Predicting Ship Heave Motions

Ship Heave Motion Prediction Using LSTM Neural Networks Without Wave Excitation Information

Jolan Keijzer

Utrecht University | Applied Data Science

Utrecht University

MARIN

Predicting Ship Heave Motions

Ship Heave Motion Prediction Using LSTM Neural Networks Without Wave Excitation Information

by

Jolan Keijzer

Student Name	Student Number
Jolan Keijzer	6920012

UU Supervisor:	Deb Panja
MARIN supervisors	Eelco Frickel and Bulent Duz
Date:	1 Juli 2022
Programme:	Applied Data Science
University:	Utrecht University

Utrecht University

Abstract

In off-shore multi-body operations, it is important to accurately determine the movements of the vessels to allow for safe operations. This is especially evident in multi-body operations that involve helicopters, due to the relative motion between the landing deck and the helicopter. To be able to augment human judgement in helicopter/off-shore vessel multi-body operations, it is crucial to have accurate predictions of the movements of the vessel. One of the key movements of off-shore vessels in these operations is the heave motion, which represents the up and down movements of the vessel. Modelling these movements mathematically is a difficult feat due to the strong non-linearity of these movements, and the complicated hydrodynamic forces and stochastic sea disturbances that are at the root of these movements. Therefore, Neural Networks bear a lot of opportunity in this regard, because of their strong ability of handling non-linearity. Especially, Long Short-Term Memory (LSTM) models are useful in this regard owing to their strong ability of handling sequences and their ability of learning both long- and short-term dependencies. However, most existing research uses wave excitation information to predict the heave motion even though ships are not often equipped with expensive wave detection systems. Therefore, this paper researches whether LSTM models are able to accurately predict heave motions 10 and 20 seconds ahead without the usage of wave excitation information. This paper shows that LSTM models are able to achieve accurate predictions without the usage of wave excitation information. Therefore, it is shown that LSTM models have the ability to augment human judgement in helicopter/off-shore vessel operations where this wave excitation information is not available. Especially in terms of 10 second ahead predictions LSTM is able to achieve promising prediction performance. However, the prediction quality of the 20 second ahead predictions is less satisfying. Finally, most existing papers used simple one-layer LSTM models, whereas this paper shows that using more complicated models lead to increased prediction performance.

Contents

Abstract **i**

1 Introduction **1**

2 Data **3**

2.1 Data Exploration 3

2.2 Data preparation 4

2.3 Ethical and legal considerations 5

3 Methods **6**

4 Results **9**

4.1 Tuning 9

4.2 Model training 10

4.3 Model evaluation 11

5 Conclusion & Discussion **14**

References **16**

A Appendix A: Code **18**

B Appendix B: Preliminary tuning results **19**

B.1 First tune cycle 19

B.2 Second tune cycle 20

1 Introduction

In off-shore multi-body operations, it is important to accurately determine the movements of the vessels to be able to safely merge or split vessels (e.g. connect two vessels, side-by-side transfer, helicopter landing, etc) (X. Zhao et al., 2004). During these operations, the success of the operation is highly contingent on ship movements like positioning, speed and heading, and the six ship motions: surge, sway, heave, roll, pitch and yaw (Cheng et al., 2019). These ship motions are visualized in figure 1.1. In general, matching up speed and heading between separate vessels is quite straightforward, whereas, matching the ship motions is more challenging. This issue is especially evident in multi-body operations that involve helicopters or drones, due to the relative motion between the landing deck and the helicopter (Yang et al., 2008). As of yet, landing operations with human pilots are guided by the human judgement of landing deck superintendents whom indicate safe landing windows based upon intuition about the ship's motions. Therefore, accurate and reliable predictions of these ship movements can help to replace or augment the human judgement in these operations (Khan et al., 2005).

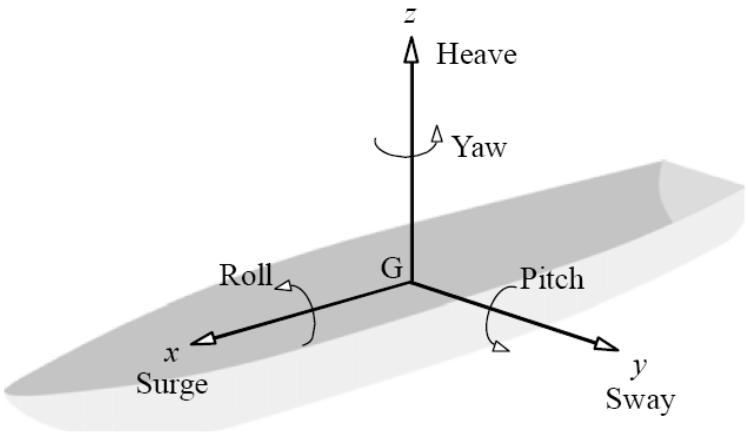

Figure 1.1: The six ship motions visualized as degrees of freedom (Tanaka, 2018)

Unfortunately, modelling ship motions mathematically is a difficult feat. This is due to the complicated hydrodynamic forces and stochastic sea disturbances that are crucial in determining ship motions and that are too complex to accurately model mathematically (Khan et al., 2005; Sørensen, 2011; Yang et al., 2008). For this reason, earlier attempts of modelling ship motions using traditional statistical methods like Kalman filter (Triantafyllou & Bodson, 1982), ARMA (Yumori, 1981) and ARIMA (Khan et al., 2004) have been unsatisfactory in terms of predicting multiple time steps ahead (Khan et al., 2005).

Generally, Neural Networks (NN) perform well at tasks that are difficult to describe by formal mathematical rules (G. Li et al., 2017) and that have non-linear components (Janiesch et al., 2021). Therefore, utilizing NN bears opportunities for ship motion prediction due to the strong non-linearity of ship motions (Sun et al., 2022) and time-series prediction more generally (Dumitru & Maria, 2013).

To achieve success in these predictions, several NN architectures have been applied to time-series and ship motion prediction. First of all, Convolutional Neural Networks (CNN) are applied because time-series can be regarded as a 1D spatial component and CNN are particularly effective in mapping spatial relationships (Selvin et al., 2017). However, albeit utilizing CNN models for time-series prediction has been effective in predicting stock price (Selvin et al., 2017), traffic and solar energy production (K. Wang et al., 2019). Their performance in ship motion prediction has been disappointing, due to it's inability to handle longer term sequences (M. Zhao et al., 2021).

Opposingly, Recurrent Neural Networks (RNN) learn from previous time steps to learn dependencies of data over time (Mandic & Chambers, 2001) and thus perform particularly well on sequential data (D'Agostino et al., 2021). Unfortunately, traditional RNNs suffer from the vanishing and exploding gradient problem when learning long-term dependencies (Bengio et al., 1994). To address this problem, the Long Short-Term Memory (LSTM) architecture was proposed, which uses a memory block with input, forget and output gates to allow for (partly) opening and closing the access to the historical error flows (Hochreiter & Schmidhuber, 1997). For this reason, LSTM models achieve higher accuracy in time-series prediction in several applications and are able to model both long- and short-term dependencies of data (Yunpeng et al., 2017). The (dis)advantages of the models are summarized in table 1.1.

