Neural Network for SSCV Hydrodynamics

A Study on the Potential of a Neural Network Based Model in Predicting Hydrodynamic Behavior of a Semi-Submersible Crane Vessel

A.J. Haenen

TU Delft

Neural Network for SSCV Hydrodynamics

A Study on the Potential of a Neural Network Based Model in Predicting Hydrodynamic Behavior of a Semi-Submersible Crane Vessel

by

A.J. Haenen

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended on Sep 25, 2018

Student number:	4064275	
Project duration:	Dec 18, 2017 – Sep 25, 2018	
Thesis committee:	Prof. dr. ir. A.P. van 't Veer,	TU Delft
	Dr. E. Lourens,	TU Delft
	Dr. ir. P. Naaijen,	TU Delft
	Ir. R. de Bruin,	Heerema Marine Contractors
	Ir. I. Rentoulis,	Heerema Marine Contractors

Preface

Dear reader,

Thank you for taking the time to read my graduation thesis. I was in search of a project that would give me the opportunity to learn something completely new to me. I found that opportunity at HMC and that something turned out to be machine learning. The past nine months have been educational and enjoyable as well as tiring and frustrating.

I want to express my graditude towards Professor van 't Veer. It took some convincing, but I am really glad you allowed me to research the potential of neural networks in predicting hydrodynamic behavior for my graduation project.

Yannis and Ruben, I couldn't have wished for better supervisors than you guys. I really appreciate that despite your endless enthousiasm, you always remained critical. By doing this, both of you have really contributed to my work. I wish you all the best.

Thank you Mingcheng, you were always helpful when I had questions regarding machine learning and Python. I would also like to thank you, Marcelo, Ruben and Yannis for attending our weekly meetings and providing me with new insights. Furthermore I would like to thank Radboud, Eelco, Job and Geert for your enthousiasm and critical attitude in the expert meetings during my first months at Heerema.

Eliz-Mari, thank you for your feedback during our meetings. It was really helpful to talk to someone outside of the company. You made me question my research path and kept me from exploring dead ends.

I would also like to thank my family and friends for their support and encouragements during this graduation project. This includes my fellow graduation students in sharing experiences, drinking bakkies pleur and making sure that this thesis work has been more than solitary research. Last I would like to thank my housemates Roel, Bernd and Tim for putting up with my increasing sloppiness and grumpiness. Also for convincing me to go out for a beer when I really needed one and your rubber spines when I had to convince you guys.

A.J. Haenen
Delft, September 2018

Abstract

Heerema Marine Contractors (HMC) is a contractor in the international offshore oil, gas and renewables industry. It is specialized in transporting, installing and removing large offshore facilities. HMC operates three crane vessels. Two of which are semi-submersibles (Thialf and Balder), the other is the monohull Aegir. A third semi-submersible, the Sleipnir, is currently under construction.

To ensure safe operations, make accurate fatigue predictions and extend operational windows, HMC relies on vessel motion calculations. Currently, vessel motion estimations are based on response amplitude operators calculated by the diffraction software package WAMIT. As HMC cannot rely on diffraction software in case of non-linear vessel motions, the need for a method capable of capturing non-linear effects arises.

The goal of this study is to determine the potential of a neural network based model in predicting hydrodynamic behavior of semi-submersible crane vessels. Hindcast weather data, vessel motion measurements and model test data are used to train several different neural network architectures. The research into the potential of neural networks in predicting hydrodynamic behavior is split into two main categories: the frequency domain and the time domain.

Within the frequency domain, wave forecasts can be used to predict a response spectrum. The neural network in this case acts as a conventional RAO. In an artificial environment, four architectures are tested and the results show that neural networks are able to make accurate predictions in a fully linear environment. When tested on project data, where the vessel sails at operational draft, the neural network predictions shows a slightly higher accuracy than the diffraction based predictions for the specific test case. Another network is tested on transit data, where the vessel sails at an inconvenient draft. The results from these tests show that there is potential for a neural network to be used as a substitute for Response Amplitude Operators.

The time domain models focus on predicting ship response based on surface height signals and/or hindcast vessel motion measurements. The first model is trained and tested on model test data from an SSCV. The input of the neural network is surface height measurements and the output is pitch motion prediction. The model shows that it is capable of predicing both first and second order pitch motions. Another time domain model has MRU roll measurements as input and it tries to predict the future 60 seconds of roll motion. Many network topologies and optimizer settings are tested but none are capable of predicting future motions.

List of Abbreviations

Adam Adaptive moment estimation

AI Artificial Intelligence

ANN Artificial Neural Netork

API Application Programming Interface

CFD Computational Fluid Dynamics

CNN Convolutional Neural Network

DCV Deepwater Construction Vessel

DP Dynamic Positioning

FEM Finite Element Method

FMS Fatigue Monitoring System

HMC Heerema Marine Contractors

LSTM Long Short-Term Memory

MLP Multi-Layer Perceptron

MRU Motion Reference Unit

NN Neural Network

RAO Response Amplitude Operator

RBF Radial Basis Function

ReLU Rectified Linear Unit

RNN Recurrent Neural Network

SDA Significant Double Amplitude

SGD Stochastic Gradient Descent

SSCV Semi-Submersible Crane Vessel

List of Symbols

a_{kj}	added mass coefficients
b_i	network biases
b_{kj}	damping coefficients
$\mathbf{b}^{(k)}$	network bias matrices
d	learning rate decay
D	Draft
f	frequency (Hz)
g	gravitational acceleration
h	water depth
h_0	distance from mean free surface height to seabed
H_s	significant wave height
I_i	node value after summation
J	network cost
k	wave number
^{k}	layer number
l	network depth
L	network loss
m_0	zeroth moment
n	layer size
S_s	Ship motion energy spectrum
S_ζ	Wave energy spectrum
t	time
T	period
T_p	peak period
U	forward speed
v_t	parameter update vector
w_{ij}	network weights
$\mathbf{W}^{(k)}$	network weight matrices
$\mathbf{x}$	network input vector
y_i	node output value
$\mathbf{Y_n}$	network output vector
$\mathbf{Y_t}$	network target output vector
z	coordinate with positive z-axis pointing upwards

ζ_0	undisturbed wave amplitude
η	learning rate
θ	represents all network parameters (weights & biases)
ξ_j	complex amplitude of the body oscillatory motion corresponding to the j-th degree of freedomn
φ	complex velocity potential
φ_R	radiation potential
φ_D	diffraction potential
φ_0	incident wave potential
φ_S	velocity potential representing the scattered disturbance of the incident wave by the body fixed at its undisturbed position
φ_j	unit-amplitude radiation potential corresponding to the j-th degree of freedom
ω	(wave) frequency (rad/s)
ω_e	encounter frequency

Contents

1 Introduction

Heerema Marine Contractors (HMC) is a contractor in the international offshore oil, gas and renewables industry. It is specialized in transporting, installing and removing large offshore facilities. HMC operates three crane vessels. Two of which are semi-submersibles (Thialf and Balder), the other is the monohull Aegir. A third semi-submersible, the Sleipnir, is currently under construction.

The initial incentive for this study was in line with earlier HMC studies to provide better vessel motion predictions based on weather data. During a Balder transit from West-Africa to Europe, it was noted that the predictions made by the fatigue damage prediction tool overestimated the wave-induced fatigue damage as was measured during the transit. This inability pertains usually to the apparent nonlinearities of the problem, such as in the case of a semi-submersible vessel operating with its pontoons close to the free surface. In diffraction software, this situation creates huge standing waves, which when a linear transfer function is applied, results in large predicted vessel motions. In reality, these standing waves are limited in height because shallow water effects cause wave breaking. Other limitations of the software package being used are also present, such as the inability to capture forward speed effects. This has been an incentive for HMC put effort into improving its fatigue damage prediction procedures. For example by identifying new RAOs for the transit draft. In the meantime, HMC has started a big monitoring effort by equipping her vessels with several sensors. Depending on the vessel, these sensors monitor and log continuously response data such as accelerations, motions and stresses, as well as loading conditions such as the vessel draft. The presence of a large amount of data and the need for a replacement method that is able to capture non-linear effects is reason to look into neural networks.

The hydrodynamic predictive capabilities of neural networks may extend beyond the use as a direct substitute of RAOs. Therefore this study is also determines the potential of neural network based models in two time domain applications. The first of which is the real time estimation of vessel motions based on a hindcast surface height signal. The network used for this application is trained on model test data. The second is the prediction of vessel motions based on the hindcast measurements of these motions.

2
Research Setup

2.1. Problem statement

During transit, the current HMC hydrodynamic prediction model fails to represent reality because the theoretical models being used do not capture the governing physics. The current prediction model is based on RAO's calculated by WAMIT. These predictions of hydrodynamic behavior are necessary for fatigue prediction, risk assessment etc.

There are two main reasons why the hydrodynamic prediction model fails to represent reality; because of a shallow draft during transit and due to forward speed. The pontoons close to the surface introduce non-linearities to the hydrodynamical problem and RAOs calculated by WAMIT do not include forward speed.

The non-linearities are caused by two phenomena that occur in shallow draft. Standing waves form above the pontoons. In reality this would cause wave breaking and turbulence which introduces non-linearities. However because these non-linear effects lack in linear wave theory, large standing waves occur and motions are often overpredicted. Secondly, in high sea-states, the pontoons can become partly submerged. This means that the waterplane area is non-constant and there are non-linearities in the stiffness.

There are calculation methods (CFD) that can capture these non-linearities, but the computational cost of these calculations is too high to provide a suitable alternative to the current hydrodynamic prediction model.

The amount of available data for the Balder provides the opportunity to test a neural network based hydrodynamical model.

2.2. Research Question

The research question for this graduation project is *"What is the potential of a neural network focused method in assessing hydrodynamic behavior based on hindcast and measurement data?"*. This is a broad question and asked in this form with the intent of looking further than an alternative to the RAO when sailing in inconvenient drafts.

There is an obvious split regarding the potential hydrodynamic applications of neural network based model. First the frequency domain, which focuses on either providing an alternative to the RAO or calculating an accurate RAO. Second the time domain, wherein short time response predictions can be made based on hindcast measurement data.

2.2.1. Secondary Research questions

Can a neural network based model be used as an alternative to the RAOs calculated by WAMIT?

This is the driving question behind the research in the frequency domain. A neural network in this form would serve as a direct alternative to the RAOs by WAMIT. The question is studied in an artificial environment, a project environment (operational draft) and a transit environment (inconvenient draft).

Can a neural network based model be used as an alternative to an RAO in the time domain?

Time domain predictions are better suited for neural network based models than high-input, high-output calculations. Therefore, it is expected that a time domain model can achieve better results than the frequency domain model. If this is indeed the case, it might also be possible to use the time domain model to reconstruct an RAO. An extra question is thus: *Can an RAO be reconstructed using a working time domain model?*.

To what extend is it possible to make predictions using a neural network based model using hindcast measurements?

Ocean waves have a random nature but are also composed of clear patterns in the short term. Autocorrelation can be used to find out to what extend hindcast response is related to future response. However, autocorrelation methods assumes a linear dependency on its own previous values. A neural network may be able to establish more 'out-of-the-box' correlations.

2.3. Research Approach

As mentioned, this research aims to determine the potential of neural networks in both the frequency and the time-domain. There is a difference in research approach for these domains.

For the frequency domain the approach to provide an answer to the research questions involves the development of three models with an increasing complexity. The first model will be based on the model calculations of the current hydrodynamic prediction model from HMC. The second model will be based on hind-cast environmental data and measurements from accelerometers, but only for the cases that the current model is capable of predicting accurately. This means avoiding inconvenient drafts and foreword speed. Therefore measurement data from different projects will be used. The third model will also be based on measurements and will include all cases, so also the non-standard, non-linear cases. For this model, transit data can be used.

The time domain approach focuses on short time response prediction. Measurement data from Motion Reference Units (MRU) provide a ship motion hindcast with which future ship motions may be predicted. Additionally there are model tests available where both free surface motion and ship motion are measured. Two separate networks are assessed in the time domain. The first is there to provide a time domain alternative to an RAO by using wave input and providing vessel motion predictions. The second uses hindcast vessel motions to make predictions of future vessel motions to see if a neural network can establish useful correlations.

3

Literature Study

3.1. Neural Networks

The field of neural networks and machine learning has risen in popularity in the last decade. This is mainly because computers that can perform demanding computational tasks have become available and, perhaps even more important, affordable. However, the field is much older than that. The first computational model for neural networks was created in 'A logical calculus of the ideas immanent in nervous activity' by McCulloch and Pitts [16] in 1943. Most of the fundamental research was published in second half of the twentieth century.

Today there are a number of open-source machine learning libraries developed by tech-companies, universities and hobbyists. Because of these open-source libraries and affordable hardware, a lot of innovation is being done by enthusiasts and start-ups, with blogs and fora often being much more common as learning platforms than books or classes.

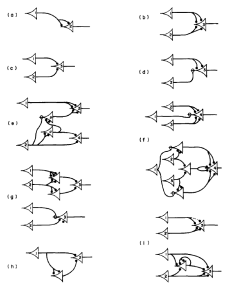


Figure 3.1: image from 'A logical calculus of the ideas immanent in nervous activity' [16]

3.2. Neural Networks in Offshore Engineering

Offshore engineering problems can be approached in two ways. Either a knowledge-based or a data-driven approach. Neural network based models are a recent addition to the latter. Research projects and early applications are realized in a wide range of fields within the offshore industry. Fields such as forecasting environmental parameters, fatigue prediction, structural response estimation, etc [4]. A few examples in these different fields are discussed in this section on a high level.

ANNs can replace statistical models used for external loads on offshore structures. These structures are exposed to waves, wind, earthquakes, impacts and operational loads. In a 2013 paper [30], long-term strain measurements on a jacket-type structure are used to train a prediction model using neural network. The use of this kind of prediction model provides a

relatively simple way to incorporate measurement data into future predictions and it factors in all encountered contributions to strain responses automatically. However, this also reveals one of the major drawbacks of neural networks. Major contributions to fatigue damage can be phenomena that happen once every couple of years or even less frequent. This can fall beyond the scope of the measurement hindcast data and are thus not part of the training data on which the network bases its predictions.

Fatigue damage monitoring systems could benefit from continuous time-domain analysis with neural network based models but these are often not viable yet because of a lack of computational power [12]. If Moore's law continues this would however become possible within a few years. For now though, fatigue monitoring systems that are based on neural networks are often still bound to the spectral approach.

One of the research fields applying neural networks within the offshore industry is wave prediction. One example is the use of an ANN to provide an alternative to wave spectra that use empirical relationships like Pierson-Moskowitz, Jonswap and Scotts. Decisions on which spectrum to use can be quite subjective and often these spectra fail to generalize actual site conditions [17]. ANNs can be a viable alternative in estimating these spectral shapes. The same is true for nonlinear interaction models for wind wave spectra [27].

Like the field of wave spectra, other fields in the industry also rely heavily on statistical methods. A 2011 paper provides a neural network based method as an alternative to the Fokker-Planck approach [5]. The Fokker-Planck equation is a partial differential equation that describes the time evolution of the probability density function of something under influence of forces.

Another possible application of neural networks within the offshore sector is workability analysis. For example the coupling of a finite element method and an artificial neural network to predict safe sea-states [29]. This has been done in a 2009 study. Here, a table of safe sea-states was generated using a finite element dynamic analysis and sea-state predictions were made by a neural network. These sea-state predictions used an input of previous significant wave heights and peak periods of preceding consecutive 3-hour periods in addition to a few fuzzy values that are related to the season.

4

Background

This chapter discussed the theoretical background of this thesis. As this research combines the fields of ship hydrodynamics and artificial neural networks, it is necessary to provide information for both.

4.1. Established Method for Assessing Ship Motions

The conventional calculations convert wave spectra to ship response spectra for different degrees of freedom. In these calculations, Response Amplitude Operators (RAO's) act as a transfer function between these two types of spectra. This section will provide a background on RAO's, the software which is used within HMC (WAMIT), the limitations of this software and previous studies to help overcome these limitations.

4.1.1. RAOs

As mentioned, an RAO acts as a transfer function between a wave spectrum and a vessel response spectrum. An RAO, like any transfer function, describes the linear relation between the input and the output. The RAO is a function of frequency and heading. There are multiple methods of assessing RAO's for ships. CFD analysis and model tests can be used to calculate RAOs accurately but these are often too expensive and time consuming. More common calculation methods are based on linear potential theory. Within HMC, the diffraction software package WAMIT is used to calculate the RAOs. WAMIT solves the boundary value problem in terms of a complex potential describing the flow. Linearizing the problem enables the decomposition of the velocity potential φ into the radiation and the diffraction components [28][15].

$$S_s(\omega) = \int_0^{\infty} \int_0^{2\pi} |RAO(\omega,\theta)|^2 \cdot S_\zeta(\omega) d\theta d\omega \tag{4.1}$$

4.1.2. Limitations of diffraction software

In the case of 'inconvenient' or 'awkward' draft, HMC has problems assessing the vessel motion behavior. These cases typically occur during transit. Transit drafts for the ##### lie between 13 and 15 m. When sailing at these drafts, the pontoons of the semi-submersible are close to the free surface. This stops the current model from being physically consistent. This can be concluded due to to the differences between hydrodynamic reaction forces and vessel inertia, the scattered and incident wave exciting forces and also by comparing the characteristic values of 15m and 22m draft [21].

The limitations of the diffraction software calculations at an inconvenient draft are caused by the significant wave elevations on top of the pontoons. These are present in the calculations but physically they cannot exist. At 15m draft the water column above the pontoons is 3m which limits the wave height to only a few meters due to the depth limited wave height [22]. Diffraction software however predicts wave elevations up to 9m for a 1m incident wave [21]. This large wave height is caused by resonant motions in the fluid between the pontoons and columns of the semi-submersible. This resonant effect can also occur in reality, however the wave height is limited due to turbulence and wave breaking. As there is a linear relationship between ship motion and wave height in the WAMIT, unrealistically high wave elevation such as the 9m wave for a 1m incident wave causes large differences between reality and the predictions based on diffraction software.

Another cause of the changes between reality and the predictions based on WAMIT is because of forward speed effect. The 3D panel method being used in WAMIT does not include forward speed. Instead, the assumption is made that the vessel response at forward speed is the same as the response to encounter frequencies [7].

$$\omega_e = \omega - \frac{U\omega^2}{g}\cos(\phi) \tag{4.2}$$

Previous HMC studies

The over-prediction of ship motions while in transit has been an issue for years and applying neural networks is certainly not the first attempt to improve predictions. Earlier work includes a thesis[21] which goal was to improve fatigue damage prediction during transits. A solution was found in identifying new RAOs for the transit draft using measurements and hindcast weather data. This was done by calibrating the RAO using its definition, which is the transfer function between the excitation and the response. While assuming that the linearity holds.

4.2. Artificial Neural Networks

A brief introduction to neural networks is provided in this section. The different components of a neural network are discussed along with the basic principles of the training process.

4.2.1. Neural networks

Neural network, or artificial neural network (ANN) is a system based on the biological neural networks such as our nervous system. These networks are used as computing systems that allow for machine learning. Tasks for these computing systems can be image recognition, machine translation and medical diagnosis. In a maritime context they can be used for tasks like dynamic positioning and propeller design.

The main layout for ANNs can be seen in Figure 4.1. It is composed of many nodes that are connected to nodes in different layers. Each network consists of an input layer, an output layer and a number of hidden layers that can range from zero to dozens or even hundreds.

4.2.2. Keras & TensorFlow

Neural network modeling is done with the use of software libraries. There are multiple open source software libraries available that allow or are dedicated for machine learning. For this project Keras is used on top of TensorFlow.

Keras is an API for neural networks written in Python. It is capable of running upon several libraries: TensorFlow, CNTK and Theano. It is developed with a focus on enabling fast experimentation[1].

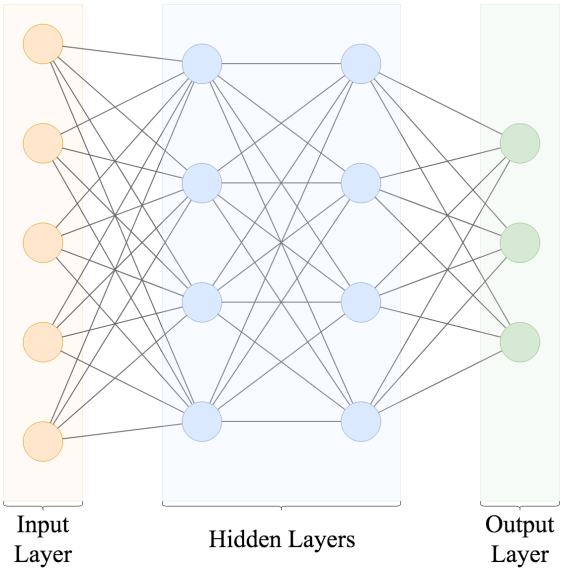


Figure 4.1: Illustration of an Artificial Neural Network

TensorFlow is an open source software library for high performance numerical computation. The library is developed within Google's AI organization. It supports machine learning and deep learning[26].

4.2.3. Nodes

An ANN is composed of nodes. Each node has a similar structure. As an input, it receives the connection 'strengths' from the nodes of the previous layer (with the input layer as an exception). The output of each node is the result of two mathematical computations. First is the summation, which is the sum of the product of all strengths and weights plus a certain bias (Eq. 4.3). The weights are values held by each connection between nodes and the biases are a constant that is held by each node.

$$I_i = \sum_j w_{ij} x_j + b_i \tag{4.3}$$

The second computation is the transfer, which is the application of a transfer function. In the machine learning field this transfer function is called the activation function. These subsequent computations through the entire network is called forward propagation (Figure 4.2).

$$y_i = f(I_i) \tag{4.4}$$

4.2.4. Activation functions

The output of each node in a network is defined by an activation function. The use of activation functions can give the network certain properties such as nonlinearities, range or better response to certain training methods. It is important to keep in mind the properties of the different activation functions. In practice however, the most suitable activation function is found mostly by trial and error. Some commonly used activation functions are presented in Figure 4.3.

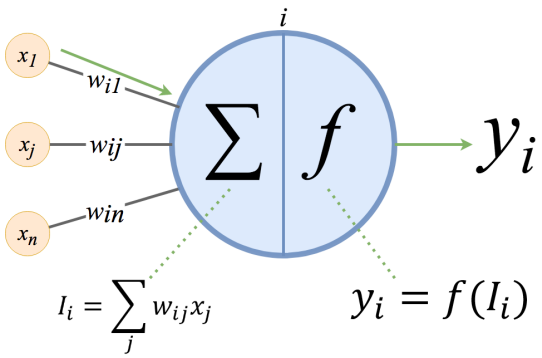


Figure 4.2: Forward propagation through node

4.2.5. Cost function

Before the ANN can learn to produce a desired output, there should be a quantification for the error. Here the loss function is introduced, this is part of the cost function or objective function. The loss function is a measure for the difference between the network output and the expected output (or 'ground truth'). As neural networks often have multiple outputs, there is need for something more comprehensive than $loss = networkoutput - expectedvalue$.

Keras provides more than a dozen 'standard' loss functions but it is also possible to implement a custom loss function. One of the most used loss functions is the quadratic loss function (Eq. 4.5). It is also known as the mean squared error (MSE). The important properties of the MSE is that the loss $L(w,b)$ is always positive and that it becomes very small in case of small errors and very large in case of large errors. These properties make it very useful in a lot of cases, but it can be susceptible to outliers in the dataset.

$$L_{MSE}(Y_n, Y_t) = \frac{1}{n} \sum_{i=1}^{n} (Y_n(i) - Y_t(i))^2 \tag{4.5}$$

The cost function or objective function is the value that flows back into the network via backpropagation (Section 4.2.6) to update the weight and bias matrices. It can consist of a single loss, but it can also be the sum of multiple losses in a batch (Section 4.2.8). Another addition to the cost function can be a regularizer (Section 5.2.2).

The computation of the cost function is performed by the use of a forward-propagation algorithm. This maps the parameters to the training loss $L(Y_n, Y_t)$. An example of such an algorithm is provided (Algorithm 1). It is the forward propagation through a typical neural network (Multilayer Perceptron) and the computation of the cost function.

$\mathbf{x}$	network input vector
$\mathbf{Y_n}$	network output vector
$\mathbf{Y_t}$	network target output vector
l	network depth
$\mathbf{W}^{(k)}$	weight matrices
$\mathbf{b}^{(k)}$	bias matrices
$\lambda\Omega(\theta)$	regularizer with θ containing all network parameters (weights and biases)
J	Cost
L	Loss

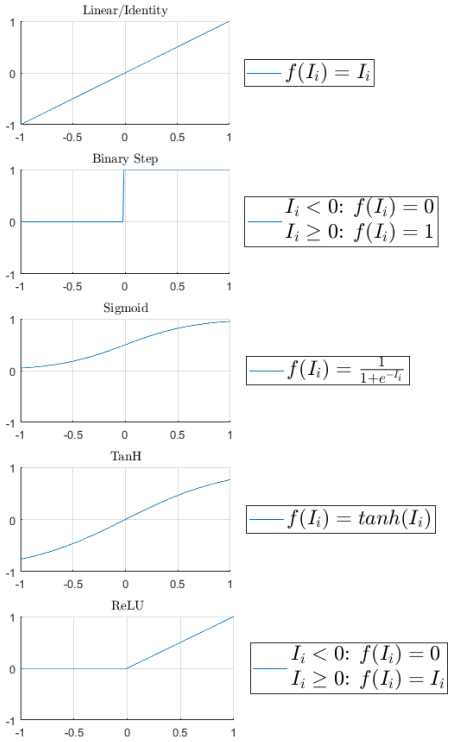


Figure 4.3: Common activation functions

Algorithm 1 Forward-Propagation Algorithm [6]

$\mathbf{h}^{(0)} = \mathbf{x}$
for $k = 1, ..., l$ **do**
  $\mathbf{a}^{(k)} = \mathbf{b}^{(k)} \mathbf{W}^{(k)} \mathbf{h}^{(k-1)}$
  $\mathbf{h}^{(k)} = f(\mathbf{a}^{(k)})$
end for
$\mathbf{Y_n}$
$J = L(\mathbf{Y_n}, \mathbf{Y_t}) + \lambda \Omega(\theta)$

4.2.6. Backpropagation

Using a neural network, information flows from the input layer to the output layer (forward propagation). This produces a prediction, or in the case of training a cost scalar. In order to train the network, the information has to flow back into the network. This is done by the back-propagation algorithm (Algorithm 2). Backpropagation is a means of computing the gradient of the cost function. This provides the gradient for the weight and bias matrices which can subsequently be altered by an optimization algorithm.

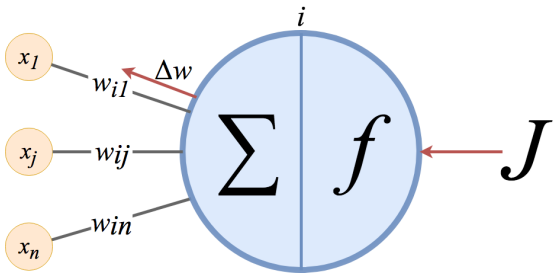


Figure 4.4: Backpropagation through node

Algorithm 2 Back-Propagation Algorithm [6]

Compute gradient on the output layer
$\mathbf{g} \leftarrow \nabla_{\mathbf{Y_n}} J = \nabla_{\mathbf{Y_n}} L(\mathbf{Y_n}, \mathbf{Y_t})$
for $k = l, l-1, \ldots, 1$ **do**
Convert gradient on the layer's output into gradient into the pre-nonlinearity activation, with element wise multiplication if f is element-wise:
$\mathbf{g} = \leftarrow \nabla_{\mathbf{a}^{(k)}} J = \mathbf{g} \odot f'(\mathbf{a}^{(k)})$
Compute gradients on weights and biases
$\nabla_{\mathbf{b}^{(k)}} J = \mathbf{g} + \lambda \nabla_{\mathbf{b}^{(k)}} \Omega(\theta)$
$\nabla_{\mathbf{W}^{(k)}} J = \mathbf{g}\mathbf{h}^{(k-1)\top} + \lambda \nabla_{\mathbf{W}^{(k)}} \Omega(\theta)$
Propagate the gradients w.r.t. the next lower-level hidden layer's activations
$\mathbf{g} \leftarrow \nabla_{\mathbf{h}^{(k-1)}} J = \mathbf{W}^{(k)\top} \mathbf{g}$
end for

4.2.7. Optimization

The goal of the learning stage of the neural network is to minimize the cost function. Approaching the minimum of this function is done with a technique called gradient descent [2]. Suppose there is a function $y = f(x)$. The derivative $f'(x)$ gives the slope of $f(x)$ at the point x. It thus tells you how a small change in the input scales with the output. The derivative can be used to minimize a function because it shows how to change x in order to lower y. This method however stagnates when $f'(x) = 0$. This is the case in local minima, local maxima and saddle points [6].

