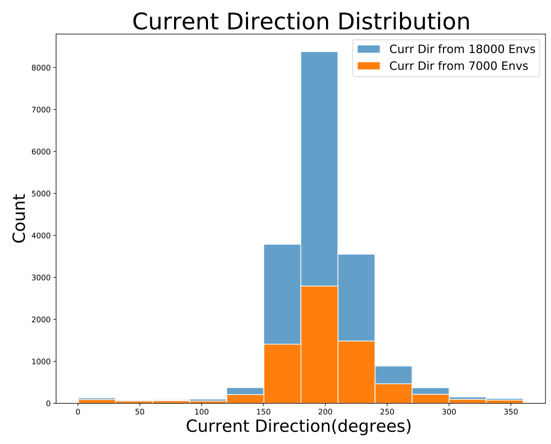
Real Environmental Conditions: Real environmental condition readings were collected from a weather station located in Campos Basin (Bacia de Campos) of Rio de Janeiro, Brazil from 2003 to 2006. These environmental conditions were recorded at an interval of 3-hours, making 18,000 distinct environmental conditions. The features of the environmental conditions recorded are described in Table 19.

In order to ensure that the dataset contained an equal proportion of each possible environmental condition severity (e.g., calm or stormy), the recorded environmental conditions were pre-processed as described below.

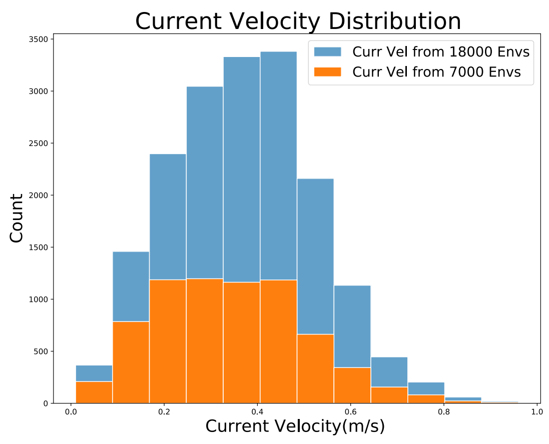
Environmental Condition Clustering: The aforementioned 18,000 recorded environmental conditions were then grouped into clusters, according to their severity, applying a K-means clustering algorithm (MACQUEEN, 1967). Silhouette scoring (ROUSSEEUW, 1987) was used to assess the quality of clusters created by the K-means algorithm, and define the best number of clusters – here, six distinct clusters – in terms of how similar the clustered samples were and the distances between the clusters, in which 1,167 different environmental conditions were randomly sampled from these six clusters, making 7,000 conditions. Figures 40 and 41 presents the distribution of the 7,000 randomly sampled environmental conditions from the 18,000 environmental conditions.

Table 19: Environmental condition dataset features

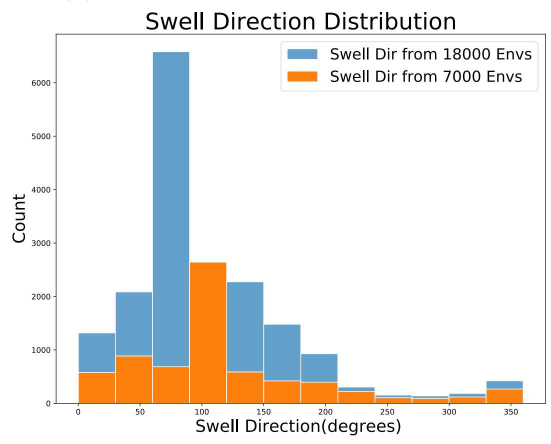
Feature	Name	Unit
Date	data	DD/MM/YY
Hour	hora	HH:MM
Wave Height	hs1	Meters [m]
Wave Period	tp1	Seconds [s]
Wave Direction	dir1	Degrees [º]
Swell Height	hs2	Meters [m]
Swell Period	tp2	Seconds [s]
Swell Direction	dir2	Degrees [º]
Wave Total Height	hstotal	Meters [m]
Wind Speed	vento_vel	Meters per Second [m/s]
Wind Direction	vento_dir	Degrees [º]
Current Speed	corr_vel	Meters per Second [m/s]
Current Direction	corr_dir	Degrees [º]



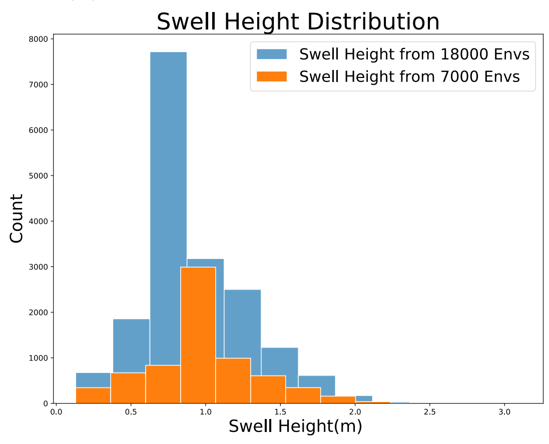
(a) Current Direction Distribution



(b) Current Velocity Distribution

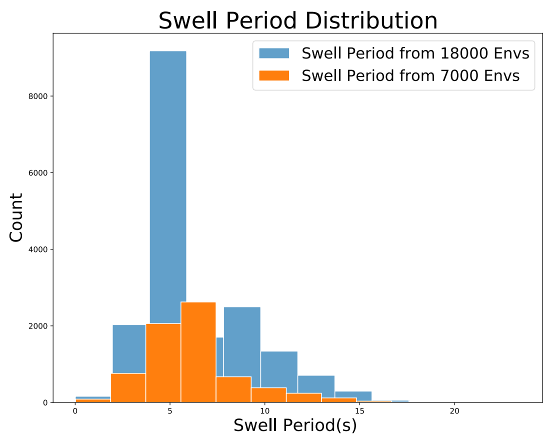


(c) Swell Direction Distribution

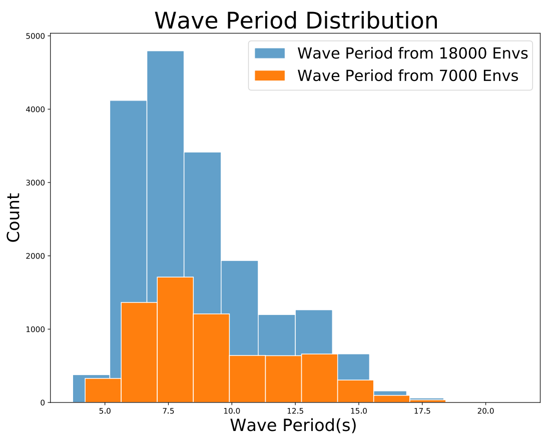


(d) Swell Height Distribution

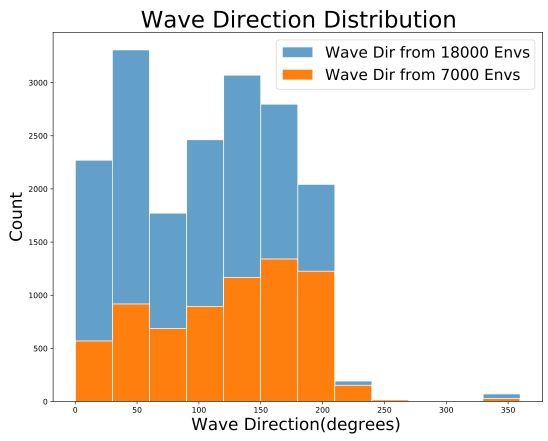
Figure 40: Histogram distribution of the 7,000 randomly sampled environmental conditions from the 18,000 environmental conditions after clustering, for the following variables: current direction (Fig. a) and velocity (Fig. b), as well as swell direction (Fig. c) and height (Fig.d).



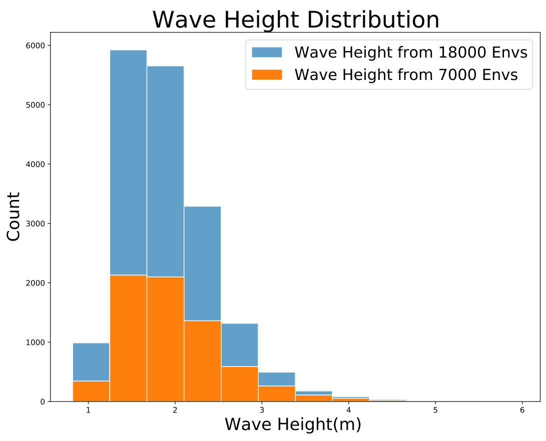
(a) Swell Period Distribution



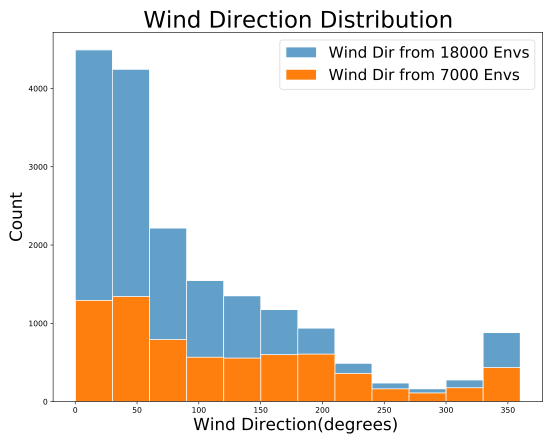
(b) Wave Period Distribution



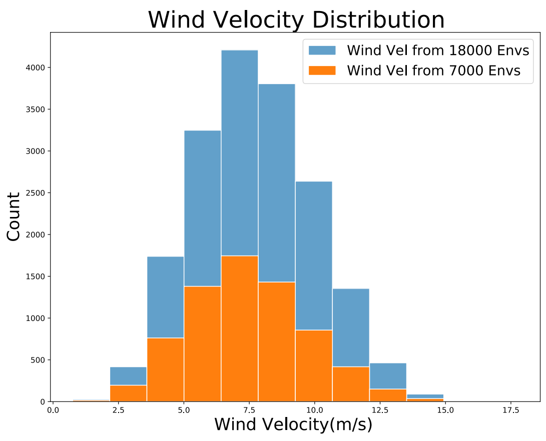
(c) Wave Direction Distribution



(d) Wave Height Distribution



(e) Wind Direction Distribution



(f) Wind Velocity Distribution

Figure 41: Histogram distribution of the 7,000 randomly sampled environmental conditions from the 18,000 environmental conditions after clustering, for the following variables: swell period (Fig. a), wave period (Fig. b), direction (Fig. c), and height (Fig. d), as well as wind direction (Fig. e) and velocity (Fig. f).

Note: the zero values of each measurement were replaced with the mean.

Environmental Condition Preprocessing: A sample from the 7000 environmental conditions is shown in Table 20, which shows wave, swell, wind and current. Wave and swell are characterized by their height, period, and direction, while wind and current are characterized by their speed and direction.

Table 20: Example of Environmental Conditions

Current Velocity	Current Direction	Wind Velocity	Wind Direction	Wave Height	Wave Period	Wave Direction	Swell Height	Swell Period	Swell Direction	Date	Time
0.18	321.17	6.90	87.20	1.73	11.51	179.80	1.11	5.05	97.40	4/ 8/2006	09:00:00
0.48	228.11	5.01	120.60	2.00	9.63	187.50	0.58	3.77	133.20	25/ 8/2004	06:00:00
0.37	216.58	3.90	60.50	2.07	15.99	129.60	0.36	3.30	66.00	10/ 6/2007	09:00:00
0.27	163.09	9.67	20.40	2.34	13.07	191.30	1.34	5.98	39.30	18/ 1/2007	03:00:00
0.26	216.56	6.30	107.40	2.72	14.40	177.10	1.00	4.56	112.00	2/ 5/2008	15:00:00
0.35	219.37	5.80	64.20	2.09	10.44	175.50	0.73	4.51	74.80	2/11/2008	12:00:00
0.43	197.07	9.36	44.0	1.44	5.44	54.30	0.42	6.14	136.40	26/11/2004	21:00:00

The angles that indicate velocity directions have been projected onto their x and y components, as described in Section 2.3.3. For the wave and the swell, their angles are multiplied by the height.

All conditions were normalized (see Section 2.3.3) before the simulation. The result can be seen in Table 21.

Table 21: Environmental condition scaled between 0 and 1

Wave Period	Swell Period	Wave Height_x	Wave Height_y	Swell Height_x	Swell Height_y	Wind_x	Wind_y	Current_x	Current_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

Simulation Cases: The simulator Dynasim takes as input the 7000 sampled environmental conditions resulting from the K-means clustering and the platform specification and draft setting to produce simulated platform motion data under these conditions. The platform specification and draft settings are shown in Table 2. Using the Dynasim simulator, platform motions with and without mooring line breakage for platform with drafts ranging from 8m to 21m (14 drafts) were generated. For motions with line breakage, one of its specified mooring line was broken at time step 5000 seconds during the simulation.

Mooring Line Breakage: The P50-FPSO model used by the Dynasim has 18 mooring lines attached to it. These 18 mooring lines were divided into 4 groups (see Figure 13) and one line from each group was broken in the simulations (see Table 22).

Table 22: Mooring line groups. Lines in bold were broken during simulation

Line Groups	
Group 1	**L1**, L2, L3, L4, L5
Group 2	L6, L7, L8, **L9**
Group 3	L10, L11, **L12**, L13
Group 4	L14, L15, L16, L17, **L18**

5.1.2 Data Preparation

The output of Dynasim simulations consist of motion times series for each of the 6DoF of the platform – surge, sway yaw, roll, heave and pitch – subject to the selected environmental conditions. This section outlines the necessary steps to prepare these time series for adequate analysis. First we explain the transformation of the coordinate system and standardization of the simulated platform motions, and then we describe how time series are transformed into a format suitable for model training.

Time Series Processing: Dynasim simulations are subject to transient effects that do not reflect actual platform motions and thus need to be removed. A cutoff period of $3600s$ was selected, meaning the first hour of the three-hour simulation is removed, leaving two hours or $7200s$ of useful data for each simulated condition (See Figure 42). Therefore, for the simulated motions with line breakage at time step 5000, after removing the transient motion, the breakage time adjusts to time step 1400 seconds.

Dynasim also provides platform motion data based on its global frame of reference, which is not aligned with that of the platform. Therefore, these motions are rotated so that e.g. motion data in the x-axis becomes the actual platform surge motion.

In order to facilitate model convergence during training, the resulting motion time series are standardized. This process consists of standardizing each variable from each time series separately, using the z-score calculation (see Section 2.3.3). For the i-th time series ($i = 1, 2, \ldots, 7000$), the j-th DoF of the platform ($j = 1, 2, \ldots, 6$) is thus standardized by calculating its z-score (see Section 2.3.3).

Windowing: These simulated time series are partitioned into multiple sections to facilitate their analysis by machine learning and statistical models, as processing the complete time series at once would require complex models which are also resilient to changes in the underlying trend of the series.

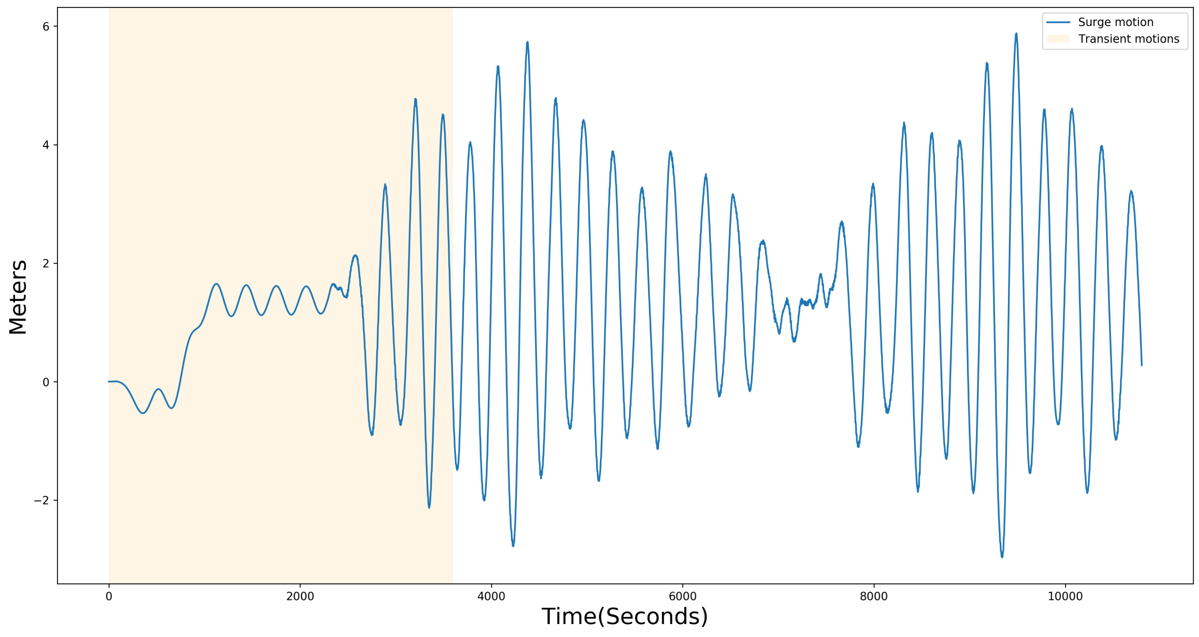


Figure 42: An example of the surge motion time series output generated by the Dynasim simulator. The first 3600 seconds (in yellow) of the 3 hour (10,800 seconds) simulation represents the range dropped due to transient motions and the remaining 7200 seconds are used to training and test the predictor module of NEMO.