Table 1.1: Advantages and disadvantages of different models

Model	Advantages	Disadvantages
Statistical methods	Low computational cost Easier configuration	Bad performance multi-step prediction Inability to handle non-linearity
Traditional RNN	Handles non-linearity well Strong ability with sequences	High computational cost Vanishing and exploding gradients
LSTM	Advantages of traditional RNN Learns long- and short-term dependencies	Highest computational cost Propensity of over-fitting

Consequently, LSTM models have been applied in several ship motion prediction studies with promising results. Be that as it may, many of these studies utilize the wave excitation to help predict the ship movements (D'Agostino et al., 2021; Duan et al., 2019; Liu et al., 2020). Even though, ships are not often equipped with wave detection systems due to the high price of these systems. For this reason, real-time wave information is most often not available. Therefore, it is paramount to research how ship motions can best be predicted without information about the wave excitation with different multi-step prediction lengths. The research question that culminates from this is *"How can ship motions best be predicted with LSTM neural networks without the usage of wave excitation information?"*

To answer this research question, a model with finely tuned hyper parameters is required. However, in most of these studies, the optimal hyper parameters of the model are chosen through grid-search. Despite the existence of model tuning algorithms. These tuning algorithms utilize Bayesian (Snoek et al., 2012) or Hyperband (L. Li et al., 2017) optimization techniques to make this process more efficient and more effective. Noting the absence of usage of these algorithms in existing papers, this paper will examine the usability of hyper parameter optimization algorithms. Finally, most LSTM ship motion prediction papers do not give clear insight into the used input lengths and prediction lengths. Noting this absence of information, this paper attempts to map the effects of these choices and the prediction quality at different time steps. Namely, this paper compares several models with different hyper parameters that are tuned and trained based upon 10 and 20 second output lengths and different input lengths.

2
Data

2.1. Data Exploration

The data used in this study is a set of 100 simulations that are conducted by the Maritime Research Institute Netherlands in the simulation program FREDYN. In these simulations, the ship motions of frigate type vessel 5415M are simulated in sea state 5 with a constant heading and vessel speed. The duration of this simulation is 11500 seconds and the first 500 seconds are deducted because the simulation needs some time to become a realistic representation. After this deduction, 11000 seconds (around 3 hours) remain per simulation, whereas, the granularity in these simulation is one observation per 0.2 seconds (5 hertz). Therefore, every simulation consists of 55.000 observations and thus the 100 simulations combined span 5.5 million observations. The features of the data consist of the six ship motions in relation to the centre of gravity and a reference point at the stern of the vessel, which represents a helicopter landing platform. Additionally, the data contains features on the wave height, heading of the ship and sea state. Importantly, this information should not be used in making predictions of the ship motions because this paper attempts to predict ship motions without the usage of wave excitation information.

To make the prediction problem more feasible, this paper solely focuses on predicting the heave motion of the reference point located at the stern of the vessel. This feature is denoted as RefZ and has similar means and standard deviations over the 100 simulations as visualized in figure 2.1, which is expected since the simulations are conducted with identical settings.

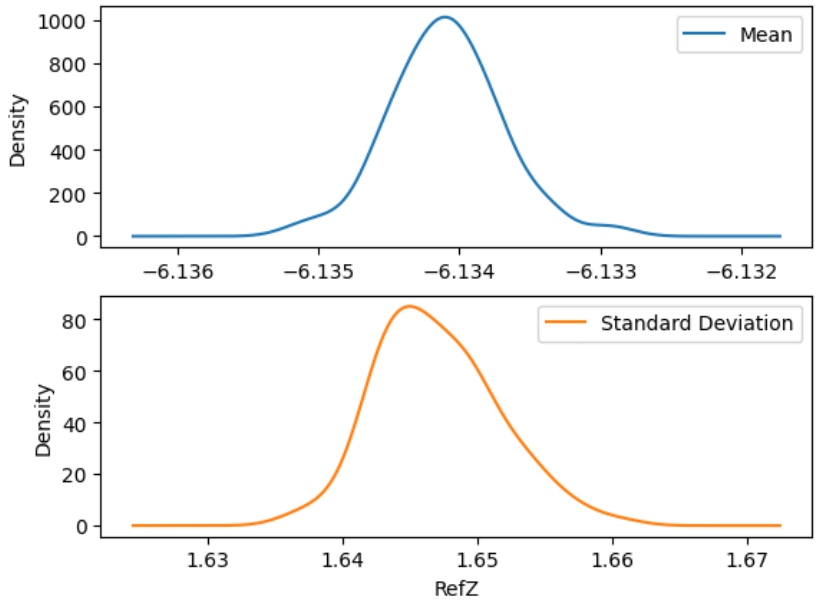

Figure 2.1: Density plots of the means and standard deviations of RefZ over the 100 simulations

2.2. Data preparation

In comparison to data retrieved from sensors, data quality as a result of simulations does not run the risk of missing data or outliers occurring. For this reason, missing data analysis and outlier analyses are not performed.

However, noting the high computational costs associated with training (Recurrent) Neural Networks (Laurent et al., 2016), it is important to consider how training costs can be lowered. Consequently, it is important to analyze whether the time step granularity of 5 hertz is required to achieve accurate predictions. As a result, a data re-sampling function is created that attempts to reduce the time step granularity while simultaneously maintaining the temporal relations and details of the peaks and valleys of the time-series. This is achieved by looping over the data in non-overlapping windows and locally fitting trend lines. Afterwards, the minimum value is selected in descending trends and the maximum value is selected in ascending trends. As becomes evident from figure 2.2, re-sampling 5 or more time steps into one observation results in a loss of the peak and valley details and changes the temporal relationships of the data. For this reason, the time step granularity should not be reduced to such high levels. In addition, the optimal level of time step granularity is determined through it's effect on model training in terms of the trade-off between prediction accuracy and model training cost, which is discussed later.