In deep learning, cost functions are very complex functions with many local minima and very flat regions. Optimization can therefore be a challenge and usually one settles for a very low local minimum instead of the global minimum (Figure 4.5).

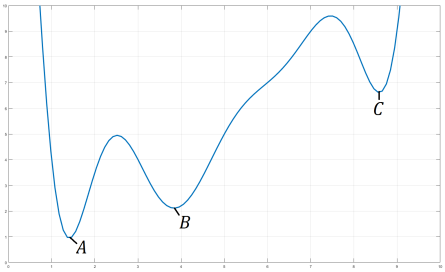


Figure 4.5: (A) Global minimum, optimal solution; (B) local minimum, acceptable solution; (C) local minimum, poor solution

Three types of gradient descent variants exist. These types represent the trade-off between accuracy and computational costs [6].

Batch gradient descent $\theta = \theta - \eta \cdot \nabla_\theta J(\theta)$
The gradient of the cost function is computed w.r.t. the entire training dataset. This leads to an accurate gradient and it guarantees to converge to a local minimum with relatively low computational cost per epoch (Section 4.2.8). However with only one parameter update per epoch, the number of epochs needed for sufficient convergence can be high. This may lead to overfitting.

Stochastic gradient descent $\theta = \theta - \eta \cdot \nabla_\theta J(\theta; x^{(i)}; y^{(i)})$
A parameter update is performed for each training example in the training dataset. This makes continuous learning possible. The individual parameter updates are much faster but the computational costs for an epoch is much higher than with batch gradient descent. As stochastic gradient descent performs parameter updates with a high variance, this increases the chances of finding another local minimum. The downside to this is that it will keep overshooting the exact minimum. This can be mitigated by slowly decreasing the learning rate.

Mini-batch gradient descent $\theta = \theta - \eta \cdot \nabla_\theta J(\theta; x^{(i:i+n)}; y^{(i:i+n)})$
Mini-batch gradient descent is the most widely used type. It computes a parameter update for every n training examples. With n being the mini-batch size. This way, the advantages of both previous methods can be combined.

In practice, people also refer to mini-batch gradient descent as stochastic gradient descent (or SGD). With the original SGD having a mini-batch of size 1. Throughout the rest of this thesis SGD refers to a vanilla mini-batch gradient descent. Meaning mini-batch gradient descent without additions. Furthermore, batch size will be used instead of mini-batch size.

Vanilla gradient descent algorithms do not guarantee convergence to an acceptable local minimum. Complex cost functions can contain sub-optimal local minima and saddle points which are difficult for SGD to escape. And convergence is not always fast enough to avoid overfitting. Numerous extensions to the vanilla gradient descent optimization algorithm are used in machine learning. All have advantages and pitfalls. Two of which are discussed in this section, these are the algorithms that proved to be useful in this project: Momentum and Adam [23].

Momentum Common around local optima are 'ravines'. These are areas with a vastly different steepness in one dimension than the others. Momentum is an addition to SGD that helps to navigate through these ravines much faster (Figure 4.6) by adding a fraction of the previous parameter update to the new one (Eq. 4.6). This may however result in very high learning rates in some dimensions which can lead to overshooting.

$$\begin{aligned} v_t &= \gamma v_{t-1} + \eta \nabla_\theta J(\theta) \\ \theta &= \theta - v_t \end{aligned} \tag{4.6}$$

Adam One of the most popular optimization algorithms for machine learning is Adam (Adaptive moment estimation) It computes adaptive learning rates for each parameter in a more comprehensive way than momentum. As it stores an exponentially decaying average of both past gradients and past squared gradients (Eq. 4.7) [13]. The main downside to Adam is that the average and variance of each parameter is stored and therefore increases the computational cost.

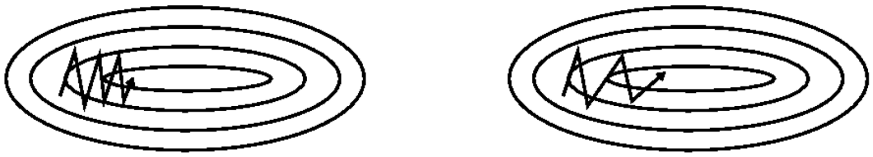


Figure 4.6: SGD without and with momentum [19]

$$
\begin{aligned}
m_t &= \beta_1 m_{t-1} + (1-\beta_1) g_t \\
v_t &= \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \\
\hat{m}^t &= \frac{m_t}{1-\beta_1^t} \\
\hat{v}^t &= \frac{v_t}{1-\beta_2^t} \\
\theta_{t-1} &= \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t
\end{aligned}
\tag{4.7}
$$

The default values for Adam ($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$) proposed by the original paper result in a robust optimization algorithm leaving the learning rate and batch size as the only hyper-parameters to be tuned.

4.2.8. Training Process

A neural network is represented by its topology (layers and activation functions) and its parameters. The parameters are all weights and biases in the network. The initial values of these parameters is commonly a random value between zero and one, but custom values can also be assigned. The training process is there to change the parameters so that a functional predictive model is established.

In the training process, data is used with which the network learns to make accurate predictions. This data is split into three groups: training data, validation data and test data. The entire training dataset is used in each iteration of the training process, also called an epoch. The concepts of forward-propagation, back-propagation and optimization together make up one iteration of the training process. After each epoch, a relatively small amount of data (usually about 10% of the training dataset), the validation training set is used to compute a validation loss. Comparing the training loss and validation loss during the training process gives an indication of the quality of the learning process. Whereby the main question is whether the model is overfitting or underfitting (Section 5.2.2). The validation loss curve is used to evaluate the model while tuning the hyper-parameters (batch size, learning rate and other optimizer parameters). Finally, the test dataset is used to provide an unbiased evaluation of the quality of the model.

5

Methodology; Network Architecture and Training

A common phrase in machine learning articles is: 'it is more of an art than a science'. It is mostly used in the context of choosing network topologies and hyper-parameters of the optimization algorithm. It is true that constructing a neural network is something you learn by experience and often a neural network is capable of using correlations beyond our comprehension. The process of creating a neural network is one of guided empiricism. Network training can often be substantially improved by applying different network architectures, optimization algorithms, training schemes and hyper-parameters. Without an effective optimization process of these parameters, wrong conclusions can be easily drawn when network performance appears to be worse than alternative methods. This is also true for other data driven methods such as genetic programming, support vector machines, instance based learning, model trees, etc [4]. but these fall outside of the scope of this thesis.

A standard multilayer feedforward network can approximate any Borel measurable function from one finite dimensional space to another to any desired degree of accuracy, provided sufficiently many hidden units are available[10]. A lack of convergence or accuracy therefore must be caused by insufficient training data, insufficient hidden nodes and/or an inadequate learning method. The response spectrum is a function of many variables and the complexity of the problem makes all three of these potential pitfalls a challenge. This chapter explains how a right network topology can help to overcome the first two of these challenges.

5.1. Neural network types

Neural networks are categorized in certain categories based on the mathematical operations. These different network types are suited for different purposes but not always uniquely used for a particular field. The main categories are discussed in this section, all of which are featured by the Keras library.

Multilayer Perceptron The Multilayer Perceptron (MLP) is an ANN in it's most simple and common form. These networks can also be referred to as feedforward neural networks or deep feedforward networks depending on the number of hidden layers. The workings of this network are already discussed in chapter 4. Common applications for MLPs are pattern classification, pattern matching, function approximation and nonlinear mapping.

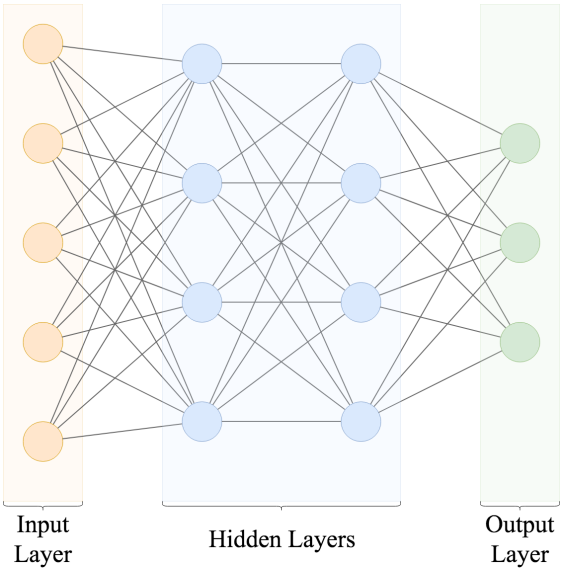


Figure 5.1: Multi-Layer Perceptron

Recurrent Neural Network The principle recurrent neural networks is that the model holds information of previous computations. This 'memory' can be useful if there is a strong correlation between the previous computation(s) and the next one. For example this can be true for time series prediction and translation models. Many different RNN architectures exist but only two are mentioned in this thesis. The basic, fully recurrent network, and the Long Short Term Memory (LSTM) network (Section 5.1.1).

In recurrent neural networks, every node in the hidden layers depends both on its input as on its output of the previous timestep (Eq 5.1). Both the input and the memorized output have individual corresponding weights. For a tanh activation function for example the node output will be calculated as in equation 5.2.

$$h_t = f(h_{t-1}, x_t) \tag{5.1}$$

$$h_t = \tanh(w_{ijh} h_{t-1} + w_{ijx} x_t) \tag{5.2}$$

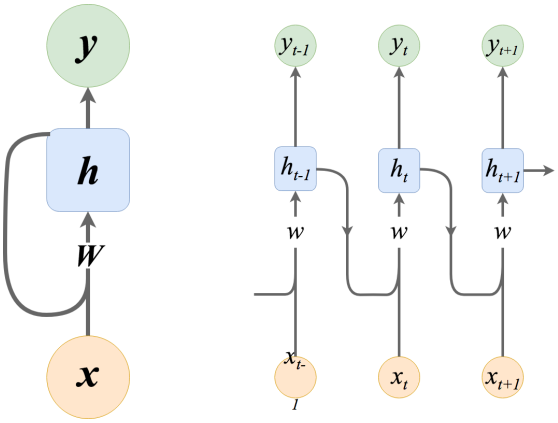


Figure 5.2: Recurrent Neural Network

Radial Bases function Neural Network A radial basis function (RBF), considered the distance of a point with respect to the center. RBF networks typically consist of three layers. An input layer containing a vector of real numbers. An output layer with a linear activation containing a scalar function of the input vector. The hidden layer has a non-linear RBF activation function.

The RBFs feature is that their response changes monotonically with distance from a central point. Typical RBFs are the multi-quadratic (Eq 5.4) and the Gaussian function (Eq 5.3).

$$h(x) = exp\left(-\frac{(x-c)^2}{r^2}\right) \tag{5.3}$$

$$h(x) = \frac{\sqrt{r^2+(x-c)^2}}{r} \tag{5.4}$$

Radial basis functions can be seen as a type of activation function that can be employed in any MLP. However they are often considered as their own type of network. In the case that basis functions are able to move, change size and/or when the network has multiple hidden layers, RBFs become nonlinear [20].

5.1.1. Long Short Term Memory

There is no specialized neural network architecture available for the frequency domain application in this research. Therefore an MLP is used for those predictions. The reason why there is no neural network category dedicated for these multi-input multi-output calculations is simply because a lack of research in this particular field. Timeseries prediction however is a much more common application of neural networks. ANNs are used in forecasting social trends, stock exchanges and climate change among many other forecasting applications. Besides neural network based models, many other forecasting methods exist, e.g. autoregression, Kalman filters and river flow forecasting [11]. Neural network based models though, have the advantage of being self-adaptive, able to generalize and able to capture nonlinearities.

Recurrent neural networks, as discussed before, are particularly good at timeseries prediction. RNNs use their feedback connections to store representations of recent input events ('short-term memory'). Learning to store information over extended time intervals in this way takes a long time and can cause problems. When backpropagating an error for a long term dependency, when the chain rule is applied and in any of the gradients approached zero, all the gradients would rush to zero exponentially due to multiplication. This is known as the vanishing gradient problem. Long Short-Term Memory proposed by Hochreiter and Schmidhuber[8] attempts to remedy this by combining a recurrent network with a gradient-based learning algorithm. This enforces a constant error backpropagation. Long Short Term Memory (LSTM) can bridge long time lags, less effected by short-term noise and functions well without tedious parameter fine-tuning. These LSTM networks do not consist of standard nodes. Instead the nodes have been replaced by modules consisting of a small network of four activation functions. A disadvantage of these modules is that they increase the number of total weights of the network by a factor of 9.

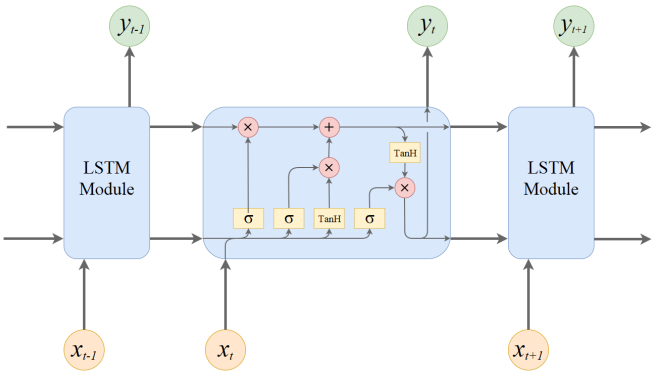


Figure 5.3: LSTM modules

5.1.2. Approach

For MLPs, there are many potentially well-performing architectures ranging from shallow to deep with all kinds of activation functions. Finding the right architecture for a particular problem can be a challenge. This should be done using a structured approach while minimizing the amount of influencing parameters.

Reducing the choice of activation functions significantly lowers the amount of network architecture options. For this research, the four most commonly used activation functions are considered: linear, sigmoid, tanh and ReLU. Not combining these different activation functions has this same effect, with the exception of a linear input and output layer when using one of the other activation functions. This is necessary because the output ranges of these activation functions are limited and some input values can cause problems (Section 4.2.4).

The optimization of an LSTM based architecture is less tedious process. The topology of the LSTM layer is bound by the amount of input parameters, there are standard activation functions in the LSTM modules and although often a fully connected linear layer can be a valuable addition, the addition of any extra layers is not likely to improve the network.

5.2. Training process

5.2.1. Overfitting

Provided that the network is complex enough and a robust optimizer is used, if a neural network keeps training on a certain dataset, eventually it will become an almost perfect transfer function between input and output data for every training example in the dataset. The downside to this is however that now the network has adapted to all outliers in the training dataset and is not able to generalize. When such a network is presented with new input data, it is not able to act as a generalized transfer function. This phenomena is called overfitting and it is most common problem in machine learning. The opposite of overfitting is underfitting. This takes place when the network does not converge to a point that it is able to grasp the complexity of the problem. This occurs when the network architecture is not complex enough and when optimizers get stuck in a sub-optimal minimum.

5.2.2. Mitigation of overfitting

The most obvious ways to reduce overfitting are enlarging the dataset and reducing the network complexity. The size of the dataset is in this case linked to the transits of the semi-submersible and it is not possible to enlarge it. Therefore it is of importance to have a network with the least amount of nodes while still being able to capture the physical problem.

Overfitting can be mitigated by a number of different methods. Some of these methods

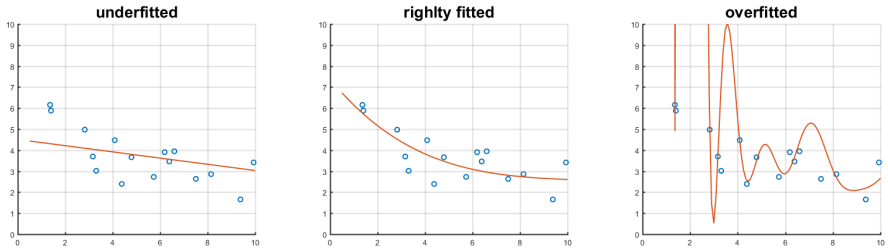


Figure 5.4: Underfitting & overfitting example

alter the topology of the network, others have to do with the training and optimization process. The ones altering the topology that are used in this research are explained below:

Regularization Overfitting can be seen as a lack if generalization caused by a model with a complexity that is too high. This can be mitigated by introducing an extra term in the cost function that discourages complexity [18]. Such a term is called a regularizer. It penalizes the sum of all weights in the network and therefore acts on the assumption that weights that only have an effect on outliers in the training examples are of no value (or even negative value) to the network when it is being applied on another dataset. Two types of regularizer are often used: L1 and L2. Where L1-regularization is the sum of the weight times a certain regularization coefficient and L2-regularization is the sum of the square of all weights times a certain coefficient.

Dropout This technique randomly leaves out a fraction of the network's nodes (along with it's connections) for each training example. This can significantly reduce overfitting [25]. In a Keras built network, dropout is implemented by introducing a dropout layer. This layer can 'cut' the connections of a certain amount of the next layer's nodes.

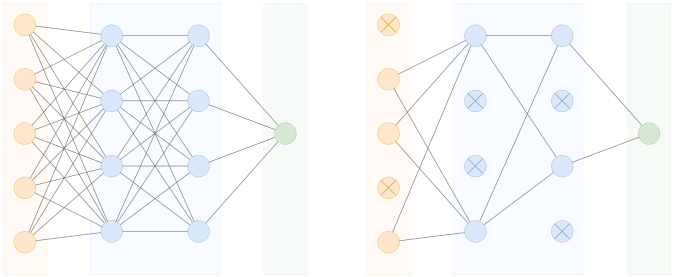


Figure 5.5: Neural network without and with dropout[25]

Accuracy The accuracy of the input data, by which is meant the size of the steps in frequency and heading, influences the potential quality of the output data. A high input resolution is therefore preferred. However the amount of parameters in the network often increases exponentially with a higher resolution. A higher amount of parameters can lead to a less efficient training process and thus to overfitting.

Besides network changes, choices made in the training process also have a major influence on chance of overfitting the network on the training data. It all comes down to the simple reasoning that more epochs result in a higher chance at overfitting. Reducing the amount of epochs is achieved by increasing the rate of convergence per epoch. This can be done by either having a more efficient training process or somehow increase the amount of training data. The methods used in this research to reduce the amount of epochs are listed and described below.

Validation set A validation set is a part (usually about 10%) of the training data set that is not used for updating the network parameters. Instead it is used to evaluate network performance every epoch. The comparison between training loss and validation loss is used to diagnose overfitting. In general, it is best to stop network training when the validation loss stops decreasing. This can however be a pitfall because this can also happen when a network is (temporarily) stuck on a saddle point of the cost function.

Learning rate The learning rate is the most influential parameter in the optimization algorithm. Taking larger steps while updating the weights can increase the rate of convergence. However this can be done up until a certain point. Too high a learning rate will cause overshooting of the local minimum and will halt convergence or even cause divergence.

Optimizer There is no single best optimizer for training a neural network. Often there is a trade-off between robustness, tedious hyper-parameter optimization and likeliness of overfitting. In general, SGD and Adam are good choices for network training and so these are the ones used for this project. Like the network architecture, it is best to limit your own choices of optimizers as it takes too much time to try everything and there are no reliable rules of thumb in this field.

6

Frequency Domain

The purpose of the research in the frequency domain is to find out whether a neural network can function as an alternative for an RAO. Either close to it's current form with a network without hidden layers or in alternative form.

6.1. Artificial Environment

6.1.1. Data

For the artificial environment, the training, validation and testing data has to be generated. HMC has Matlab toolbox functions to either define a spectrum using Jonswap, Pierson-Moskowitz, etc. or retrieve a spectrum from environmental databases such as Argoss. Argoss is a metocean consultancy and weather forecasting company that provides offshore weather forecasts for HMC. Forecasts are based on dedicated weather and shallow water wave models validated by satellite observations and/or in-situ data.

Defining custom spectra provides the opportunity to easily make case studies to test the networks ability to make predictions on spectra that are different than it's training spectra. In other words, the ability to interpolate and extrapolate can be evaluated. On the other hand, using environmental databases will provide more realistic data which will make it easier to develop a network architecture that is more likely to succeed on measurement data. Since both methods are available, both ways will be used to create data.

The Argoss Data consists of the following parameters:

Time Time
Hs Significant wave height (Wind, Swell, Total)
Hsdir Heading (Wind, Swell, Total)
Tz Zero crossing period (Wind, Swell, Total)
Tm Mean Period (Wind, Swell, Total)
Tp Peak Period (Wind, Swell, Total)
Vw Wind speed
Vwdir Wind direction
s3d Full polar wave spectrum

6.1.2. Architectures

The aim for the frequency spectrum model is to provide an alternative to the RAOs calculated by WAMIT. It would make sense then that the input would be a 2D (frequency and heading) wave spectrum and that the output would be a response spectrum for a certain degree of freedom. This makes the network a direct substitute for an RAO. However this is not the only

valuable network architecture. A 1D spectrum (frequency, heading = constant) can also be useful in model tests and maybe even in a sea-state with low spreading. Also since in practice often only the significant double amplitude is used, a network providing only this number would also prove useful. One of the disadvantages of using the entire spectrum as an input is the amount of input nodes. Therefore, a network using only wave parameters as input values might have a better chance at convergence. A network with wave parameters as input and the significant double amplitude as output is also tested. The different network input and output combinations are listed in Table 6.1.

Name	Input	Output
1D	$S_W(\omega)$	$S_R(\omega)$
2D	$S_W(\omega,\Theta)$	$S_R(\omega)$
2D-SDA	$S_W(\omega,\Theta)$	SDA
Parametric	Sea state parameters	SDA

Table 6.1: Network input/output combinations

6.1.3. Cases

The main goal for the artificial environment is the exploration of topologies and hyper-parameters. The liberty that comes with unrestricted amount of data with an adjustable parameters ranges allows for more insight driven exploration of these topologies and hyper-parameters. In aid of gaining more insight, different 'cases' are provided to the model. These different cases are not used in any way in optimizing the network topology or hyper-parameters, they serve only as a means to determine network capabilities when confronted with different datasets. The following six cases are used in the artificial environment:

Standard The standard case for the neural network is a dataset composed of 80% training data and 20% test data (Figures 6.1 & 6.5). Sea state parameters $H_s[m]$ and $T_p[s]$ range from $0-8$ and $1-20$ respectively and the main heading is random. The topology of the network and the order of magnitude for the hyper-parameters is based on this case.

Interpolation A lack of overlap of training data and test data can occur in real-world applications of the network. The interpolation case is used to assess the networks ability to interpolate. The dataset (Figures 6.3 & 6.7) is composed of 80% training data and 20% test data. For the test data set, sea state parameters $H_s[m]$ and $T_p[s]$ range from $4-5$ and $9-11$ respectively. For the training data set, sea state parameters $H_s[m]$ and $T_p[s]$ range from $2-7$ and $7-13$ respectively, excluding the ranges $4-5$ for H_s and $9-11$ for T_p. The main headings are random.

Extrapolation A lack of overlap of training data and test data can occur in real-world applications of the network. The extrapolation case is used to assess the networks ability to extrapolate. The dataset (Figures 6.2 & 6.6) is composed of 80% training data and 20% test data. For the training data set, sea state parameters $H_s[m]$ and $T_p[s]$ range from $4-5$ and $9-11$ respectively. For the test data set, sea state parameters $H_s[m]$ and $T_p[s]$ range from $2-7$ and $7-13$ respectively, excluding the ranges $4-5$ for H_s and $9-11$ for T_p. The main headings are random.

Scarce To assess the functionality of the network when trained on a limited number of datapoints (Figures 6.4 & 6.8). The sea state parameters $H_s[m]$ and $T_p[s]$ range from $0-8$ and $1-20$ respectively. The main headings are random.

Spectral wave climate same location In an effort to bring the artificial environment closer to reality, the spectral wave climate of the North sea from the Argoss database is used to generate the dataset. This provides a realistic spreading in H_s and T_p. The main headings are random.

Spectral wave climate different location The second case based on Argoss data is used to assess the performance of a network trained on a specific spectral wave climate in making predictions for a spectral wave climate of a different environmental area. The two data sets used for this case are North Sea (Snorre) and Trinidad (Cassia). H_s and T_p are plotted in Figures 6.9 & 6.10). Heading is randomized.