Each of these sections is called a *window*. The time-series generated are divided into windows that share a common duration or *width* w and which may or may not overlap (i.e. contain a period of time which is also contained in another window). The window generation procedure is bound by a parameter s denoted *stride*, which is the distance between the beginning of one window and that of the next one. The k-th window of a time series is thus generated by taking data from the $k \cdot s$ to the $k \cdot s + w$ instant in the series. This definition highlights how window overlap is controlled by the stride: a stride $s < w$ implies there is overlap between adjacent windows.

Window width and stride need to be carefully determined so that the model have enough information to make predictions about global trends in the time series while also supplying information on the short- and mid-term variations in the series and balancing model computational cost.

Filtering: A third order Butterworth low-pass filter (BUTTERWORTH, 1930) with a cutoff frequency of $0.02Hz$ was applied to the remaining 7200s time-series motion in order to smooth out high frequency oscillations that do not provide relevant information about platform dynamics and that are detrimental to model training. Figure 43 shows a sample of the surge in blue and sway in red motions before and after filtering.

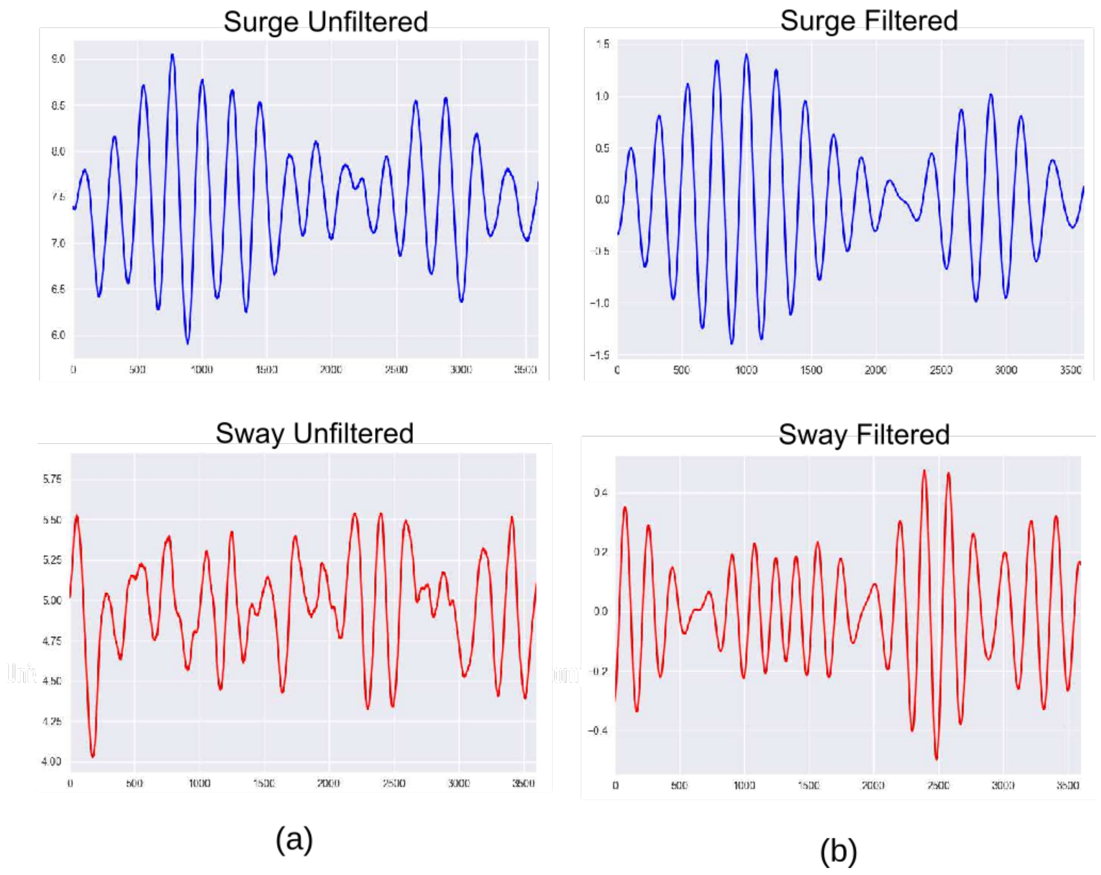


Figure 43: Sample of the surge motion in blue and the sway motion in red before (a) and after (b) filter application.

This concludes the data engineering and preparation phase conducted.

5.2 Motion Predictor Module

This section focuses on the predictor module of the NeMo system. The predictor module is composed of an MLP neural network model (as described in Section 4), which is trained with simulated platform motion data (generated as described in Section 5.1.1) and then used to forecast these motions. The hypothesis is that, as the predictor is trained with the simulated FPSO-P50 motions without failures, differences detected between the predicted series and the conditions indicate failures in the platform's mooring system.

The MLP network that plays the role of predictor was implemented to use the previous 600 seconds of horizontal motions – surge, sway and yaw – to predict the next 100 seconds of the same horizontal motions, as shown in Figure 44. We call *forecast window*, w_f, the future period predicted by the predictor. Here, $w = 600s$ and $w_f = 100s$.

Predictor Module Configuration

The predictor module is composed of an input layer, 3 hidden layers and an output layer, whose nodes are fully connected to those of the preceding and following layers, as described in Section 4. An illustration of the predictor architecture is shown in Figure 44. The input layer receives 600 seconds of horizontal motions (surge, sway and yaw), which amount to 1800 data points (1 point for every second). The output layer predicts 100 seconds of the same motions, which is equivalent to 300 data points. The first to third hidden layers have 7200, 3600 and 1800 neurons in each layer, respectively.

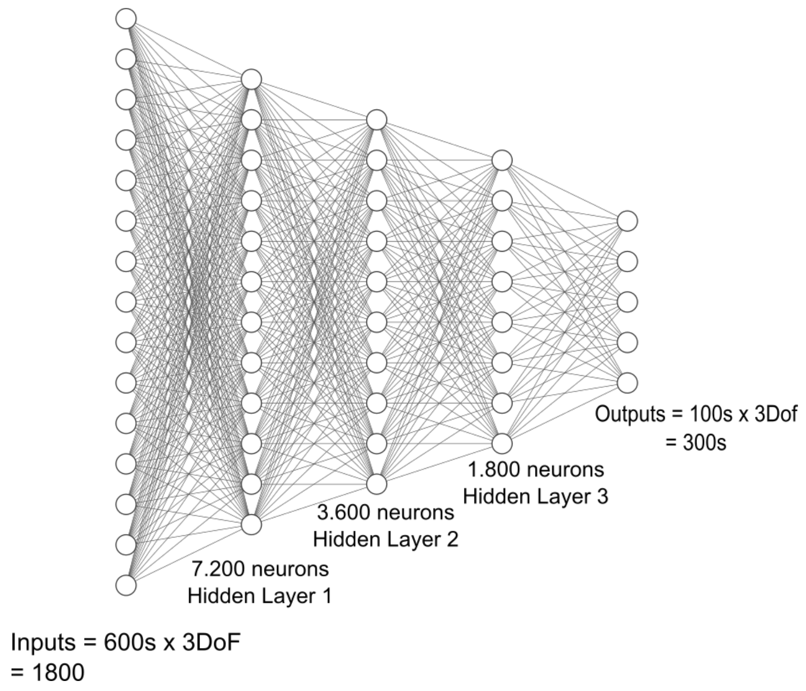


Figure 44: Motion predictor model. The MLP network receives 600s input of the surge, sway and yaw motions (3DoF) and predicts 100s of the same motions. The numbers under the layers represent the number of neurons in the best-known MLP architecture.

The rectified linear (ReLU) activation function was used in all the layers with the exception of the output layer, which used linear activation function to make the prediction. The Adam optimizer algorithm with a learning rate of $1 \cdot 10^{-7}$ was used to optimize the MLP network. The model training was stopped at epoch 3000 when the model started over-fitting.

5.2.1 Train and Test Dataset

An adequate sampling methodology is critical to guarantee the ML model's ability to learn and generalize to new data. Therefore, from the 7000 simulated environmental conditions as explained in Section 5.1.1, a subset of 3000 random environmental conditions were used for training and 1500 for validation, both selected from multiple platform drafts and the remaining environmental conditions separate from those used for training and validation were used as the test dataset. From these motions, the time series values associated with the three horizontal degrees of freedom – sway, surge and yaw – were employed and the first $3600s$ were removed and the remaining $7200s$ were filtered as described in Section 5.1.2.

For each environmental condition, parts of the time series that depict the platform motion generated by the Dynasim were sampled. Each of these parts comprises a data window that corresponds to a training unit. A training unit comprises of a set of $< input - output >$ unit, which is defined by a partition of the time series of the data window. Here, a training unit comprises of an input window of $w = 600$s and an output window of $w_f = 100$s, making a data window of 700s. The procedure of creating training units entails sliding the data window over the Dynasim-generated time series, with strides s of 100s. Windows were defined in Section 5.1.2. The set of all data window input-output pairs from the selected simulated motions form the training and validation sets for the MLP network.

In the training phase, data windows are independent of one another and can be drawn at random within the remaining 7400 seconds period from the same environmental condition. Similarly, data windows are collected from other selected environmental conditions, and the collection of all these training units forms a training set.

The MLP predictor module training data were composed of 3000 different platform motions, selected from different environmental conditions from different drafts, which generated $213,000$ windows. Similarly, the validation data were composed of 1500 different environmental conditions, consisting of $106,500$ windows were used to validate the network (See Appendix B for more on data window creation).

The MLP predictor module was trained for 3000 epochs, where an epoch defines the number of time the MLP model is trained on the entire training data set, and early stopping API from Keras[1] was employed in the training phase to stop the MLP model from over-fitting. After training, the trained MLP model is used to predict platform motion without and with mooring line breakage in the testing phase.

In our proposal, a batch consists of 32 training units, i.e., 32 pairs of input-output time windows.

[1]https://keras.io/

The test phase of the MLP network also requires preprocessing the selected simulated environmental conditions different from those selected for training and validation of the MLP network into test sets, consisting of input-output pairs and creating test data windows of equivalent training window size .i.e., 700 seconds. Test data windows with input window (w) of 600s are fed into the trained MLP network which then makes a $100s$ forecast of the output for the given test window.

For every test environmental condition 10 testing units were used and they are provided as input to the trained MLP network. For each test unit the error calculator module compares the networks forecast with the actual simulated motion and uses the difference to make error metric indexes. This process is repeated for all the sets of test data.

5.2.1.1 Error Calculation Module

Different metrics need to be generated to detail the difference between predicted and simulated motions in order to evaluate the performance of the proposed MLP model, since these same metrics are used as input for the line breakage classifier. In order to generate representative statistics about the complete time series, the model is used to generate continuous motion predictions for the duration of the $7200s$ time series.

As illustrated in Figure 45, for every $w = 600s$ input (represented in blue), a $w_f = 100s$ prediction (represented in orange) is calculated (Figure 45a) and the input is subsequently shifted by $s = 100s$ to generate the next prediction (Figure 45b). The resulting predicted motions are grouped into 200 second intervals without overlap (Figure 45c) for error metric calculation, in order to average out transient effects that arise when line breakage occurs and thus facilitate breakage detection.

Thus, for each 200 seconds of predicted Y_{pred} and corresponding simulated Y_{sim} horizontal motions, their difference Δ is calculated for each point by

$$\Delta = Y_{pred} - Y_{sim}. \tag{5.1}$$

For each Δ, the Root Mean Squared Error (RMSE), mean (ME) and median (MedE) are calculated. MedE is found by ordering the set of Δ values from lowest to highest and finding the exact middle. RMSE and ME are widely used error metrics in ML and are defined as

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} {\Delta_i}^2}, \quad \text{ME} = \frac{1}{n} \sum_{i=1}^{n} \Delta_i. \tag{5.2}$$

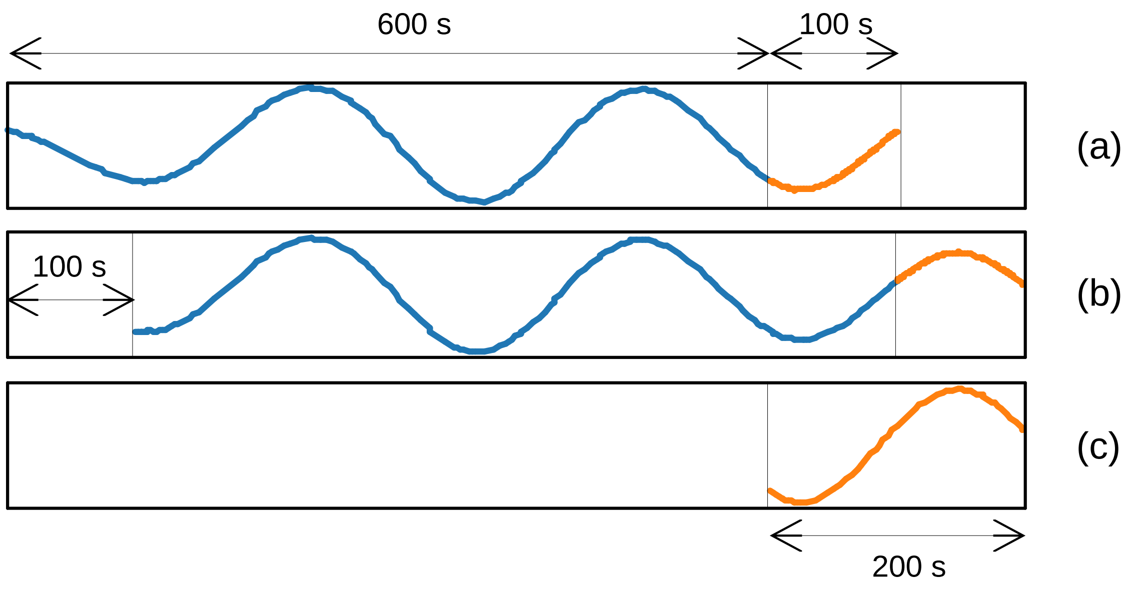


Figure 45: Illustration of the continuous prediction generation process and subsequent partitioning.

In cases where line breakage occurs, although MLP predictions generally differ significantly from simulated motions with breakage, erratic behaviour may occur and the network may temporarily generate predictions that follow these motions, which makes it necessary to consider error metrics from a longer time span than that of a single window. Therefore, the metrics belonging to the window with greatest RMSE are chosen to represent the entire 7200 second time series, and are used both to report MLP performance and to serve as input for the line breakage classifier.

5.2.2 Results

After the model architecture search step (see Section 4.2.2), different draft combinations during training was tested, in order to assess which group of drafts was optimal to train the predictor module of the NeMo system, to learn the complex motions for all drafts motion generated. Therefore, to evaluate the model's capacity to generalize, the test set included all possible drafts, that is, drafts from $8m$ to $21m$.

As training with all possible drafts (total of 14 different drafts, considering meter to meter variations) would be very costly, two different draft combinations were tested to find the minimum number of drafts needed that allows the predictor module to learn and generalize to every draft. One model was trained with motions for drafts 8, 16 and 18, while another model was trained with drafts 8, 12, 16 and 20.

3-draft model: This model had its training data constituted by 1,000 random movements of each draft, 8, 16 and 18, making 3,000 platform movements. The validation data consisted of 1,500 random movements of the same drafts, that is, resulting in 500 movements of each draft.

4-draft model: This model had its training data constituted by 750 random movements of each draft, 8, 12, 16 and 20, making 3,000 platform movements and was validated in 1,500 random movements of the same drafts, that is, 375 movements of each draft.

These two models were trained and tested on drafts motions without breakage. Figure 46 shows the average surge RMSE error score obtained for all the drafts of the two motion predictor models trained with 3 drafts (8, 16, 18) and 4 drafts (8, 12, 16, 20).