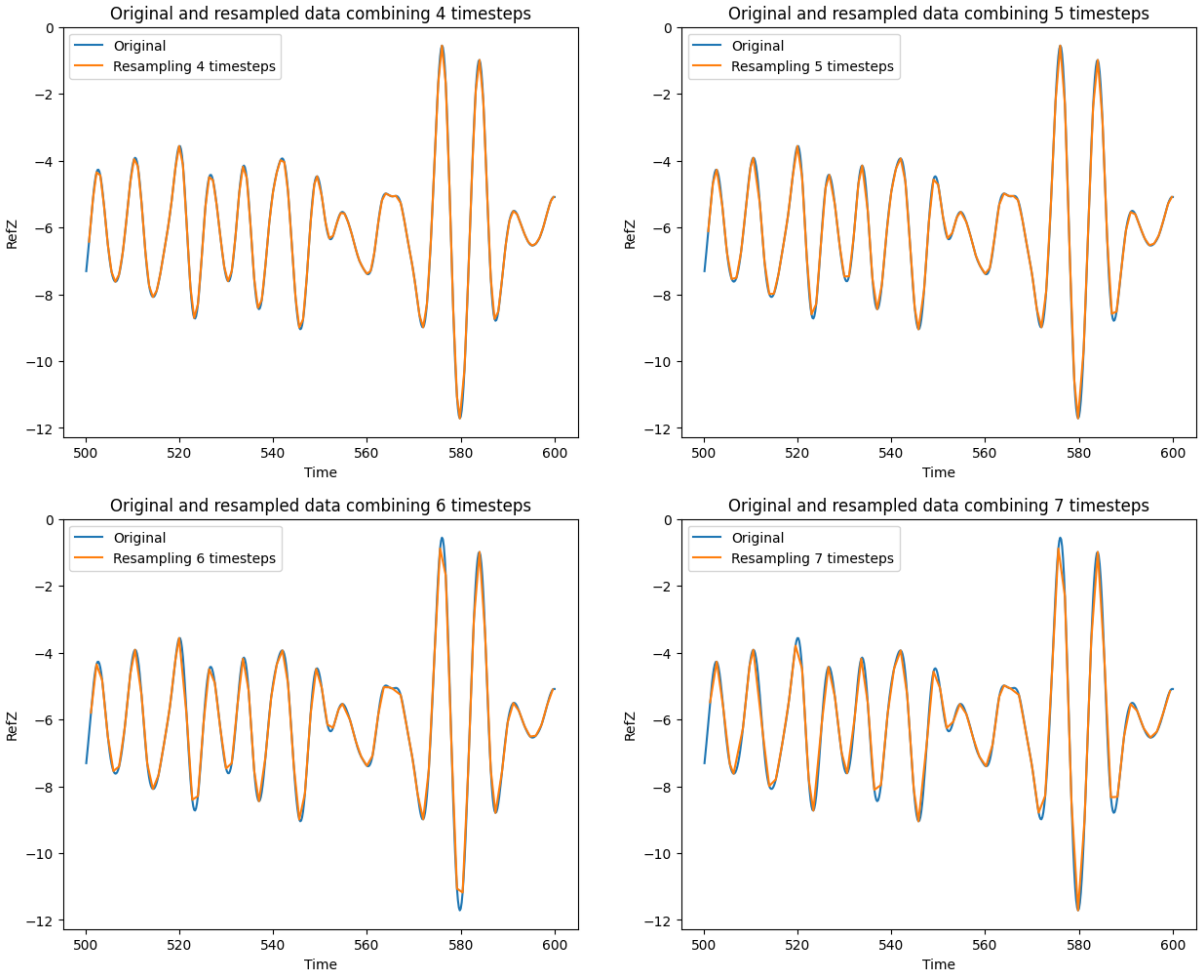

Figure 2.2: Line-plots of original data at 0.2 second granularity and combining 4 to 7 of these time-steps

Moreover, to be able to train an LSTM model for time-series prediction training, validation and test examples need to be created. To achieve this for LSTM models, several methods are available as visualized in figure 2.3. Noting that this paper attempts to predict several time-steps into the future based on sequences of historical data two options are available, namely many-to-one and many-to-many. In utilising the many-to-one method for time-series prediction previous predictions are used to predict further into the future. Unfortunately, this method results in the accumulation of errors, due to the predictions being based upon other predictions (Elsworth & Güttel, 2020). Hence, this paper utilizes the many-to-many method of creating examples. This is done by taking the RefZ of observations of a specified input length as X and future observations of RefZ of a specified output length as Y to make the time-series problem supervised. An example is shown in the fourth image in figure 2.3. In this figure, the rectangles represent vectors of information and the arrows represent functions applied by the model. With regards to the colors, red represents input, blue represents output and green represents the internal state of the LSTM network. In contrast to this image, this paper does not have overlap between the input and output.

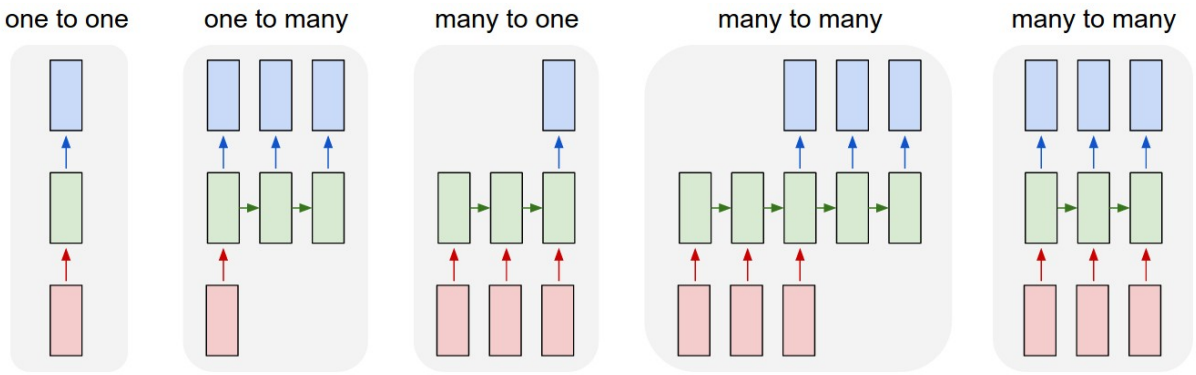

Figure 2.3: Visualization of methods of training LSTM models (Karpathy, 2015)

Finally, noting that the data is sourced from 100 separate simulations, the sequences can not simply be merged. This is due to the absence of temporal relationships between the end of one simulation and the start of the next. To deal with this, examples are sourced from every separate simulation to create distinct train, validation and test sets. This split is done by taking simulations 1 to 70 as training set, 71 to 85 as validation set and 86 to 100 as test sets. Due to the large size of these data sets and the large overlap between adjacent training examples due to the many-to-many method, sub-sampling is applied using strides. In short, this stride determines how many time-steps are skipped before a new example is created.

2.3. Ethical and legal considerations

Generally, ethical considerations are less prevalent in data that is sourced from simulations and that does not contain personal information. Nonetheless, due to the simulated nature of the data it is important to consider that if results from this paper are applied to helicopter landing decisions, the models require extensive training and testing on real-world data and extensive peer-reviewing.

In terms of legal considerations, the data set used in this paper is created by the Maritime Research Institute Netherlands (MARIN). A key condition of the usage of this data is that it can not be shared without explicit consent of MARIN.

3
Methods

As mentioned before, LSTM models show promising results in predicting ship motions. The effectiveness of LSTM in predicting ship motions is explained by a combination of the inherent properties of recurrent connections in RNN architectures in general (Mandic & Chambers, 2001) and the usage of memory blocks in LSTM models more specifically (Bengio et al., 1994). This combination allows for learning both short- and long-term dependencies in the data and makes LSTM models especially effective in ship motion prediction (D'Agostino et al., 2021). However, previous literature most often uses wave excitation information in their predictions. Even though, most off-shore vessels are not equipped with wave detection sensors. Thus, the research question that originated from this is *"How can ship motions best be predicted with LSTM neural networks without the usage of wave excitation information?"*

To translate this research question into a data science question, it is important to note that finely tuned hyper parameters are key in achieving high prediction accuracy. Surprisingly, existing literature primarily utilizes grid-search to determine the hyper parameters of the model. Even though, tuning algorithms can help to make this process more efficient and effective. In addition, the existing papers are not open about the decision making in terms of input and output lengths. Finally, as explained in the data section, this paper solely attempts to predict the heave motion at the reference point. Therefore, the following data science question is formulated: *"How can the input length and the hyper-parameters of an LSTM model best be configured to allow for accurate predictions of ship heave motion (RefZ) for 10 and 20 seconds into the future?"*

To provide a meaningful answer to this question, this paper utilizes a tuning algorithm instead of grid-search methods to determine the optimal hyper parameters in a more effective and efficient manner. In this paper, the Keras-tuner is applied. The reason for this decision is three-fold. First of all, the Keras-tuner is built specifically for tuning of hyper parameters in the widely used library Keras, which helps to improve the replicability of this study. Second, the wide usage of Keras has resulted in clear documentation and community support for the Keras-tuner. Third, the Keras-tuner library is especially flexible in tuning options and allows for straightforward sub-classing to expand tuning options even further.

Moreover, in the Keras tuner, multiple optimization algorithms are available of which Hyperband and Bayesian Optimization are the most advanced. Even though Hyperband is much more efficient than Bayesian Optimization, due to the smart allocation of resources, it tends to converge to sub optimal solutions (J. Wang et al., 2018). For this reason, both methods are applied and compared to allow for efficient exploration with Hyperband and increased reliability on converging to an optimal solution with Bayesian Optimization.