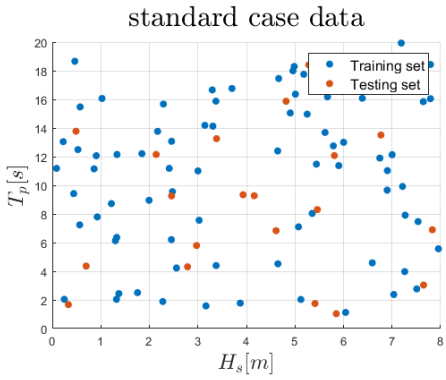


Figure 6.1: Standard case with 100 datapoints

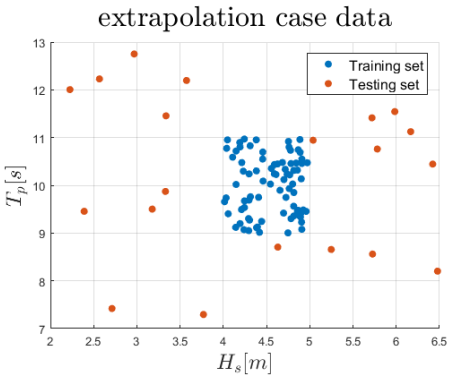


Figure 6.2: Extrapolation case with 100 datapoints

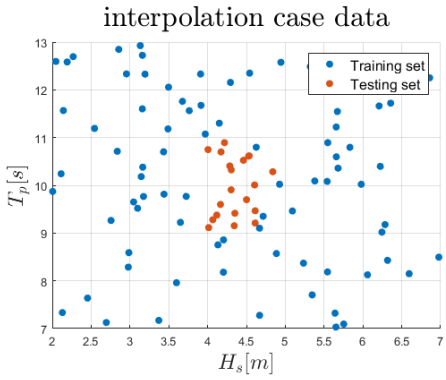


Figure 6.3: Interpolation case with 100 datapoints

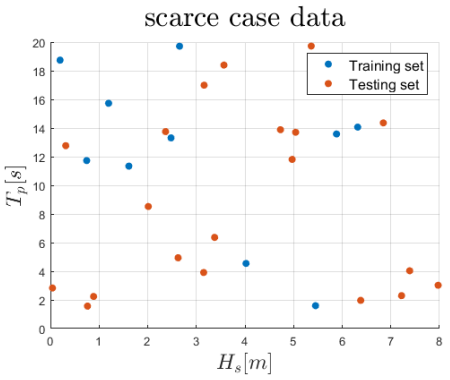


Figure 6.4: Scarce case with 20 datapoints

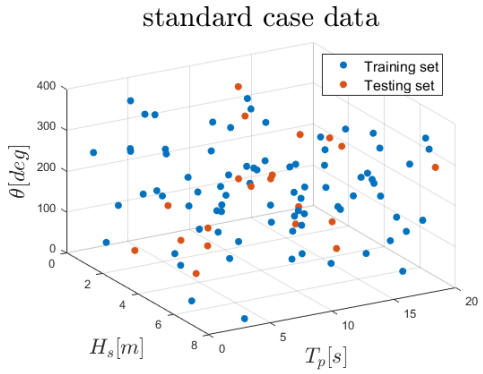


Figure 6.5: Standard case with 100 datapoints

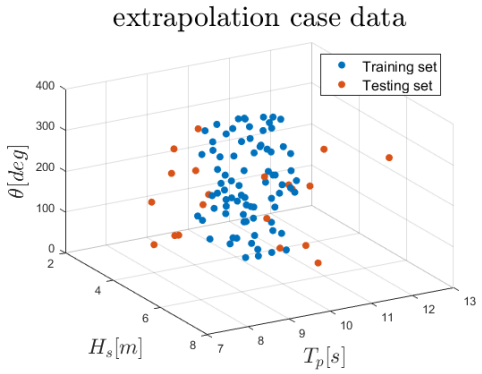


Figure 6.6: Extrapolation case with 100 datapoints

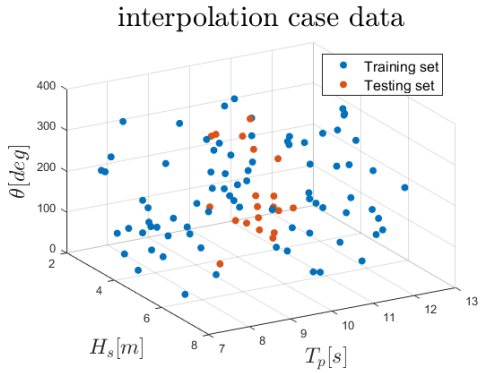


Figure 6.7: Interpolation case with 100 datapoints

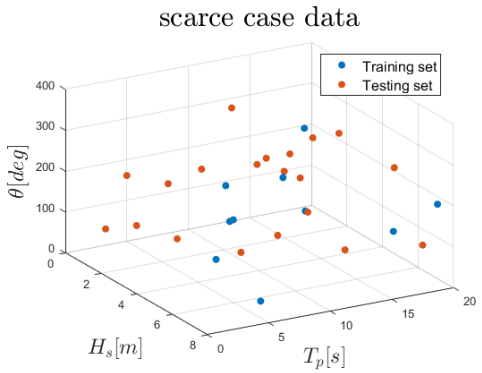


Figure 6.8: Scarce case with 20 datapoints

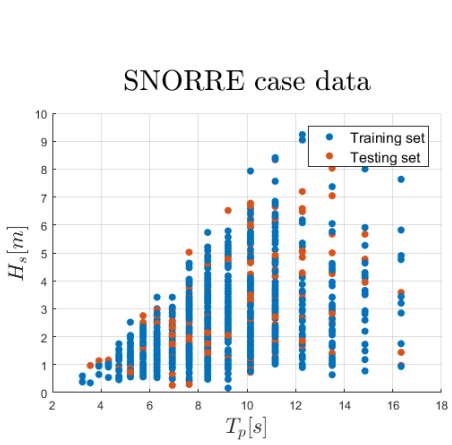


Figure 6.9: Snorre, 1000 datapoints

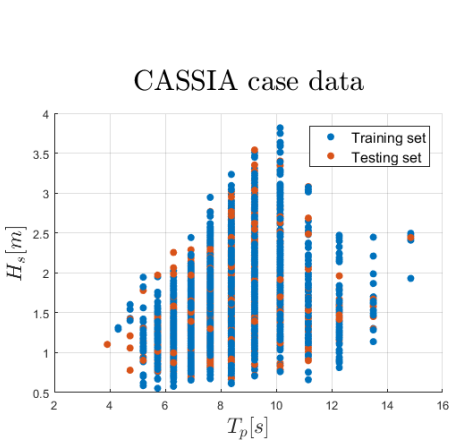


Figure 6.10: Cassia, 1000 datapoints

6.1.4. 1D

The 1D model is only used for the artificial environment as there is a lack of measurement data for this model. Model tests and possibly sea states that are almost unidirectional are the only kinds of useful data for this model. It is of course able to make predictions in an artificial environment. Also, the relatively low amount of input parameters causes the weight and bias matrices to be small enough to be assessable. This makes it possible to gain some insight in network behavior.

The 1D model in the artificial environment has to be capable of a fairly simple computation; a linear transfer function for all corresponding frequencies. A neural network for this application can be designed by hand and would not require machine learning. It is however a nice introductory exercise in the method and it can still answer some questions regarding

network behavior. Because of the simplicity of the computation, the number of potential topologies is fairly low: Either a network without hidden layers and a linear activation function, or a network with one hidden layer and two linear activation functions. The second option will also require a search for the optimal hidden layer size. The network without a hidden layer would have the RAO on the diagonal of its weight matrix in the case that the input and output layer have the same size and represent the same frequency bins (Figure 6.11). The network with one hidden layer on the other hand, is a universal approximator (Appendix B). This enables the network to represent the RAO as well. While both topology types are able to represent the necessary computation, performance can still differ in the form of optimizer performance.

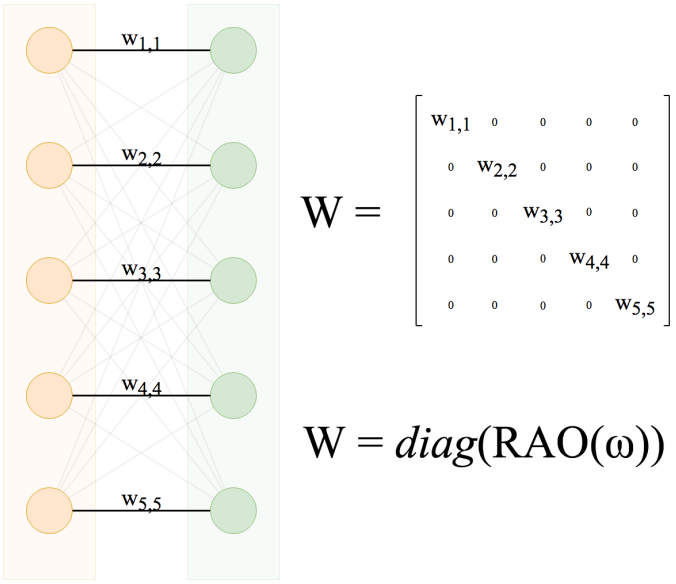


Figure 6.11: Diagonal of weight matrix W correspond with values of $RAO(\omega)$

Standard case

The general methodology of finding the optimal topology and optimizer for this particular model is as follows. First, the a small dataset of 100 training/testing examples is used to find the right setting, and then a larger dataset of 1000 training/testing examples is used to assess the model. Initially, using the smaller dataset, ten topology options are tested with a robust optimizer (Adam, with default hyper-parameters). These topology options are displayed in Table 6.2.

Option	1	2	3	4	5	6	7	8	9	10
Number of layers	2	3	3	3	3	3	3	3	3	3
Hidden layer size	n.v.t.	4	8	16	32	64	128	256	512	1024
Activation	Linear	Linear	Linear	Linear	Linear	Linear	Linear	Linear	Linear	Linear
Batchsize	5	5	5	5	5	5	5	5	5	5
Epochs	100	100	100	100	100	100	100	100	100	100
Optimizer	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam

Table 6.2: 1D model topology/optimizer options

None of these topology options resulted in satisfying results within a reasonable amount of epochs. The results of the best performing topology from this series are shown in Figure 6.12. It appears that non-corresponding frequencies still get excited by these networks. A possible reason for this is that the learning rate is too high and therefore will keep overshooting

the desired value of 0 (instead of a random value between 0 and 1) for biases and unrelated weights. A lower learning rate would however result is much lower convergence rates.

Alternative solutions for this problem are intializing the weights and biases at 0 and/or restrict the value ranges of the weights and biases. Because the first of these solutions is less invasive, this is the preferred option. All topology options from Table 6.2 are tested again with weights and biases initialized at 0.

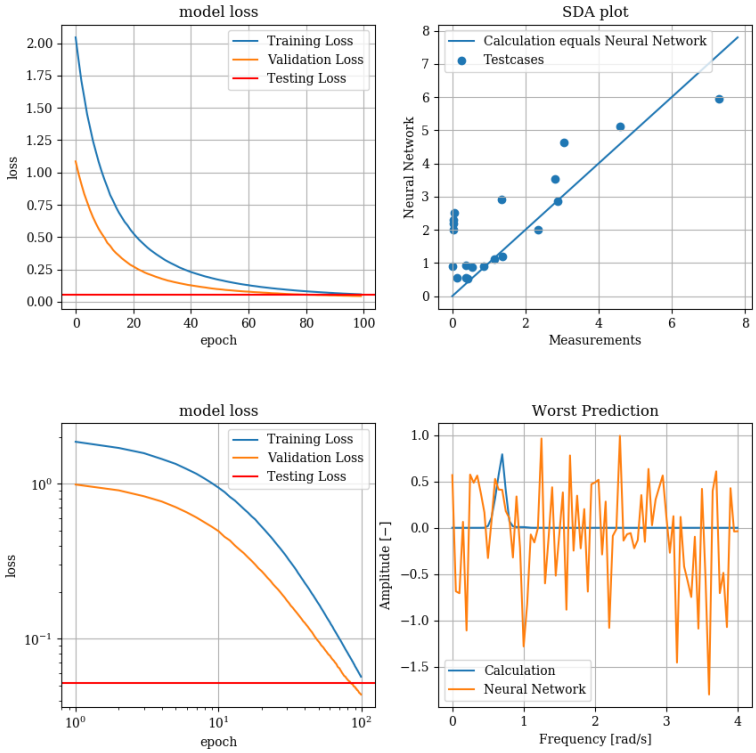


Figure 6.12: 1D model, option 4 initial results

Tests with the initialized parameters show a clear pattern shown in Figure 6.13. The trend seen in this comparison is faster and further convergence with an increasing amount of hidden nodes. With the topology without a hidden layer (model 0010) deviating from this trend. All topologies are still converging at 100 epochs but not all show a sufficient rate of convergence and show a clear limit to their maximum capability. This is true for the models with no, or a low (4, 8 and 16) amount of hidden nodes.

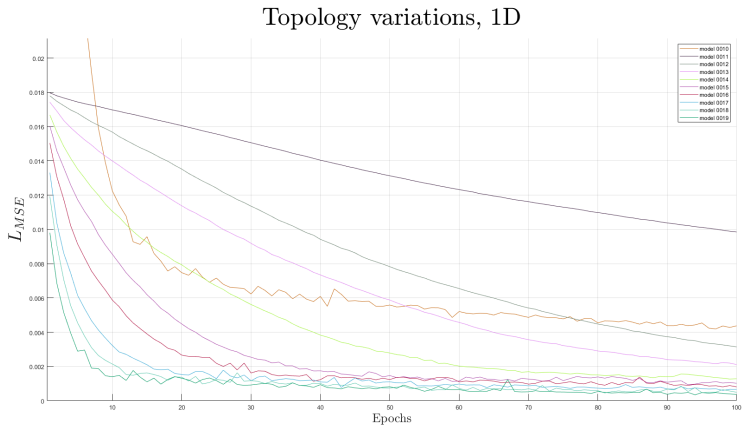


Figure 6.13: 1D model, option comparison

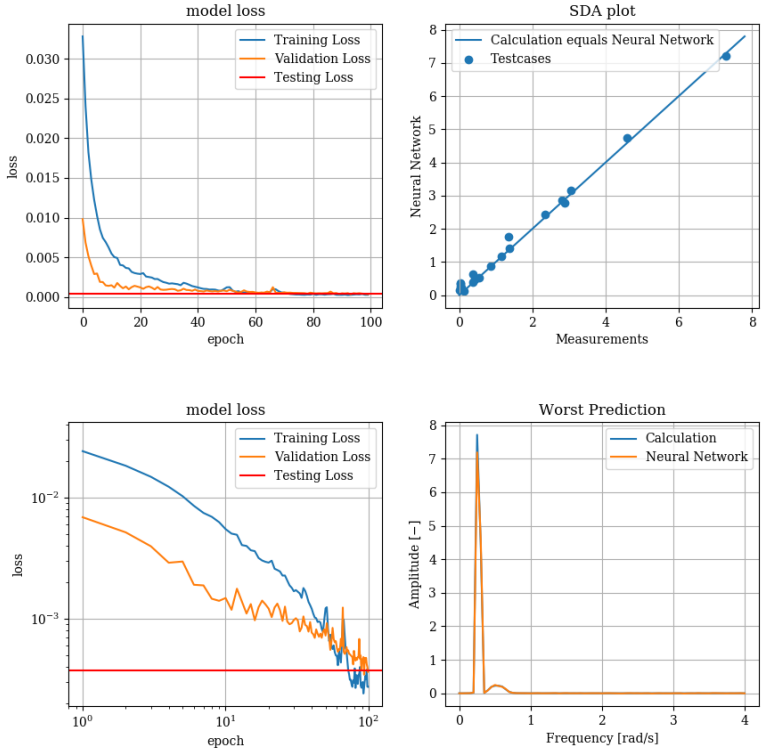


Figure 6.14: 1D model, option 10 for standard case, 100 examples

The next step; applying the neural network with topology option 10 to a larger (1000 examples) dataset, shows similar results (Figure 6.15. The model loss curve suggests that the learning rate is too high after about 10 epochs as it starts to oscillate heavily. The network still has some problems when dealing with low SDA spectra as can be seen in the SDA plot. The reason for this is that, even though the parameters are initialized at 0, small values are assigned to irrelevant parameters by the optimizer. In low SDA cases these small values are significant and cause inaccurate results.

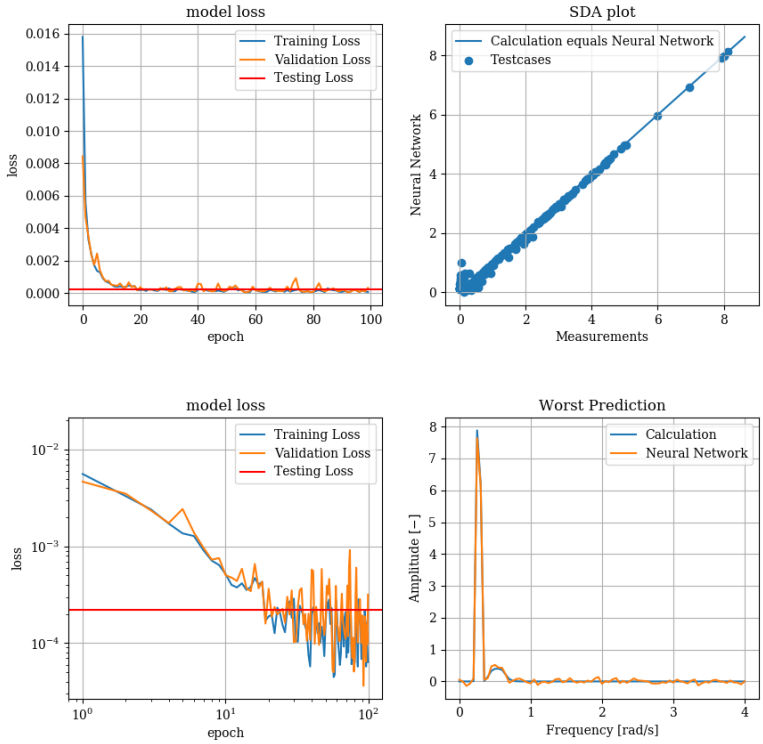


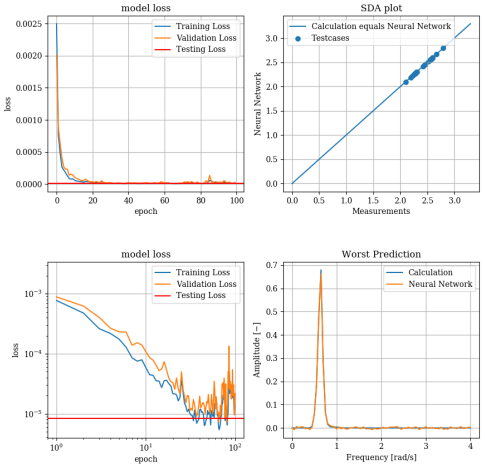
Figure 6.15: 1D model, option 10 for standard case, 1000 examples

Alternative cases

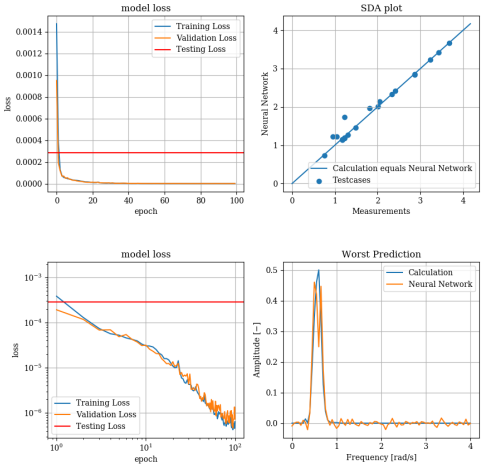
The topology found for the standard case is now used to evaluate the models extrapolation and interpolation capabilities, performance on a low amount of data and data that is more representative of the real world conditions. The specifics of these data sets are discussed earlier in Section 6.1.3. To avoid the convergence issues seen in the model loss of Figure 6.15, a small (100 examples) dataset is used for the alternative cases. With the obvious exception of the scarce case.

The interpolation case depicted in Figure 6.16a shows a near perfect match between the Neural network SDA and the diffraction based calculations. As the more extreme values in the training set cover a larger part of te frequency domain than the ones in the test set, this is no surprise. The extrapolation case on the other hand shows some deviating results because of the training set not containing values in the entire frequency domain, but most of the test examples are on the diagonal.

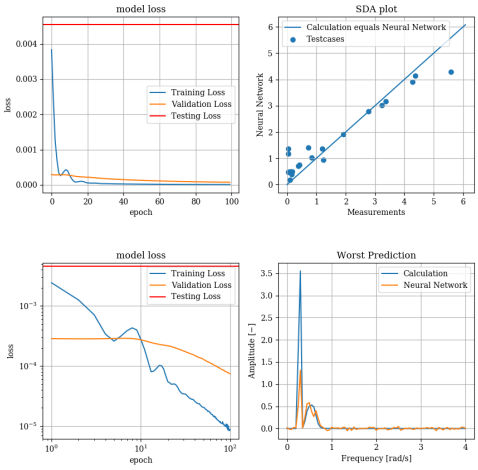
In the results from the scarce case (Figure 6.16c), the validation loss curve is converging well but a higher learning rate or increasing the amount of epochs both result in overfitting on the small dataset. Still, with this very short learning process, the right trend is clearly visible in the SDA plot. This shows that such a simple neural network can be trained on a very low amount of data.



(a) Interpolation case



(b) Extrapolation case



(c) Scarce case

Figure 6.16: 1D model, alternative cases

6.1.5. 2D

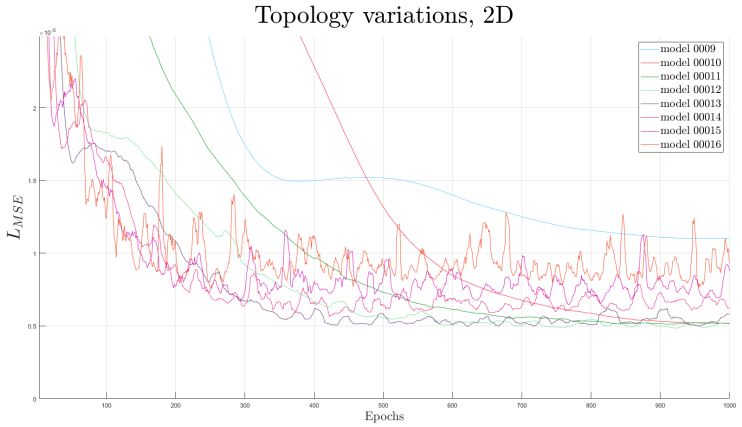
For the 2D model, there is a significant increase in the amount of nodes of the input layer. The size of the input layer in this model is equal to the amount of frequencies and the amount of headings used to describe the wave spectrum. For a wave spectrum described by bins of a 0.05 rad/s frequency step and a 5 ° heading step, the amount of input nodes is $81 \cdot 73 = 5913$. With a large input layer, the amount of parameters in the network can easily skyrocket when increasing the size of the hidden layer(s). Because this can lead to long training times, searching for the right architecture is best not done with the entire training set. While a network designed for a small data set may not be the optimal architecture for a case where a large amount of data is available, it is still a lot more robust than other methods of training time reduction like reducing the amount of epochs or increasing batch size.

standard case

The success of the shallow network for the 1D model implies that this would also be a good option for the 2D case. As it is the same type of computation; a linear transfer between each heading-frequency bin. A threat for this topology however is that it is possible that not all bins get excited in the training examples. This means that if the initial value of all parameters is 0, this will still be (close to) 0 after the training process for the un-excited heading-frequency combinations.

The same topology options are tested as in the 1D case. With random initial values and with the parameters initialized at 0.

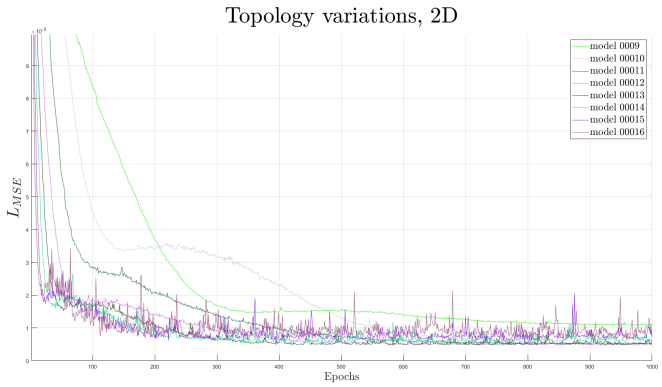
(a) initial parameter value: random (0-1)



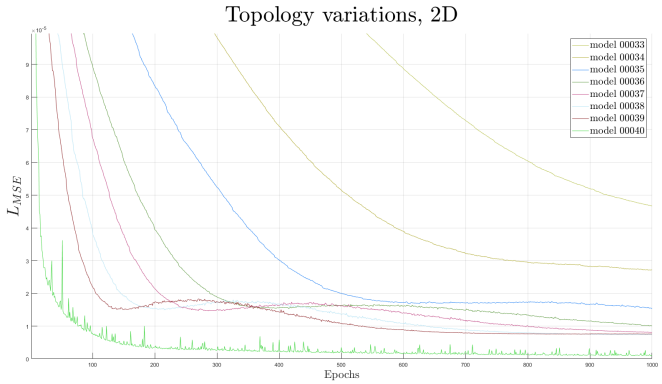
(b) initial parameter value: 0

Figure 6.17: 2D, topology options 1-8
Validation loss curves

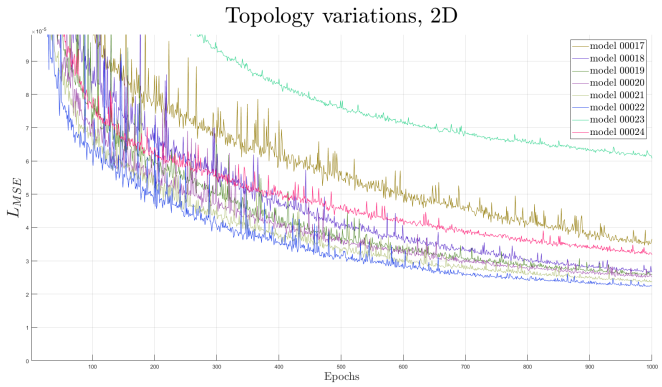
Results displayed in Figure 6.17 show a similarity with the 1D case. Initializing the network parameters with 0 values significantly improves their convergence. However, in Figure 6.17b, the networks with a high amount of nodes do not converge as far as the other networks. Instead their loss curves start to oscillate, but not diverge. This indicates optimizer issues and therefore, variations in optimizer and optimizer settings are applied. The variations are: Adam with lr = 1e-5, SGD with lr = 1 and SGD with lr = 10. Figure 6.18 shows the comparison of these optimizer settings.



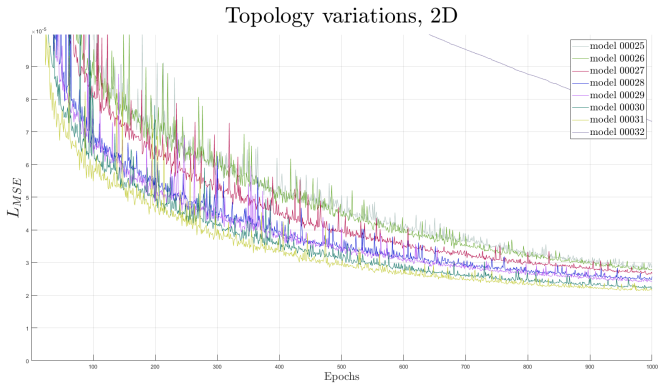
(a) Adam lr = 0.0001 (default)



(b) Adam lr = 0.00001



(c) SGD lr = 1



(d) SGD lr = 10

Figure 6.18: 2D, topology options 1-8, initial parameter value: 0 , optimizer variations
Validation loss curves

The optimizer with the best results is Adam with a learning rate ten times lower than the default value (Figure 6.18b). This also results in the correlation between hidden layer size and network loss; just like with the 1D model, a higher amount of hidden nodes results in a higher accuracy. The high amount of hidden nodes, in combination with a large input layer, causes a very high amount of network nodes. For example, the amount of network parameters in a model with one hidden layer consisting of 1024 nodes is 6,138,961 (Equation 6.1). The consequence of this higher number of nodes is that the sum of all parameters is spread out over more weights and biases while the sum itself does not scale linearly with the amount of parameters. As the average value of network parameters decreases, the optimizer requires lower stepsizes when approaching a minimum. This explains why the default values in Figure 6.17b cause oscillatory behavior.

$$\begin{aligned} n_\theta &= n_{W_1} + n_{B_1} + n_{W_o} + n_{B_o} \\ 6,138,961 &= 5913 \cdot 1024 + 1024 + 1024 \cdot 81 + 81 \end{aligned} \tag{6.1}$$

n_θ number of network parameters

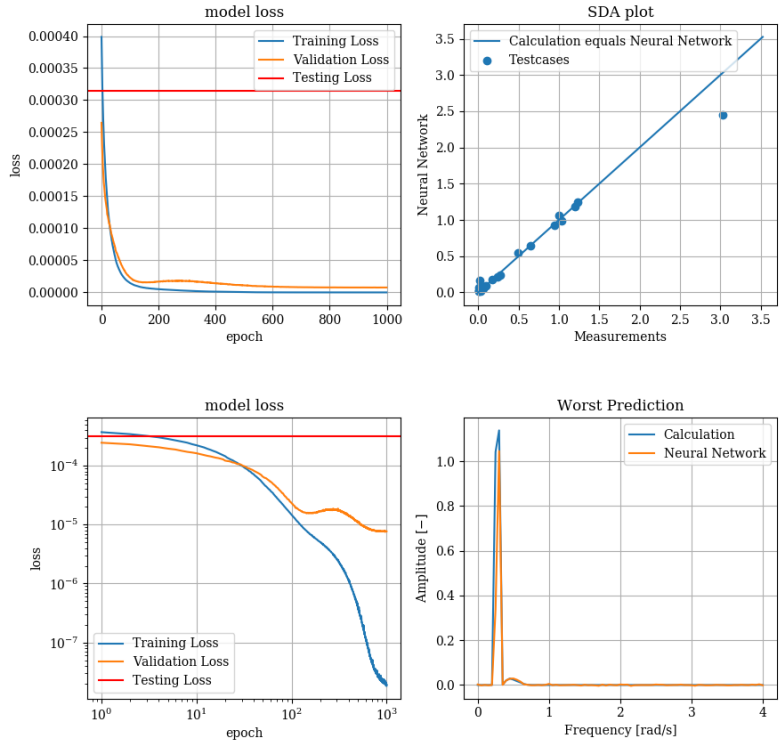
n_{W_1} number of weights in first hidden layer

n_{B_1} number of biases in first hidden layer

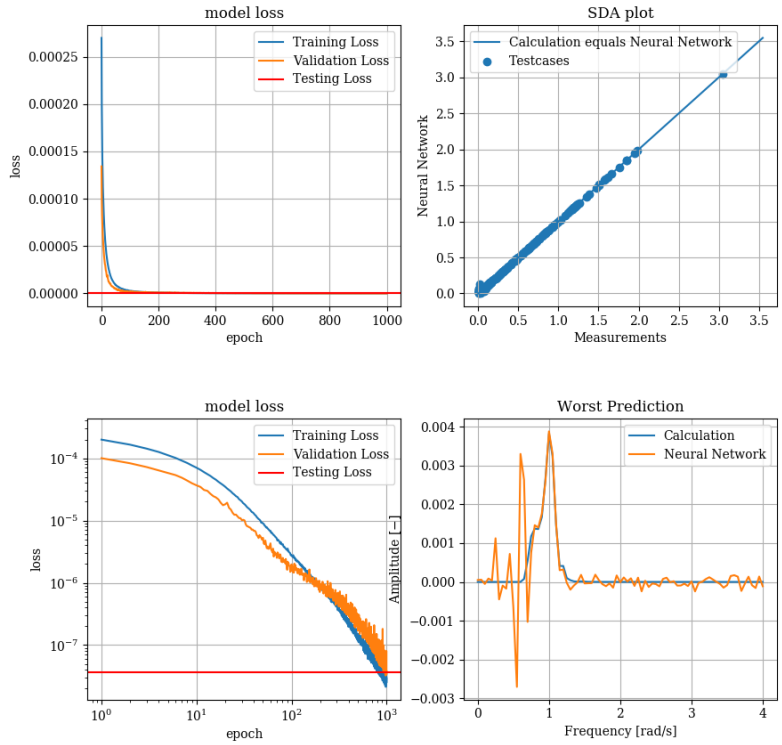
n_{W_o} number of weights in output layer

n_{B_o} number of biases in output layer

For the final model, the network with the highest amount of network nodes is chosen. The optimizer is Adam with a lower learning rate (lr=1e-5). The network is trained and tested separately on both a 100 example dataset and a 1000 example dataset. The validation curve of the smaller dataset (Figure 6.19a) stops converging before the training process is done while the training loss curve still converges. This shows that the network is overfitting on the training dataset. Using the larger dataset, overfitting issue is solved and the SDA plot shows good results. However, the validation curve is oscillating which indicates that the learning rate is too high for a loss below 10^{-4}. The consequence of this can be clearly seen in the 'Worst Prediction' plot; the learning rate is too high to deal with very small values. This issue can be mitigated by lowering the learning rate after a certain amount of epochs, or using learning rate decay which gradually lowers the learning rate. Because this is only an issue for low SDA sea-states and the general trend of the SDA curve shows a good match with the WAMIT results, the model is considered satisfactory and no further changes are made to the network.