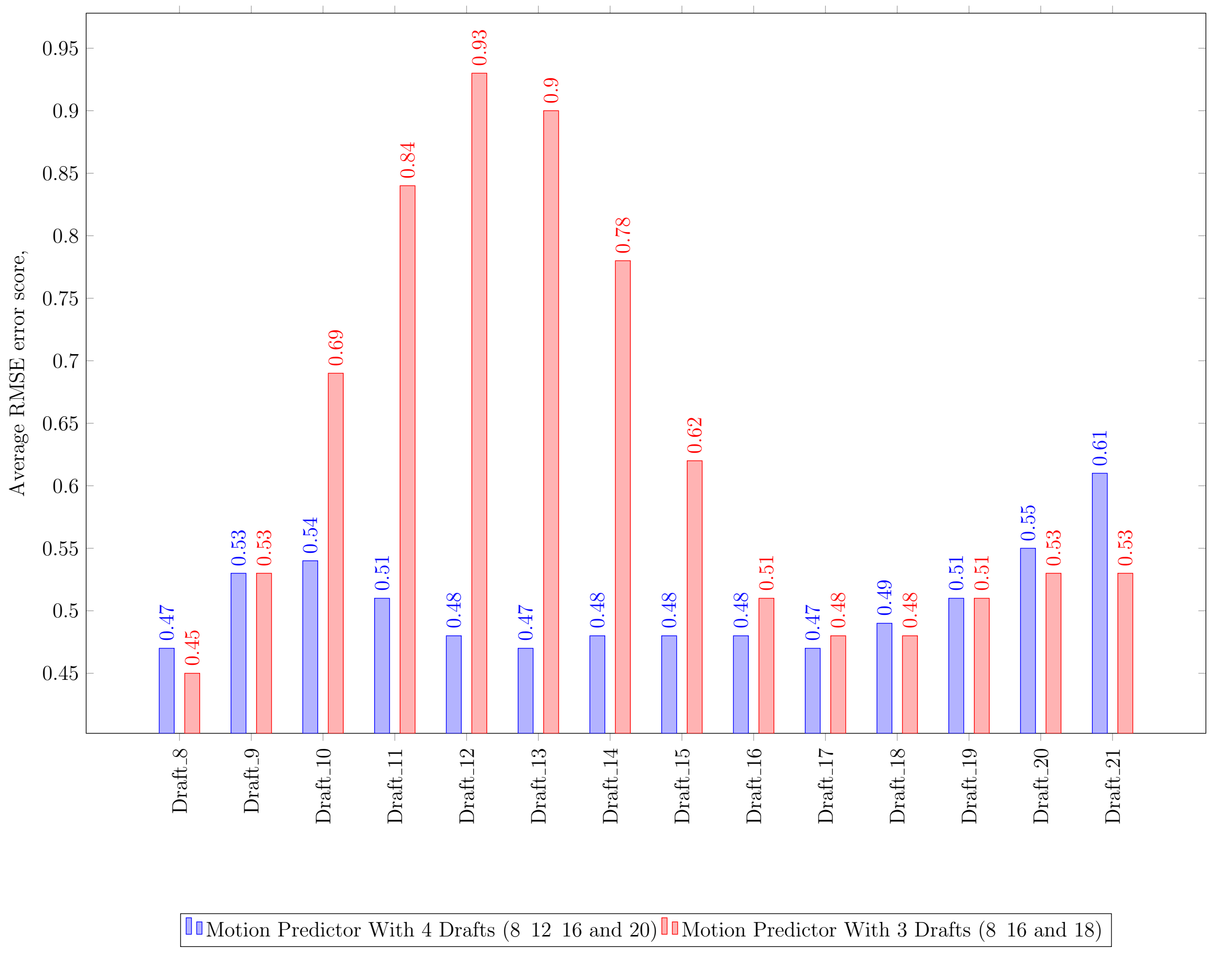


Figure 46: Comparing motion predictors trained on 3 and 4 drafts. The surge average RMSE error scores for the model trained with 3 drafts are illustrated in red, and for the model trained with 4 drafts they are in blue.

As it can be seen in Figure 46, the model trained with 4 drafts (in blue) had better scores when compared to the model trained with 3 drafts (in red) overall. Thus, it was asserted that the minimum number of drafts needed is 4, as this choice achieved an adequate performance for all drafts. Therefore, the MLP predictor trained with 4 drafts was chosen to be used during the next steps in this work.

Figure 47 shows a box plot of the RMSE for the surge motion prediction of 300 platform motions with intact and broken mooring lines, encompassing every draft from 8 to 21 m. The model used to generate the predictions was trained with drafts 8, 12, 16 and 20, as described previously. The figure shows that the trained model was able to generalize for drafts not included during the training phase, and it was also able to distinguish between motions with and without breakage for the different drafts, since there is a large difference between the RMSEs for motion with intact and broken lines. For motions with intact mooring lines, the error was consistently smaller than 3, this is demarcated by the area enclosed in a red rectangle in the figure; moreover, for motions with breakage the error score was regularly greater than 3. Breaks were made in lines L1, L9, L12 and L18 for each of the drafts. Therefore, there is a clear separation between scenarios with and without line failure.

In the figure, it also shows that as the platform becomes heavier, the range between the minimum and maximum values of each boxplot increases. This can be seen when comparing the boxplot of draft 8 with breakage in line 1, annotated as d8-bl01 in the legend, with the boxplot of draft 21 with breakage in line 1, denoted as d21-bl01.

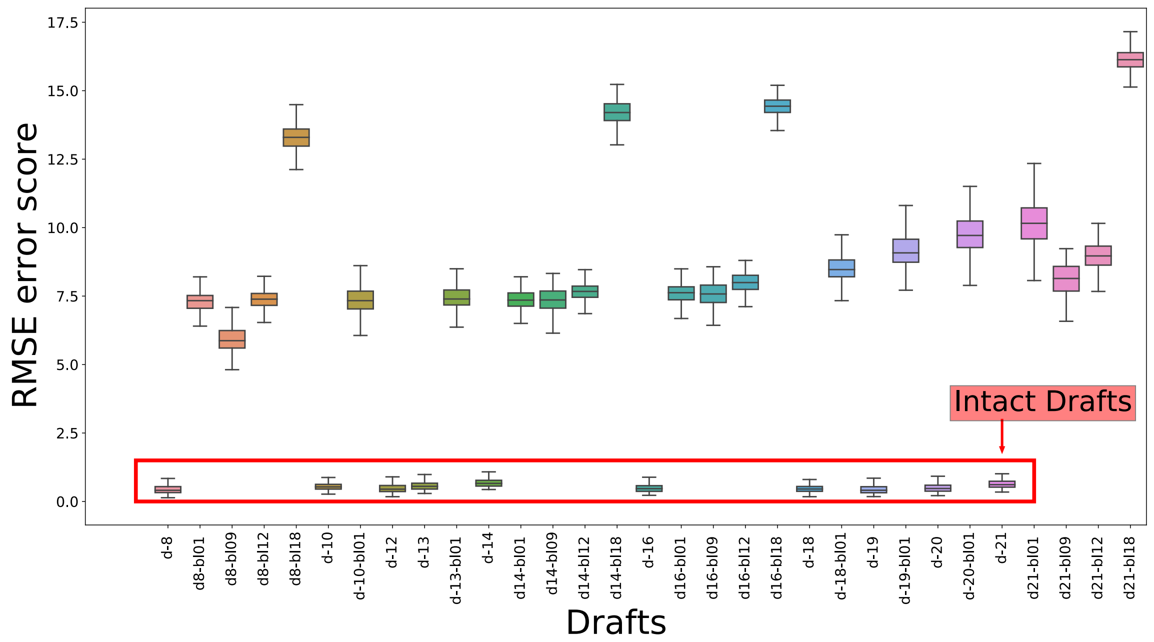


Figure 47: Box plot of surge RMSE of 300 platform motions with intact and broken mooring lines, evaluated for the model trained with drafts 8, 12, 16 & 20m. The motions without breakage are enclosed in the red rectangle and the motions with breakage (L1, L9, L12 and L18 for each draft) are above it.

Figures 48 and 49 shows a box plot of the mean error and median for the surge of

300 platform motions with intact and broken mooring lines respectively. In these figures there is also a good separation between the scenarios with and without line failure. It can also be observed that line failures from different lines lead the error score to be positive or negative. For motions with intact mooring lines the errors are between ± 3, this can be seen enclosed in a red rectangle in the figure. For motions with breakage in line 1 and 18, for all the drafts used during the test, the error scores are negative, while for motions with breakage in line 9 and 12 the error scores are positive. This can be attributed to the physical configuration of the platform mooring lines.

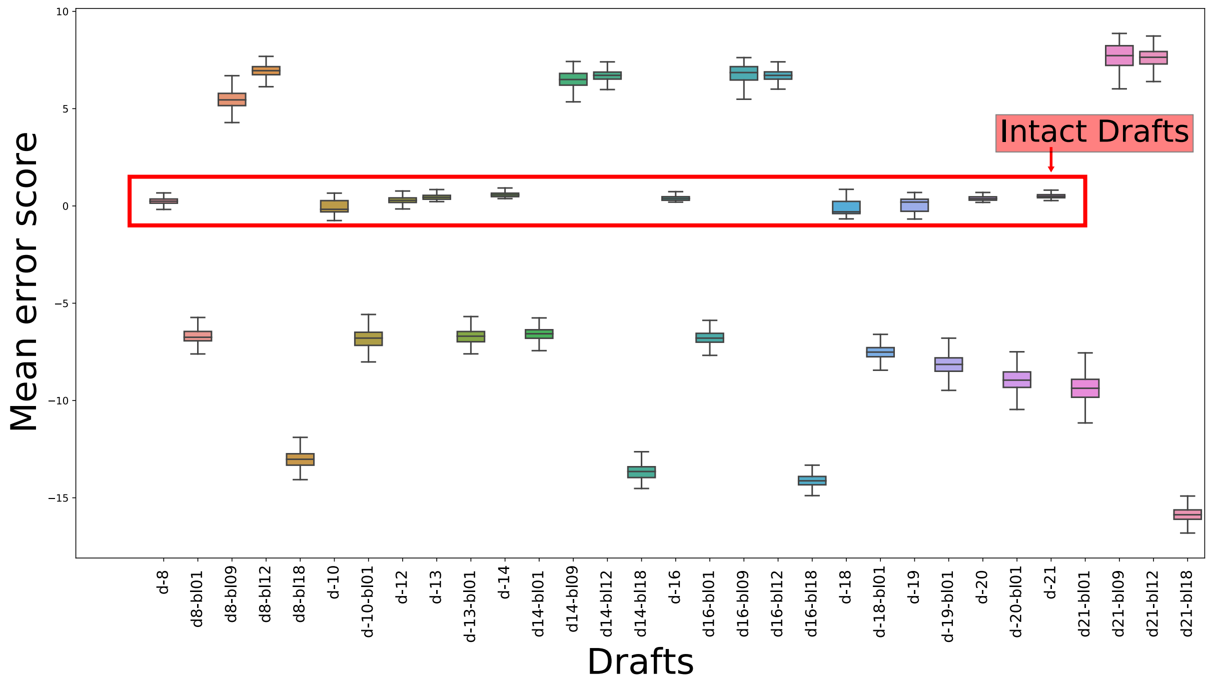


Figure 48: Box plot of mean error for the surge of 300 platform motions with intact and broken mooring lines for the trained predictor. The motions without breakage are enclosed in the red rectangle and motions with breakage lead to error scores with positive or negative values. Each draft is grouped by a distinct color.

In order to further examine the model efficiency, the motion predictions on drafts 8, 12, 16 and 20 for the environmental condition with wave height of 1.57 meters, swell height of 0 meters and wind velocity of 5.79 meters per second (see Table 23), with intact mooring lines, was represented.

Table 23: Analyzed Environmental conditions

Index	Current Velocity	Current Direction	Wind Velocity	Wind Direction	Wave Height	Wave Period	Wave Direction	Swell Height	Swell Period	Swell Direction
6489	0.12	172.65	5.79	152	1.57	8.01	167.5	0	0	0

In the figures representing the predictions, the orange line represents the predictor modules prediction, while the blue line is the simulated platform motion. Figure 50 shows

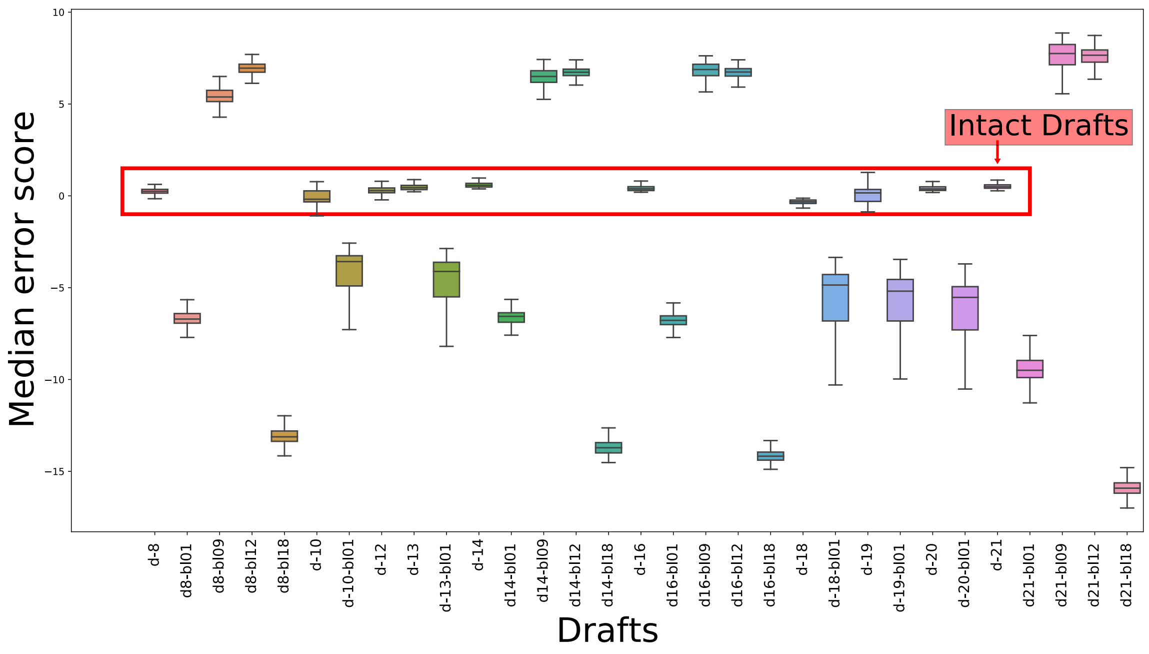


Figure 49: Box plot of Median error for the surge of 300 platform motions with intact and broken mooring lines for the trained predictor. The motions without breakage are enclosed in the red rectangle and motions with breakage lead to error scores with positive or negative values. Each draft is grouped by a distinct color.

predictors' prediction on draft 8, Figure 51 on draft 12, Figure 52a on draft 16 and Figure 52b on draft 20. The result shows it was able to predict the frequency and amplitude of the platform motions, serving as a good predictor of platform motion.

For motions with mooring line breakage in line 1, MLPs predictions on draft 8, 12, 16 and 20 are shown in Figure 53, Figure 54, Figure 55a and Figure 55b respectively. In these figures, the orange line is the MLP prediction, and the blue line is the simulated platform motion; furthermore, the red line shows when breakage occurred, at time step 1400s. This result shows that predictor was able to predict the platform motion closely before mooring line breakage occurs at time step 1400s, after which it was not able to predict the motions properly. As it can be seen in these figures, an offset occurs after breakage; therefore, NeMo can use this information to detect when mooring line breakage happens.

The error scores for different draft settings of the same environmental condition with intact and broken mooring line L1 are shown in Table 24. It is noticeable that the RMSE of the surge, sway and yaw motions for all drafts with intact mooring lines are one order of magnitude smaller than those with breakage. This behaviour also holds true for Mean and Median error scores calculated for these motions, as shown in Table 25 and 26: the error scores for all drafts with intact mooring lines are one order of magnitude smaller than

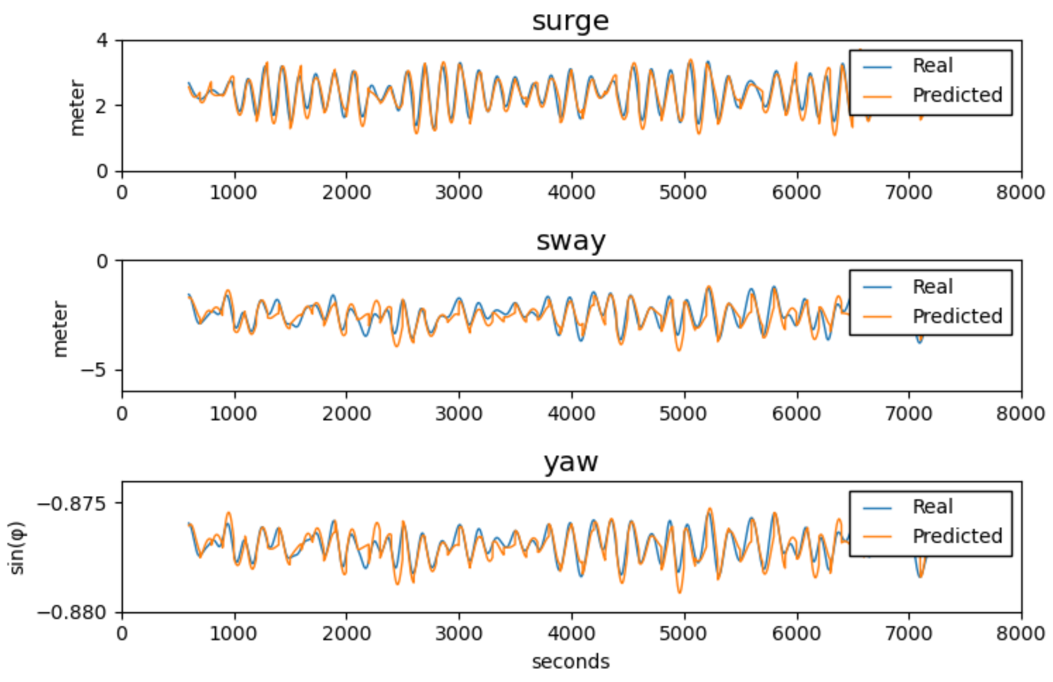


Figure 50: Illustration of the MLP prediction on draft 8 for the environmental condition with index 6489, with intact mooring lines.

Note: the first 3600 seconds (transient motions) have been removed, and thus the prediction is done on the remaining 7200 seconds for all subsequent tests.