In order to make use of the Keras-tuner, it is important to clearly define which parameters should remain constant, which parameters the tuner should attempt to tune and what type of decisions the tuner can make within a specific parameter. Deciding upon this so-called search space is highly dependent on the model that is used and the domain problem at hand. In short, this paper decides upon this search space through a combination of literature review, experience and experimentation. A summary

of the selected search space and argumentation for this selection is provided in table 3.1. First, some tuning cycles are performed using this entire search space. Afterwards, the results of these tune cycles are analysed to spot trends of convergence around specific hyper parameter configurations. Finally, based upon this analysis a smaller search space is defined around these configurations to come to a more finely tuned model.

Successively, the most promising model of the tuning cycle is trained. During this training process, the model is trained with varying input lengths and sampling strides to discover the optimal amount of input. Upon increasing the input length to high amounts the stride is increased accordingly to prevent infeasible training times.

Table 3.1: Search space definition and argumentation

Hyper parameter	Options	Argumentation
Number of layers	Integer between 1 and 6	Optimal number of layers highly contingent on use case, thus tune with large range
Units per layer	Integer between 16 and 512	Optimal units per layer highly contingent on use case, thus tune with large range
Learning rate	Choice of [1e-2, 5e-3, 1e-3, 5e-4, 1e-4]	Optimal learning rate highly contingent on use case, thus tune with large range
Batch size	Integer between 64 and 1024	Optimal batch size highly contingent on use case, thus tune with large range
Epoch amount	Early stop based on validation loss	On the one hand, as many epochs as possible are desired, but over-fitting might occur. Use early stop based on validation loss to prevent over-fit
Activation function (dense layer)	Default: linear	LSTM layers already contain 3 sigmoid and 1 tanh activation function (Greff et al., 2016). Dense layer requires continuous output values and thus linear activation is the only built-in option.
Optimizer	Choice between Adam and SGD	Adam has shown to be a good combination of the advantages of RMSprop and Adadelta and thus is the preferred option. However, albeit Adam converges quicker, SGD often generalizes better (Reddi et al., 2019; Zhang, 2018). Therefore, both Adam and SGD are examined.
Dropout	Float between 0 and 0.5	In LSTM models dropout an be applied before, in or after LSTM layers to prevent over-fitting. Placing dropout before layers is shown to have the best performance in most use cases (Bluche et al., 2015)
Batch normalization	No	Regular batch normalization not possible since it is computed per batch and thus does not consider recurrent connections (Cooijmans et al., 2016). Only possible through reparametrizing the LSTM, which is outside the scope of this paper

The raw RefZ data is normalized to allow for better model convergence and performance (Jayalakshmi & Santhakumaran, 2011). In this paper, MinMax normalization in the interval of [0, 1] is applied. This method normalizes the raw RefZ values to values in this interval.

$$Y_{minmax} = \frac{Y - minY}{maxY - minY} \tag{3.1}$$

To evaluate the performance of the proposed model, the Root Mean Squared Error (RMSE) metric is used. Moreover, as a novel contribution, this paper calculates the RMSE per time step to determine the RMSE for every position that is predicted in the future. This is done to gain a better understanding of the strengths and weaknesses of the model. The implementation of $RMSE_{timestep}$ is shown in Equation 3.3. In this metric, instead of calculating the RMSE over all the prediction errors, the RMSE is calculated separately on the errors at time step t.

$$RMSE = \sqrt{\frac{1}{n}\Sigma_{i=1}^{n}\Big(d_i - f_i\Big)^2} \tag{3.2}$$

$$RMSE_{timestep} = \sqrt{\frac{1}{n^t}\Sigma_{i=1}^{n^t}\Big(d_i^t - f_i^t\Big)^2} \tag{3.3}$$

To summarize, the workflow of this research is visualized in figure 3.1. Even though this figure visualizes this research as a linear process, it is important to note that the actual process is iterative. Especially determining the used stride in sub-sampling, the input length, and the training and tuning are iterative processes.

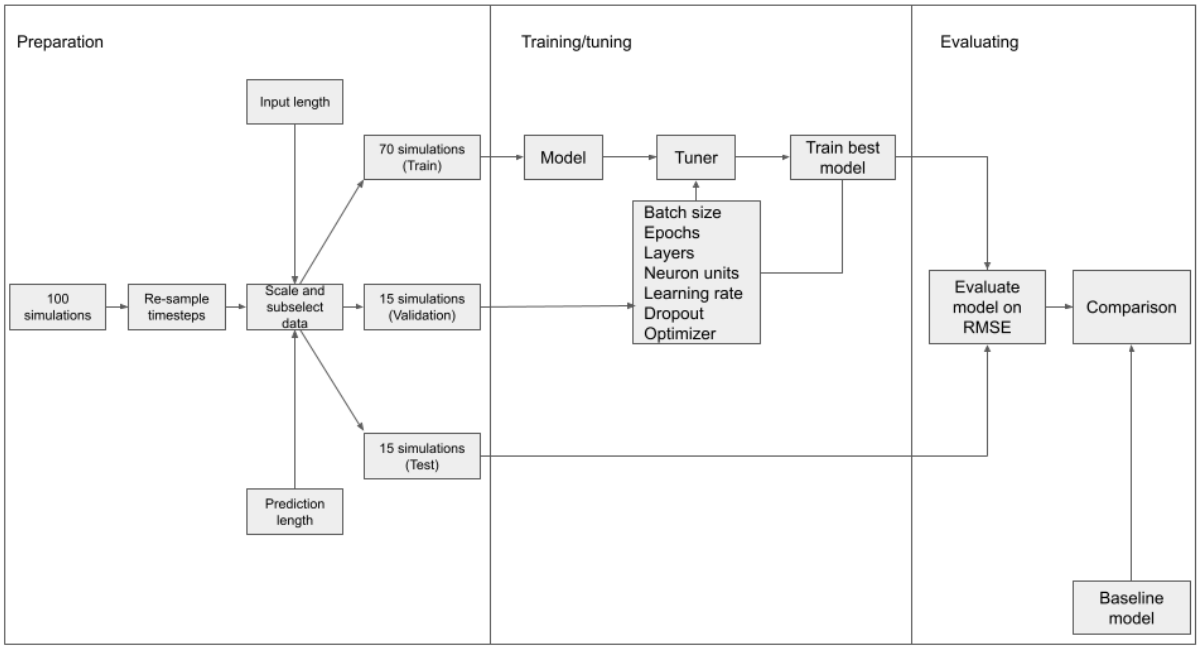

Figure 3.1: Visualization of the workflow of this research

4
Results

4.1. Tuning

As previously mentioned, the tuning process is an iterative process and in this research this process is divided in multiple tuning cycles. In this section the final tuning cycle is outlined, whereas the preceding cycles are discussed in appendix A. Initial results of these two earlier tuning cycles are as follows.

- Dropout is not beneficial in this application, because this prevents the model from converging and surprisingly leads to more over-fitting
- The Adam optimizer converges quicker and also leads to better performance than SGD
- Optimal number of layers and nodes unclear
- Most well performing models converge to a validation loss slightly above 0.005

Taking these initial results into account, it becomes clear that additional tuning is required to determine what the optimal number of layers and nodes is. To do this, it is beneficial to switch from the Bayesian Optimization algorithm to the Hyperband algorithm. The reason for this is that Hyperband is able to test a large number of models more efficiently due to smart resource allocation (J. Wang et al., 2018) by incrementally assigning more resources to the best performing models (L. Li et al., 2017).