(a) 100 examples



(b) 1000 examples

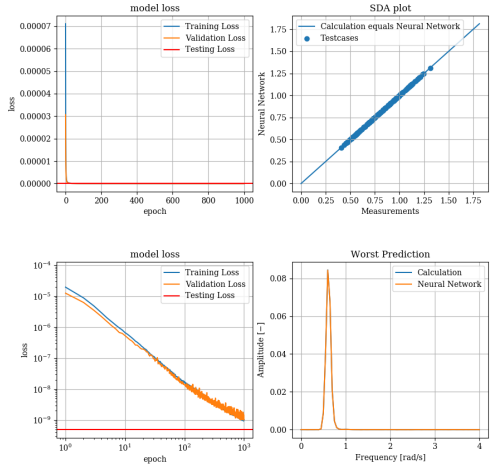
Figure 6.19: topology option 1, standard case, Adam lr=1e-5

Alternative cases

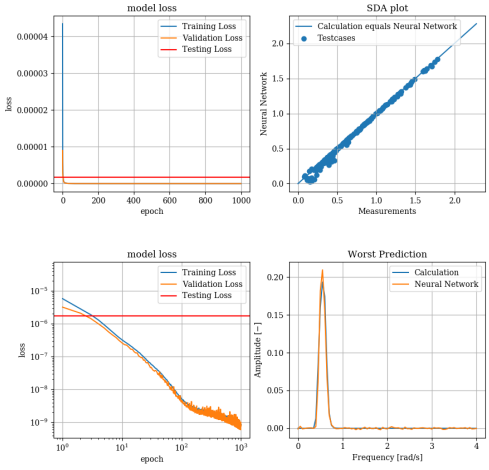
Training and testing the network on the alternative cases shows the same results as with the 1D network; the interpolation case shows near perfect results, while the extrapolation case shows less good, but still acceptable results. In this case, the network especially has trouble making predictions in low SDA sea-states.

For the scarce case the results are also similar to the 1D case. The general trend of the test examples is good, but the validation curve shows clear signs of overfitting.

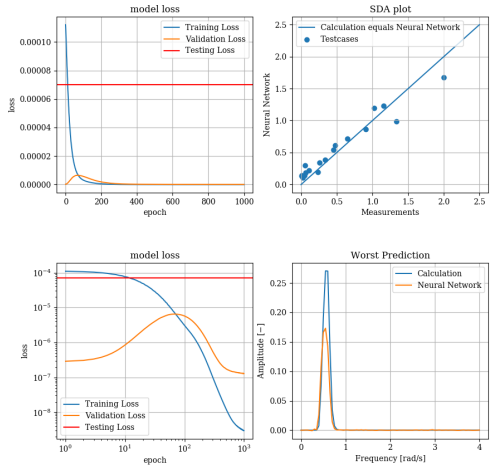
The network trained and tested on the North Sea data shows good results, as can be expected from the standard case results. However the network does show a lot of fluctuations on the loss curves. In an attempt to mitigate this, the network trained on the North Sea data and tested on the Trinidad data is trained in larger batches (50 training examples) and more epochs (10,000). The network results show that the loss curves only have very limited fluctuations and the SDA plot results follow the right trend, although this dataset seems to be much harder to predict than the extrapolation case.



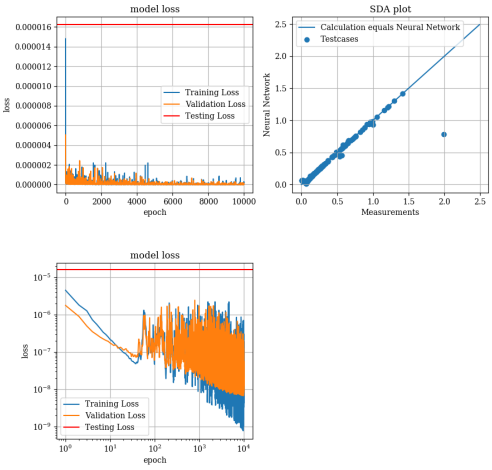
(a) Interpolation case



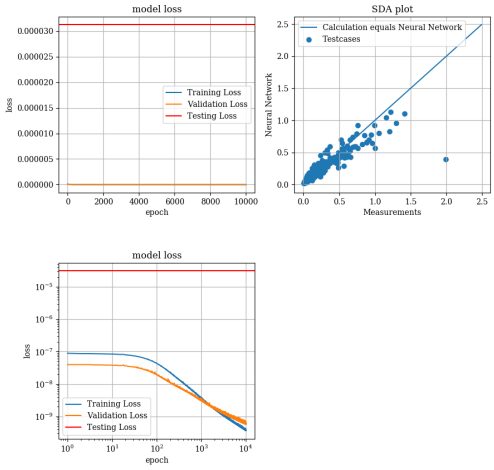
(b) Extrapolation case



(c) Scarce case



(d) North Sea



(e) North Sea - Trinidad

Figure 6.20: 2D model, alternative cases

6.1.6. 2D-SDA

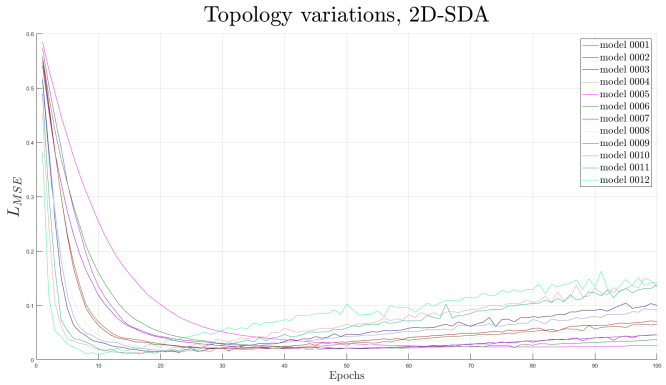
The 2D-SDA model aims to predict only a single value which represents the entire response spectrum. The computation that this model should represent consists of a linear transfer function per heading-frequency bin and a form of integration. As the linear transfer was best represented by two linear activation functions, one would expect that this network should have at least one extra layer to represent the integration. However it is possible that the network can integrate both computations in the first two layers.

The first topology options that are tested vary in network depth and hidden layer size. The first twelve linearly activated networks with 1-3 hidden layers and 32-256 nodes per hidden layer as can be seen in Table 6.3.

Option	1	2	3	4	5	6	7	8	9	10	11	12
Number of layers	4	4	4	4	3	3	3	3	5	5	5	5
Hidden layer size	32	64	128	256	32	64	128	256	32	64	128	256
Activation	Linear	Linear	Linear	Linear	Linear	Linear	Linear	Linear	Linear	Linear	Linear	Linear
Batchsize	5	5	5	5	5	5	5	5	5	5	5	5
Epochs	100	100	100	100	100	100	100	100	100	100	100	100
Optimizer	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam

Table 6.3: 2D-SDA model topology options 1-12

(a) Topology options 1-4



(b) Topology options 1-12

Figure 6.21: Initial variations in layer size and network depth, validation loss curves

Out of the tests of the first twelve topology options (Figure 6.21), a few conclusions can be drawn. The first is that the results get better with each added hidden layer. In the next topology variations, more layers need to be added to see whether this trend will continue.

The second conclusion is that this model is prone to overfitting and that this effect increases with a higher amount of nodes. Option 12, with 256 nodes per layer, does however reach the lowest loss. Therefore there seems to be a trade-off between rate of convergence and robustness (lower chance of overfitting).

The reason for this overfitting is related to the network output and the size of the training dataset. Because only the SDA is predicted, and the training set, excluding the validation data is 72 cases, the network can settle for learning these 72 values instead of the physical relationship between the input and output data. As network capability (amount of nodes) increases, the network becomes more able to memorize these 72 values.

The next set of topology options explores the trend seen in the first 12 options: higher network depth leads to better convergence. Because of the overfitting issue, all depth variations are tested with both 32 and 256 nodes per layer (Table 6.4).

Option	13	14	15	16	17	18	19	20	21	22
Number of layers	6	7	8	9	10	6	7	8	9	10
Hidden layer size	32	32	32	32	32	256	256	256	256	256
Activation	Linear	Linear	Linear	Linear	Linear	Linear	Linear	Linear	Linear	Linear
Batchsize	5	5	5	5	5	5	5	5	5	5
Epochs	100	100	100	100	100	100	100	100	100	100
Optimizer	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam

Table 6.4: 2D-SDA model topology options 13-22

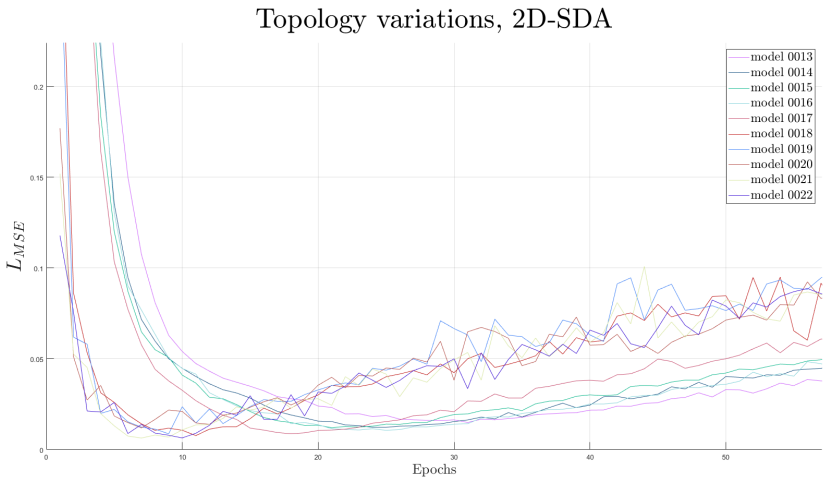


Figure 6.22: Topology options 13-22, validation loss curves

Figure 6.22 shows that the overfitting issue also exists with the more narrow networks. Overfitting does happen after more epochs, but the wider networks achieve are still converged further at the moment that overfitting starts. There does seem to be a positive effect of a higher network depth, but it is hard to conclude because of oscillatory behavior. For the next set of tests, the learning rate is lowered in an effort to mitigate these oscillations in the validation loss curve. In this next set of tests the topologies shown in Table 6.5.

Option	23	24	25	26	27	28	29	30
Number of layers	6	7	8	9	10	15	20	25
Hidden layer size	256	256	256	256	256	256	256	256
Activation	Linear	Linear	Linear	Linear	Linear	Linear	Linear	Linear
Batchsize	5	5	5	5	5	5	5	5
Epochs	100	100	100	100	100	100	100	100
Optimizer	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam
Learning rate	1e-5	1e-5	1e-5	1e-5	1e-5	1e-5	1e-5	1e-5

Table 6.5: 2D-SDA model topology options 23-30

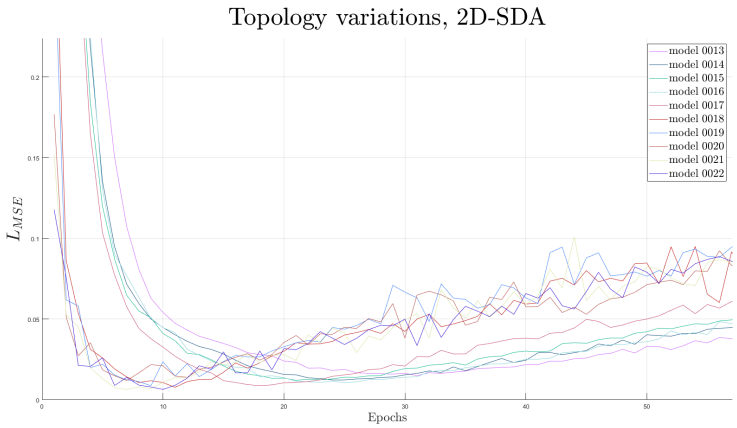


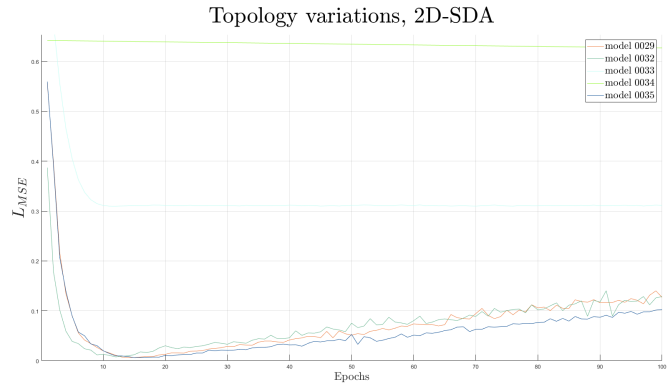
Figure 6.23: Topology options 13-22, validation loss curves

The correlation between network performance and network depth is clear in Figure 6.23. This trend halts at about 20 nodes and increasing network depth from that point on does not lead to further convergence.

A final effort to increase network convergence includes different activation functions. The sigmoid, ReLU and Tanh activation functions are tested. Besides these different activation functions, a narrow network with 20 layers is also tested. Learning rates are adjusted for the narrow network because the amount of parameters in this network is much lower. For the ReLU activation function, learning rate is also adjusted, preliminary tests showed much better convergence with a higher learning rate.

Option	32	33	34	35
Number of layers	20	20	20	20
Hidden layer size	32	256	256	256
Activation	Linear	Sigmoid	ReLU	TanH
Batchsize	5	5	5	5
Epochs	100	100	100	100
Optimizer	Adam	Adam	Adam	Adam
Learning rate	1e-4	1e-5	1e-3	1e-5

Table 6.6: 2D-SDA model topology options 32-35



(a) Full image

(b) Zoomed in

Figure 6.24: Topology options 29,32:35, validation loss curves

The narrow network of 32 nodes per layer performs as well as the wide network. Where the narrow network only has 207,000 network parameters and the wide network 2,6 million. Increasing the amount of nodes per layer greatly increases the computational cost with only a small gain in accuracy.

Sigmoid and ReLU activation functions are clearly not able to represent the computations of this particular model. TanH on the other hand, performed slightly better than the linear activation functions and therefore shall be used on the bigger dataset and for the alternative cases.

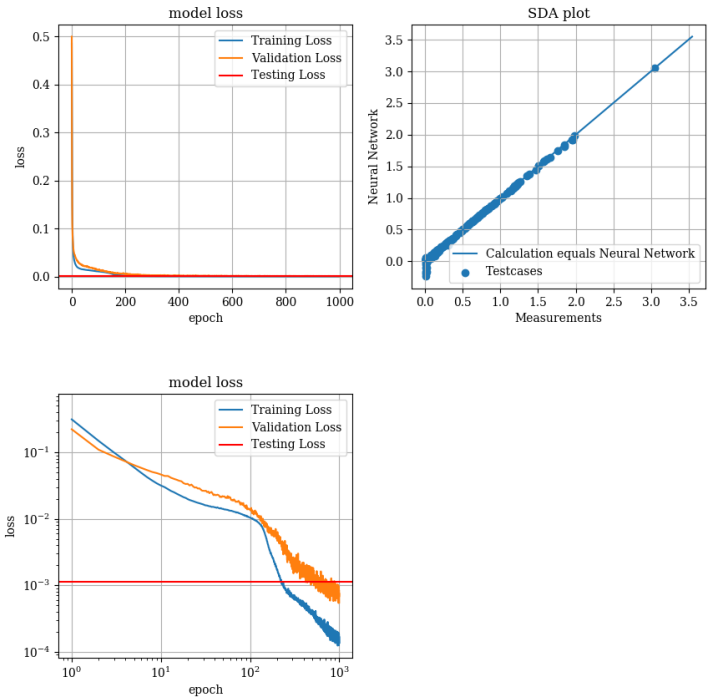


Figure 6.25: 1000 examples, 2D-SDA standard case

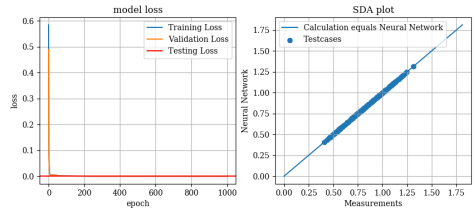
Figure 6.25 shows that the network is capable of imitating the two computations discussed at the beginning of this chapter. There is however an issue with sea-states that have an SDA of (nearly) 0. Here, the network outputs negative values in some cases. This could be solved by implementing constraints.

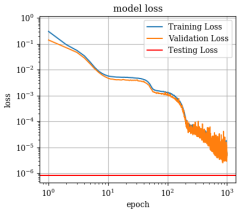
As the 2D model with the full response spectrum as an output gives satisfactory results for the artificial environment. The 2D-SDA model, which contains a lot less information in it's output, is essentially not a useful addition. However should the results of the 2D model be disappointing when dealing with real measurement data, this model type is a potential alternative.

Alternative cases

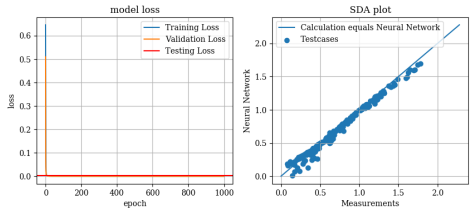
The network, with it's topology designed for the standard case, is also tested on the alternative cases: interpolation, extrapolation and scarce. The results are presented in Figure 6.26. In the interpolation case, with the network trained on high and low sea-states with high and low peak periods, the model performs well. With the absence of very low sea-states, the only downside seen in the standard case is also gone. The model performs less well when the extrapolation case is applied. But the general trend is still clear. The network is thus capable of making somewhat accurate predictions in sea-states it has not encountered before.

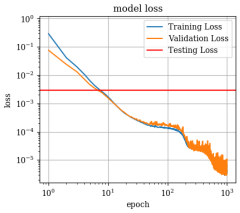
As encountered earlier in topology optimization, the single output parameter makes this network prone to overfitting. The difference in trend of the training loss and validation loss curves in Figure 6.26c also indicated overfitting and explains the outliers in the SDA plot.



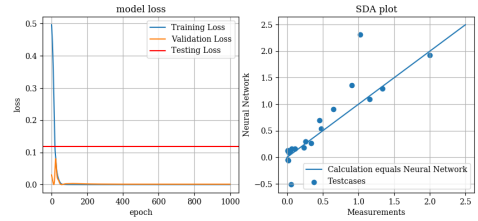


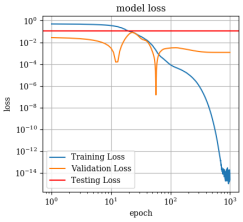
(a) Interpolation case





(b) Extrapolation case





(c) Scarce case

Figure 6.26: 2D-SDA model, alternative cases

6.1.7. Parametric

One of the issues with the 2D model is the amount of input and output nodes. Too many nodes will cause training difficulties that can only be overcome by more data (and in some cases time). The amount of data however is limited. Decreasing the amount of input nodes will lower the accuracy and reducing the amount of output nodes reduces the resolution.

In some applications, the vessel motions are not represented by the entire frequency response spectrum but by the Significant Double Amplitude (SDA). This is a value that represents the entire spectrum in one integrated value. This value of course holds less information and it is possible for two spectra to have the same SDA, but it many practical applications it is a useful measure. As for the input spectrum, the wave spectrum can also be represented by a few variables instead of the entire spectrum in all heading and frequency components. A simple spectrum can be represented by the significant wave height (Hs), the peak period (Tp) and the main wave direction (Θ). More complex spectra can be represented by multiple sets of these values. For example one set for swell, and two sets for wind waves:

$Hs_s, Tp_s, \Theta_s, Hs_1, Tp_1, \Theta_1, Hs_2, Tp_2, \Theta_2)$.

Using this 'parametric' model, the amount of nodes can be decreased dramatically. Only the question arises: does this limited amount variables hold enough information to describe the complex nature of the SDA value?

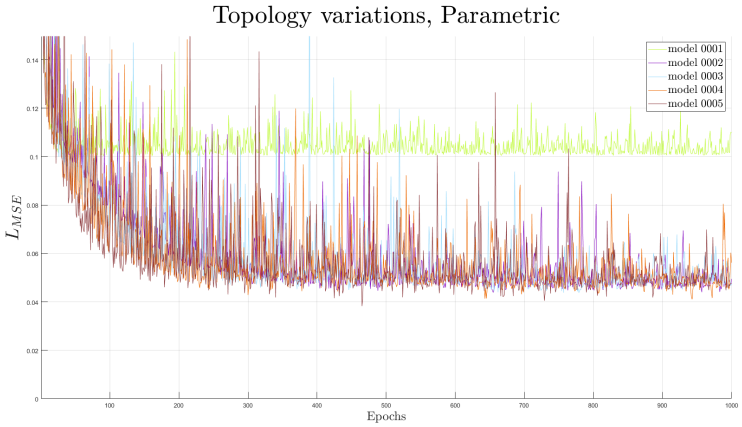


Figure 6.27: Topology options 1:5, validation loss curves

The validation loss curves in Figure 6.27 show faster convergence with an increase in the amount of nodes per layer. But it does not show further convergence. To keep training time low, further tests are done with 32 nodes per hidden layer. In a later stage, optimization through layer size will be applied again. In an effort to increase the sudden fluctuations in the validation loss curves, learning rate is decreased to 1e-5.

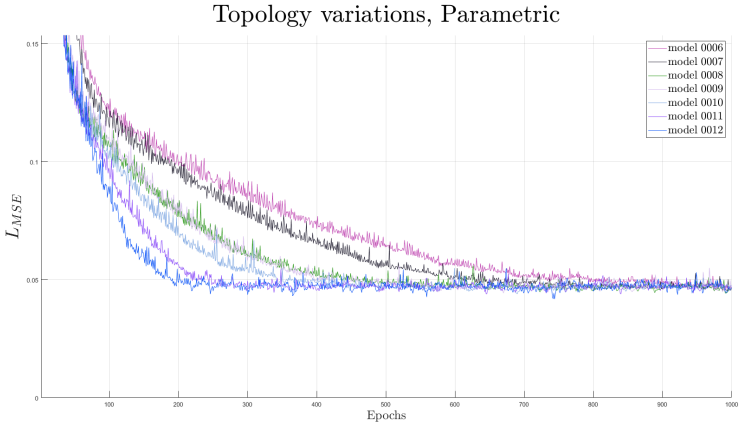
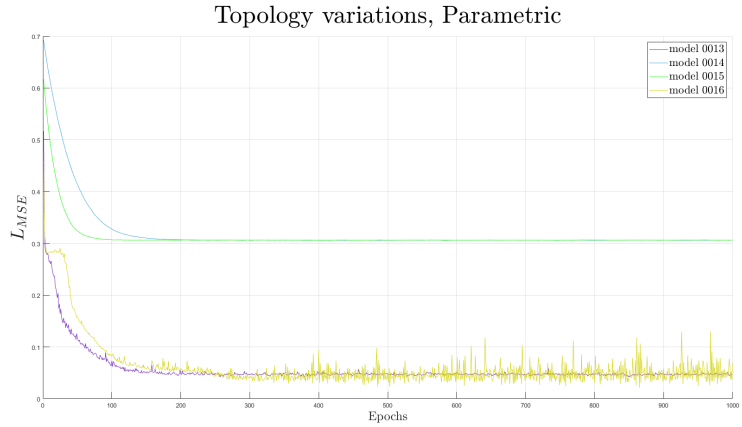
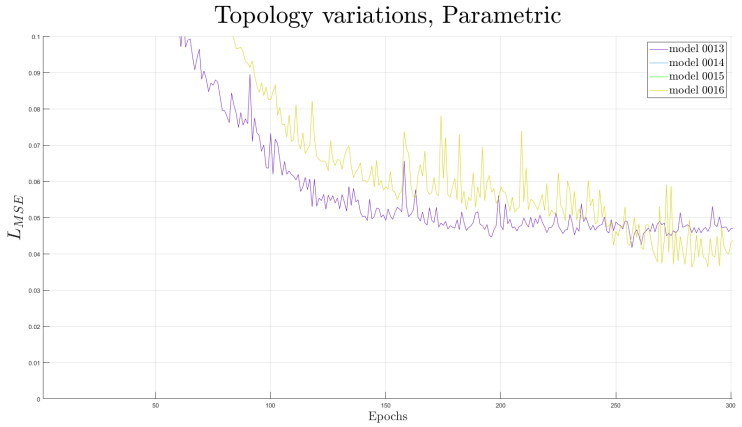


Figure 6.28: Topology options 6:12, validation loss curves

Figure 6.28 shows that adding more layers, has the same effect as adding more nodes per layer; the rate of convergence is higher, but all loss curves stop decreasing at the same loss value. This means that either the limit of correlation between input and output has been reached or a network consisting of linear activations is not capable of finding this correlation. This is determined by performing tests with sigmoid, ReLU and TanH activations.



(a)



(b) Zoomed in

Figure 6.29: Topology options 13:16, validation loss curves

Variations in activation functions show that the sigmoid and ReLU functions stop converging fairly quick. This is due to the biases taking over as the dominant parameter. Therefore, all predictions are the same no matter the network input. The TanH function on the other hand is a slight improvement to the linear activation. As different activation functions can react differently to layer sizes, the TanH function is tested with 256 nodes per layer in Figure 6.30.