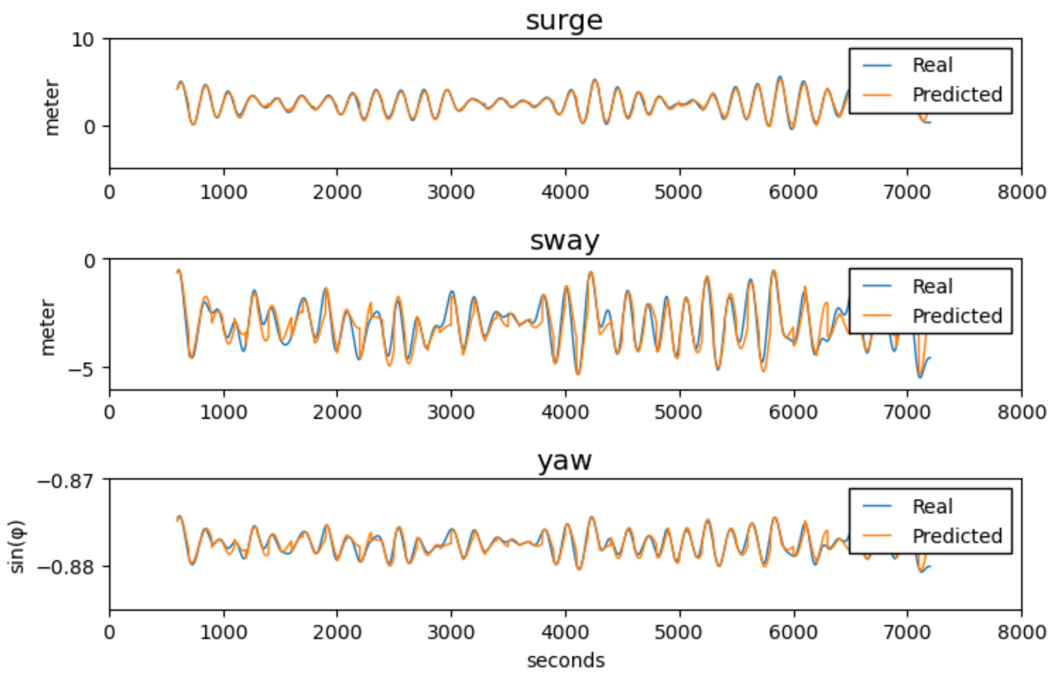
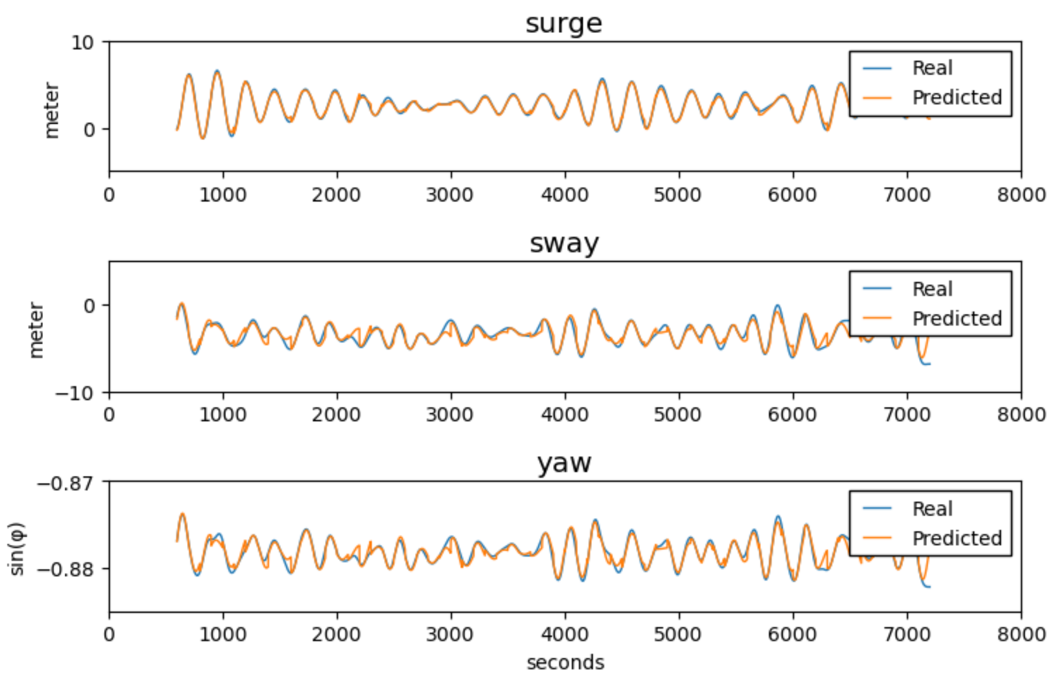
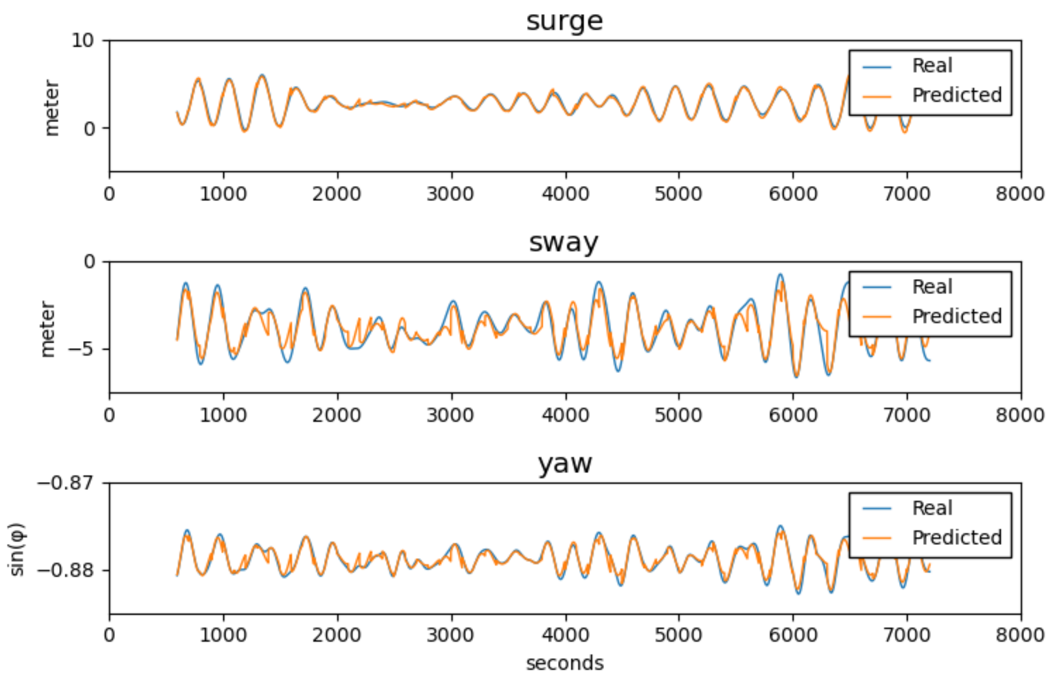


Figure 51: Illustration of the MLP prediction on draft 12 for the environmental condition with index 6489, with intact mooring lines.



(a) Draft 16



(b) Draft 20

Figure 52: Illustration of the MLP prediction for the environmental condition with index 6489, with intact mooring lines, for draft 16 in Figure (52a) and for draft 20 in Figure (52b).

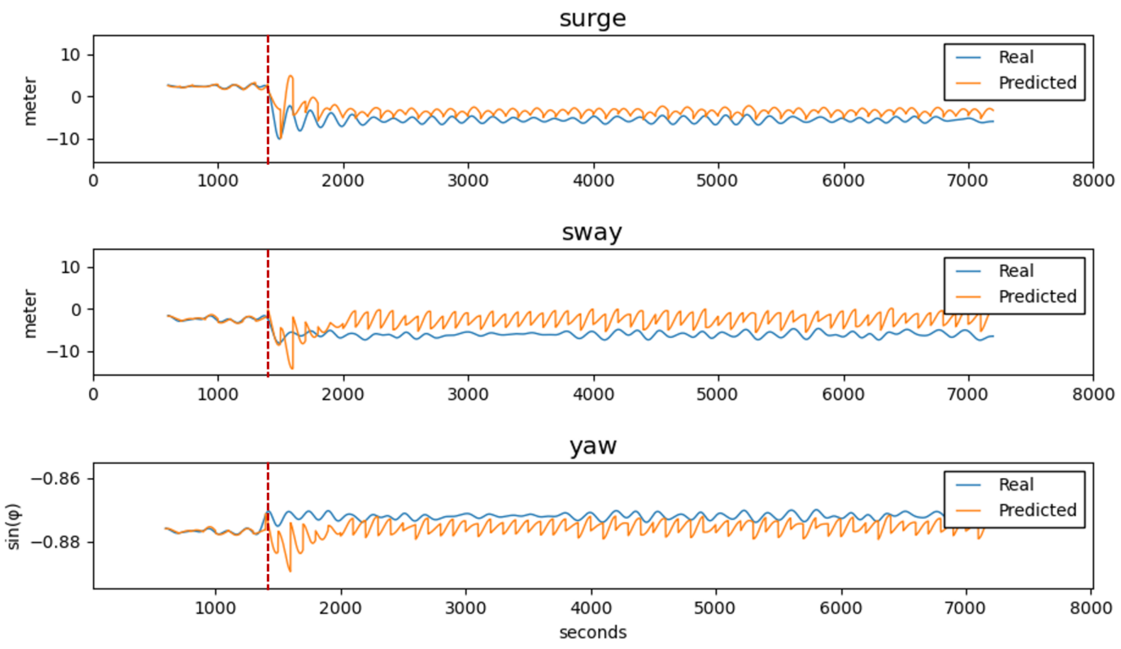


Figure 53: Illustration of the MLP prediction on draft 8 for the environmental condition with index 6489, with breakage in mooring line 1.

Note: The first 3600 seconds (transient motions) have been removed, resulting in a change in the timing of the line breakage from occurring at time step 5000 seconds to occurring at 1400 seconds for all tests with line breakage.

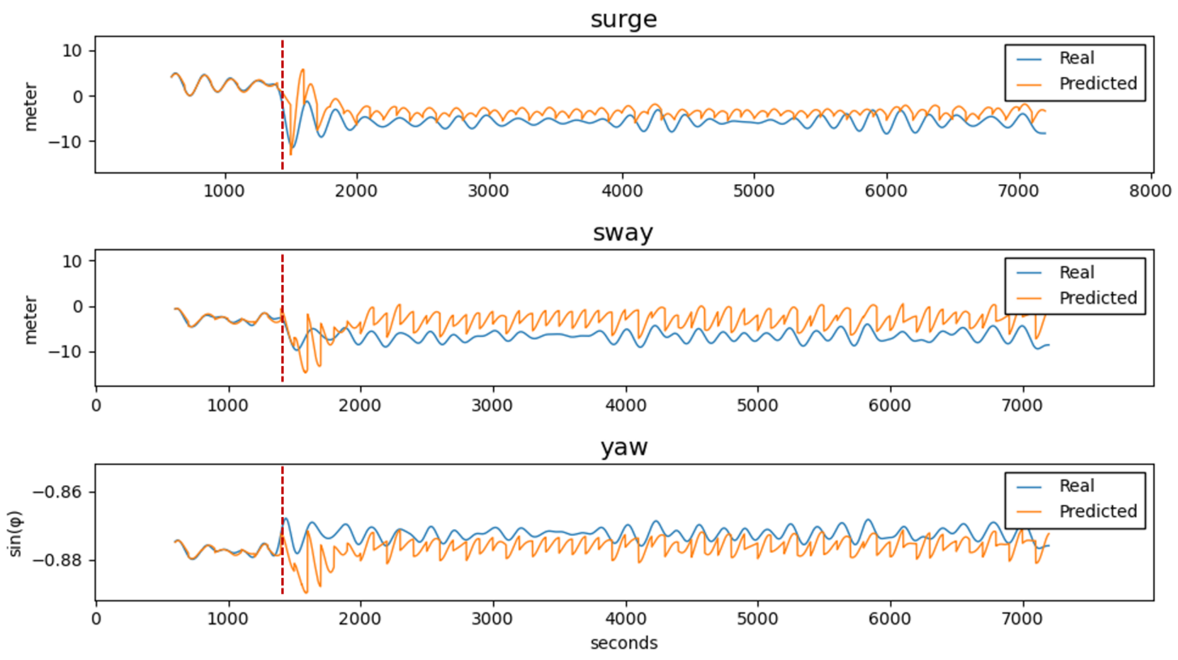
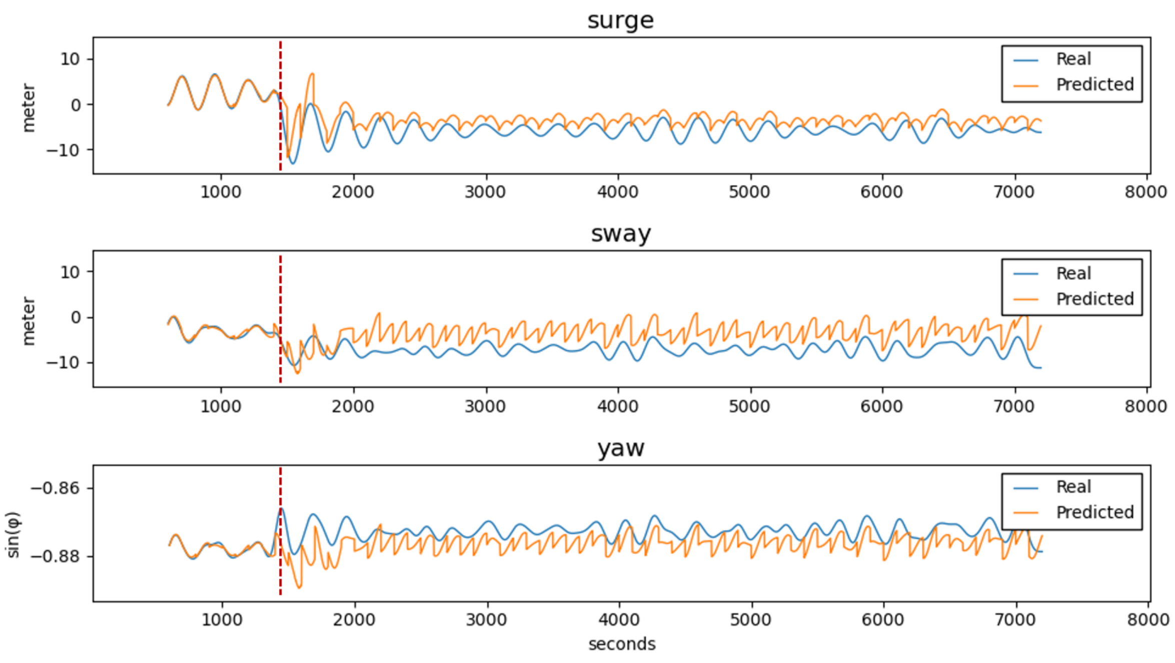
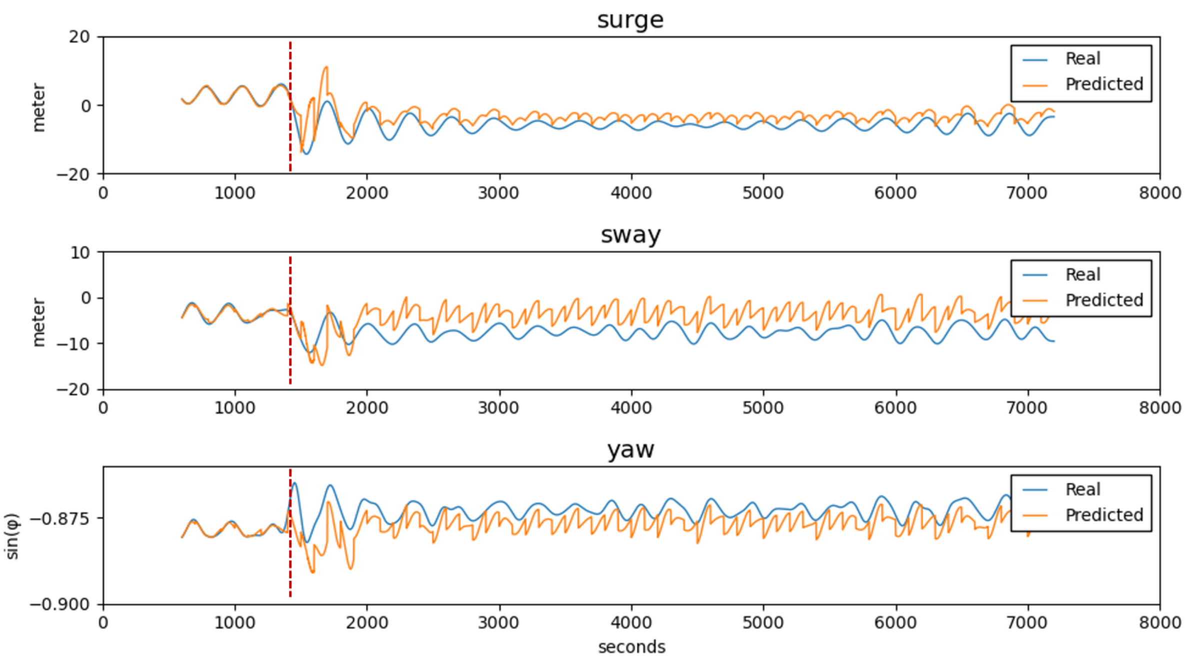


Figure 54: Illustration of the MLP prediction on draft 12 for the environmental condition with index 6489, with breakage in mooring line 1.

those with breakage. Therefore, it is clear that there is always a significant difference between error scores in situations with intact and compromised mooring lines under a



(a) Draft 16 with breakage in line 1



(b) Draft 20 with breakage in line 1

Figure 55: Illustration of the MLP prediction for the environmental condition with index 6489, with breakage in mooring line 1 for draft 16 in Figure (55a) and for draft 20 in Figure (55b).

certain environmental condition.

Table 24: RMSE of the NeMo MLP predictor error scores for test environment index 6489 on different drafts, with and without Broken Line (BL).