Importantly, this final tuning cycle does not include the optimizer nor dropout in the search-space, because the preceding tuning cycles have shown that dropout is not beneficial and the Adam optimizer is superior. The results of this tuning cycle are shown in table 4.1, which indicates that the Hyperband algorithm was able to converge to superior models. The chosen model has 3 layers with 368, 368 and 240 neurons per layer. In addition, the learning rate is 0.0005 and the batch size is 64.

Table 4.1: Tuning trial summary top 10 ordered by validation loss selection from 484 trials with Hyperband

Rank	Validation loss	Number of layers	Units/layer	Learning rate	Batch Size
1	0.00509	3	368, 368 and 240	0.0005	64
2	0.00509	3	368, 368 and 240	0.0005	64
3	0.00511	2	96 and 320	0.001	64
4	0.00513	3	480, 208, 32	0.001	64
5	0.00513	2	96 and 320	0.001	64
6	0.00521	2	302 and 48	0.001	64
7	0.00523	3	480, 208 and 32	0.001	64
8	0.00526	2	32 and 368	0.005	64
9	0.00532	3	368, 368 and 240	0.0005	64
10	0.00538	2	32 and 368	0.005	64

4.2. Model training

During the training process, the tuned model is trained on varying input lengths and varying strides to determine the optimal amount of input for accurate predictions. In contrast, the output length remains constant at 25 time steps (20 seconds). To concretize, in line with the model tuning, the first model has an input length of 50 and a stride of 5. Afterwards, the input length and stride are incrementally increased and decreased to analyze the effects. Importantly, the models use early stopping as a mechanism to prevent over-fit, which halts model training if validation loss does not improve for a specified number of epochs. Upon activation of the early stop mechanism the weights are restored to the epoch with the best validation loss. This so-called patience varies per model, due to varying training processes per model.

The input configurations and their respective training validation loss histories are visualized in figure 4.1. This figure only contains the validation loss to allow for easier interpretation. The horizontal lines indicate the validation loss of the best epoch that the early stop mechanism restores to and thus indicates the performance of the trained model. The upwards moving validation-losses are a result of this early stop mechanism and do not represent the final performance of the model.

This visualization shows that the best performing model is the 200 input stride 1 model closely followed by the 200 input stride 5 model. Additionally, comparison of the different models shows that an increase of input to the model improves validation performance both in terms of input length and lowering the stride. However, increasing the input length and lowering the stride both result in increased computational costs with an epoch of the 200 input and stride 1 model requiring 25 minutes. Therefore, a trade-off exists between computational costs and model performance (Huang et al., 2017). This increased computational cost of increasing input length is managed by simultaneously increasing the stride. Nonetheless, doubling the input length and stride together does not cancel each other out as evidenced by the 400 input and stride 10 model. Finally, the 400 input model also shows that increasing the input length to very high levels does not necessarily result in the best performance. Be that as it may, it is possible that the 400 input model would perform better upon decreasing the stride.

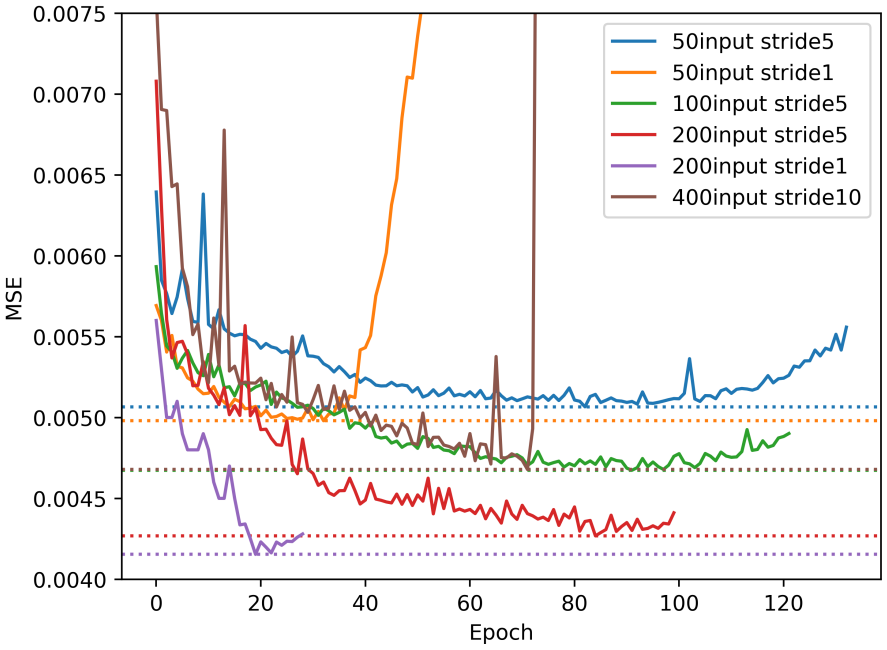

Figure 4.1: Visualization of validation loss histories of trained models

4.3. Model evaluation

The model evaluation on the test set is summarized in table 4.2, which contains RMSE calculated on both 10.4 and 20 second ahead predictions to allow for more detailed comparison. An important note is that is that the validation loss during training is based upon the MSE on normalized data, whereas for model evaluation on the test set RMSE is used. Moreover, table 4.2 displays both the RMSE calculated on MinMax normalized data and unnormalized data due to the variation in metrics used in existing papers (D'Agostino et al., 2021; Liu et al., 2020).

The model performance achieved on the test set is very similar to the validation loss achieved during training. The 200 input stride 1 model achieves the best test performance closely followed by the 200 input stride 5 model. Interestingly, the relative differences in model performance are more evident in the 10.4 second ahead predictions than in the 20 second ahead prediction. This indicates that the best models primarily perform better in the earlier time steps. Figure 4.2 confirms this finding and shows that the model performance becomes increasingly similar after time step 15.

Table 4.2: Model evaluation test RMSE statistics on MinMax scaled predictions and unscaled predictions

Model	RMSE MinMax (10.4 sec / 13 steps)	RMSE MinMax (20 sec / 25 steps)	RMSE unscaled (10.4 sec / 13 steps)	RMSE unscaled (20 sec / 25 steps)
50 input stride 5	0.04717	0.07110	0.76604	1.15472
50 input stride 1	0.04573	0.07050	0.74271	1.14507
100 input stride 5	0.04352	0.06827	0.70674	1.10874
200 input stride 5	0.03810	0.06538	0.61888	1.06181
200 input stride 1	0.03679	0.06449	0.59757	1.04744
400 input stride 10	0.04520	0.06852	0.73419	1.11291

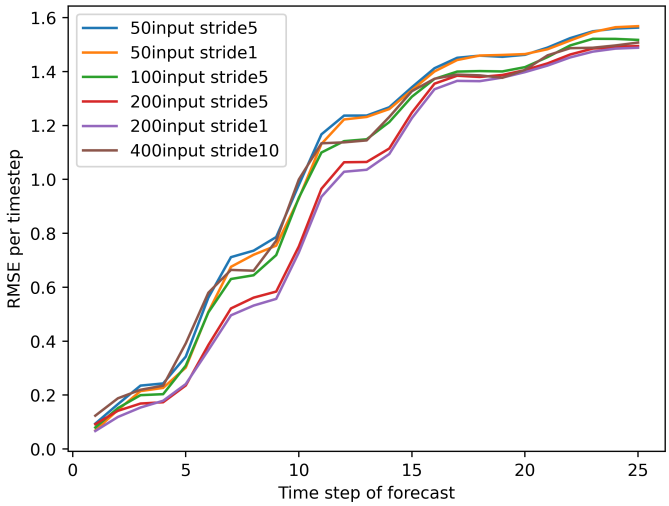

Figure 4.2: Visualization of RMSE unscaled per time step for trained models calculated with Equation 3.3