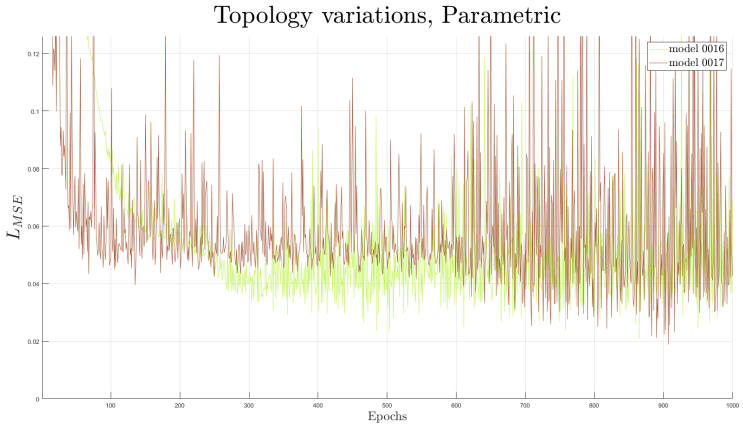
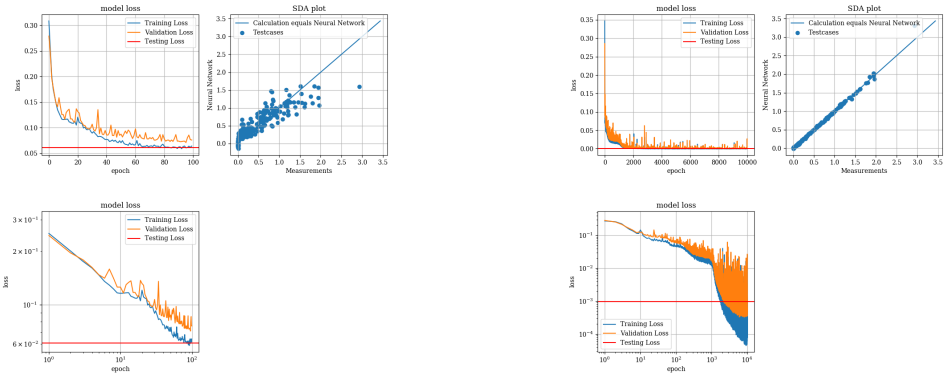


Figure 6.30: Topology options 16 and 17, validation loss curves

As the increase of layer size did not increase network performance, the layer size of 32 nodes is used for the final network topology. The network is used on the 1000 example dataset. The result of which is shown in Figure 6.31a. The SDA plot shows a clear trend around the diagonal, but with a large spread. Because the validation curve is still converging, an effort is made to increase convergence. Increasing learning rate however, results in divergence and the network training time is at the maximum that is still workable. Increasing the amount of epochs therefore is only acceptable if the batch size is also increased. This training process is presented in Figure 6.31b. The result of this effort is a good match in the SDA plot, but a learning process that does not seem to be very robust. Increasing robustness of this training process can be achieved by tuning optimizer hyper-parameters or using a different optimizer altogether. As the SDA plot results are satisfactory, no further changes are made in the optimizer or the network.

The SDA plot shows that the model is capable of adapting to only knowing three parameters instead of the full wave spectrum in heading-frequency bins. This does not necessarily mean that it is capable of this to the same extend when dealing with real world predictions and measurements. For the artificial environment, a random set of wave spectrum parameters is used to create a 2D spectrum using the Ochi-Hubble method. In reality, wave spectra can not be described with the same simplicity. However the method does show promise as an alternative to the 2D method if it should not work with real world data.

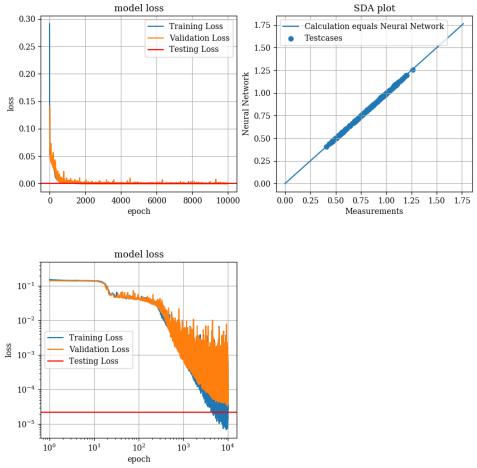


(a) (b)

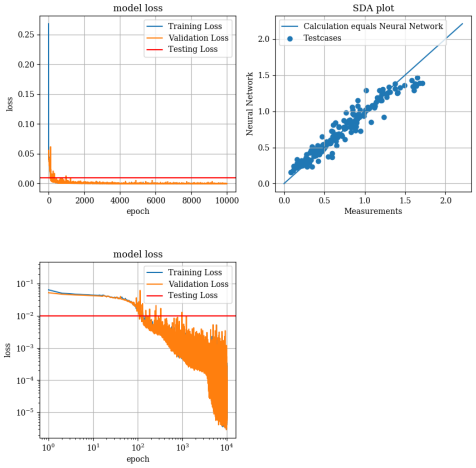
Figure 6.31: Standard case, 1000 training examples, validation loss curves

Alternative cases

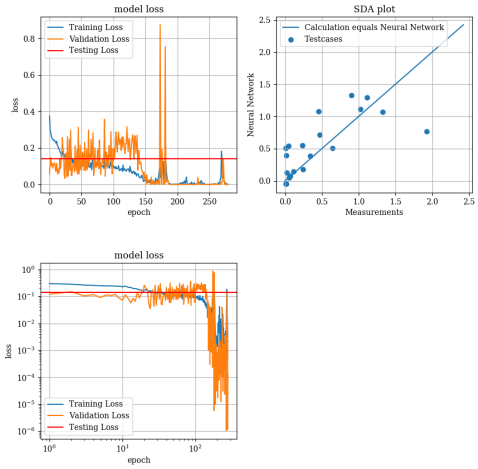
In the alternative cases, this model shows similar behavior than the other architectures, but a little more extreme. The interpolation case yields good results and shows a near-perfect SDA plot. The extrapolation case and scarce case on the other hand show less good results. It shows that this model is not the most robust and is sensitive to large input variations because of the low amount of input parameters. The real world cases show similar network performance as the extrapolation case. Unlike with the 2D model, where the Snorre-Cassia case showed much less convergence than the extrapolation case.



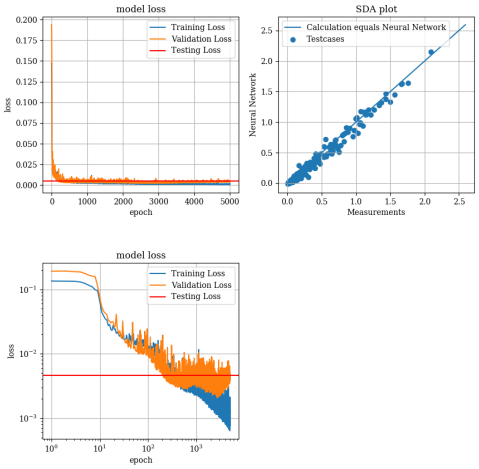
(a) Interpolation case



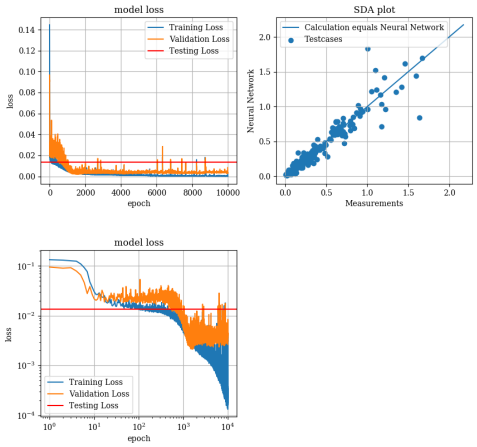
(b) Extrapolation case



(c) Scarce case



(d) North Sea



(e) North Sea - Trinidad

Figure 6.32: Parametric model, alternative cases

6.1.8. Including ship parameters

The calculation that needs to be mirrored by the neural network has been kept rather simple by keeping the RAO constant. This implies that the ship parameters such as draft and vessel speed remained constant. Whether it is possible to include these parameters as input nodes in the network is discussed in this section.

There are multiple topologies possible which would include extra ship parameters, all have their advantages and disadvantages. The following options are discussed: Linear Network, Multiple Linear Networks, Deep Network, Function Approximation, Integrated Variables.

Linear Network This network is established by simply adding an extra dimension to the input layer. This way, the network is able of using the same simple topology as before. However, the size of the input layer is now dependent on the resolution of θ, ω and D. Adding this extra dimension to the already sizable input layer, can lead to unworkable sizes.

Another disadvantage of this topology is the limited predictive capability. As the network is just filling in a very extensive RAO. It is likely that it will not perform well in sea-states that it has not encountered yet.

Multiple Linear Networks As only a few vessel drafts are common, it is also a possibility to have separate networks for each draft or each set of ship parameters. This way, the network cannot be used in predicting situations in unfamiliar drafts due to the lack of data, but it can become very accurate in predicting motion behavior in common vessel drafts.

Deep Network In an effort to construct a network capable of making predictions in unfamiliar sea-states and/or drafts, using a deep network topology is one of the main possibilities. The aim of a deep network is to make the network capable of making complex, non-linear connections between the input and output layer.

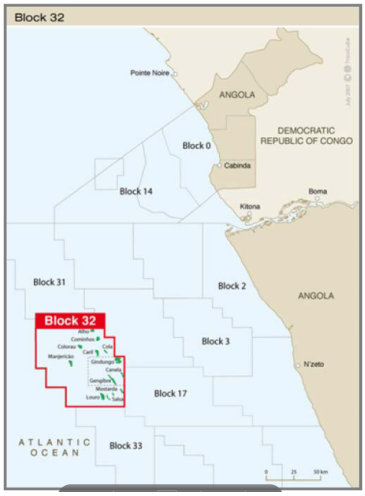
Parametric The implementation of draft or other ship parameters can lead to large input sizes. Using only a few parameters to describe the wave spectrum avoids this issue. The downside is that there is much less detail in the spectrum and some wind swell spectral contributions could be completely lost.

6.2. Measurement Data

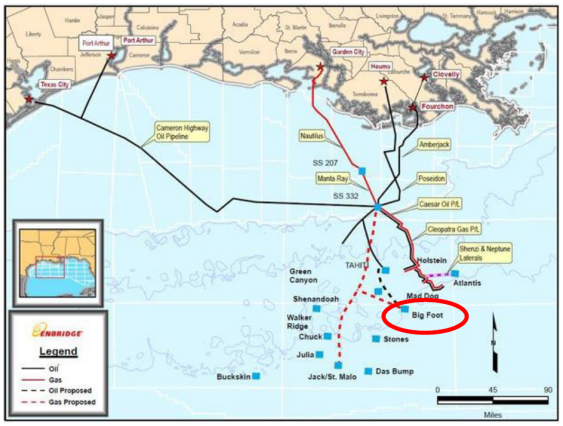
In the previous sections, multiple types of models are tested on a artificial environment. Both with completely random data and data that mirrors certain sea states. Now that is established which methods have potential, promising models can be tested with measurement data. The models will be tested on roll motions of a few specific transits and a project.

Projects

The project data is used to train the network on the 'standard' case where the vessel sails at operational draft. Roll motion MRU measurement data is taken from two projects: 'Kaombo' and 'Bigfoot'. For the training data, the Kaombo project is used, which is a project near the coast of Angola (Figure 6.33a). It totals 822 training examples. For the test data, Bigfoot data is used. Bigfoot is a project in the Gulf of Mexico. It totals 276 examples.



(a) Kaombo-B32



(b) Bigfoot

Figure 6.33: Projects

Transit

For the non-standard case, the dataset consists of transit data of the Balder from 2010-2011. Like the the projects, roll motion MRU measurements are used. The main differences in ship parameters compared to the projects are a shallow draft and forward speed. In Figure 6.34, the three transits of that period are plotted. The dataset is randomly seperated into test-data and training data. The training dataset is composed of 563 training examples and a test dataset composed of 161 examples. This amounts to roughly $70 + 20 = 90$ days worth of data.

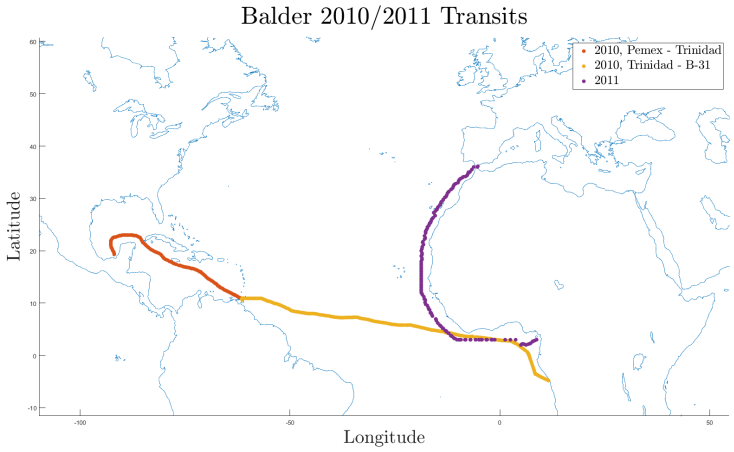


Figure 6.34: Balder transits used for neural network data

6.3. Standard cases; project

6.3.1. 2D

The search for the right network topology in the 2D case gets more extensive when using real measurement data. In the artificial environment, heading is random and a large enough dataset will likely cover most of the heading-frequency bins. Real world data however does not have a uniform distribution in heading. Instead the data is more clustered and thus amplifies the weakness of the single hidden layer network.

A deep network that does not rely on a linear correlation for every heading-frequency

bin, may perform better on training data wherein a relatively small part of the 2D spectrum is covered. The downside to this is that there is a huge amount of possible topologies for this model. The restrictions on the considered topologies discussed in Section 5.1.2 are applied for pragmatic reasons.

Nevertheless, the first set of tests is performed on single-hidden layer topologies with an increase in the amount of nodes. An overview is depicted in Table 6.7.

Option	1	2	3	4	5	6	7
Number of layers	3	3	3	3	3	3	3
Hidden layer size	16	32	64	128	256	512	1024
Activation	Linear	Linear	Linear	Linear	Linear	Linear	Linear
Batchsize	5	5	5	5	5	5	5
Epochs	100	100	100	100	100	100	100
Optimizer	Adam	Adam	Adam	Adam	Adam	Adam	Adam

Table 6.7: project; topology options 1-7

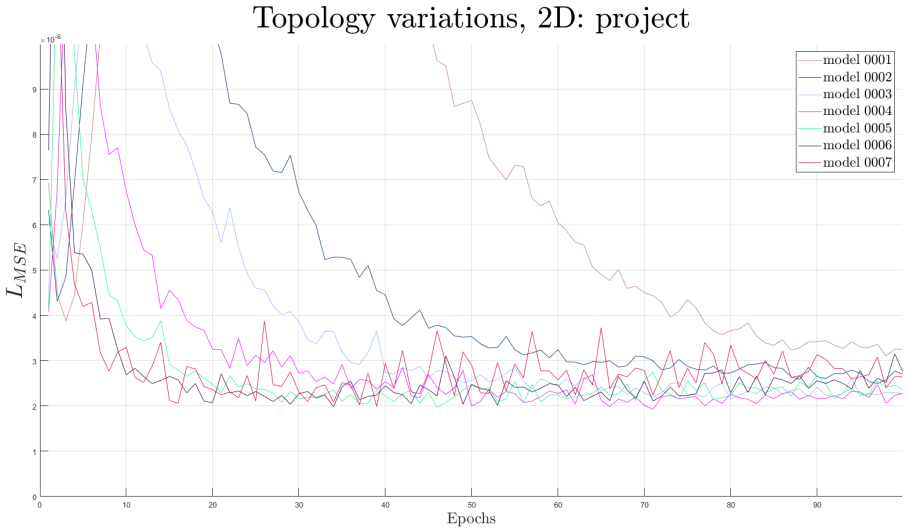


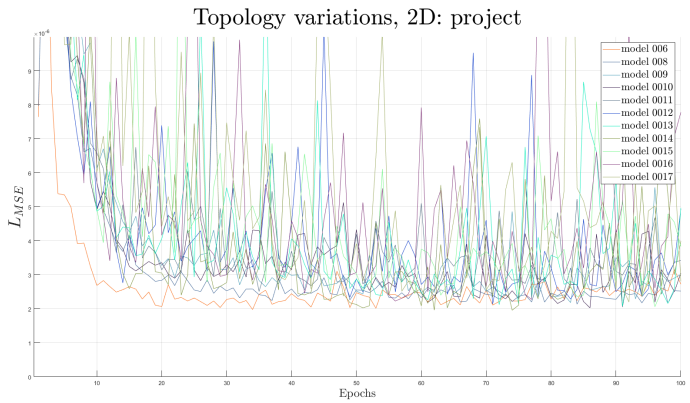
Figure 6.35: Topology options 1-7, validation loss curves

The validation loss curves in Figure 6.35 show a clear trend of a higher rate of convergence with a higher amount of nodes per layer. Since this trend slows down significantly after 128 nodes per layer, this becomes the amount of nodes for the next series of topology options. Notice that a higher rate of convergence does not lead to further convergence because all validation loss curves eventually settle on the same asymptote.

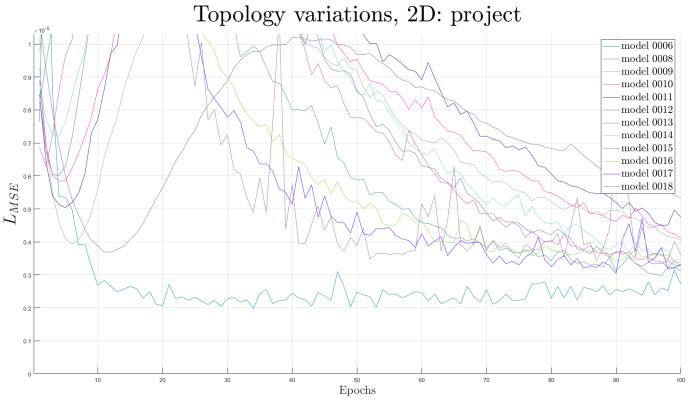
The next series of topology options is exploring the effect of network depth on the validation loss. This series, consisting of option 8-17, is depicted in Table 6.8.

Option	8	9	10	11	12	13	14	15	16	17
Number of layers	4	5	6	7	8	9	10	15	20	25
Hidden layer size	128	128	128	128	128	128	128	128	128	128
Activation	Linear	Linear	Linear	Linear	Linear	Linear	Linear	Linear	Linear	Linear
Batchsize	5	5	5	5	5	5	5	5	5	5
Epochs	100	100	100	100	100	100	100	100	100	100
Optimizer	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam

Table 6.8: project; topology options 8-17



(a) Default Adam (lr = 1e-4)



(b) Adam, lr = 1e-5

Figure 6.36: Topology options 6, 8-17, validation loss curves

The effects of network depth are limited as can be seen in Figure 6.36a. The curve with the highest rate of convergence is model 6, which has only the one single layer. Because of the high oscillatory behavior, the same series 8-17 is run with a lower learning rate. This yielded the same result (Figure 6.36b). Therefore, A single layer neural network is trained on the full dataset of the Kaombo project.

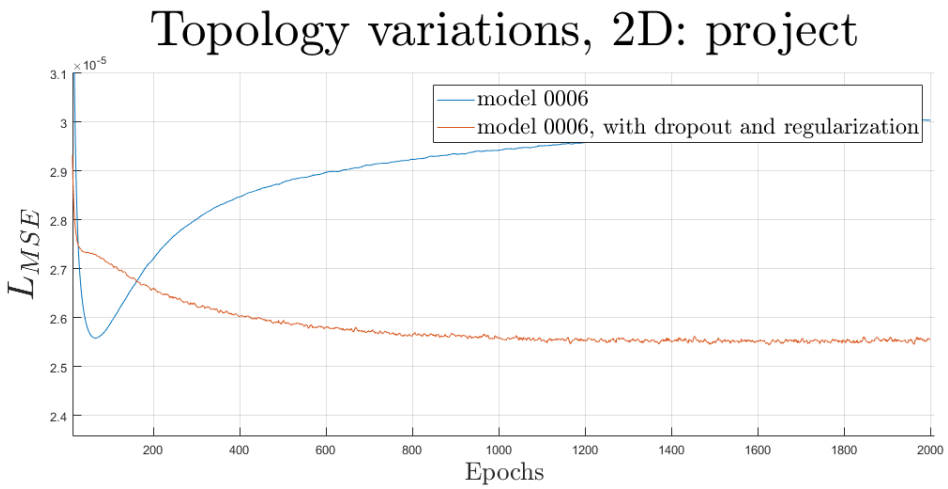


Figure 6.37: Project; all data, validation loss curves

Using the model on the entire dataset however shows overfitting however (Figure 6.37, blue curve). Overfitting mitigation measures, as described in Section 5.2.2 are applied in an attempt to reduce overfitting. The red curve in the graph represents the validation loss curve of the model with dropout and weight regularization and overfitting is indeed stopped. The results of this model are shown in Figure 6.38.

The training and validation loss curves both show that the network is converging. this, together with the absence of large fluctuations show that the network is robust and no overfitting takes place. The SDA plot shows a spread around the $y(x) = x$ line. The tests in the artificial environments show that a neural network is capable of making near perfect predictions in a fully linear environment. The spread is therefore caused by uncertainties of weather data, vessel manoeuvres, deck activity and non-linear hydrodynamic effects.

The results from the SDA plot show that the neural networks predictions have a similar accuracy as the diffraction based predictions. However, it is hard to draw conclusions from the SDA plot with different dots overlapping. Therefore the difference between the measurement SDAs and the predicted SDAs are depicted in Figures 6.39a & 6.39a by means of the contours of the distribution. The plots show that the neural network has a tendency to overpredict the SDA value but has fewer outliers than the diffraction method predictions.

In order to quantify the predictive capabilities of both the neural network and the diffraction method, the mean squared error (Equation 6.2) and a normalized mean squared error (Equation 6.4) are calculated. The results of which is depicted in Table 6.9 and show that the neural network performs better according to both measures.

	RAO	Neural network
MSE	0.0057	0.0040
NMSE	0.082	0.013

Table 6.9: MSE & NMSE for project

$$MSE(\mathbf{A},\mathbf{B}) = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{A}_i - \mathbf{B}_i)^2 \tag{6.2}$$

$$NMSE(\mathbf{A},\mathbf{B}) = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{\mathbf{A}_i - \mathbf{B}_i}{\mathbf{A}_i}\right)^2 \tag{6.3}$$

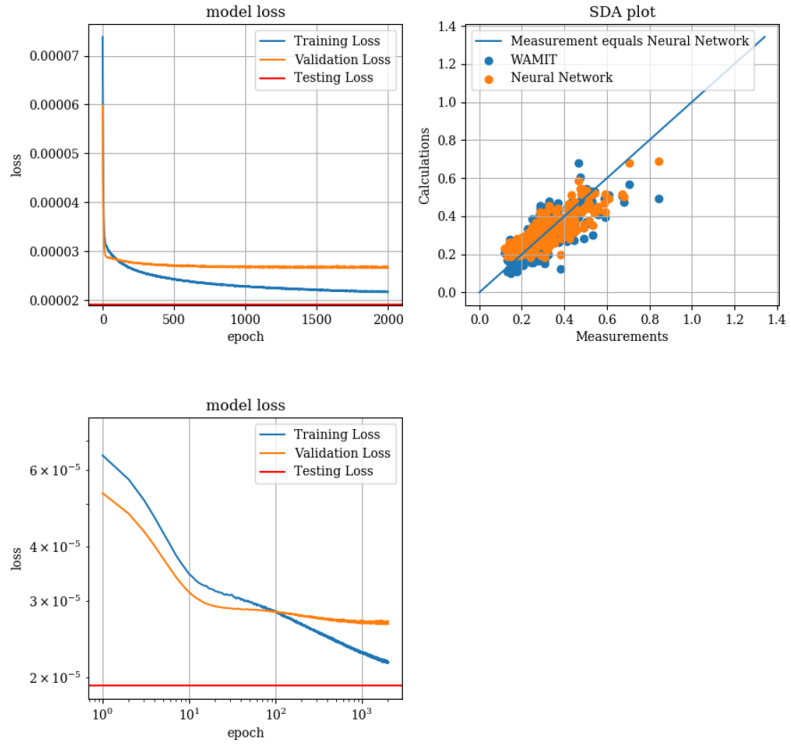
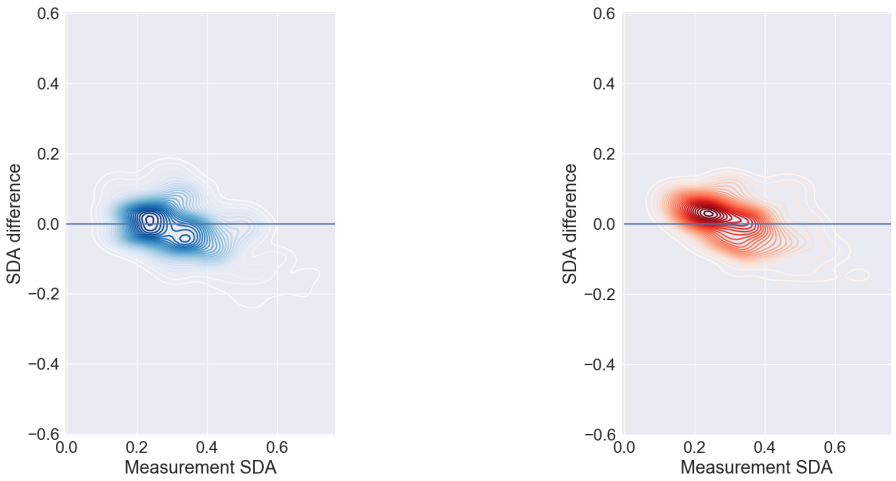


Figure 6.38: Test on model with overfitting mitigation measures



(a) WAMIT RAO (b) Neural network

Figure 6.39: SDA error distribution ($SDA_{measured} - SDA_{predicted}$)

6.4. Non-standard cases; transit

6.4.1. 2D

The success of the single hidden layer network in the project environment is reason to start the topology selection process in the transit environment with a similar network. The first batch of topology variations consist of single hidden layer networks with varying layer size and learning rate. An overview of these topology variations is displayed in Table 6.10.

Option	1	2	3	4	5	6	7	8	9	10	11	12
Number of layers	3	3	3	3	3	3	3	3	3	3	3	3
Hidden layer size	16	32	64	128	256	512	16	32	64	128	256	512
Activation	Linear	Linear	Linear	Linear	Linear	Linear	Linear	Linear	Linear	Linear	Linear	Linear
Optimizer	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam
Learning rate	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0002	0.0002	0.0002	0.0002	0.0002	0.0002

Table 6.10: Topology variations 1-12

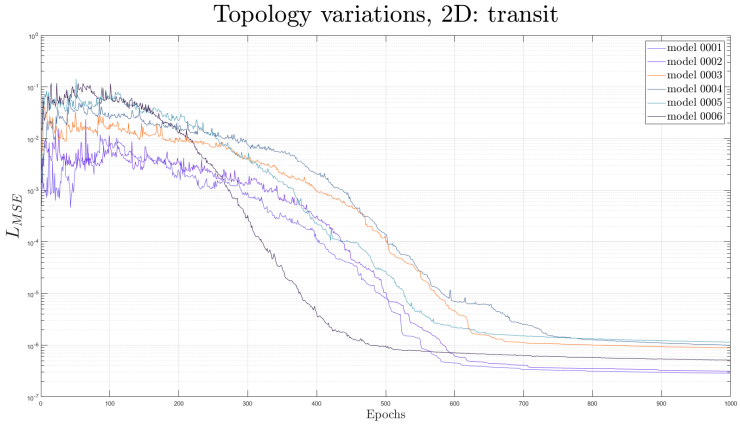


Figure 6.40: Topology options 1-6, validation loss curves

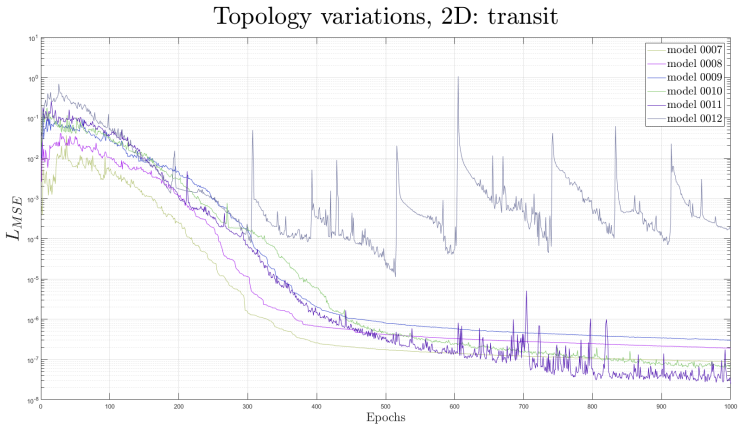


Figure 6.41: Topology options 7-12, validation loss curves

The validation loss curves of the first twelve topology variations show that the network with 256 nodes and the learning rate of 0.0002 convergences the furthest. However the validation loss curve contains high peaks as the rate of convergence decreases This indicates that the learning rate is too high. A learning rate decay may solve this problem. The network

containing 512 nodes diverges in an early stage, which might be by the same cause. Therefore, both these layer sizes are used in the next batch of topology variations. The following variations are displayed in Table 6.11 and contain different decay values.