Draft	Surge	Sway	Yaw
Draft 8	0.39	0.67	7.48e-4
Draft 12	0.41	0.85	8.26e-4
Draft 16	0.34	0.89	8.51e-4
Draft 20	0.59	0.99	8.57e-4
Draft 8 with BL 01	5.07	4.65	1.03e-2
Draft 12 with BL 01	6.32	4.95	1.14e-2
Draft 16 with BL 01	5.96	5.20	9.85e-3
Draft 20 with BL 01	8.69	5.35	1.10e-2

Table 25: Mean error scores of the NeMo MLP predictor for test environment index 6489 on different drafts, with and without Broken Line (BL).

Draft	surge	sway	yaw
Draft 8	0.26	-0.54	-5.90e-4
Draft 12	0.19	-0.57	-5.67e-1
Draft 16	0.17	0.67	5.64e-4
Draft 20	0.26	0.79	6.23e-4
Draft 8 with BL 01	-4.39	-4.42	9.22e-3
Draft 12 with BL 01	-5.22	-4.60	1.02e-2
Draft 16 with BL 01	-4.91	-4.85	9.10e-3
Draft 20 with BL 01	-7.22	-5.01	1.06e-2

Table 26: Median error scores of the NeMo MLP predictor for test environment index 6489 on different drafts, with and without Broken Line (BL).

Draft	surge	sway	yaw
Draft 8	0.23	-0.56	-6.86e-4
Draft 12	0.22	-0.47	-5.72e-4
Draft 16	0.17	0.56	3.44e-4
Draft 20	0.31	0.62	3.48e-4
Draft 8 with BL 01	-4.96	-4.60	9.37e-3
Draft 12 with BL 01	-5.85	-4.72	1.05e-2
Draft 16 with BL 01	-5.34	-4.85	9.50e-3
Draft 20 with BL 01	-8.61	-5.13	1.10e-2

5.2.3 Discussion

In chapter 4 we proposed to build the NeMo system, hypothesizing that the change in platform motions can be a good indicator of when mooring line failure occurs. Results presented in Section 5.2.2 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 the predictor model was only trained to respond well to platform motions with all mooring lines intact, this irregular motion causes a difference between the simulated and predicted motions, leading to a significantly higher error score in case of line failure.

The box plot in Figure 47, generated based on the RMSE error scores, shows a clear separation between the cases with intact and broken mooring lines. As this error depends on the quadratic difference between the conditions, it is not possible to assess how different line breakages influence this error to be positive or negative. This issue no longer occurs when using the mean errors, as shown in Figure 48. For this metric, the error score sign changes depending on which line is broken. This test with the predictor trained on 3000 random environmental conditions from 4 drafts and tested on every draft shows that the model was able to generalize for all drafts and detect the difference in motion between situations with compromised and intact mooring lines.

A test sample on the MLP prediction accuracy in Figures 50, 51, 52a and 52b showed the model was capable of predicting the platform motions adequately for different draft settings in a specific set of test data; in Figures 53, 54, 55a and 55b NeMo is able to detect when line breakage occurs, as it exhibits an offset between its predictions and the simulated platform motions. After mooring line breakage, the model was unable to predict the platform motions properly because the platform motions with a compromised mooring system are unknown to the model.

The predictor error scores in Tables 24, 25 and, 26 show a clear separation between the cases with intact and broken mooring lines, since the error score calculated for motions with breakage are approximately 10 times bigger than those without breakage.

It can be concluded that the mean and median errors can be used for multi-group mooring line failure detection, while the RMSE error can be used for binary mooring line breakage detection. Therefore, all these metrics will be used as input to the classifier module of the NeMo system, since all of them provide useful information.

5.3 Line breakage classifier module

The predictor described in Section 5.2 is trained to predict platform motion for different environmental conditions and based on past motions without line breakage. It is then used to predict these motions for any mooring system state (with or without broken lines). Since training was performed only on motions without breakage, a significant difference between predicted and simulated motions is expected, which should allow for breakage detection and classification.

A classifier is used to determine whether a line breakage occurred in a particular simulated platform motion, and, if failure is detected, to indicate in which line group it occurred. This model takes as input the error scores of the MLP predictors, and outputs breakage probability for each line group, as well as probability of no line breakage, as shown in the Figure 56. A multinomial logistic regression (MLR) model which uses the one vs rest approach was chosen to implement the aforementioned classifier.

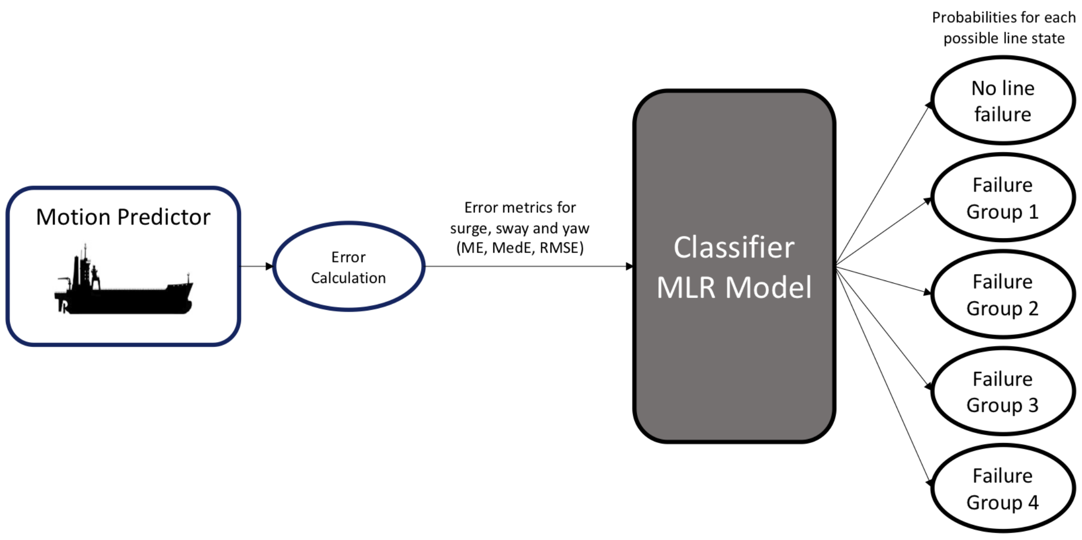


Figure 56: A classifier is used to define whether a platform motion indicates a line failure, it also indicates the group where breakage happened, by generating probabilities for each possible line state.

The classifier input consists of the error scores (RMSE, mean and median), for each of the three horizontal motions. The desired output consists of the class the platform motion belongs to, either with no line failure present (referenced as class **No_B**) or with failure in Group **1**, Group **2**, Group **3** or Group **4** (see Table 22).

5.3.1 Prediction Error Exploratory Analysis

The minimum (Min) and maximum (Max) values of the surge RMSE were explored under all environmental conditions for all drafts to better understand the error score correlation with both the draft and line failure groups. Figure 57 shows these statistics for all motions in group **No_B**, which highlight the proximity between the smallest maximum RMSE – 3.26, obtained for draft 9 – and the greatest maximum RMSE – 4.82, obtained for draft 21 – differs by 1.56.

Figure 58 illustrates the Min and Max RMSE for motions in Group **1** (with breakage

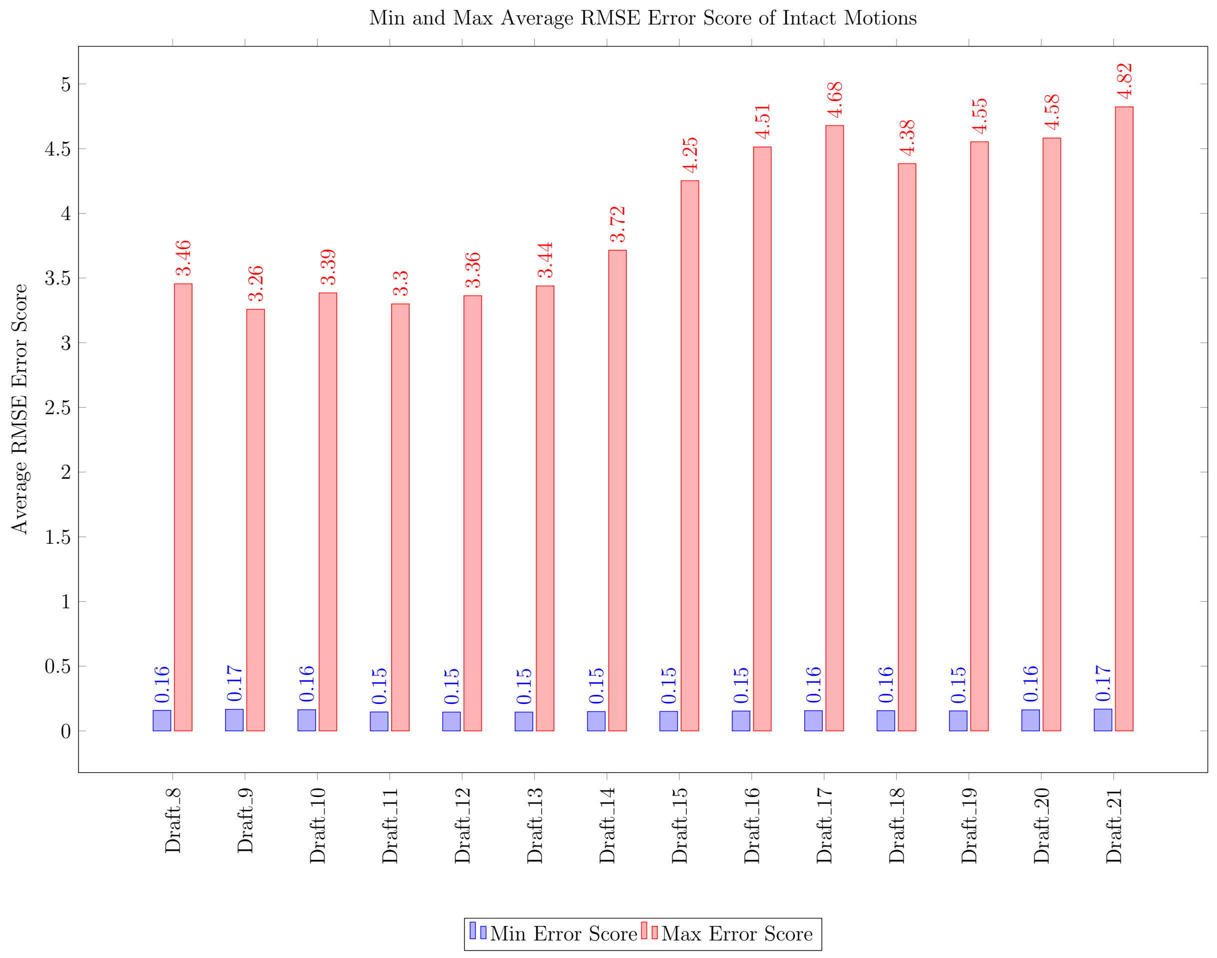


Figure 57: Min and Max average RMSE scores of platform motion surge for each draft, with lines intact. The blue bar is for the Min values of each draft and the red bar is the Max values for the same draft.

in line 1) across all simulated drafts. These error scores exhibit a positive correlation with draft, increasing as the platform becomes heavier. In this case, the difference between the smallest Max – 12.01, obtained for draft 8m – and the biggest Max – 17.32, obtained for draft 21m – was 5.31, well above the 1.56 difference reported for motions without breakage. This difference can be attributed to the load of platform with it being light weighted in draft 8 and heavy weighted in draft 21m.

This pattern of the error score increasing as the platform becomes heavier holds true for the Max and Min platform motion error scores obtained for cases with breakage in lines 12 and 18 as well. Figure 59 shows a complete picture of the surge Max RMSE for each draft with breakage in each of the selected lines (represented by bars), along with the Max RMSE obtained for the same draft with intact mooring lines (at the top of each box), allowing for a better comparison of the Max score obtained for cases with and without mooring line breakage. The clear separation between error scores for cases

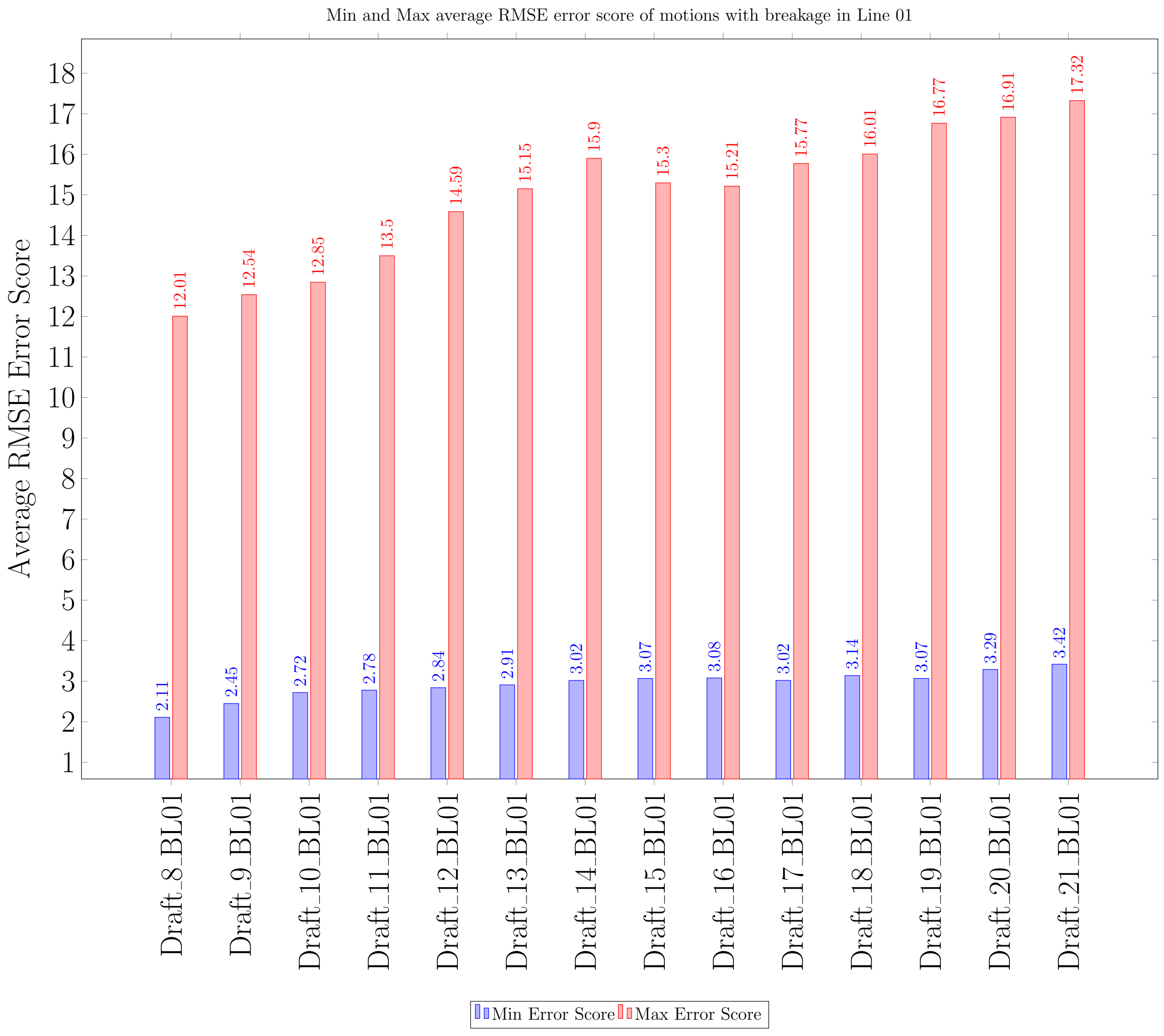


Figure 58: Average RMSE surge Min and Max scores for each draft with breakage in line 01. The blue bar are for the Min values of each draft and the red bar are the Max values of the same draft.

with and without failure suggests the classifier will be able to differentiate between these conditions.

5.3.2 Train and Test Datasets

Dynasim was used to simulate a total of 490,000 platform motions (7000 for each draft), with and without breakage as listed in Table 27. In the simulations with breakage, the line failure occurred in the time 5000 seconds.