To analyze the now-casting abilities of the model, a set of 9 selected prediction versus actual plots is displayed in figure 4.3. Importantly, these plots only show 50 percent of the input sequence (100 out of 200 time steps) to make them easier to interpret. The selection is based upon 9 evenly spaced positions in the test set and thus are selected in quasi-random nature. Notably, the predictions at the peaks and valleys (highest and lowest values in RefZ movements) of the time-series are predicted higher and deeper than the observed pattern. This is not problematic because in helicopter and off-shore vessel multi-body operations it is preferred to over-estimate the peaks and valleys. The reason for this is that under-estimating a peak or valley might lead to unintended contact. In addition, the predicted versus actual plots highlight the accuracy of the ten second ahead predictions and underline the difficulty of predicting further ahead. This is in line with what is expected from the previous analyses and *a priori* assumptions. Finally, it becomes apparent that the model has difficulty predicting quickly altering RefZ directions and performs better at predicting more continuous movements.

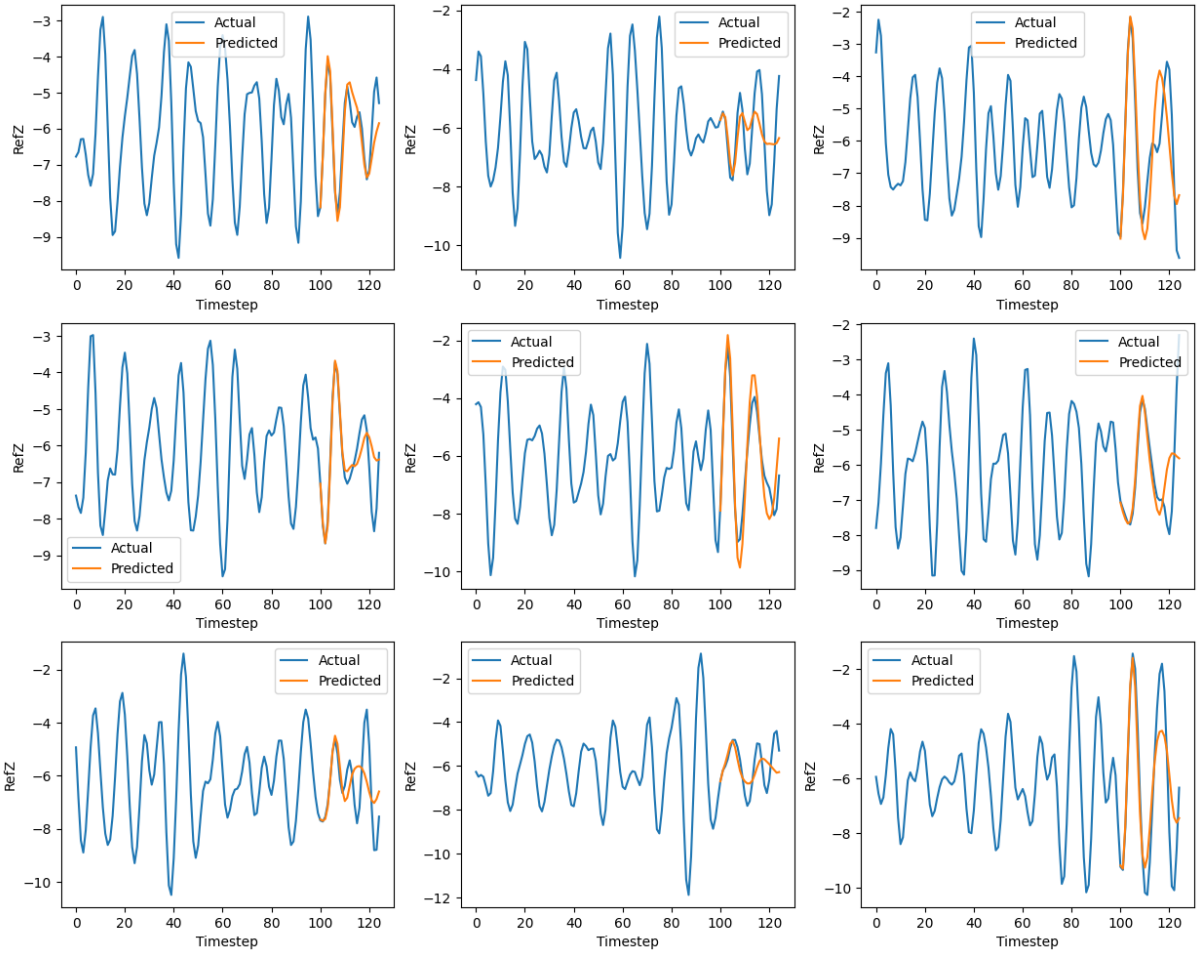

Figure 4.3: Predicted versus actual plots

Finally, an important metric for assessing the now-casting performance of a model is to determine whether the prediction errors are normally distributed around a mean of 0 (D'Agostino et al., 2021). This is useful to determine whether the predictions of the model tend to over- or under-estimate. If the model is biased in this way, the model requires additional training. This might appear counter intuitive in relation to the previous paragraph. However, if the mean is not centered around 0 the model tends to over- or under-estimate from every position in the time-series and not only at the peaks and valleys. In this case, the prediction errors of the model are normally distributed around a mean of 0 as visualized in figure 4.4. This underlines the good quality of the predictions of the model.

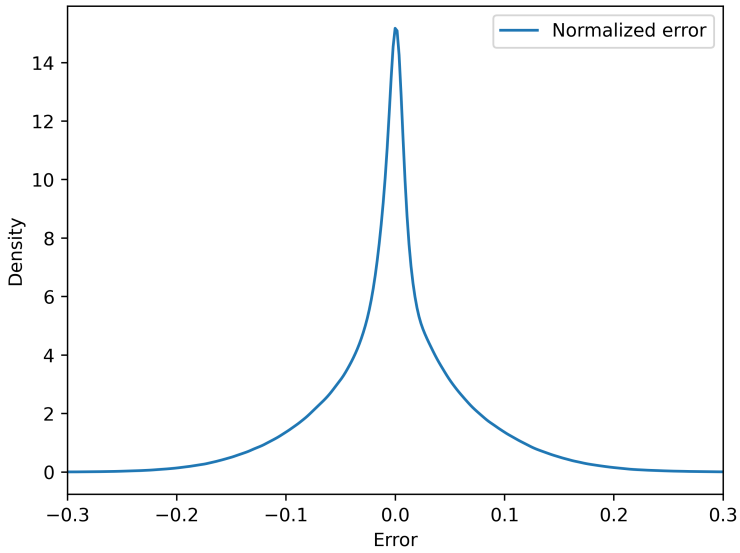

Figure 4.4: Density plot of the distribution of the prediction errors

5
Conclusion & Discussion

Augmenting or replacing human judgement in helicopter/off-shore vessel operations requires accurate predictions of the movements of the vessel. A useful model for accurate time-series predictions is the LSTM model. Many efforts in the field use wave excitation information to make these predictions, which is surprising due to the high costs of wave detection systems for ships. For this reason, it is important to discover whether accurate predictions can be made by an LSTM model without using wave excitation information. To allow for accurate predictions, the correct inputs and finely tuned hyperparameters are key.