Option	13	14	15	16	17	18	19	20
Number of layers	3	3	3	3	3	3	3	3
Hidden layer size	256	512	256	512	256	512	256	512
Activation	Linear	Linear	Linear	Linear	Linear	Linear	Linear	Linear
Optimizer	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam
Learning rate	0.0002	0.0002	0.0002	0.0002	0.0002	0.0002	0.0002	0.0002
Decay	1e-4	1e-4	1e-5	1e-5	1e-6	1e-6	1e-7	1e-7

Table 6.11: Topology variations 13-20

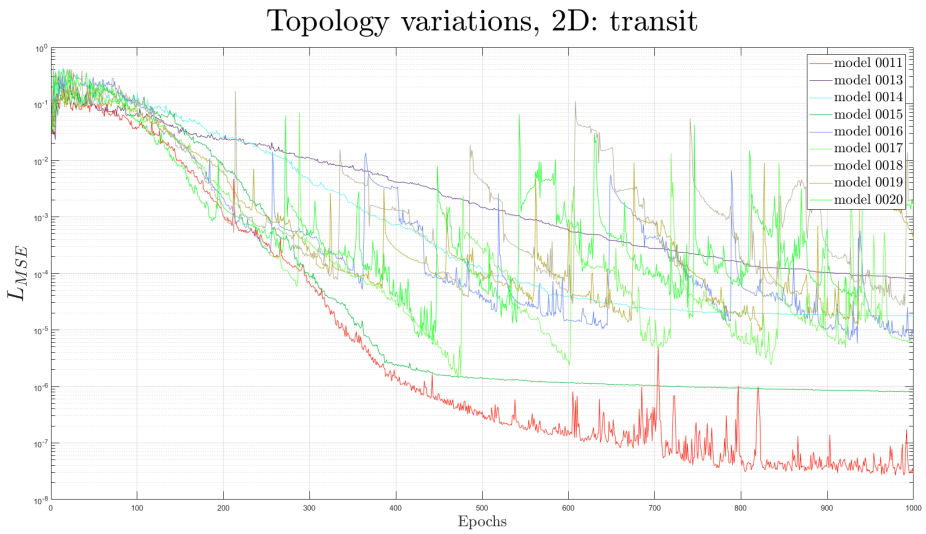
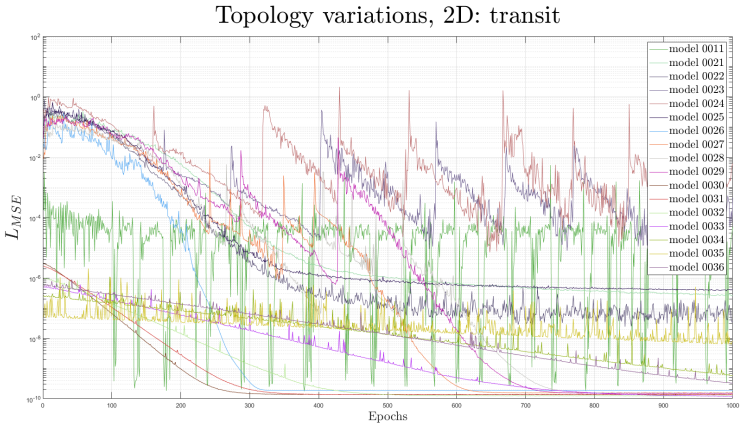


Figure 6.42: Topology options 13-20, validation loss curves

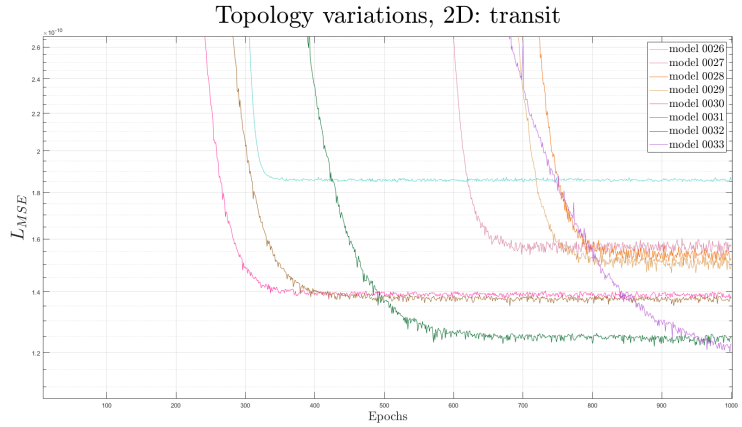
In Figure 6.42, the new topology variations are compared. Although learning rate decay is able to reduce fluctuation of the loss curve in some cases, non of the networks containing decay converge further than the model without decay. Another method of increasing convergence has to be found. In the next batch of variations, multiple network additions are tested: dropout, regularization and parameter constraints (all are discussed in Chapter 5). All tested variations are shown in Table 6.12.

Option	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
Number of layers	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
Hidden layer size	256	256	256	256	256	256	256	256	256	256	256	256	256	256	256	256
Activation	Linear	Linear	Linear	Linear	Linear	Linear	Linear	Linear	Linear	Linear	Linear	Linear	Linear	Linear	Linear	Linear
Optimizer	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam
Learning rate	0.0002	0.0003	0.0003	0.0004	0.0003	0.0002	0.0002	0.0002	0.0002	0.0002	0.0002	0.0002	0.0003	0.0003	0.0004	0.0003
Decay					1e-5											
Dropout	10%		10%	10%	10%				10%	10%	10%	10%	10%	10%	10%	10%
Weights regularizer						1e-6	1e-7	5e-8	5e-8	5e-8	5e-8	2e-8	1e8	5e-9	1e-9	1e-8
Weights constraint										>0	>0	>0	>0	>0	>0	>0
Bias constraint						=0	=0	=0	=0	=0	=0	=0	=0	=0	=0	=0

Table 6.12: Topology variations 21-36



(a)



(b)

Figure 6.43: Topology options 21:36, validation loss curves

The validation loss curves are displayed in Figure 6.43a and for readability reasons, the best performing networks are again shown in Figure 6.43b. A combination of all introduced measures (network 30) results in the best network performance.

As a network with only a single hidden layer has only limited capability in adapting to non-linear physical phenomena, a batch with an increasing number of hidden layers is tested along with an additional network with TanH activation functions. This batch of topology variations is depicted in Table 6.13.

Option	37	38	39	40	41	42	43	44	45	46	47
Number of layers	4	5	6	7	8	9	10	15	20	25	25
Hidden layer size	256	256	256	256	256	256	256	256	256	256	256
Activation	Linear	Linear	Linear	Linear	Linear	Linear	Linear	Linear	Linear	Linear	TanH
Optimizer	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam
Learning rate	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
Dropout	10%	10%	10%	10%	10%	10%	10%	10%	10%	10%	10%

Table 6.13: Topology variations 37-47

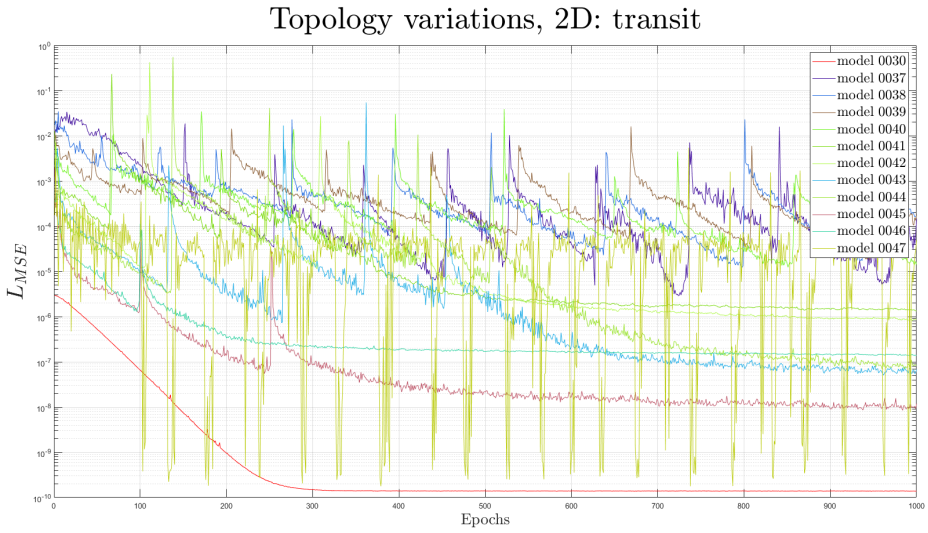


Figure 6.44: Topology options 37-47, validation loss curves

As shown in Figure 6.44, extending the network with more layers does not lead to better predictions. Therefore, no further changes are made to the network. The single hidden layer neural network with measures to increase robustness and decrease the chances of overfitting is used to make predictions on the test data set. Serving as a benchmark are predictions made by means of an identified RAO (Section 4.1.2). A acceleration RAO, available at HMC, was originally calibrated on roll accelerations. For the purpose of serving as a benchmark for the neural network, it is transformed to a motion RAO by dividing its amplitude components by the squared frequency.

The results are shown in Figure 6.45. Figures 6.46a & 6.46b show the SDA difference distributions. The majority of the test examples show a good trend on the diagonal of the SDA plot. However, a part of the test examples is significantly under-predicted. A part of the spread around the $y(x) = x$ line can be ascribed to the same causes as in the project data tests: uncertainties of weather data, vessel manoeuvres, deck activity and non-linear hydrodynamic effects. The main two differences in the circumstances compared to the project data are a shallow draft and forward speed. Extra non-linearities are hereby introduced. The forward speed also changes the encounter frequency of the incoming waves. As the network input (a 2D wave spectrum) is adjusted for heading but not for forward speed, the network will adapt to the mean forward speed (or a speed close to this). But the network is not capable of adjusting for changes in forward speed as it is not provided as an input.

The neural network predictions show a closer match with the measurements than those of the identified RAO. To quantify this, the mean squared error (Eq. 6.4) and the normalized mean squared error (Eq. 6.5) are calculated. The NMSE is modified to limit the effect of the very low SDAs by dividing by a value of at least 0.1. The calculated values are depicted in Table 6.14.

The test for the non-standard case/transit case show that there is potential for a neural network to act as an RAO when sailing at inconvenient draft, for the vast majority of the test examples clearly follows the SDA plot diagonal. However, like the RAO based method, the network accuracy is also reduced significantly during transits.

	RAO	Neural network
MSE	0.707	0.057
NMSE	3.45	0.16

Table 6.14: MSE & NMSE for transit

$$MSE(\mathbf{A},\mathbf{B}) = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{A}_i - \mathbf{B}_i)^2 \tag{6.4}$$

$$NMSE(\mathbf{A},\mathbf{B}) = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{\mathbf{A}_i - \mathbf{B}_i}{max(\mathbf{A}_i, 0.1)}\right)^2 \tag{6.5}$$

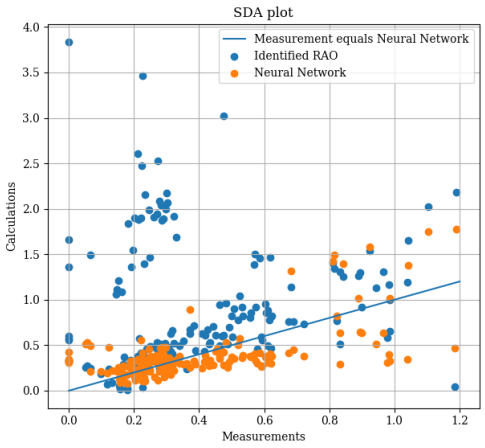
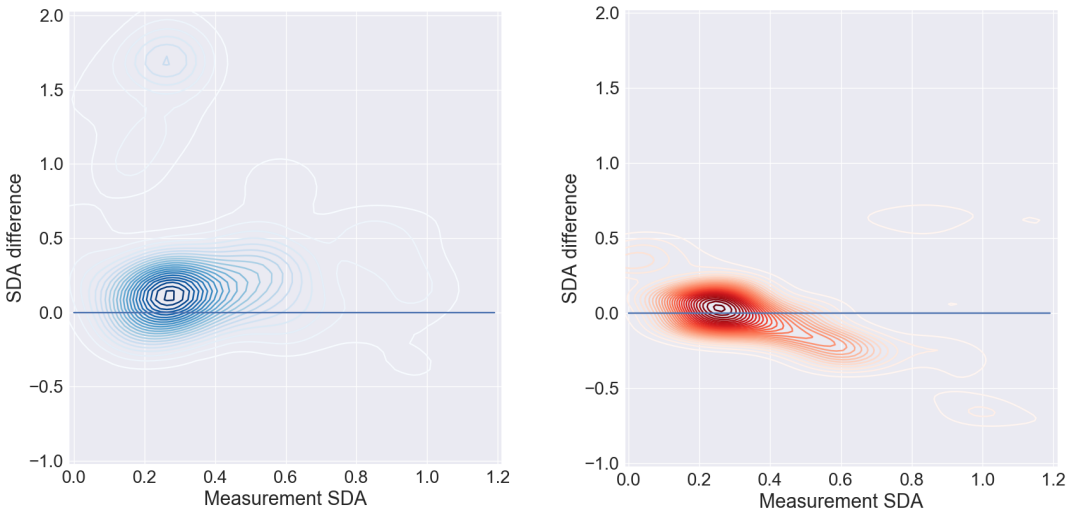


Figure 6.45: Non-standard case, test SDA plot



(a) Identified RAO

(b) Neural network

Figure 6.46: SDA error distribution ($SDA_{measurement}$-$SDA_{predicition}$)

7

Time Domain

This chapters discusses the research into two potential uses of a neural network in predicting or estimating hydrodynamic behavior. The first uses model test data two learn a neural network the link between surface elevation and ship motion. The second uses MRU hindcast data to try and predict the future of the same signal.

7.1. Wave ⇒ Response

In this section, model test data is used to train a neural network. The goal of this is to determine the potential of a NN based model in acting as a real time transfer function between wave input and vessel motion output.

7.1.1. Data

For this assessment of neural network capability, data is used from two model tests conducted at Marin. During these modeltests, both surface height and pitch motion are measured. Each modeltest lasts 3.5 hours and the measurement sample rate is 4 Hz. Both datasets are splitt, with the first 80% (Figures 7.1 & 7.5) as training data and the last 20% (Figures 7.3 & 7.7) as test data. As the signal contains noise, this is mitigated by using the Matlab 'smoothdata' function. This returns a moving average of the signal. The result of this can be seen in two snapshots of the datasets in Figures 7.2, 7.4, 7.6 & 7.8 .

For both modeltests, the model is at operational draft ($T = ##m$ in full scale) and the model heading is 0°. The waves are uni-directional and come from 180°. The seastate conditions are depicted in Table 7.1. The sea-state 2 is particularly interesting because the pitch motion of the vessel model is at a lower frequency than that of the waves. This can be observed in Figure 7.6. The motion that is observed is caused by second order wave drift forces.

Sea-state nr.	Duration $[hours]$	$4\sqrt{m_0}$ $[m]$	T_p $[s]$	Shape factor γ
1	##	##	##	##
2	##	##	##	##

Table 7.1: Marin modeltest seastates

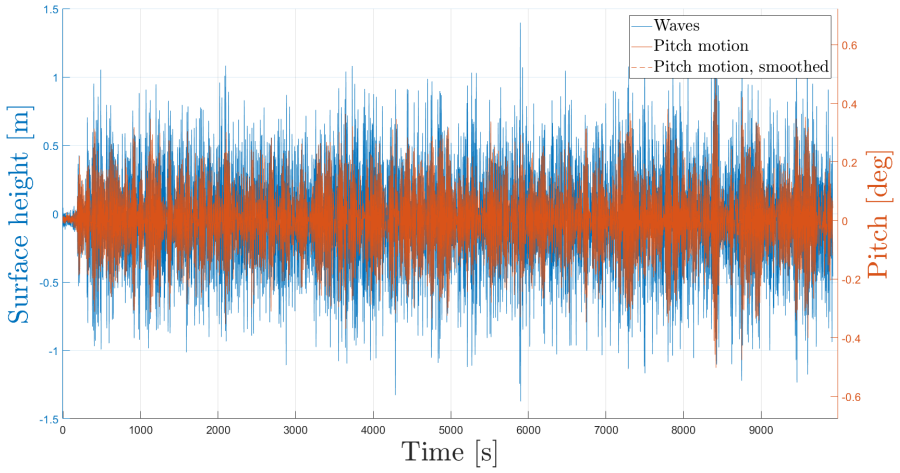


Figure 7.1: Modeltest training data, full timeseries

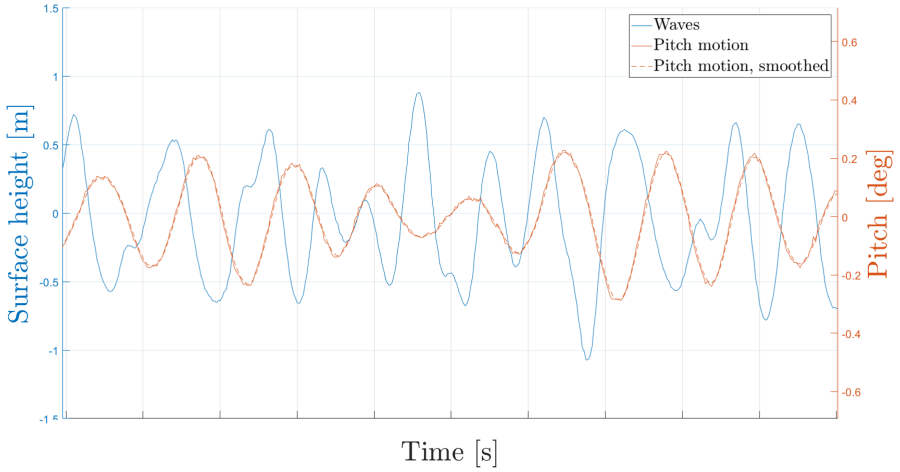


Figure 7.2: Modeltest training data, zoom-in

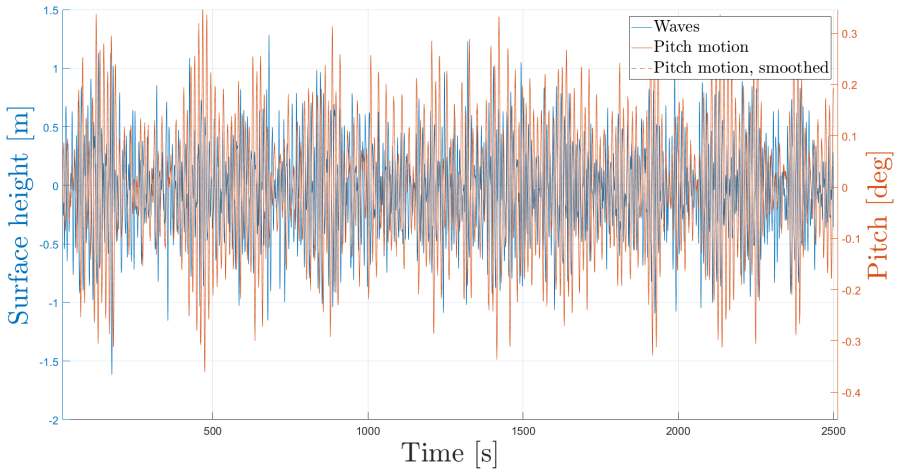


Figure 7.3: Modeltest test data, full timeseries

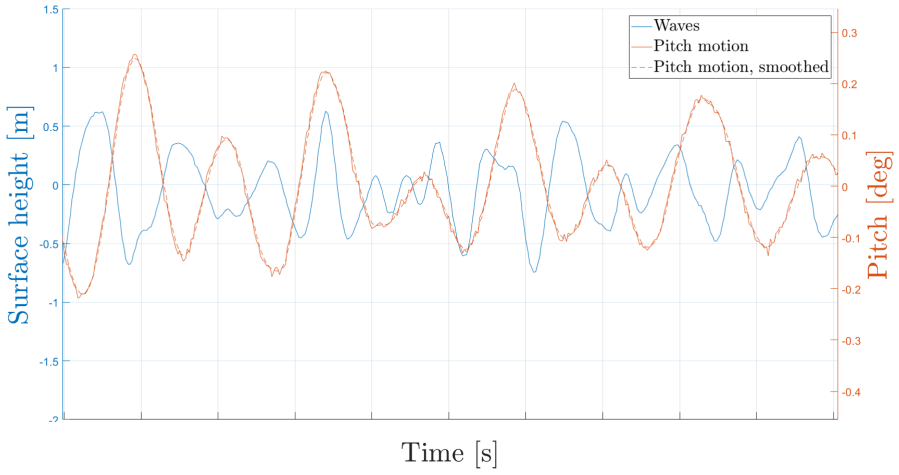


Figure 7.4: Modeltest test data, zoom-in

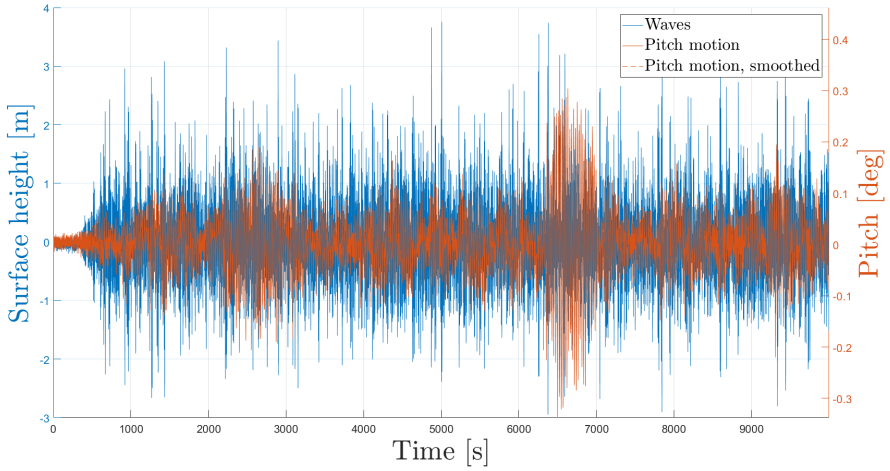


Figure 7.5: Modeltest training data, full timeseries

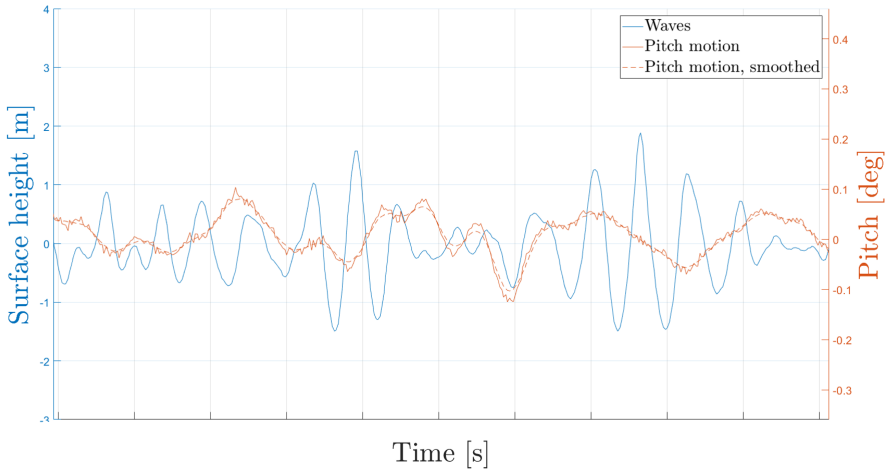


Figure 7.6: Modeltest training data, zoom-in

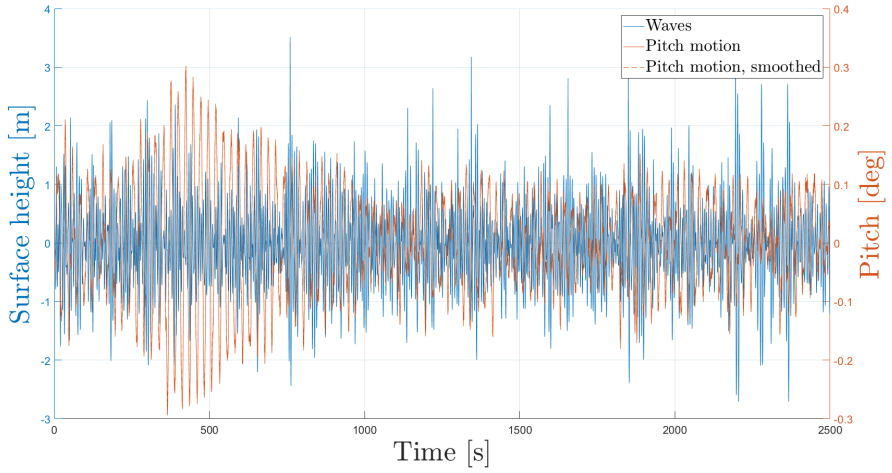


Figure 7.7: Modeltest test data, full timeseries

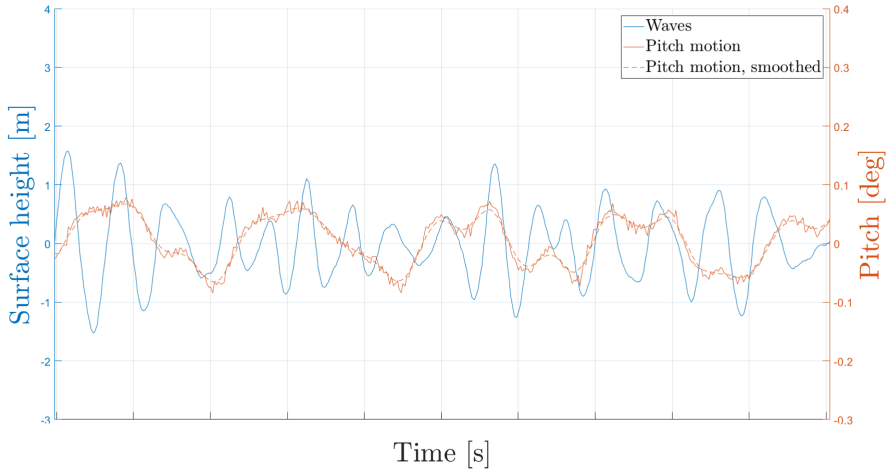


Figure 7.8: Modeltest test data, zoom-in

7.1.2. Topology selection

As the second sea-state contains second order responses, the computations that the neural network has to mimic are more complex than for the first sea-state. Therefore it is likely that a network that can make accurate predictions in the second sea-state, will also be able to do so in the first sea-state. That is why the topology selection process is done using the second sea-state, containing the second order responses.