To train and test the MLR classifier, a subset of all these simulated platform motions was used. Using the `train_test_split` function from the Scikit-learn (PEDREGOSA et al., 2011) Python library, the complete dataset (see Table 27) was stratified such that

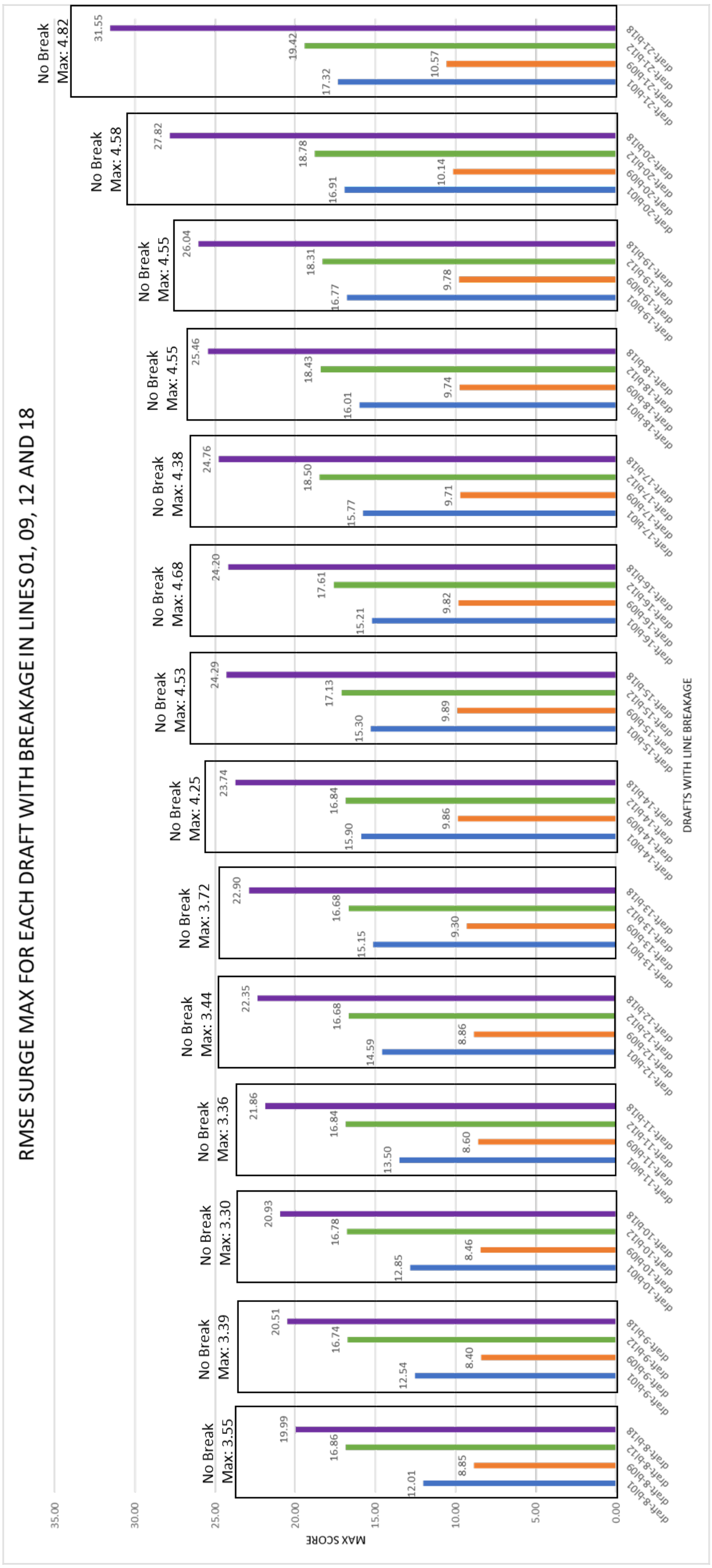


Figure 59: Surge RMSE Max score for each draft with breakage in lines 01, 09, 12, and 18. Blue bars are for breakage in line 01, orange bars for breakage in line 09, green bars for breakage in line 12, and purple bars for breakage in line 18, for each draft (8m – 21m). The Max error score for each draft without breakage is presented at the top of each box representing each draft.

80% of the platform motions, resulting in 392,000 motions (see Table 28), were used for training and the remaining 20%, resulting in 98,000 motions (see Table 29), for testing the classifier. A detailed explanation of how the training and testing datasets were obtained using the stratified sampling method can be found in Appendix C. The stratified sampling method adopted aims to randomly sample equal or close to equal number of samples from each class of a given population. MLP prediction error metrics for each of the selected environmental conditions, along with mooring status (i.e without breakage or with breakage in one of the four lines selected in each group) composed the input-output pairs for both training and test sets.

Table 27: All simulated platform motions from Dynasim.

Draft	Number of platform motions	Line Status	Group
Drafts 8 – 21	98,000 (7000/draft)	No Failure	No_B
Drafts 8 – 21	98,000 (7000/draft)	Failure L1	1
Drafts 8 – 21	98,000 (7000/draft)	Failure L9	2
Drafts 8 – 21	98,000 (7000/draft)	Failure L12	3
Drafts 8 – 21	98,000 (7000/draft)	Failure L18	4
Total number of platform motions	490,000		

Table 28: Training data for classification

Group	Drafts	No of Samples from each Draft	Total Sample for each Group
No_B	8 - 21	5,600	14 * 5,600 = 78,400
1	8 - 21	5,600	14 * 5,600 = 78,400
2	8 - 21	5,600	14 * 5,600 = 78,400
3	8 - 21	5,600	14 * 5,600 = 78,400
4	8 - 21	5,600	14 * 5,600 = 78,400
Total no. of platform motions	392,000		

Table 29: Testing data for classification

Group	Drafts	Total Sample For Each Group
No_B	8 - 21	19,600 (1400/draft)
1	8 - 21	19,600 (1400/draft)
2	8 - 21	19,600 (1400/draft)
3	8 - 21	19,600 (1400/draft)
4	8 - 21	19,600 (1400/draft)
Total no. of platform motions	98,000	

5.3.3 Experiments and Results

Results of the trained MLR classifier, represented by a confusion matrix, is show in Table 30. The confusion matrix illustrates model accuracy and it should be interpreted such that numbers in the main diagonal of the matrix represent correct classification, and numbers in the vertical axis of other cells represent incorrect classification.

Table 30: MLR Confusion Matrix

Actual Label / ***Predicted Label***	**1**	**2**	**3**	**4**	**No_B**
1	19589	2	0	8	1
2	3	19597	0	0	0
3	0	0	19600	0	0
4	7	0	0	19593	0
No_B	8	2	0	0	19590

In one instance, when the classifier predicted the simulation to belong to Group **No_B**, it misclassified a simulation which actually belonged to Group **1**. For predictions to Group **4**, it misclassified 8 simulations which belonged to Group **1**. For predictions to Group **3**, there were no misclassifications. For prediction to Group **2**, it misclassified 4 platform simulations: 2 belonging to Group **No_B** and 2 to Group **1**. Finally, for predictions to Group **1**, the classifier misclassified 18 simulations: 8 belonging to Group **No_B**, 7 to Group **4** and 3 to Group **2**.

The MLR classifier misclassified only 31 out of 98000 motions (see Table 31). Table 32 shows the total count of misclassified motion and it shows the classifier found classifying motions of group **1** most difficult.

Table 31: Error metrics of the MLR classifier

	Class No_B	Class 1	Class 2	Class 3	class 4
Accuracy	99.99	99.99	99.99	1	99.99
Precision	1	1	1	1	1
Recall	1	1	1	1	1
F1-score	1	1	1	1	1

Table 32: Count of Total Number of Misclassified Platform Motion Group

Misclassified Group	Count
1	18
2	4
3	0
4	8
No_B	1
Total	31

A sample of the misclassified platform motion drafts and mooring line status, together with the probabilities assigned by the MLR classifier, and the final prediction for each motion, along with the actual group the motions belong to are shown in Table 33. The complete misclassified motions can be found in Table 38 in Appendix C

It is worth noting that the classifier is often in doubt between the different groups, demonstrating this doubt in the form of the probabilities associated with each group. For example, in Table 33, the first misclassified condition had a predicted probability of 0.589 for it to belong to Group **1**, when in reality it belonged to Group **No_B**. However, note that the classifier also believes that Group **No_B** may be the correct group, but demonstrates less certainty in this fact. The presentation of the classification results were quite conservative, as it was decided to offer a sharp answer to the classes, despite the doubt shown by the classifier. Classification errors would be reduced if it were allowed to provide answers, for example for case No.1 of the Table 33, such as: "There is a possibility that there was a break in line 1, but there is also a good possibility that there was no break."

Table 33: A sample of MLR misclassified platform motions.

	Assigned group probability								
No.	G.1	G.2	G.3	G.4	No_B	Sum	Predicted group	Actual group	Draft
1	0.589	0.008	0.000	0.000	0.403	1	G.1	No_B	d-9
2	0.362	0.485	0.000	0.000	0.153	1	G.2	No_B	d-14
3	0.823	0.000	0.000	0.000	0.177	1	G.1	No_B	d-15
4	0.544	0.000	0.000	0.000	0.456	1	G.1	No_B	d-19
5	0.586	0.000	0.000	0.414	0.000	1	G.1	G.4	d-8-bl18
6	0.715	0.285	0.000	0.000	0.000	1	G.1	G.2	d-10-bl09
7	0.783	0.000	0.000	0.217	0.000	1	G.1	G.4	d-13-bl18
8	0.523	0.477	0.000	0.000	0.000	1	G.1	G.2	d-12-bl09
9	0.159	0.000	0.000	0.841	0.000	1	G.4	G.1	d-17-bl01
10	0.482	0.000	0.000	0.518	0.000	1	G.4	G.1	d-21-bl01

Analysis of the measured environmental conditions recorded for the 31 misclassified motions shows the classified platform motions all come from random period of time. A sample of the conditions is shown in Table 34, the complete measured conditions can be found in Table 39 in Appendix C.1. It was hypothesized that the wrong classifications result from erroneous motion prediction by the MLP model for these environmental conditions.

5.3.4 Discussion

The implemented classifier, taking the error scores of the predictor MLP as input, had an accuracy of nearly 100%, misclassifying 31 out of $980,000$ platform motions. Analysis of the misclassified motions revealed the classifier found classifying motions belonging to

Table 34: Sample of misclassified environmental conditions measured.

Current Velocity	Current Direction	Wind Velocity	Wind Direction	Wave Height	Wave Period	Wave Direction	Swell Height	Swell Period	Swell Direction	Date	Time
0.18	321.17	6.90	87.20	1.73	11.51	179.80	1.11	5.05	97.40	04/08/2006	09:00:00
0.48	228.11	5.01	120.60	2.00	9.63	187.50	0.58	3.77	133.20	25/08/2004	06:00:00
0.37	216.58	3.90	60.50	2.07	15.99	129.60	0.36	3.30	66.00	10/06/2007	09:00:00
0.27	163.09	9.67	20.40	2.34	13.07	191.30	1.34	5.98	39.30	18/01/2007	03:00:00
0.26	216.56	6.30	107.40	2.72	14.40	177.10	1.00	4.56	112.00	02/05/2008	15:00:00
0.35	219.37	5.80	64.20	2.09	10.44	175.50	0.73	4.51	74.80	02/11/2008	12:00:00
0.43	197.07	9.36	44.0	1.44	5.44	54.30	0.42	6.14	136.40	26/11/2004	21:00:00
0.53	172.44	9.41	27.60	1.72	6.63	28.80	0.59	8.96	143.10	16/11/2009	00:00:00
0.64	173.91	10.49	29.90	2.51	7.10	30.00	1.93	12.95	195.70	18/10/2005	00:00:00
0.45	194.00	9.63	15.60	1.57	5.71	34.40	0.93	8.00	95.10	01/12/2005	06:00:00
0.48	175.25	12.330	18.40	2.24	6.39	24.40	0.67	8.55	114.10	15/01/2009	09:00:00

Group **1** most difficult.

This is hypothesized to be caused by the fact that the Max surge RMSE of draft 13m to 21m without breakage (see Figure 57) were bigger than the Min surge RMSE of draft 21m with breakage in line 1 (see Figure 58), which could lead to misclassification of these motions.

Analysis of the measured environmental conditions recorded for the 31 misclassified motions showed they come from random periods of time and environmental cs.

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 and the structural integrity of the platform. The classifier had some false-negative predictions, which means it still needs improvements.

However, it is worth mentioning that it was chosen to present classification results in a very conservative way, simply adopting as the estimated class the one with the highest probability in the MLR output. A more careful look could also be given to the output that presents the second highest probability, indicating more than one possible decision and showing the degree of certainty in the class provided by the classifier. With this consideration, the number of false-negatives could be further relegated.

6 CONCLUSION AND FUTURE WORK

In this thesis, we proposed to build a machine learning (ML) model, capable of identifying mooring line breakage. Our proposed system which we refer to as NeMo, was developed to detect mooring line failure of FPSO-P50 with 14 different draft levels (from 8 to 21 meters) and also make a multi-class mooring line failure classification for five mooring line groupings.

Our implemented NeMo system divides the mooring failure identification problem into sub-modules, by first training an MLP network to predict future motions of FPSO-P50, using previous motions. The difference between the MLP prediction and the actual motion is then used as input to a classifier module which classifies and indicates the probability of the platform mooring system status, that is broken or not broken. For cases with line break NeMo indicates which line group was broken, showing the break probabilities of each group.

A hypothesis was made that the platform motion changes its intensity as well as its motion frequency after a line failure occurs. Since NeMo was only trained to respond well to platform motions with all mooring lines intact, this irregular motion can then be seen as a difference between the simulated and predicted motions, leading to a significantly higher error score in case of line failure.

We summarize below our main findings and contributions :

1. The implemented NeMo system was able to predict the platform motions for the test data given to it under normal conditions, where the platform mooring lines were intact for 14 different draft levels, and for test data where there was a failure in the mooring line. The predictor module was able to detect the occurrence of failure by displaying an offset between the prediction of the model and the simulated platform motions, for all motions and all 14 platform drafts.
2. An ablation study was carried out regarding the impact of network entry, with the objective of finding the minimum number of platform motion variables necessary to

have the desired accuracy, and also to determine the necessary number of different drafts that should be used to train networks to detect line breaks. Three models were investigated and the average RMSE error of 1000 test motions were used to compare the performance of these models. **Model A** used all the 6DoF motion as input to the network when training, **Model B** used only 3DoF and **Model C** used surge, sway, and a yaw constant at 210° as input when training. **Model B** was the best model out of the three models and it revealed that using only the horizontal motions as input was sufficient for mooring line breakage identification. The error difference between the three models were minimal which showed even with only the surge, sway and a constant yaw motion, it is possible to train a network to detect mooring line failure. Furthermore, it was found the **Model A** which used all the 6DoF motion as input to the model had the longest training time.

3. A study to find the optimal number of neurons for each hidden layer of the predictor module of the NeMo system was investigated. Optimization method such as Bayesian Optimization, state-of-the-art in neural network hyperparameter optimization (FEURER; HUTTER, 2019); and Random Search were used. The manually determined number of neurons in the hidden architecture developed was found to be the best, thus validating the selected number of neurons in each hidden layer to be optimal.

4. A study to find the minimum number of platform draft motion combinations that would be sufficient for the NeMo motion predictor module to learn the complex platform motions for all set of draft, that is, drafts from $8m$ to $21m$ was conducted. The result showed that using only 4 draft combination (8m, 12m, 16m and 20m draft data) was sufficient to train a network that is able to generalise to unseen data.

5. The error scores of the model on all the test data were used to generate box plots of the errors – RMSE, mean and median of the 3 motion variables; sway, surge and yaw. The box plots generated based on the RMSE error scores show a clear separation between the cases with all mooring lines intact for all the draft levels and the cases that had mooring line failures, showing to be a great solution for binary classification.

6. The box plots generated with the mean and median error scores also show a clear separation between the cases with all of its mooring lines intact and the cases with mooring line failures. The result of the plot indicates the mean and median error

scores can be used to detect which group of mooring lines shows a failure. It can be concluded that the mean and median error can be used to detect multi-group mooring line failure.

7. The Multinomial Logistic Regression (MLR) classifier developed used the error scores – RMSE, mean and median of the 3 motion variables; sway, surge and yaw – of all the test data gotten from the NeMo predictor module as input, to classify the mooring line status from platform motions in 14 different drafts. The classification provides the motion's probability of belonging to each of the five groups: No_B (intact mooring lines), G1 (Line 1 broken), G2 (Line 9 broken), G3 (Line 12 broken) and G4 (Line 18 broken). The MLR classifier classified the motion accurately except for 31 motions which it misclassified out of $98,000$ test motion.

The following research questions were posed in Section 1.2. Answers to these questions were derived from the result of the developed and implemented NeMo system:

RQ1 Can a failure in the mooring system of an offshore platform be detected using only the data regarding the motion and load of this platform?

Ans1 Yes, the various platform motions generated by the Dynasim simulator, for different environmental and platform loading conditions, revealed a notable difference between the response of a platform with an intact mooring system and with a compromised mooring system.

RQ2 Is a machine learning system based on Multilayer Perceptron Neural Networks capable of predicting the future motion of a vessel having as information only the previous motion and the platform load?

Ans2 Yes, the implemented NeMo system showed that it is possible to predict the future motion of a vessel (in our case, a FPSO-P50) having as information only the previous motion and the platform load. The motion predictor submodule of the NeMo system, composed of a fully connected 3-hidden-layer MLP network, showed that using only 600 seconds of previous platform motions, it is possible to predict 100 seconds of future motion.

RQ3 Would a classifier be able to detect which mooring group a broken line belongs to based solely on the discrepancy between the motion predicted by the MLP and that measured on the platform?

Ans3 Yes, the results of the implemented classifier module showed that a classifier can not only detect which mooring group a broken line belongs to based only on the discrepancy between the motions, but it can also provide probability estimates of which group of lines it belongs to.

As a final observation, it is also conceivable to use this platform motion mechanism for other marine applications, as it is not restricted to the shape of offshore platforms but can be used for any dynamic floating objects with any mooring system.

In conclusion, the objective of this work was achieved.

6.1 Future Work

For the NeMo predictor model, as future work, it would be interesting to carry out an evaluation of state-of-the-art ML models, such as Temporal Fusion Transformers (LIM et al., 2021) and Exponential Smoothing Neural Networks (SMYL; DUDEK; PELKA, 2021), in order to assess whether they are suitable for detecting mooring line break. If appropriate, comparative analyzes with the current MLP model can be performed.

In addition, one should also evaluate end-to-end models that directly classify whether or not a group of mooring lines fails. In this case, the model input would be the observed window of the platform motion series and the model would directly indicate the status of the mooring lines. The InceptionTime model (FAWAZ et al., 2019b) could be an alternative for this study.