This paper demonstrates that an LSTM model performs excellent for 10 second ahead predictions and moderately for 20 second ahead predictions. Additionally, it substantiates the required input and hyperparameters to achieve these predictions. In particular, it is shown that dropout prevents the model from converging on the training data and surprisingly leads to more over-fitting. Moreover, contrary to *a priori* expectations, models trained with the Adam optimizer generalize better than models trained with SGD. This is a helpful conclusion since Adam also converges quicker, which further accentuates the usefulness of Adam in this application. Furthermore, it is evident that a model with 3 LSTM layers with 368, 368 and 240 neuron units per layer achieves the best performance. The used learning rate and batch size are 0.0005 and 64 respectively. In addition, increasing the amount of data that is fed to the model by increasing the input length and decreasing the stride is beneficial for the performance of the model. The optimal input for this model is shown to be 200 input length and stride 1 for both 10 and 20 second ahead prediction. However, increasing the amount of input does result in higher computational costs. Therefore, a trade-off exists between increasing the amount of input and training speed of the model. Finally, this research exhibits that LSTM models can achieve accurate prediction performance without requiring the usage of wave excitation information. Thus, this paper highlights the potential of LSTM for utilization in real-world applications that lack this information.

With regards to these real-world applications, it is important to note that the current prediction performance does not allow for usage of this model in efforts towards autonomous operations. However, this research does show that the quality of the predictions allow to augment the human judgement involved in helicopter/off shore vessel operations. Especially in 10 second ahead predictions the LSTM model performs adequately to assist landing deck operators. Even though this prediction performance is adequate, it should be considered that the current model only predicts one of the six ship motions and uses simulated data in merely one sea-state. As a consequence, further research into ship motion prediction with LSTM models should focus on the following three subject areas. First, future research should focus on predicting ship motions based on real world data instead of simulated data. Second, to be applied to real-world applications, it is key that the model is able to handle several sea-states. Third, it is important to research prediction of all 6 ship motions at the same time *without* wave excitation information, which has already been shown to be effective *with* the usage of wave excitation information.

Equally important, it should be noted that the tuning process is guided by a predetermined search-space. Even though this search-space is defined according to existing literature and experimentation,

a neural network can most often be tuned further (e.g. learning rate scheduling, usage of layer normalization, etc.). Nonetheless, this research attempted to consider as many hyperparameter options as possible with the limited time available. The same holds for the number of different input configurations that are analysed. With more computational resources and time, more high input length models with low stride could be examined. Future research could investigate the effects of these changes. Moreover, in the final tuning cycle, this research switched from Bayesian Optimization to the Hyperband tuning algorithm to try out several model configurations more efficiently. This tuning cycle showed that the Hyperband tuning algorithm was not only more efficient, but also achieved superior performance. The superior performance of the Hyperband tuning algorithm was in contrast to the *a priori* assumptions and could have allowed this research to mitigate the increased computational costs at larger amounts of input. Hence, future research should utilize the increased efficiency and effectiveness of the Hyperband tuner to speed up tuning and training. Finally, this paper shows that the best performing models use 2-3 layers, whereas, most existing papers in ship motion predictions with LSTM use models with one layer. Therefore, future research could benefit from trying more complicated models than those applied previously.

Due to the simulated and non-personal nature of the data, ethical challenges are less prevalent in this research. On the other hand, the simulated nature does result in the necessity to first test the model performance on actual data before being used in augmenting human judgement. Additionally, with simulated data, it is important that the data generating mechanisms are modelled in a correct fashion and with integrity. The data is simulated by the Maritime Research Institue Netherlands (MARIN), whom accrued over 85 years of modelling and simulation capabilities and therefore the data is expected to fulfill these requirements.

References

Bengio, Y., Simard, P. & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, *5*(2), 157–166.

Bluche, T., Kermorvant, C. & Louradour, J. (2015). Where to apply dropout in recurrent neural networks for handwriting recognition? *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, 681–685.

Cheng, X., Li, G., Skulstad, R., Major, P., Chen, S., Hildre, H. P. & Zhang, H. (2019). Data-driven uncertainty and sensitivity analysis for ship motion modeling in offshore operations. *Ocean Engineering*, *179*, 261–272.

Cooijmans, T., Ballas, N., Laurent, C., Gülçehre, Ç. & Courville, A. (2016). Recurrent batch normalization. *arXiv preprint arXiv:1603.09025*.

D'Agostino, D., Serani, A., Stern, F. & Diez, M. (2021). Recurrent-type neural networks for real-time short-term prediction of ship motions in high sea state. *arXiv preprint arXiv:2105.13102*.

Duan, S., Ma, Q., Huang, L. & Ma, X. (2019). A lstm deep learning model for deterministic ship motions estimation using wave-excitation inputs. *The 29th International Ocean and Polar Engineering Conference*.

Dumitru, C. & Maria, V. (2013). Advantages and disadvantages of using neural networks for predictions. *Ovidius University Annals, Series Economic Sciences*, *13*(1).

Elsworth, S. & Güttel, S. (2020). Time series forecasting using lstm networks: A symbolic approach. *arXiv preprint arXiv:2003.05672*.

Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R. & Schmidhuber, J. (2016). Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems*, *28*(10), 2222–2232.

Hochreiter, S. & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780.

Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., Fischer, I., Wojna, Z., Song, Y., Guadarrama, S. et al. (2017). Speed/accuracy trade-offs for modern convolutional object detectors. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 7310–7311.

Janiesch, C., Zschech, P. & Heinrich, K. (2021). Machine learning and deep learning. *Electronic Markets*, *31*(3), 685–695.

Jayalakshmi, T. & Santhakumaran, A. (2011). Statistical normalization and back propagation for classification. *International Journal of Computer Theory and Engineering*, *3*(1), 1793–8201.

Karpathy, A. (2015). The unreasonable effectiveness of recurrent neural networks. http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Khan, A., Bil, C., Marion, K. & Crozier, M. (2004). Real time prediction of ship motions and attitudes using advanced prediction techniques. *Congress of the International Council of the Aeronautical Sciences. International Council of the Aeronautical Sciences*.

Khan, A., Bil, C. & Marion, K. E. (2005). Ship motion prediction for launch and recovery of air vehicles. *Proceedings of OCEANS 2005 MTS/IEEE*, 2795–2801.

Laurent, C., Pereyra, G., Brakel, P., Zhang, Y. & Bengio, Y. (2016). Batch normalized recurrent neural networks. *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2657–2661.

Li, G., Kawan, B., Wang, H. & Zhang, H. (2017). Neural-network-based modelling and analysis for time series prediction of ship motion. *Ship technology research*, *64*(1), 30–39.

Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A. & Talwalkar, A. (2017). Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research*, *18*(1), 6765–6816.

Liu, Y., Duan, W., Huang, L., Duan, S. & Ma, X. (2020). The input vector space optimization for lstm deep learning model in real-time prediction of ship motions. *Ocean Engineering*, *213*, 107681.

Mandic, D. & Chambers, J. (2001). *Recurrent neural networks for prediction: Learning algorithms, architectures and stability*. Wiley.

Reddi, S. J., Kale, S. & Kumar, S. (2019). On the convergence of adam and beyond. *arXiv preprint arXiv:1904.09237*.

Selvin, S., Vinayakumar, R., Gopalakrishnan, E., Menon, V. K. & Soman, K. (2017). Stock price prediction using lstm, rnn and cnn-sliding window model. *2017 international conference on advances in computing, communications and informatics (icacci)*, 1643–1647.

Snoek, J., Larochelle, H. & Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, *25*.

Sørensen, A. J. (2011). A survey of dynamic positioning control systems. *Annual reviews in control*, *35*(1), 123–136.