Unlike the network topology variations in the frequency domain, network performance in the time domain is judged on the training loss curve. The dataset is large and the amount of epochs relatively low. This reduces the risk of overfitting and therefore the training loss curve provides a relatively reliable insight in network performance. The training loss curve saved regularly during the training process. These curves vary in length in this chapter. In general, some training runs are stopped once the training loss curve was long enough to draw a conclusion. Some cases runs are longer because they ran overnight or other moments of absence.

The topology variations for this model are displayed in Tables 7.2 & 7.3. Changes to topology and optimizer are made after each training run instead of in batches. This is because

training times are much longer than the frequency tests. The training run times for these variations vary between 2-12 hours.

Variation	1	2	3	4	5	6	7	8	9	10	11	12
Input seconds	15	30	45	60	75	90	105	120	135	150	165	180
Input timesteps	60	120	180	240	300	360	420	480	540	600	660	720
LSTM modules	60	120	180	240	300	360	420	480	540	600	660	720

Table 7.2: Topology variations 1-12

Variation	13	14	15	16	17	18	19
Input seconds	180	15	300	600	600	600	600
Input timesteps	720	60	1200	2400	2400	2400	2400
LSTM modules	60	720	60	60	60	60	60
Decay					1e-5	1e-4	1e-3

Table 7.3: Topology variations 13-19

The training curves of de modeltest network are shown in Figure 7.9. The first twelve topologies are a series of increasing input size and network size. For each network input datapoint, an extra LSTM is added. Each step results in an increase in network performance. At 180 seconds, or 720 timesteps, the amount of network nodes is at its maximum. The amount of network parameters (Equation 7.1) for 720 LSTM modules and 720 input values is $4,150,080$. The increase in network performance is caused by either the increase in input values, the increase in LSTM modules or both. Further variations conclude that the increase in input values outweighs the increase in LSTM modules, but that both have a positive influence on network performance. A balance between network performance and computational cost is found at 600 seconds of input signal and 60 LSTM modules. As symptoms of a too high learning rate show up after a few epochs, the effect is mitigated using learning rate decay. The comparison of network performance of variation 15-19 is displayed in Figure 7.10. Network variation 18 (Table 7.3) is selected as the final network. A tuning of optimizer hyperparameters will probably lead to better network performance as only a few variations are tried, but as network training takes a long time, no further changes are made to the network.

$$\begin{aligned} n_\theta &= 4 \cdot ((n_{input} + 1) \cdot n_{LSTM} + n_{LSTM}^2) \\ 4,150,080 &= 4 \cdot ((720 + 1) \cdot 720 + 720^2) \end{aligned} \tag{7.1}$$

n_θ number of network parameters

n_{input} number input values

n_{LSTM} number LSTM modules

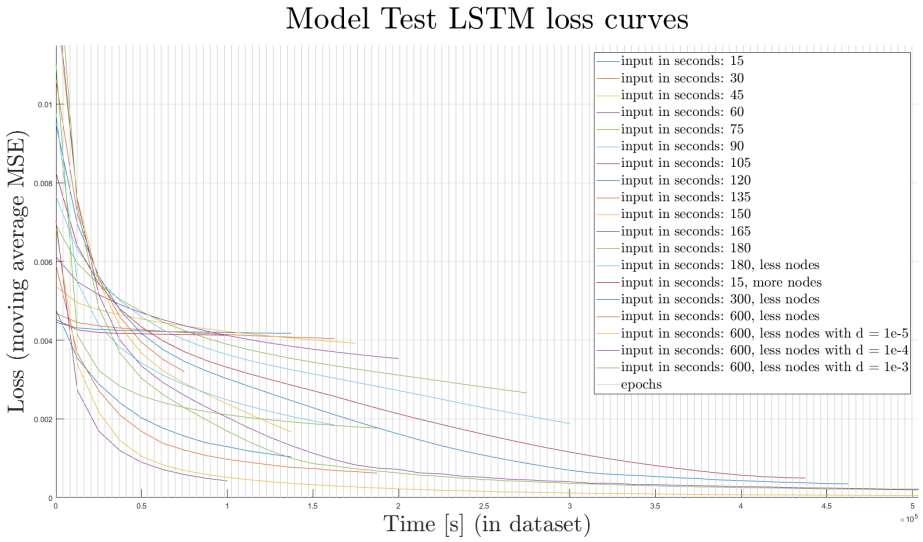


Figure 7.9: Modeltest neural network variations, training loss curves

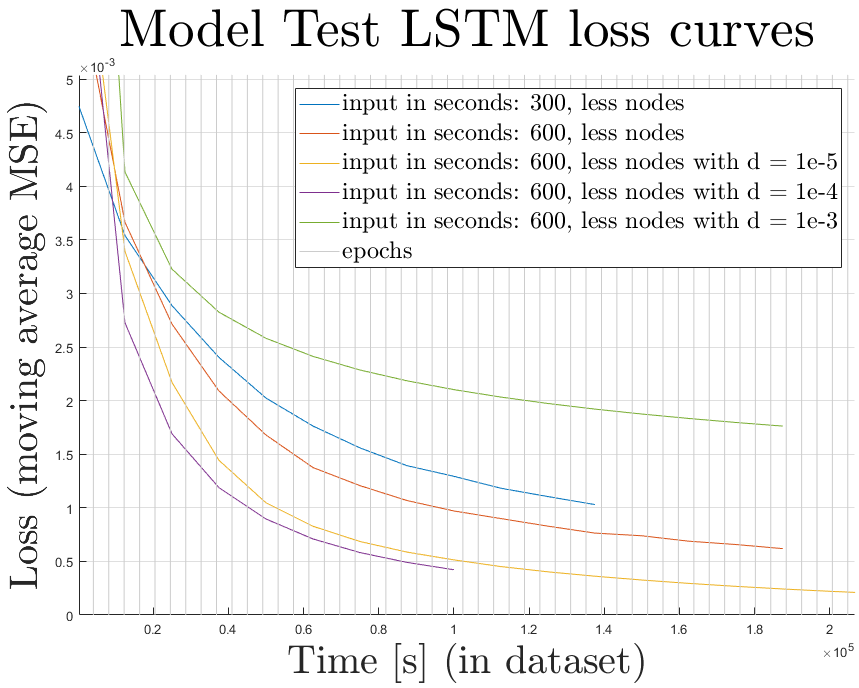


Figure 7.10: Modeltest neural network variations (selection), training loss curves

7.1.3. Results

The final network topology (variation 18, Table 7.3) is tested on the test dataset from the second sea-state modeltest. The neural network predictions of the network are compared to the measurements in Figure 7.11. In the first stage (in grey), the network is not able to make accurate predictions. The reason for this, is the large amount of input values of the network. The input consists of 2400 timesteps containing 10 minutes worth of surface height data. Therefore, the first ten minutes of the test, the network input signal is thus not large enough and gets supplemented by zeros.

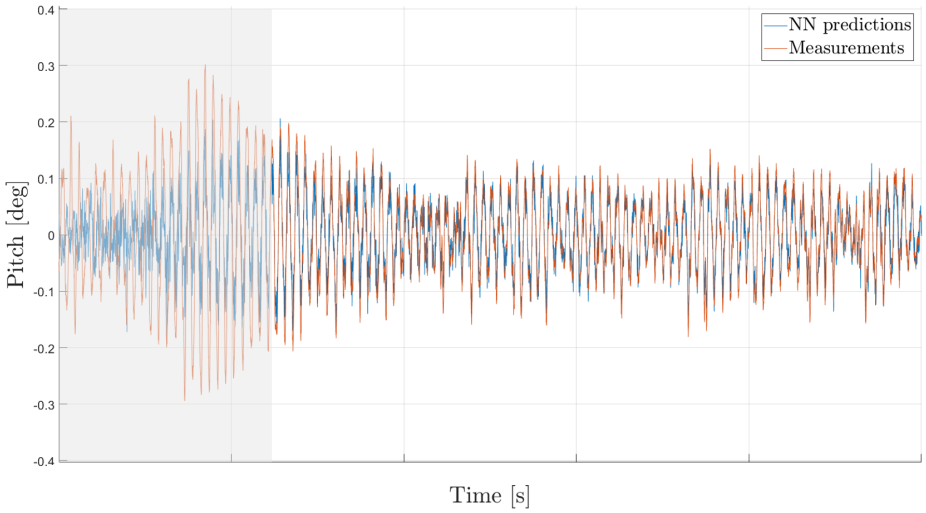


Figure 7.11: Modeltest sea-state 2, final network predictions

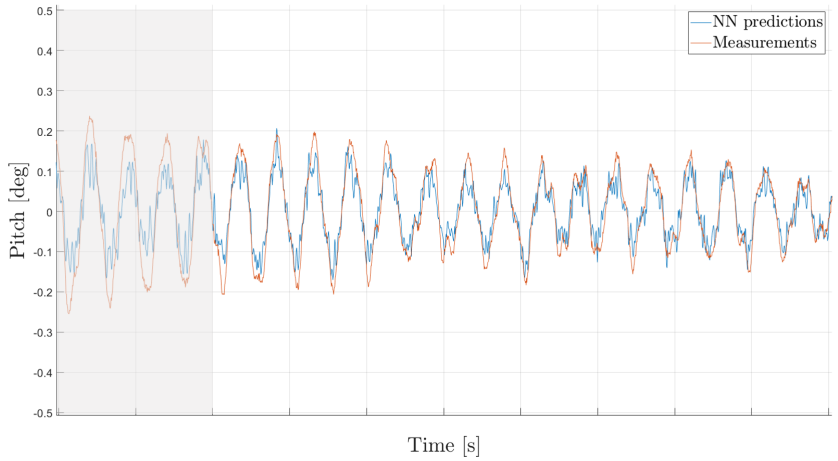


Figure 7.12: Modeltest sea-state 2, seconds 500-1000

The LSTM modules have an internal memory which extends the time that the zeros supplemented to the network input in the first stage have an effect. To determine how many seconds this internal memory lasts, a second test is done starting 750 seconds later in the timeseries. The full timeseries comparison is depicted in Figure 7.13. A zoom-in shows that the two neural networks produce the same ouput, starting 1350 seconds into the timeseries (Figure 7.14). This means the internal memory of the LSTM network does not outlast the effect of the supplemented zeros in the network.

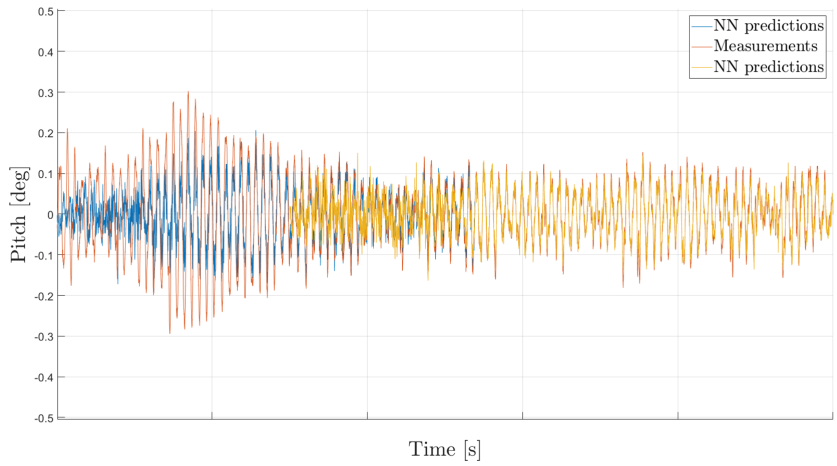


Figure 7.13: LSTM internal memory check, full timeseries

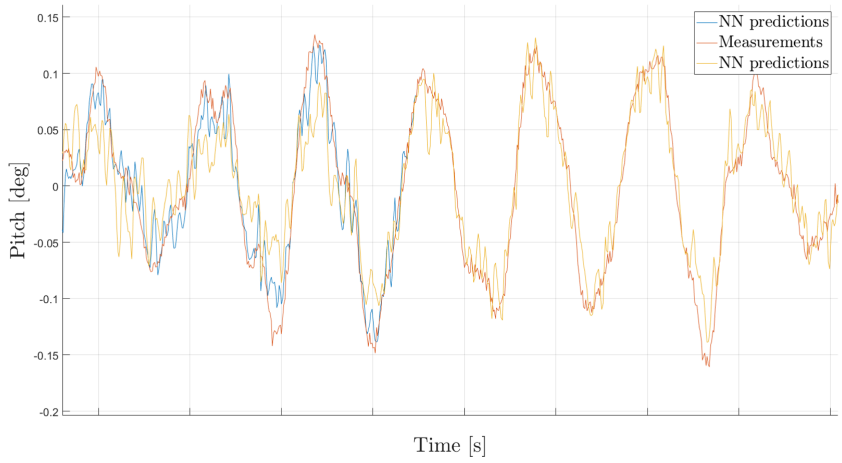


Figure 7.14: LSTM internal memory check, zoom in

A part of the first ten minutes of the timeseries is characterized by large vessel motions. After ten minutes, when there are no supplemented zeros left in the input, the network tends to slightly underpredict the vessel motions (Figure 7.12). That is a result of the large vessel motions, which are rare in the training set. Overall, the network predictions are in phase and the amplitudes also match quite close. The predictions of the full timeseries is displayed in Appendix D. The network makes accurate predictions for this sea-state and can capture second order vessel motions caused by wave drift forces. However, because of the dominance of the second order vessel motions, the first order response is of low priority in the network and there are instances where these first order motions are not captured.

The network is also tested on the first sea-state, where the first order response is dominant. Because these first order vessel motions can be described by diffraction software, the neural network output is compared with RAO based predictions as well as measurements. The results of which are displayed in full in Figure 7.15, a zoom-in of seconds 500-1000 in Figure 7.16 and the the rest in Appendix D.

As both the diffraction based prediction as the neural network based prediction correspond well with the measurement data, it is hard to judge which method produces the best

results. Quantification of the timeseries correlation by means of the Pearson correlation coefficient is a way to solve this issue [**?**]. Equation 7.2 shows the correlation coefficient definition with **A** being the measurement timeseries and **B** the predicted timeseries. μ and σ represent the mean and the standard deviation, respectively, for the corresponding timeseries. As the Pearson correlation coefficient uses a form of normalization (the correlation coefficient between $A = sin(t)$ and $B = 2 * sin(t)$ is 1), the two prediction methods are also judged on the basis of the mean squared error (Equation 7.3).

$$\rho(\mathbf{A},\mathbf{B}) = \frac{1}{N-1}\sum_{i=1}^{N}\left(\frac{\mathbf{A}_i - \mu_A}{\sigma_A}\right)\left(\frac{\mathbf{B}_i - \mu_B}{\sigma_B}\right) \tag{7.2}$$

$$L_{MSE} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{A}_i - \mathbf{B}_i)^2 \tag{7.3}$$

The Pearson correlation coefficient is calculated for prediction methods for the test dataset from $600s$ to $2500s$. The correlation coefficient for the diffraction method predictions is $\rho_D = 0.8636$. For the neural network predictions is $\rho_N = 0.8988$. The mean squared error is calculated for the same timeseries and yields $L_{MSE,D} = 0.0039$ for the diffraction method and $L_{MSE,D} = 0.0030$ for the neural network.

Both means of quantifying the results show a better performance by the neural network compared to the diffraction method. However both methods perform well in the sea-state where the first order responses are dominant and it would be premature to conclude, based on a single model test, that the neural network has better predictive capabilities in all sea-states where the first order responses are dominant. The neural network does show it is more robust as it performs well in both sea-states.

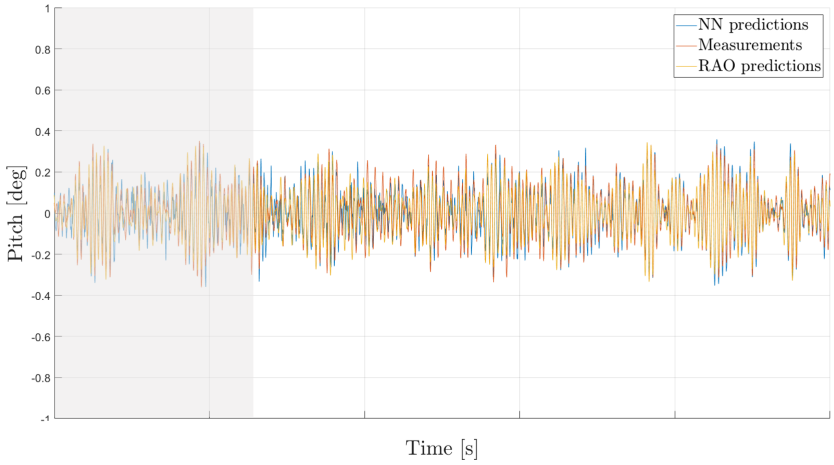


Figure 7.15: Modeltest sea-state 1, final network predictions

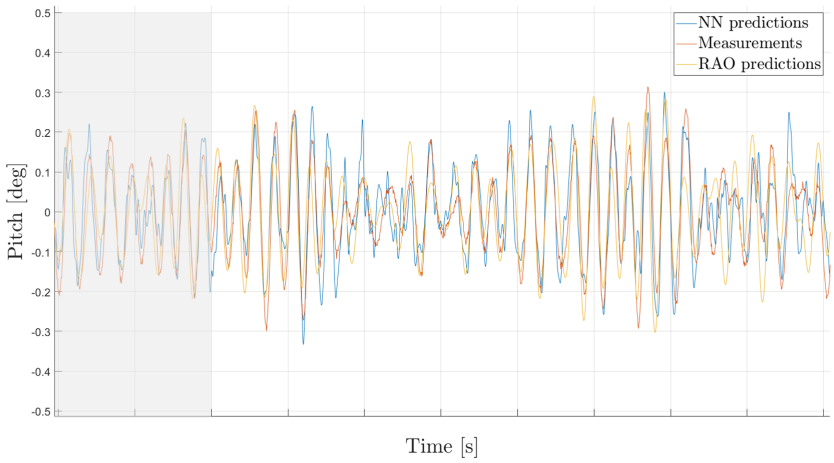


Figure 7.16: Modeltest sea-state 2, seconds 500-1000

7.1.4. Potential application

A neural network based model that is capable of acting as a real time transfer function between waves and vessel motion is valuable. Suppose one can predict future incoming waves. The conventional method, meaning the use of an RAO, is able to predict first order vessel motions in a range of sea states and vessel parameters that allow for accurate linear computations. A neural network based model, as shown in this chapter, has the potential to replace the conventional method and allow for accurate predictions in a larger range of sea states. The prediction of incoming waves has recently become a possibility. HMC makes use of FutureWaves, which is able to predict incoming waves in the near-future. FutureWaves is a real-time wave and vessel motion forecasting system that uses ocean-surface radar images [14].

7.2. Motion Hindcast ⇒ Future Motion

In this section, MRU data is used to train a neural network. The goal of this is to determine the potential of a NN based model in predicting future vessel motions based on motion measurements.

7.2.1. Data

HMC has equipped her vessels with several sensors continuously logging data such as accelerations, motions, stresses, draft and more as part of a big monitoring effort. The Fatigue Monitoring System (FMS) on board of the DCV Balder has been logging since 2009. The Motion Reference Units (MRUs) enables HMC to log the vessel motions in all six degrees of freedom. Only roll-motion is taken into account in this assessment into network potential. There is no reason why this type of network configuration would be more or less suitable for any particular degree of freedom. Therefore it is not of value to assess all DoFs seperately, when merely assessing the potential of this method. For the MRU model, data is selected from a 2010 transit. In the selection, the draft of the vessel is relatively constant (Figure 7.17). The signal of the roll motion is depicted in Figure 7.18 & 7.19.

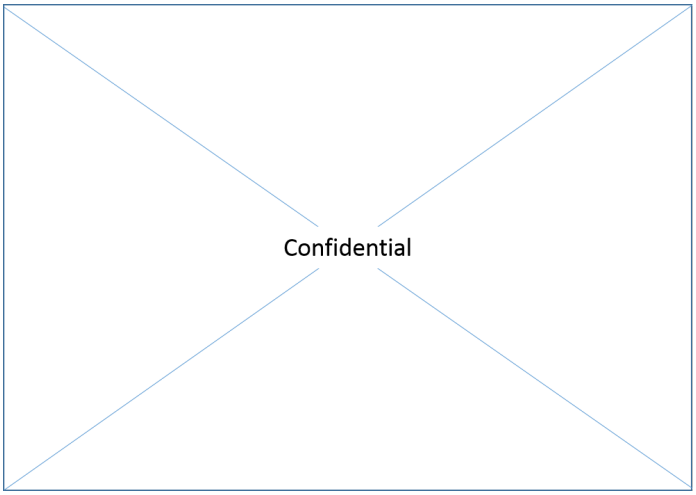


Figure 7.17: MRU dataset selection by draft

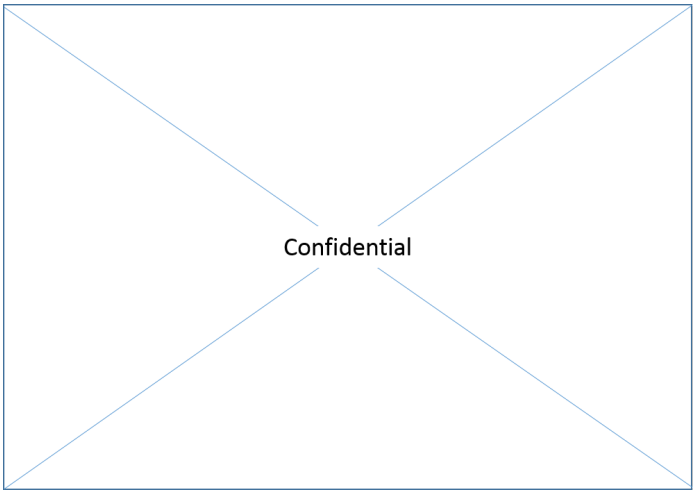


Figure 7.18: MRU dataset roll signal

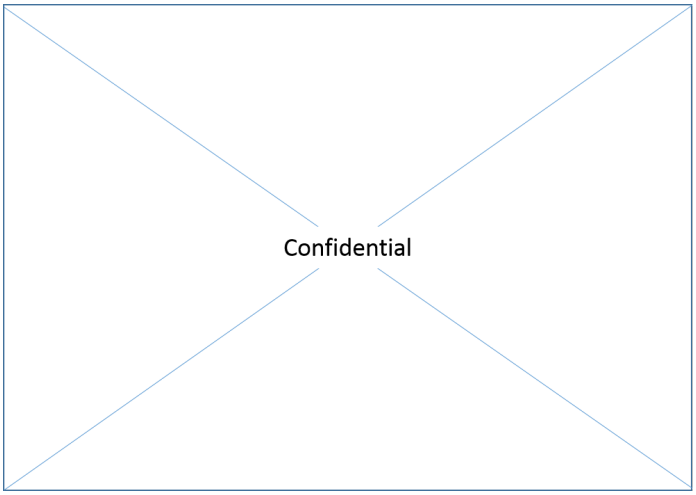


Figure 7.19: zoom-in roll signal

The dataset is not separated into training and test data. Because the dataset is this large, there is no need for a lot of epochs in the training process. This means that the network rarely gets fed the same data twice. Therefore there is no risk of overfitting. This also means that the neural network should be able to adapt itself to a different sea-state at the same rate that sea-states change in the real ocean environment.

MRU Filtering

The MRU signals for the vessel motion 25 Hz. The signal is filtered in Matlab using the smoothdata() function as can be seen in figure 7.20. This function outputs the running average of the signal.

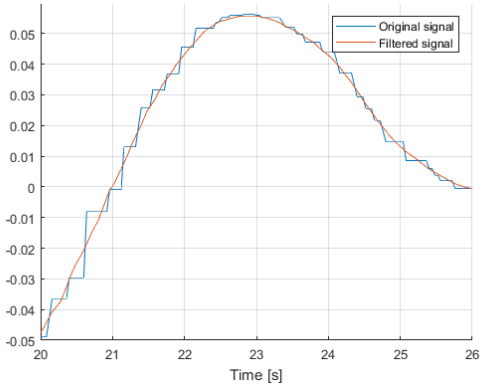


Figure 7.20: Filtering MRU signal

The 25Hz is a unnecessary high frequency for the MRU signals. This will cause the amount of input nodes to be higher which will increase training time and cause training difficulties. The signal frequency should be high enough to accurately capture the waves but any higher than the necessary accuracy will result in training complications. The sample rate was set on 3.125Hz as seen in figure 7.21. For readability, only four frequencies are shown in the figure although more have been considered.

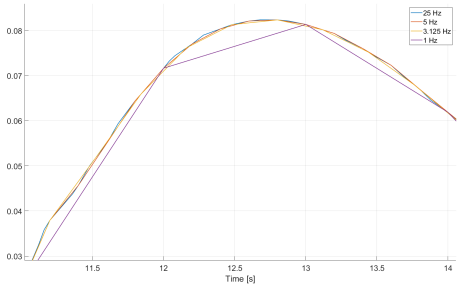


Figure 7.21: Different sample frequencies for MRU heave signal

7.2.2. Topology selection

Neural network training times for this model are long, as with the model test model. Therefore, the same strategy is applied in which topology variations are not made in batches, but after each training run. The topology variations are depicted in Table 7.4.

Variation	1	2	3	4	5	6	7	8
Input seconds (approx)	15	30	45	60	75	90	105	120
Input timesteps	45	90	135	180	225	270	315	360
LSTM modules	45	90	135	180	225	270	315	360
Optimizer	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam
Learning rate	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001

Table 7.4: MRU model, topology variations 1-8

During training, the loss curves seem to converge slowly, but none of the topology changes seem to significantly increase network performance. In Figure 7.22, the training curves are displayed and compared to the moving average of the variance of the roll motion. Figure 7.23, shows a simplified version in which all training curves are represented by their mean. The moving average variance of the roll motion is a measure for how difficult it is for the network to predict the sea-state. A strong correlation can be seen between the two curves and this indicates that the networks are not actually converging.

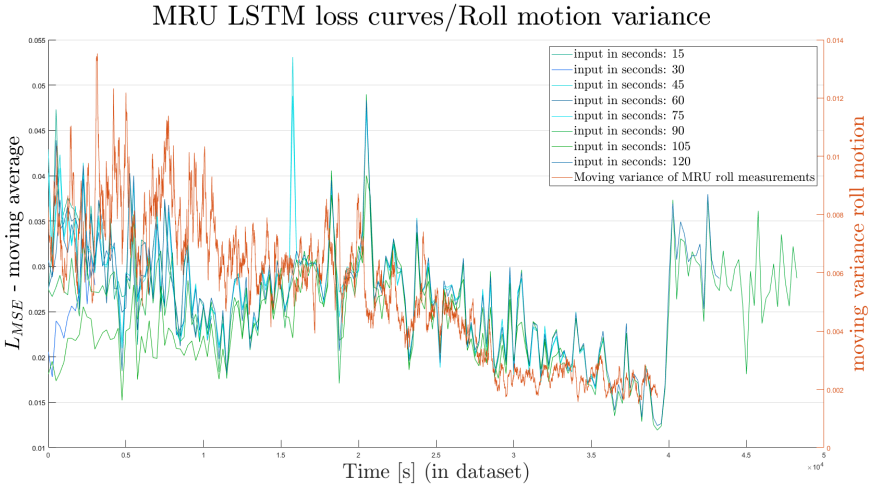


Figure 7.22: Training loss curves (left y-axis) versus roll motion variance (right y-axis)

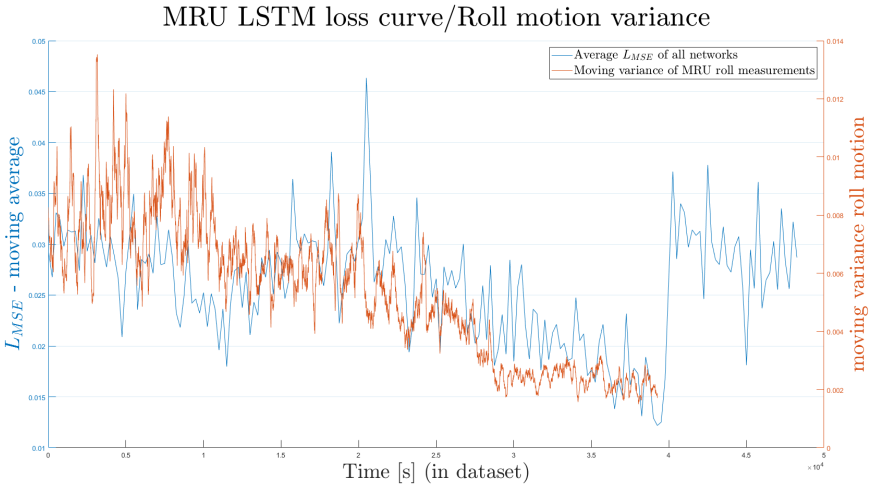


Figure 7.23: Mean of training loss curves (left y-axis) versus roll motion variance (right y-axis)

Network topology variations 1-8 follow the same strategy to quickly reduce their training loss; they repeat the last input value as their output value. This ensures a quick descent down the cost function. The networks all end up in a local minimum out of which the optimization algorithm is unable to escape. Further topology changes do not yield better results. Table 7.5 and 7.6 show all tested topology variations. These include MLPs, multi-layer LSTM networks and optimizer changes. All these variations performed worse than the first eight topologies and either ended up in the same local minimum, or diverged in an early stage.