For the line breakage MLR classifier, the result showed the classifier made good multi-label classifications, identifying which group of lines that failed belongs to with very few false negatives. As a next step, an ensemble of classifiers should be investigated to increase the confidence of the classification prediction.

Finally, training, testing and evaluating the NeMo predictor on real full scale data would be of great interest to be carried out in future work. The difficulty here lies in obtaining real quality data for training and evaluating models. It is worth noting that real annotated data on the breaking of mooring lines is extremely difficult to obtain, but that would provide an excellent opportunity for the development of new systems based on machine learning and artificial intelligence, which could bring a great increase in performance and safety of existing offshore platforms.

REFERENCES

AKIBA, T.; SANO, S.; YANASE, T.; OHTA, T.; KOYAMA, M. Optuna: A next-generation hyperparameter optimization framework. *CoRR*, abs/1907.10902, 2019. Available from:⟨http://arxiv.org/abs/1907.10902⟩.

AL-QIZWINI, M.; BARJASTEH, I.; AL-QASSAB, H.; HAYDER, R. Deep learning algorithm for autonomous driving using googlenet. In: IEEE. *2017 IEEE Intelligent Vehicles Symposium (IV)*. 2017. p. 89–96. Available from:⟨https://doi.org/10.1109/IVS.2017.7995703⟩.

AYERS, R. R.; O'HEAR, N. Accuracy of polyester subrope damage detection by rov inspection and assessment of remaining rope strength and life. *Final Report to MMS by Stress Engineering Services, Inc. and tension technology Inc., CONTRACT M07PX13203*, 2007.

BERGSTRA, J.; BARDENET, R.; BENGIO, Y.; KÉGL, B. Algorithms for hyper-parameter optimization. In: *Advances in Neural Information Processing Systems*. [s.n.], 2011. p. 1–9. Available from:⟨https://proceedings.neurips.cc/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf⟩.

BISHOP, C. M. *Pattern recognition and machine learning*. [S.l.]: Springer, 2006.

BROCKWELL, P. J.; DAVIS, R. A. *Introduction to time series and forecasting*. [S.l.]: Springer, 2002.

BUTTERWORTH, S. On the theory of filter amplifiers. *Wireless Engineer*, v. 7, n. 6, p. 536–541, 1930.

CHADEGANI, A. A.; SALEHI, H.; YUNUS, M. M.; FARHADI, H.; FOOLADI, M.; FARHADI, M.; EBRAHIM, N. A. A comparison between two main academic literature collections: Web of science and scopus databases. *Asian Social Science*, v. 9, n. 5, p. 18–26, 2013. Available from:⟨https://doi.org/10.5539/ass.v9n5p18⟩.

CHIANG, K.-W.; HOU, H.; NIU, X.; EL-SHEIMY, N. Improving the positioning accuracy of dgps/mems imu integrated systems utilizing cascade de-noising algorithm. p. 809–818, 2004.

CHUNG, M.; KIM, S.; LEE, K.; SHIN, D. H. Detection of damaged mooring line based on deep neural networks. *Ocean Engineering*, Elsevier, v. 209, p. 107522, 2020.

DOMINGOS, P. A few useful things to know about machine learning. *Communications of the ACM*, ACM New York, NY, USA, v. 55, n. 10, p. 78–87, 2012. Available from:⟨https://doi.org/10.1145/2347736.2347755⟩.

ELSKEN, T.; METZEN, J. H.; HUTTER, F. Neural architecture search: A survey. *Journal of Machine Learning Research*, v. 20, n. 55, p. 1–21, 2019. Available from:⟨https://doi.org/10.48550/arXiv.1808.05377⟩.

FAWAZ, H. I.; FORESTIER, G.; WEBER, J.; IDOUMGHAR, L.; MULLER, P.-A. Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, v. 33, n. 4, p. 917–963, 2019. Available from:⟨https://doi.org/10.48550/arXiv.1809.04356⟩.

FAWAZ, H. I.; LUCAS, B.; FORESTIER, G.; PELLETIER, C.; SCHMIDT, D. F.; WEBER, J.; WEBB, G. I.; IDOUMGHAR, L.; MULLER, P.-A.; PETITJEAN, F. Inceptiontime: Finding alexnet for time series classification. *CoRR*, abs/1909.04939, 2019. Available from:⟨http://arxiv.org/abs/1909.04939⟩.

FEURER, M.; HUTTER, F. *Hyperparameter Optimization*. Cham: Springer International Publishing, 2019. 3–33 p. ISBN 978-3-030-05318-5. Available from:⟨https://doi.org/10.1007/978-3-030-05318-5_1⟩.

FONTAINE, E.; KILNER, A.; CARRA, C.; WASHINGTON, D.; MA, K.; PHADKE, A.; LASKOWSKI, D.; KUSINSKI, G. Industry survey of past failures, pre-emptive replacements and reported degradations for mooring systems of floating production units. In: OTC. *Offshore Technology Conference*. 2014. Available from:⟨https://doi.org/10.4043/25273-MS⟩.

FRAZIER, P. I. *A Tutorial on Bayesian Optimization*. 2018. Available from:⟨https://doi.org/10.48550/arXiv.1807.02811⟩.

GAUTHIER, S.; ELLETSON, E. Mooring line monitoring to reduce risk of line failure. In: ONEPETRO. *The 24th International Ocean and Polar Engineering Conference*. [S.l.], 2014.

GLOROT, X.; BORDES, A.; BENGIO, Y. Deep sparse rectifier neural networks. In: JMLR WORKSHOP AND CONFERENCE PROCEEDINGS. *Proceedings of the 14th international conference on artificial intelligence and statistics*. [S.l.], 2011. p. 315–323.

GUMLEY, J. M.; HENRY, M. J.; POTTS, A. E. A novel method for predicting the motion of moored floating bodies. In: ASME. *The 35th International Conference on Ocean, Offshore and Arctic Engineering*. 2016. Available from:⟨https://doi.org/10.1115/OMAE2016-54674⟩.

JAISWAL, V.; RUSKIN, A. Mooring line failure detection using machine learning. In: OTC. *Offshore Technology Conference*. 2019. Available from:⟨https://doi.org/10.4043/29511-MS⟩.

JAMALKIA, A.; ETTEFAGH, M. M.; MOJTAHEDI, A. Damage detection of tlp and spar floating wind turbine using dynamic response of the structure. *Ocean Engineering*, Elsevier, v. 125, p. 191–202, 2016. Available from:⟨https://doi.org/10.1016/j.oceaneng.2016.08.009⟩.

JAMES, G.; WITTEN, D.; HASTIE, T.; ROBERT, T. *An introduction to statistical learning*. [S.l.]: Springer, 2013. v. 112.

KINGMA, D. P.; BA, J. *Adam: A Method for Stochastic Optimization*. 2017.

KWON, D.-S.; JIN, C.; KIM, M.; KOO, W. Mooring-failure monitoring of submerged floating tunnel using deep neural network. *Applied Sciences*, Multidisciplinary Digital Publishing Institute, v. 10, n. 18, p. 6591, 2020. Available from:⟨https://doi.org/10.3390/app10186591⟩.

LEE, K.; CHUNG, M.; KIM, S.; SHIN, D. H. Damage detection of catenary mooring line based on recurrent neural networks. *Ocean Engineering*, Elsevier, v. 227, p. 108898, 2021. Available from:⟨https://www.sciencedirect.com/science/article/pii/S0029801821003334⟩.

LIM, B.; ARIK, S. O.; LOEFF, N.; PFISTER, T. Temporal fusion transformers for interpretable multi-horizon time series forecasting. *ArXiv*, 2021. Available from:⟨https://doi.org/10.48550/arXiv.1912.09363⟩.

LOH, W.-Y. Classification and regression trees. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, Wiley Online Library, v. 1, n. 1, p. 14–23, 2011. Available from:⟨https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/widm.8⟩.

LUNDERVOLD, A. S.; LUNDERVOLD, A. An overview of deep learning in medical imaging focusing on mri. *Zeitschrift für Medizinische Physik*, v. 29, n. 2, p. 102–127, 2019. ISSN 0939-3889. Special Issue: Deep Learning in Medical Physics. Available from:⟨https://www.sciencedirect.com/science/article/pii/S0939388918301181⟩.

MA, K.-T.; LUO, Y.; KWAN, C.-T. T.; WU, Y. *Mooring system engineering for offshore structures*. [S.l.]: Gulf Professional Publishing, 2019.

MA, K.-t.; SHU, H.; SMEDLEY, P.; L'HOSTIS, D.; DUGGAL, A. A historical review on integrity issues of permanent mooring systems. In: OTC OFFSHORE TECHNOLOGY CONFERENCE. *Offshore technology conference*. 2013. Available from:⟨https://doi.org/10.4043/24025-MS⟩.

MACQUEEN, J. Some methods for classification and analysis of multivariate observations. In: OAKLAND, CA, USA. *5th Berkeley Symp. Math. Statist. Probability*. [S.l.], 1967. v. 1, p. 281–297.

MÄRTENS, M.; IZZO, D. Neural network architecture search with differentiable cartesian genetic programming for regression. *CoRR*, abs/1907.01939, 2019. Available from:⟨http://arxiv.org/abs/1907.01939⟩.

MAZAHERI, S.; MESBAHI, E.; DOWNIE, M. J.; INCECIK, A. Seakeeping Analysis of a Turret-Moored FPSO by Using Artificial Neural Networks. In: . [s.n.], 2003. (International Conference on Offshore Mechanics and Arctic Engineering, Volume 1: Offshore Technology; Ocean Space Utilization), p. 275–284. Available from:⟨https://doi.org/10.1115/OMAE2003-37148⟩.

MEYER, E. S.; CHARACKLIS, G. W.; BROWN, C.; MOODY, P. Hedging the financial risk from water scarcity for great lakes shipping. *Water Resources Research*, v. 52, n. 1, p. 227–245, 2016. Available from:⟨https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1002/2015WR017855⟩.

MEYES, R.; LU, M.; PUISEAU, C. W. de; MEISEN, T. *Ablation Studies in Artificial Neural Networks*. 2019.

MITCHELL, T. M. *Machine Learning*. New York: McGraw-Hill, 1997. ISBN 978-0-07-042807-2.

MOUNT, J.; ZUMEL, N. *Practical data science with R*. [S.l.]: Simon and Schuster, 2019.

NISHIMOTO, K.; FUCATU, C. H.; MASETTI, I. Q. Dynasim—A Time Domain Simulator of Anchored FPSO. *Journal of Offshore Mechanics and Arctic Engineering*, v. 124, n. 4, p. 203–211, oct 2002. ISSN 0892-7219. Available from:⟨https://doi.org/10.1115/1.1513176⟩.

PEDREGOSA, F.; VAROQUAUX, G.; GRAMFORT, A.; MICHEL, V.; THIRION, B.; GRISEL, O.; BLONDEL, M.; MüLLER, A.; NOTHMAN, J.; LOUPPE, G.; PRETTENHOFER, P.; WEISS, R.; DUBOURG, V.; VANDERPLAS, J.; PASSOS, A.; COURNAPEAU, D.; BRUCHER, M.; PERROT, M.; DUCHESNAY, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, v. 12, p. 2825–2830, 2011. Available from:⟨https://arxiv.org/abs/1201.0490⟩.

PRISLIN, I.; MAROJU, S. Mooring integrity and machine learning. In: OTC. *Offshore Technology Conference*. 2017. Available from:⟨https://doi.org/10.4043/27866-MS⟩.

QIAO, D.; LI, P.; MA, G.; QI, X.; YAN, J.; NING, D.; LI, B. Realtime prediction of dynamic mooring lines responses with lstm neural network model. *Ocean Engineering*, v. 219, p. 108368, 2021. ISSN 0029-8018. Available from:⟨https://www.sciencedirect.com/science/article/pii/S0029801820312750⟩.

REY, D.; NEUHÄUSER, M. *Wilcoxon-Signed-Rank Test*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. 1658–1659 p. ISBN 978-3-642-04898-2. Available from:⟨https://doi.org/10.1007/978-3-642-04898-2_616⟩.

ROUSSEEUW, P. J. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, v. 20, p. 53–65, 1987. Available from:⟨https://doi.org/10.1016/0377-0427(87)90125-7⟩.

SAAD, A. M.; SCHOPP, F.; BARREIRA, R. A.; SANTOS, I. H. F.; TANNURI, E. A.; GOMI, E. S.; COSTA, A. H. R. Using neural network approaches to detect mooring line failure. *IEEE Access*, IEEE, v. 9, p. 27678–27695, 2021. Available from:⟨https://doi.org/10.1109/ACCESS.2021.3058592⟩.

SAAD, A. M.; SCHOPP, F.; FILHO, A. N. Q.; CUNHA, R. D. S.; SANTOS, I. H. F.; BARREIRA, R. A.; TANNURI, E. A.; GOMI, E. S.; COSTA, A. H. R. FPSO Mooring Line Failure Detection Based on Predicted Motion. Jun 2021. Available from:⟨https://doi.org/10.1115/OMAE2021-62413⟩.

SIDARTA, D. E.; KYOUNG, J.; O'SULLIVAN, J.; LAMBRAKOS, K. F. Prediction of offshore platform mooring line tensions using artificial neural network. In: ASME. *36th International Conference on Ocean, Offshore and Arctic Engineering*. 2017. Available from:⟨https://doi.org/10.1115/OMAE2017-61942⟩.

SIDARTA, D. E.; O'SULLIVAN, J.; LIM, H.-J. Damage detection of offshore platform mooring line using artificial neural network. In: ASME. *37th International Conference on Ocean, Offshore and Arctic Engineering*. 2018. Available from:⟨https://doi.org/10.1115/OMAE2018-77084⟩.

SIDARTA, D. E.; TCHERNIGUIN, N.; LIM, H.-J.; BOUCHARD, P.; KANG, M.; LERIDON, A. An adaptive method to further improve the tolerance of an ann-based detection system for mooring line failure. In: ASME. *International Conference on*

Offshore Mechanics and Arctic Engineering. 2021. v. 85116, p. V001T01A033. Available from:⟨https://doi.org/10.1115/OMAE2021-63326⟩.

SIRÉTA, F.-X.; ZHANG, D. Smart mooring monitoring system for line break detection from motion sensors. In: INTERNATIONAL SOCIETY OF OFFSHORE AND POLAR ENGINEERS. *The 13th ISOPE Pacific/Asia Offshore Mechanics Symposium*. 2018. Available from:⟨https://onepetro.org/ISOPEPACOMS/proceedings-abstract/PACOMS18/All-PACOMS18/ISOPE-P-18-054/20345⟩.

SMYL, S.; DUDEK, G.; PELKA, P. Es-drnn: A hybrid exponential smoothing and dilated recurrent neural network model for short-term load forecasting. *CoRR*, abs/2112.02663, 2021. Available from:⟨https://arxiv.org/abs/2112.02663⟩.

SULLER, T.; GOMES, E.; OLIVEIRA, H.; COTRIM, L.; SA'AD, A.; SANTOS, I.; BARREIRA, R.; TANNURI, E.; GOMI, E.; COSTA, A. Evaluation of neural architecture search approaches for offshore platform offset prediction. In: *Anais do XVIII Encontro Nacional de Inteligência Artificial e Computacional*. Porto Alegre, RS, Brasil: SBC, 2021. p. 326–337. Available from:⟨https://sol.sbc.org.br/index.php/eniac/article/view/18264⟩.

SUTTON, R. S.; BARTO, A. G. *Reinforcement learning: An introduction*. [S.l.]: MIT press, 2018.

UDDIN, M. A.; JAMEEL, M.; RAZAK, H. A.; ISLAM, A. B. M. S. Response prediction of offshore floating structure using artificial neural network. *Advanced Science Letters*, American Scientific Publishers, v. 14, n. 1, p. 186–189, 2012. Available from:⟨https://doi.org/10.1166/asl.2012.4049⟩.

WANG, T.; ZHANG, M.; JI, H.; QICHEN, L. Damage identification of mooring lines using rbf neural network. In: ONEPETRO. *The 14th ISOPE Pacific/Asia Offshore Mechanics Symposium*. 2020. Available from:⟨https://onepetro.org/ISOPEPACOMS/proceedings-abstract/PACOMS20/All-PACOMS20/ISOPE-P-20-159/449402⟩.

YU, K.; SCIUTO, C.; JAGGI, M.; MUSAT, C.; SALZMANN, M. Evaluating the search phase of neural architecture search. *CoRR*, abs/1902.08142, 2019. Available from:⟨http://arxiv.org/abs/1902.08142⟩.

ZAKI, M. J.; JR, W. M. *Data Mining and Machine Learning: Fundamental Concepts and Algorithms*. [S.l.]: Cambridge University Press, 2019.

ZOPH, B.; VASUDEVAN, V.; SHLENS, J.; LE, Q. V. Learning transferable architectures for scalable image recognition. *CoRR*, abs/1707.07012, 2017. Available from:⟨http://arxiv.org/abs/1707.07012⟩.

APPENDIX A – ACADEMIC ACHIEVEMENTS

This section outlines the academic achievements and courses carried out during the course of the Ph.D degree program.