Sun, Q., Tang, Z., Gao, J. & Zhang, G. (2022). Short-term ship motion attitude prediction based on lstm and gpr. *Applied Ocean Research*, *118*, 102927.

Tanaka, Y. (2018). Active vibration compensator on moving vessel by hydraulic parallel mechanism. *International Journal of Hydromechatronics*, *1*(3), 350–359.

Triantafyllou, M. S. & Bodson, M. (1982). Real time prediction of marine vessel motions, using kalman filtering techniques. *Offshore Technology Conference*.

Wang, J., Xu, J. & Wang, X. (2018). Combination of hyperband and bayesian optimization for hyperparameter optimization in deep learning. *arXiv preprint arXiv:1801.01596*.

Wang, K., Li, K., Zhou, L., Hu, Y., Cheng, Z., Liu, J. & Chen, C. (2019). Multiple convolutional neural networks for multivariate time series prediction. *Neurocomputing*, *360*, 107–119.

Yang, X., Pota, H., Garratt, M. & Ugrinovskii, V. (2008). Ship motion prediction for maritime flight operations. *IFAC Proceedings Volumes*, *41*(2), 12407–12412.

Yumori, I. (1981). Real time prediction of ship response to ocean waves using time series analysis. *OCEANS 81*, 1082–1089.

Yunpeng, L., Di, H., Junpeng, B. & Yong, Q. (2017). Multi-step ahead time series forecasting for different data patterns based on lstm recurrent neural network. *2017 14th web information systems and applications conference (WISA)*, 305–310.

Zhang, Z. (2018). Improved adam optimizer for deep neural networks. *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*, 1–2.

Zhao, M., Zhang, J. & Rashid, M. H. (2021). Predicting the drift position of ships using deep learning. *The 2nd International Conference on Computing and Data Science*, 1–5.

Zhao, X., Xu, R. & Kwan, C. (2004). Ship-motion prediction: Algorithms and simulation results. *2004 IEEE international conference on acoustics, speech, and signal processing*, *5*, V–125.

A
Appendix A: Code

The annotated code for this paper is found at this Github: https://github.com/QuintenSand/Thesis-Marin. The Github page is a combination of the efforts for three thesis projects and therefore some code that is found on this Github page is not used in the paper that lies before you. Below an explanation is provided on which code is used in this paper.

The LSTM_tutorial Python notebook provides an explanation on all steps that are used to tune and train the LSTM model. This notebook uses several functions that are found in the marin_functions directory. First, the data aggregation file is used to sub-sample the data from 0.2 second time steps to 0.8 second time steps. Second, the data preparation file contains the functions that are used to create the examples in the training, test and validation sets. Third, the model forecast file contains the functions that are used to create predicted vs actuals plots and is used to compute RMSE statistics. Furthermore, this marin_functions directory also contains the code that is used to create the visualizations that are used in this paper. The creation of these visualization is divided into two files. Namely, a file for the visualizations of the data and method sections, and a file for the visualizations of the results section.

B
Appendix B: Preliminary tuning results

B.1. First tune cycle

The first tuning cycle was performed with the following search space and settings. Importantly, this tuning cycle was performed with only one simulation as input and a re-sampling combining two timesteps to 0.4 seconds per timestep. Moreover, in this tuning cycle the optimizer was held constant as Adam. An analysis of the results of this tuning cycle is provided below.

- Number of layers: between 1 and 6
- Units per layer: between 16 and 256
- Dropout per layer: between 0 and 0.5
- Learning rate: between 0.01 and 0.0001
- Batch size: between 16 and 128

Interestingly, as shown in table B.1 the tuning algorithm clearly converges to the same set of hyper parameters. It becomes apparent that a one layer model with the maximum number of nodes performs best. Additionally, this tuning cycle shows that dropout is not beneficial in this application. The reason for this is that it prevents the model from converging on the training data and even results in a higher validation loss. Finally, nearly all models in the top 10 have a learning rate of 0.0001 and a batch size of 16. However, it is important to note that this tuning cycle is only performed on one simulation, uses a different level of re-sampling (0.4 sec per timestep instead of 0.8 sec) and only attempts to predict 10 seconds into the future.

Table B.1: Tuning trial summary top 10 ordered by validation loss selection from 21 trials from Bayesian Optimization

Rank	Validation loss	Number of layers	Units/layer	Dropout/layer	Learning rate	Batch Size
1	0.00371	1	256	0	0.0001	16
2	0.00387	1	240	0	0.0001	16
3	0.00388	1	256	0	0.0001	16
4	0.00391	1	256	0	0.0001	16
5	0.00399	1	256	0	0.0001	16
6	0.00428	1	256	0	0.0001	16
7	0.00439	1	256	0	0.0001	32
8	0.00479	1	32	0	0.0001	16
9	0.00480	1	256	0	0.01	16
10	0.00482	1	256	0	0.0001	32

B.2. Second tune cycle

During the second tuning cycle the improved version of creating training examples from multiple files is used and therefore much more data is used for the tuning process. Additionally, in this tuning cycle the re-sampling is 0.8 seconds per timestep and a stride of 5 is used. This stride of 5 indicates that there are 5 timesteps between every training example. Importantly, the input length is 50 and the output length is 25, which translates to 40 and 20 seconds respectively.

Noting that dropout had adverse effects on model convergence and over-fitting, dropout is excluded from the search-space. In addition, the first tuning cycle shows that the tuning does not converge on a high number of layers and thus this search-space is reduced to a maximum of 3 layers. Moreover, noting the highly increased amount of data used, the search-space with regards to batch size is increased to a maximum of 1024 and a higher minimum of 64. As became evident in the first tuning cycle, the maximum number of nodes was used in most of the models in the top-10. Therefore, in this tuning cycle the maximum number of nodes is increased from 256 to 512. Finally, this tuning cycle includes a choice of optimizer between Adam and SGD. The search-space is summarized below:

- Number of layers: between 1 and 3
- Units per layer: between 16 and 512
- Optimizer: Adam or SGD
- Learning rate: between 0.01 and 0.0001
- Batch size: between 64 and 1024

The results of this tuning cycle are displayed in table B.2. From this summary, it becomes apparent that the best performing models all have Adam as an optimizer and use a learning rate of 0.0001. Furthermore, most well performing models have a batch size of 64. In contrast with the first tune cycle, now the best performing models have multiple layers, which is in line with expectations because longer input and output lengths are used and the timesteps changed from 0.4 to 0.8 sec per timestep. Lastly, the number of units used per layer vary quite a lot and thus an additional tune cycle is required. In this final tune cycle Hyperband is used because Hyperband is more efficient in trying a large number of possible models and the Bayesian Optimization tuning cycle shows us the desired performance level.

Table B.2: Tuning trial summary top 10 ordered by validation loss selection from 29 trials from Bayesian Optimization

Rank	Validation loss	Number of layers	Units/layer	Optimizer	Learning rate	Batch Size
1	0.00515	3	512, 16 and 256	Adam	0.0001	64
2	0.00516	3	192, 384 and 208	Adam	0.0001	64
3	0.00521	1	304	Adam	0.0001	64
4	0.00521	2	272 and 16	Adam	0.0001	64
5	0.00522	3	352, 208 and 16	Adam	0.0001	256
6	0.00522	2	336 and 16	Adam	0.0001	64
7	0.00525	3	16, 176 and 16	Adam	0.0001	64
8	0.00537	3	96, 176 and 112	Adam	0.0001	512
9	0.00547	1	16	Adam	0.001	126
10	0.00549	1	16	Adam	0.01	64