Disciplines

- **PMI5932-1/1 Elaboração de Artigos Científicos para Mineração, Petróleo e Gás**, Workload: 120h, Credits: 8, **Grade: A.**
- **PMI5040-1/2 Fundamentos de Exploração de Petróleo**, Workload: 120h, Credits: 8, **Grade: B.**
- **PMI5926-1/3 Aplicação de ROVs em Engenharia de Petróleo e de Minas**, Workload: 120, Credits: 8, **Grade: A**.
- **PCS5119-1/1 Tópicos Avançados em Engenharia de Computação**, Workload: 30, Credits: 2, **Grade: A**.
- **PCS5024-1/4 Aprendizado Estatístico**, Workload: 120, Credits: 8, **Grade: C**.
- **PCS5120-1/1 Tópicos em Engenharia de Computação**, Workload: 30, Credits: 2, **Grade: B**.
- **Atividade do Programa Publicação do trabalho "Using Neural Network Approaches to Detect Mooring Line Failure"** no Instituto de Engenheiros Eletricistas e Eletrônicos Access (IEEE Access), p. 27678 - 27695, v. 9, n. 20334031, DOI: 10.1109/ACCESS.2021.3058592, 2021, Workload: 120, Credits: 8.

Honours

- Best paper at the XI Workshop de Pós-Graduação de Engenharia da Computação 2020 (WPGEXI workshop) titled: Detecção De Falhas Na Amarração De Plataformas Flutuantes Pela Previsão Do Movimento (Detection Of Failures In Mooring Lines Of Floating Platforms By Forecast Of The Movement) by Sa'ad A.M., Costa, A.H.R

Publications

- Saad, A. M., Schopp, F., Barreira, R. A., Santos, I. H., Tannuri, E. A., Gomi, E. S., & Costa, A. H. R. (2021). Using Neural Network Approaches to Detect Mooring Line Failure. IEEE Access, 9, 27678-27695. DOI: 10.1109/ACCESS.2021.3058592
- Saad, A. M., Schopp, F., Queiroz Filho, A. N., Cunha, R. D. S., Santos, I. H., Barreira, R. A., Tannuri, E. A., Gomi, E. S., & Costa, A. H. R. (2021, June). FPSO Mooring Line Failure Detection Based on Predicted Motion. In International Conference on Offshore Mechanics and Arctic Engineering (Vol. 85116, p. V001T01A002). American Society of Mechanical Engineers. DIO:10.1115/OMAE2021-62413
- Suller, T. M., Gomes, E. O., Oliveira, H. B., Cotrim, L. P., Sa'ad, A. M., Santos, I. H., Barreira, R. A., Tannuri, E. A., Gomi, E. S., & Costa, A. H. R (2021, November). Evaluation of Neural Architecture Search Approaches for Offshore Platform Offset Prediction. In Anais do XVIII Encontro Nacional de Inteligência Artificial e Computacional (pp. 326-337). SBC. DOI:10.5753/ENIAC.2021.18264
- Ojonugwa, A., Odloak, D., Saad, A. M., Kassab Jr. F. (2022, June). State Estimation of Gas-Lifted Oil Well Using Nonlinear Filters. In Sensors vol.22, No.13, pp.4875. DOI:10.3390/s22134875,
- Dalhatu, A. A., Sa'ad, A. M., Cabral, R., and De Tomi, G. (September, 2022). Remotely Operated Vehicle Taxonomy and Emerging Methods of Inspection, Maintenance and Repair Operations: an overview and outlook. In American Society of Mechanical Engineers, Journal of Offshore Mechanics and Arctic Engineering. DOI:10.1115/1.4055476

APPENDIX B – NEMO TRAINING UNIT

A good training set is critical to the success of the models ability to learn. This section describes how training and testing dataset window was calculated. Using the same procedure as explained in Section 5.2.1, a subset of 3000 different environmental conditions was used for training and another 1500 conditions for validation were selected for the proposed MLP predictor module.

A NN learns by the iterative process of feeding the input of a data unit to a network and comparing the calculated output to the expected output. The network changes its weights and biases until the error difference between the networks' output and the expected output is minimal. The adaption is done using back-propagation (BP) algorithm to reduce the networks' error.

B.1 MLP Training

In this work, 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 600 s and an output of 100 s. The training stride was 100 s. The number of training units for each environmental file can be calculated by the formula

$$k_u = (Tr_{Total} - Tr_{out})/Tr_{Stride} \tag{B.1}$$

where k_u is the number of training units, Tr_{Total} is the total simulation time, Tr_{out} refers to the output partition time of the training unit and Tr_{Stride} is the training stride.

After the first 3600s from the 3 hr simulation generated using dynasim is removed as described in Section 5.1.2 applying the formula results in 71 training units for a single environmental conditions:

$$k_u = (7200s - 100s)/100s = 71 \tag{B.2}$$

In total, 71 units x 3000 environmental conditions $= 213,000$ training units for all environmental conditions were used as the training data set. The validation data was composed of 1500 environmental conditions. Each environmental condition consisted of 71 validation units, adding up to a total of $106,500$ validation units.

APPENDIX C – NEMO MLR CLASSIFIER ADDITIONAL MATERIALS

An overview of how data balancing for the NeMo MLR classifier implemented in section 5.3 is detailed in this section.

Dataset Balancing for MLR classifier: The amount of simulated platform motions is imbalanced. Platform motions labeled No-failure has a smaller number of simulated motion data than those with line failures when combined(see Table 35). To balance the number of platform motions, a **stratified sampling** approach was employed, the method aims to randomly sample equal or close to equal number of samples from each class of a given population.

Table 35: All simulated platform motions from Dynasim.

Draft	Number of platform motions	Line Status	Group
Draft 8 – 21	98,000	No Failure	No_Break
Draft 8 – 21	98,000	Failure L1	A
Draft 8 – 21	98,000	Failure L9	B
Draft 8 – 21	98,000	Failure L12	C
Draft 8 – 21	98,000	Failure L18	D
Total number of platform motions	490,000		

Since there are 14 drafts (8 to 21) with 7000 platform motion error scores for each draft, and for each draft there are 5 platform motions simulated; motion without breakage and 4 motions with breakage in lines 1, 9, 12, 18. Therefore for each draft, there are 7000 * 5 scores which amounts to 350000 error scores, by multiplying 35000 by the 14 drafts a total of 490000 scores is gotten (see Table 35). Hence, 80% of 490000 scores making 392000 scores was used for training of the classifier (see Table 36) and 20% of 490000 making 98000 motions was used for testing the classifier. Table 37 shows the platform motions used for testing.

The stratified sampling method samples by randomly selecting $1,120$ scores from each draft. For example, it selects $1,120$ scores from draft 8 motions for each mooring line status. Given there are 5 mooring line status (no failure, failure in lines 1, 9, 12 and 18), $1,120$ scores by 5 this gives a total of $5,600$ scores then by multiplying $5,600$ scores by the total number of drafts which is 14, the total becomes $78,400$ and by multiplying $78,400$ by the 5 groups (No Break, Group A, Group B, Group C and, Group D) we get $392,000$ which was used for training of the classifier (see Table 36).

Table 36: Example of training data for classification

Group	Drafts	No of Samples from each Draft	Total Sample for each Group
No Break	8 - 21	5,600	14 * 5,600 = 78,400
Group A	8 - 21	5,600	14 * 5,600 = 78,400
Group B	8 - 21	5,600	14 * 5,600 = 78,400
Group C	8 - 21	5,600	14 * 5,600 = 78,400
Group D	8 - 21	5,600	14 * 5,600 = 78,400
Total no. of platform motions	392,000		

Table 37: Example of testing data for classification

Group	Drafts	Total Sample For Each Group
No Break	8 - 21	19,600
Group A	8 - 21	19,600
Group B	8 - 21	19,600
Group C	8 - 21	19,600
Group D	8 - 21	19,600
Total no. of platform motions	98,000	

C.1 Complete misclassified motions from the MLR classifier

The table shows the complete 31 platform motions the MLR in Section 5.3 misclassified. The table shows the MLR prediction for each misclassified motion, labeled as predicted group, together with the actual group the motions belongs to and draft of the misclassified sample. It also shows the probability of each sample it predicted to belong to a particular group.

Table 38: Complete Misclassified Motions from the MLR Classifier

A	B	C	D	No_B	sum	predicted_Group	actual_Group	Draft
0.481582	0.000	0.000	0.518418	0.000	1	D	A	d-21-bl01
0.523100	0.476888	0.000	0.000010	0.000002	1	A	B	d-12-bl09
0.543884	0.000	0.000	0.000014	0.456102	1	A	No_B	d–19
0.714743	0.285257	0.000	0.000001	0.000	1	A	B	d-10-bl09
0.588754	0.008088	0.000	0.000002	0.403157	1	A	No_B	d–9
0.015532	0.000	0.000	0.984468	0.000	1	D	A	d-14-bl01
0.822961	0.000	0.000	0.000005	0.177034	1	A	No_B	d–15
0.585678	0.000	0.000	0.414322	0.000	1	A	D	d-8-bl18
0.782901	0.000	0.000	0.217099	0.000	1	A	D	d-13-bl18
0.159140	0.000	0.000	0.840860	0.000	1	D	A	d-17-bl01
0.362308	0.484746	0.000	0.000001	0.152945	1	B	No_B	d–14
0.932041	0.060170	0.000	0.000	0.007788	1	A	No_B	d–21
0.239460	0.000	0.000	0.760540	0.000	1	D	A	d-17-bl01
0.293493	0.706507	0.000	0.000	0.000	1	B	A	d-19-bl01
0.152633	0.847367	0.000	0.000	0.000	1	B	A	d-20-bl01
0.861314	0.104325	0.000	0.000	0.034360	1	A	No_B	d–14
0.851781	0.148202	0.000	0.000017	0.000	1	A	B	d-13-bl09
0.579196	0.000	0.000	0.420804	0.000	1	A	D	d-11-bl18
0.950370	0.006540	0.000	0.000	0.043090	1	A	No_B	d–17
0.468475	0.000	0.000	0.531525	0.000	1	D	A	d-16-bl01
0.460407	0.000	0.000	0.000	0.539593	1	No_B	A	d-8-bl01
0.146590	0.853250	0.000	0.000	0.000160	1	B	No_B	d–17
0.768183	0.000	0.000	0.231817	0.000	1	A	D	d-8-bl18
0.004694	0.000	0.000	0.995306	0.000	1	D	A	d-14-bl01
0.000487	0.000	0.000	0.999513	0.000	1	D	A	d-20-bl01
0.764241	0.000	0.000	0.000010	0.235749	1	A	No_B	d–20
0.873650	0.006486	0.000	0.000006	0.119859	1	A	No_B	d–11
0.784140	0.000	0.000	0.215860	0.000	1	A	D	d-9-bl18
0.219683	0.000	0.000	0.780317	0.000	1	D	A	d-15-bl01
0.538073	0.000	0.000	0.461927	0.000	1	A	D	d-14-bl18
0.599952	0.000	0.000	0.400048	0.000	1	A	D	d-14-bl18

The Table 39 shows the complete misclassified environmental conditions misclassified by the MLR classifier.

Table 39: Complete Misclassified Enviromental Conditions Measured

Combination ID	Current Velocity	Current Direction	Wind Velocity	Wind Direction	Wave Height	Wave Period	Wave Direction	Swell Height	Swell Period	Swell Direction	Date	Time
165	0.18	321.17	6.90	87.20	1.730	11.51	179.80	1.11	5.05	97.40	4/ 8/2006	09:00:00
170.0	0.480	228.110	5.010	120.600	2.0	9.630	187.500	0.580	3.770	133.200	25/ 8/2004	06:00:00
483.0	0.370	216.580	3.900	60.500	2.070	15.990	129.600	0.360	3.300	66.0	10/ 6/2007	09:00:00
492.0	0.270	163.090	9.670	20.400	2.340	13.070	191.300	1.340	5.980	39.300	18/ 1/2007	03:00:00
548.0	0.260	216.560	6.300	107.400	2.720	14.400	177.100	1.0	4.560	112.0	2/ 5/2008	15:00:00
983.0	0.350	219.370	5.800	64.200	2.090	10.440	175.500	0.730	4.510	74.800	2/11/2008	12:00:00
1281.0	0.430	197.070	9.360	44.0	1.440	5.440	54.300	0.420	6.140	136.400	26/11/2004	21:00:00
1548.0	0.530	172.440	9.410	27.600	1.720	6.630	28.800	0.590	8.960	143.100	16/11/2009	00:00:00
1983.0	0.640	173.910	10.490	29.900	2.510	7.100	30.0	1.930	12.950	195.700	18/10/2005	00:00:00
2179.0	0.450	194.0	9.630	15.600	1.570	5.710	34.400	0.930	8.0	95.100	1/12/2005	06:00:00
2259.0	0.480	175.250	12.330	18.400	2.240	6.390	24.400	0.670	8.550	114.100	15/ 1/2009	09:00:00
2781.0	0.470	179.950	8.110	14.500	1.820	11.350	119.700	1.450	5.240	23.200	1/ 8/2004	09:00:00
2915.0	0.830	194.620	6.740	354.800	1.500	9.180	121.800	0.970	4.200	9.300	3/12/2007	09:00:00
2983.0	0.440	191.010	7.190	12.0	1.390	6.820	87.400	1.290	5.110	1.200	28/ 2/2004	12:00:00
3048.0	0.170	147.090	9.200	22.600	1.500	10.400	151.100	1.380	4.750	42.200	16/ 4/2005	09:00:00
3170.0	0.320	194.750	8.430	13.600	1.790	15.220	116.200	1.690	5.720	18.0	23/ 5/2007	06:00:00
3181.0	0.100	113.700	4.580	21.0	1.750	14.010	153.400	0.490	3.340	12.600	12/ 4/2008	00:00:00
3190.0	0.510	178.330	7.620	30.0	1.920	10.590	115.300	1.360	5.660	39.200	2/ 2/2005	03:00:00
3259.0	0.290	196.280	7.730	12.500	1.690	10.690	168.900	1.070	4.510	30.200	29/ 7/2005	12:00:00
3483.0	0.260	199.680	6.170	61.700	1.200	7.590	115.600	0.750	5.030	65.300	25/10/2009	00:00:00
3499.0	0.490	195.370	8.330	21.500	1.340	13.300	178.200	1.330	5.360	25.100	15/ 5/2004	18:00:00
3559.0	0.230	166.660	5.450	159.400	1.480	8.020	109.0	0.0	0.0	0.0	1/ 4/2009	00:00:00
3625.0	0.380	208.430	9.730	126.700	2.020	6.510	121.100	0.830	7.090	70.0	12/12/2006	21:00:00
3659.0	0.280	226.830	7.410	142.700	2.050	8.060	142.100	0.0	0.0	0.0	5/ 5/2005	15:00:00
4155.0	0.390	172.960	5.810	25.500	1.620	7.480	57.300	0.0	0.0	0.0	1/12/2006	03:00:00
4548.0	0.200	209.310	8.090	120.600	2.770	14.110	146.400	0.0	0.0	0.0	21/ 5/2007	00:00:00
4659.0	0.340	200.120	9.520	123.500	1.580	7.070	100.0	1.210	11.990	209.900	13/11/2007	03:00:00
4681.0	0.410	167.260	6.400	281.700	1.560	8.340	65.800	0.670	3.390	304.200	10/ 4/2004	18:00:00
4687.0	0.500	142.600	11.010	304.900	2.150	8.060	53.700	1.720	5.540	332.800	12/ 7/2009	15:00:00
4759.0	0.130	64.410	5.840	208.500	1.570	9.190	93.100	0.850	4.120	221.700	12/ 6/2009	18:00:00
4983.0	0.270	199.230	5.550	285.400	1.660	8.380	36.400	0.550	3.530	295.400	14/ 1/2007	09:00:00
4999.0	0.440	178.610	3.930	282.200	1.520	9.360	158.0	0.280	2.890	259.100	11/12/2004	09:00:00
5172.0	0.350	169.350	6.0	209.700	1.0	7.130	52.900	0.660	4.720	220.300	10/12/2004	18:00:00
5559.0	0.310	198.260	4.320	226.300	1.900	14.490	83.300	0.180	2.170	248.700	25/ 6/2007	03:00:00
5675.0	0.150	163.080	6.900	236.600	1.930	13.0	177.800	1.070	5.120	240.500	22/ 1/2007	12:00:00
5983.0	0.570	213.710	11.120	161.100	2.570	8.0	180.0	0.0	0.0	0.0	14/ 9/2005	06:00:00
6059.0	0.190	282.120	11.200	183.500	1.940	6.050	185.200	1.330	8.700	53.0	17/ 1/2004	09:00:00
6103.0	0.140	289.010	7.560	168.200	3.280	11.920	185.700	0.0	0.0	0.0	27/ 4/2005	21:00:00
6148.0	0.200	5.210	7.320	196.300	2.620	10.430	198.200	0.0	0.0	0.0	6/ 5/2006	18:00:00
6155.0	0.260	358.330	8.640	207.800	2.420	10.470	187.400	0.0	0.0	0.0	5/ 5/2006	15:00:00
6159.0	0.240	229.480	10.880	157.500	2.600	6.040	174.900	0.0	0.0	0.0	28/ 7/2007	15:00:00
6175.0	0.160	229.690	10.470	169.400	2.770	7.520	162.300	0.0	0.0	0.0	6/ 2/2005	15:00:00
6548.0	0.260	196.140	6.240	200.0	2.130	10.780	186.200	0.0	0.0	0.0	15/ 5/2006	06:00:00
6624.0	0.240	243.570	8.920	195.600	1.730	6.390	207.500	1.170	7.470	58.700	21/11/2005	09:00:00
6655.0	0.220	243.380	7.590	175.200	3.760	14.090	205.500	0.0	0.0	0.0	17/ 6/2008	12:00:00
6690.0	0.370	220.130	8.480	185.200	1.580	6.180	194.400	1.290	10.660	139.0	17/ 9/2005	21:00:00