Variation	9	10	11	12	13	14	15	16	17	18
Input seconds (approx)	60	60	60	60	60	60	60	60	60	60
Input timesteps	180	180	180	180	180	180	180	180	180	180
Layers	3	7	12	22	12	22	12	22	12	22
Layer size	500	500	500	500	500	500	500	500	500	500
Activation function	Linear	Linear	Linear	Linear	TanH	TanH	ReLU	ReLU	Sigmoid	Sigmoid
Optimizer	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam
Learning rate	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001

Table 7.5: MRU model, topology variations 9-18

Variation	19	20	21	22	23
Input seconds (approx)	60	60	60	60	60
Input timesteps	180	180	180	180	180
Layers	3	3	3	4	5
LSTM modules (per layer)	180	180	180	Linear	Linear
Optimizer	SGD	SGD	Adam	Adam	Adam
Learning rate	0.001	0.01	0.1	0.0001	0.0001

Table 7.6: MRU model, topology variation 19-23

8

Discussion

In this chapter, answers to the research questions from Chapter 2 are given based on the test results of the different neural networks.

Can a neural network based model be used as an alternative to the RAOs calculated by a diffraction based method?

The results from the artificial environment in Chapter 6 show that, in theory, all four proposed neural network architectures are capable of acting as an alternative to an RAO. As the 2D network provides the most information and gives accurate results in the artificial environment, it is the best option for tests with real world data.

A distinction is made between projects and transits. During projects, when the vessel sails at operational draft, the diffraction based calculations are accurate enough to provide reliable predictions for workability analysis, fatigue damage prediction, etc. During transits, the vessel sails at an inconvenient draft and nonlinearities and other effects make that the diffraction based calculations fail to represent reality.

The neural network based model when tested on project data is able to attain the same accuracy of the diffraction based calculations. This makes it a viable alternative for these cases. The spread around the $y(x) = x$ line in the SDA plot (Figure 6.45) is most probably due to uncertainties such as weather data, vessel manoeuvres, deck activity and non-linear hydrodynamic effects.

For the model that was tested on transit data, the results show that the vast majority of the predictions follow the right trend, however the SDA of some test examples show a significant under-prediction by the network. A part of the spread around the $y(x) = x$ line (Figure 6.45) can be ascribed to the same causes as in the project data tests: uncertainties such as weather data, vessel manoeuvres, deck activity and non-linear hydrodynamic effects. The main two differences in the circumstances compared to the project data are a shallow draft and forward speed. Extra non-linearities are hereby introduced. The forward speed also changes the encounter frequency of the incoming waves. As the network input (a 2D wave spectrum) is adjusted for heading but not for forward speed, the network will adapt to the mean forward speed (or a speed close to this). But the network is not capable of adjusting for changes in forward speed as it is not provided as an input.

A neural network based model shows promise as an alternative to a diffraction based calculation method. Training a neural network in this way does however take a large amount of data and using the model when encountering sea-states that are not in the training dataset may cause reduced accuracy.

Can a neural network based model be used as an alternative to an RAO in the time domain?

One of the researched potential time domain uses of a neural network is a model that acts as a real time tranfer function between waves and vessel motion. Two datasets from model tests at operational draft, containing surface height and pitch measurements are used to train and test the network. In the first dataset, a sea-state is present that invokes first order vessel motion responses. The sea-state from the second dataset invokes mainly second order responses.

A network topology with 2400 input values containing a 10 minute, 4Hz surface height signal and 240 LSTM modules and a single linearly activated node is chosen after a topology optimization process. Results are displayed in Appendix D. The results from both datasets show that the network is able to predict vessel motions with an acceptable accuracy.

Out of the modeltest results can be concluded that there is potential in a neural network based model acting as a real time transfer function. It can predict first order vessel motions with a similar accuracy than the diffraction based method. Unlike the conventional, RAO based, method it is not limited to linear computations as it showed it is able to predict vessel motions that are caused by second order wave drift forces.

To what extend is it possible to make predictions using a neural network based model using hindcast measurements?

The model that uses hindcast measurements to try and predict the future vessel motions shows the same result for all topology variations. The network gets stuck in a local minimum in which the network reproduces the input as a best guess of the future vessel motions. This is either because the optimization algorithm is unable to avoid this local minimum or because there is a lack of correlation between hindcast vessel motions and future vessel motions. The last implies that ocean waves are simply too random to make any meaningful prediction of the future.

9 Conclusions

Frequency domain

- All four proposed architectures (1D, 2D, 2D-SDA & Parametric) tested in the artificial environment show satisfactory results.
- The 2D architecture is tested on project data with the Balder at operational draft. A single hidden layer neural network with dropout and regularization was the best performing network after a topology selection process.
- The 2D architecture is tested on project data with the Balder at operational draft. The neural network predictions show similarly good results as the predictions based on the RAOs.
- The 2D architecture is tested on transit data with the Balder at transit draft. The neural network predictions show a closer match to the measurements than the predictions based on the idenified RAOs.

Time domain: Wave ⇒ Response

- An LSTM network is trained and tested on two different datasets from model tests. The first contains a sea-state that invokes first order pitch motions, the second contains a sea-state that invokes second order pitch motions.
- The results from the first dataset show that the neural network can predict first order pitch motions with similar accuracy as a diffraction based method.
- The results from the second dataset show that the neural network can predict second order (pitch) motions that a first order diffraction based model is not able to.

Time domain: Motion Hindcast ⇒ Future Motion

- Both LSTM and MLP networks are trained and tested on an MRU timeseries containing roll motions. The network input is the 20 minute hindcast, the network output is the prediction for the 60 seconds into the future.
- All tested topologies either diverge, or end up in a local minimum in which the network mimics the networks input. Attempts to predict the future motions based only on the past motions were therefore unsuccesful.

Findings regarding neural network

- Single hidden layer neural network outperform deep neural networks in many cases. Even though is forced to linearize the problem as it is limited in introducing non-linearities.
- Applying constraints to network parameters has proven to be a useful tool in improving network convergence.
- Initializing weights and biases at 0 significantly increases rate of convergence in the frequency domain architectures. In these networks, the large majority of weights and biases should be 0. The optimizer algorithm is not good at approaching 0 as it will keep overshooting. Initializing the parameter values at 0 solves this issue.
- The overfitting mitigation measures Dropout and Regularization increase the robustness of the training process.
- Topology selection by structurally varying network parameters is an effective method. This process relies heavily on the recognition of overfitting and optimizer issues.

10

Recommendations

Frequency Domain

- Make seperate networks per draft range using the final topologies from Sections 6.3 & 6.4. There will be a trade-off between the maximum network accuracy and the amount of training data when choosing a draft range. These networks can be made for any vessel.
- Implement alongside conventional method as neural networks remain very sensitive to errors in their input data and may not always be able to extrapolate past experience into sea-states and/or ship parameters that it has not encountered in the past.
- Create largest amount of input data possible to mitigate overfitting.
- Adjust transit model input to encounter frequency.

Time domain: Wave ⇒ Response

- Determine if it is possible to construct an accurate RAO from a network trained on model test data. It may be possible by feeding the network sine waves in a range of frequencies. The amplitude and phase difference between input and output can be used to construct an RAO.
- The current modeltest networks are trained in a few epochs. This cannot be done real time. Develop a network training process in which the network is trained multiple times on recent data, while also being trained real time.
- Test if the network works with input from FutureWaves, using the dominant wave heading.
- Develop a network that uses the entire heading spectrum that FutureWaves provides.
- If computational capability allows it, increase the amount of LSTM modules and input values (longer timeseries). This should increase network performance.
- Tune optimizer hyperparameters to further increase network performance.

Time domain: Motion Hindcast ⇒ Future Motion

- The prediction of future values of a timeseries using its hindcast values will cause the network to end up in a local minimum during training. This should not be pursued further.

- Using MRU measurements as an extra network input to potentially increase performance of the model test based network is an option.

Other neural network applications

- A lot of research is currently being done in the field of neural networks. It is becoming easier to implement because of open source libraries and increased computational capabilities. Machine learning is not, and will never be the solution to everything. It should be seen as a new tool available to engineers. An example for a potential applications at HMC is the coupling of FEM and an ANN to predict safe sea-states [29].

A

Vector Fitting

A.0.1. Vector Fitting

Earlier research within Heerema [24] has included an effort to reduce the amount of elements needed to accurately describe hydrodynamic properties. This is done by means of the 'vector fitting' method. This method may also provide an alternative for the wave and response spectra and thus provide a ways to decrease the amount of necessary input and output nodes.

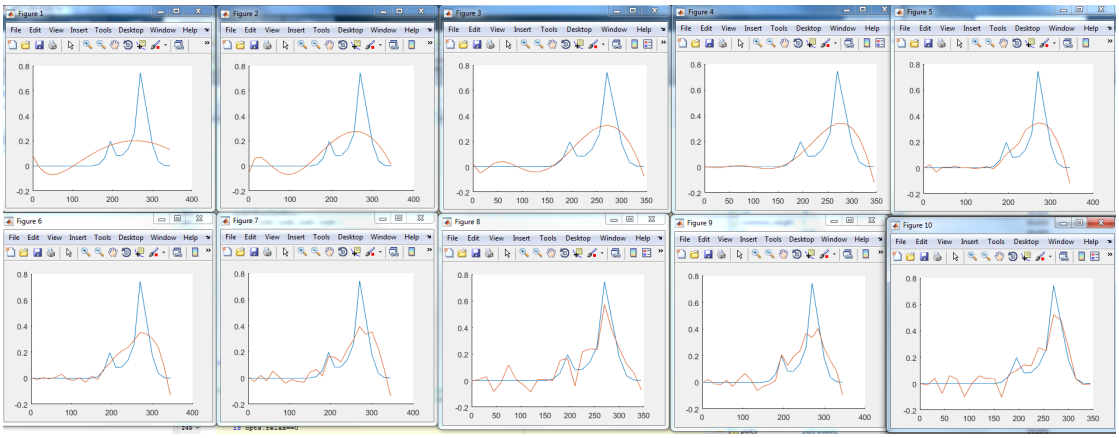


Figure A.1: Vector fitting trials

The vector fitting method turned out not to be applicable in this case. The main reasons for this are the high amount of zeroes and the low resolution. The zeroes form a problem because the vector fitting method produces a form of polynomial which in general can not accurately describe flat lines. And the low resolution is a problem because the surface of the wave spectrum is already described with too few values to produce a smooth line. The method works best when one wants to go from hundreds of values to dozens of values. Not when one wants to go from dozens to only a few. Simply because the complexity of the wave spectrum needs at least as many parameters as the current method used.

B

Universal approximation theorem

The universal approximation theorem states that a neural network with a single hidden layer that contains a finite number of nodes, can approximate any continuous function on a bounded and closed subset of Euclidean space. This does contain some restraints on the activation function.

In 1989, this was proven for the sigmoid activation function by George Cybenko [3]. A visual representation of the proof by Cybenko is depicted in Figure B.1. It shows how a neural network with two hidden nodes and a sigmoidal activation function is capable of representing a step function. Any function within the bounds of the universal approximation theorem can be approximated by a finite number of step functions. From this follows that any continuous function on a bounded and closed subset of Euclidean space can be approximated by a neural network with a single hidden layer with a finite number of nodes and a sigmoidal activation function.

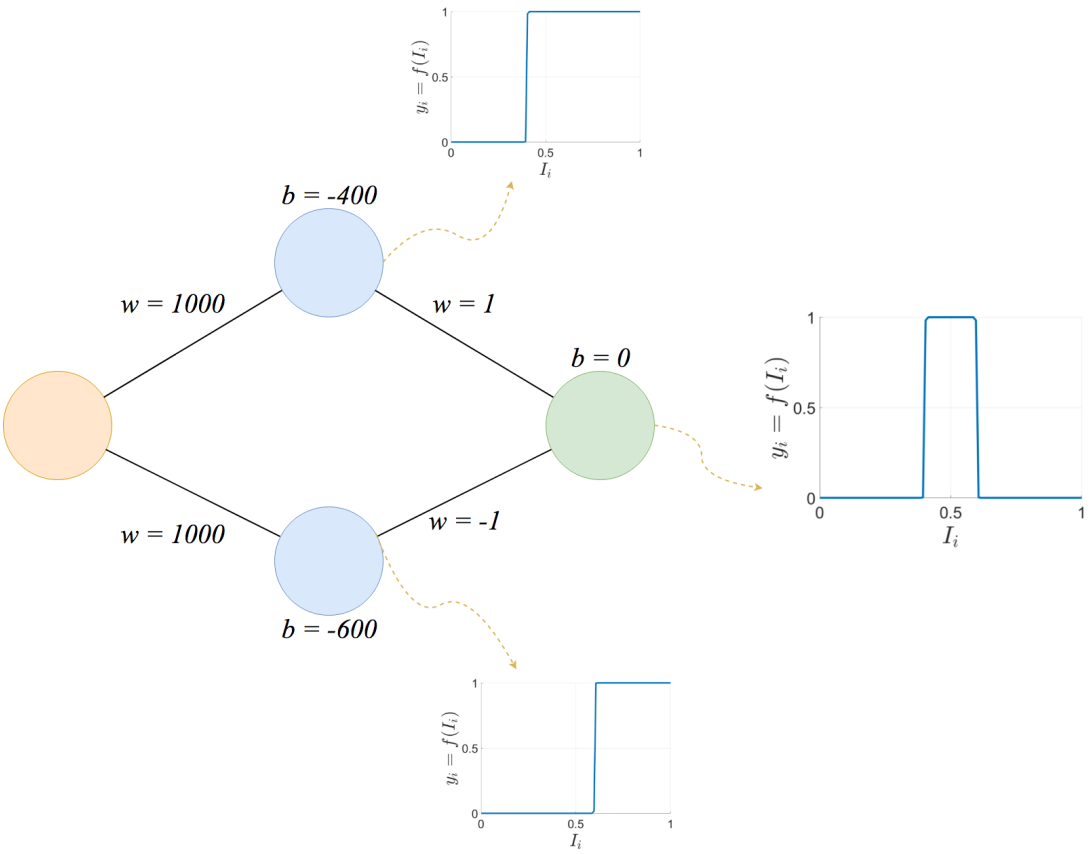


Figure B.1: Stepfunction by a single hidden layer neural network using $f(I_i) = \frac{1}{1+e^{-I_i}}$

Later, the proof was delivered that single hidden layer networks are universal approximators as long as the activation function is continuous, bounded and non-constant [9].

Having established that neural networks are universal approximators does not give any guarantee that a neural network can give accurate predictions. It is guaranteed under the condition that training data is representative for the test data, enough training data is provided, the training process is not bound by computational cost and that the neural network is not bound to a certain size.

C

MLP for timeseries

Recurrent neural networks such as LSTM networks are dedicated for time series prediction. Multilayer perceptrons are not. Figure C.1 includes a visualization of how a timeseries is 'fed' to an MLP network.

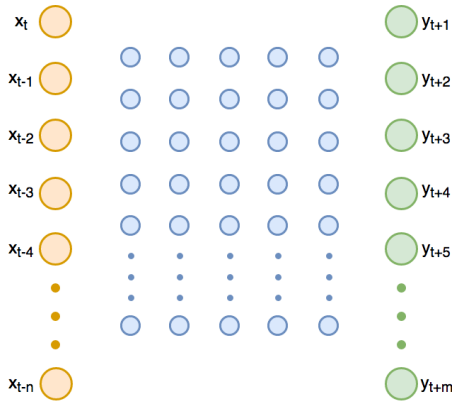


Figure C.1: MLP timeseries

D

Modeltest predictions

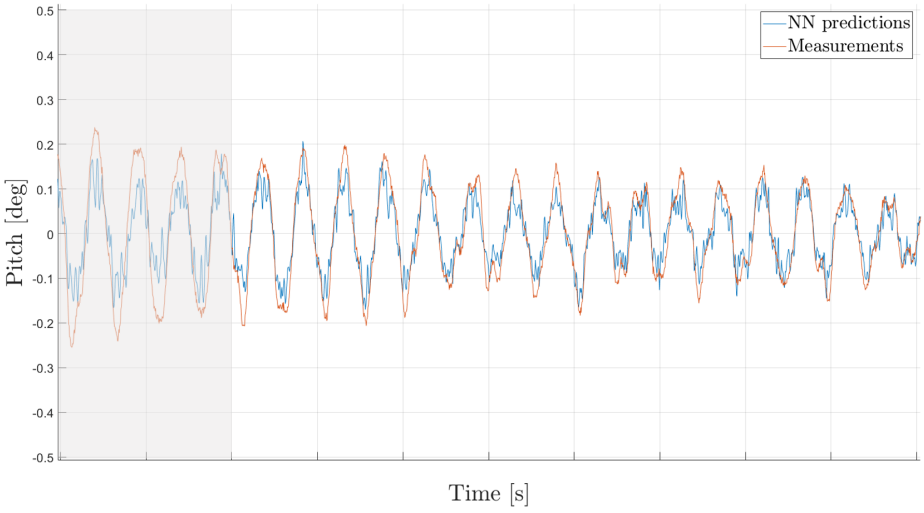
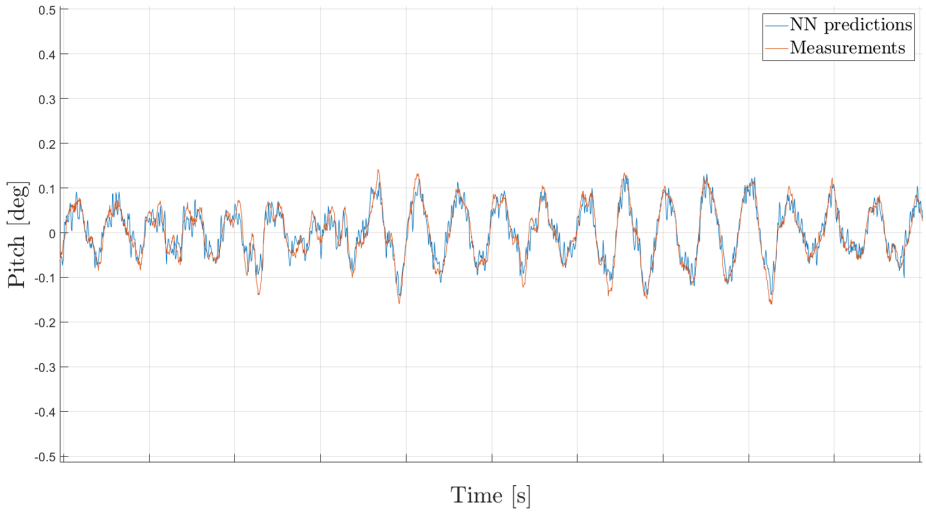
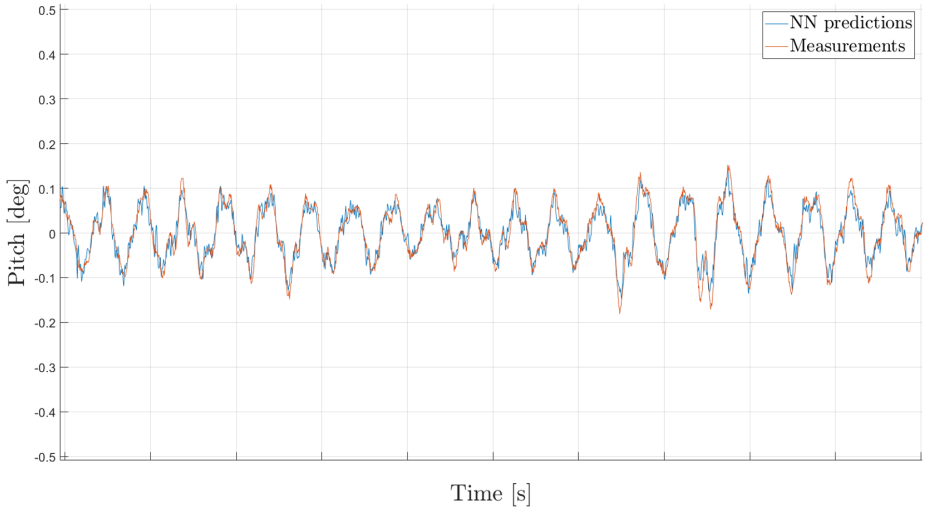


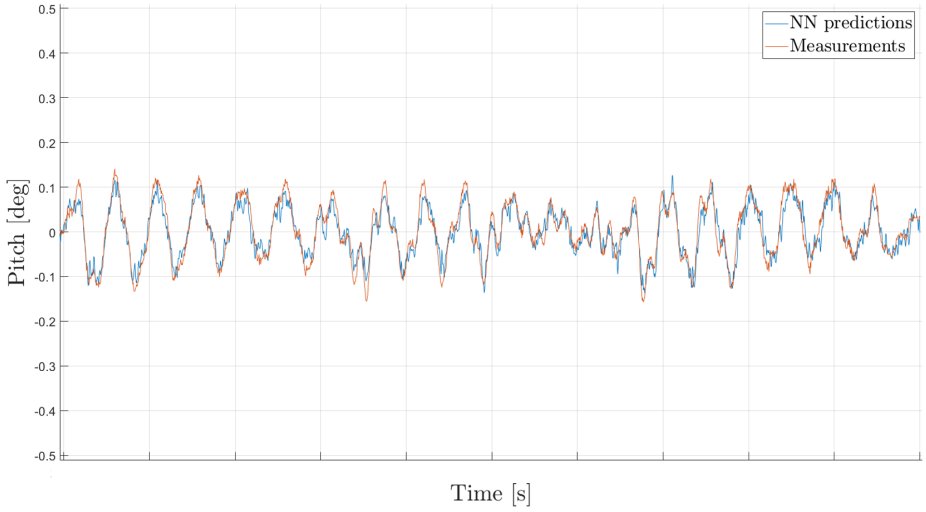
Figure D.1: Modeltest sea-state 2, neural network predictions



(a)

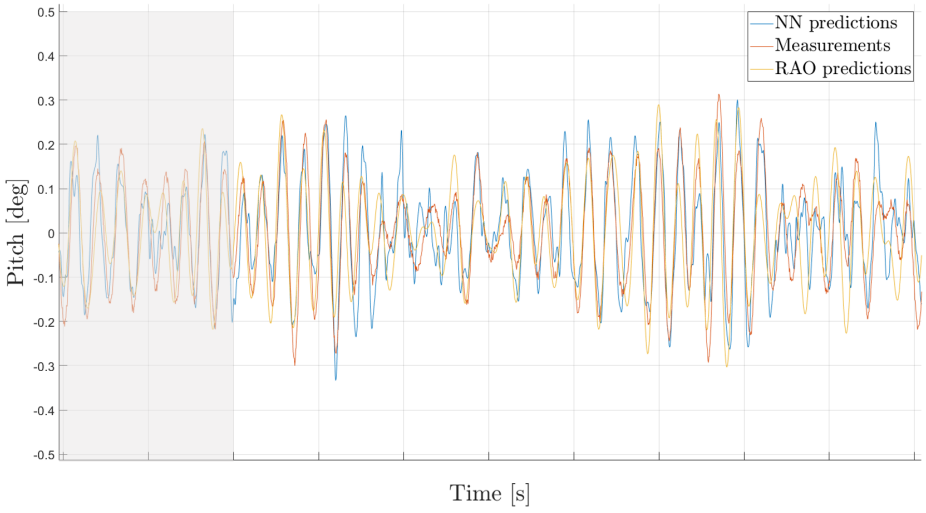


(b)

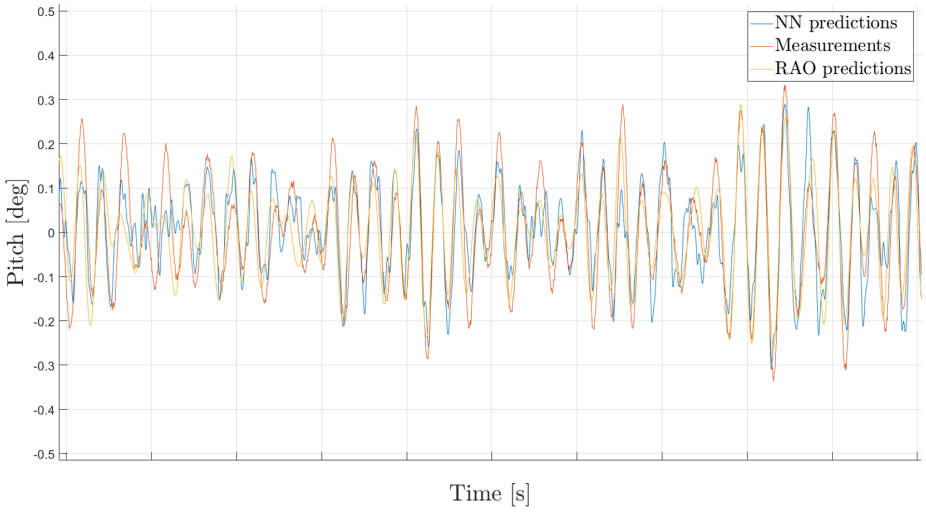


(c)

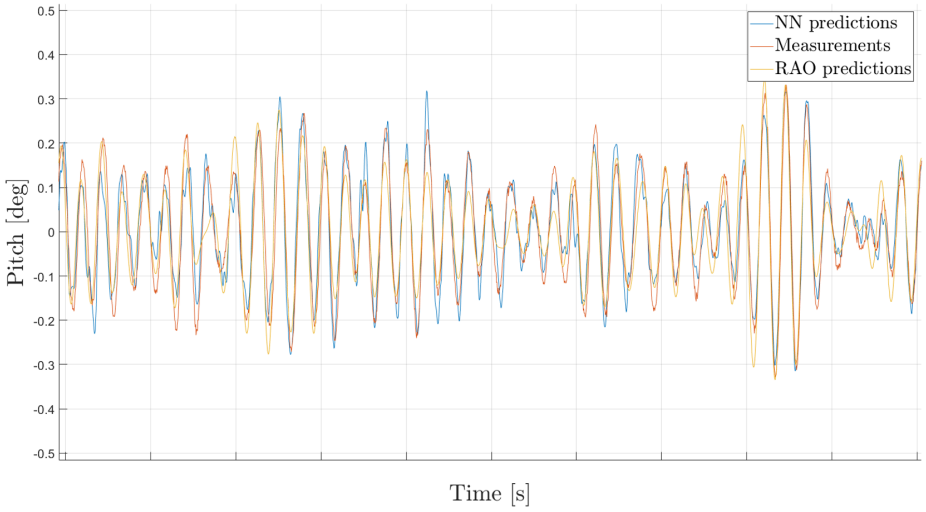
Figure D.2: Modeltest sea-state 2, neural network predictions



(a)



(b)



(c)

Figure D.3: Modeltest sea-state 1, neural network and diffraction predictions

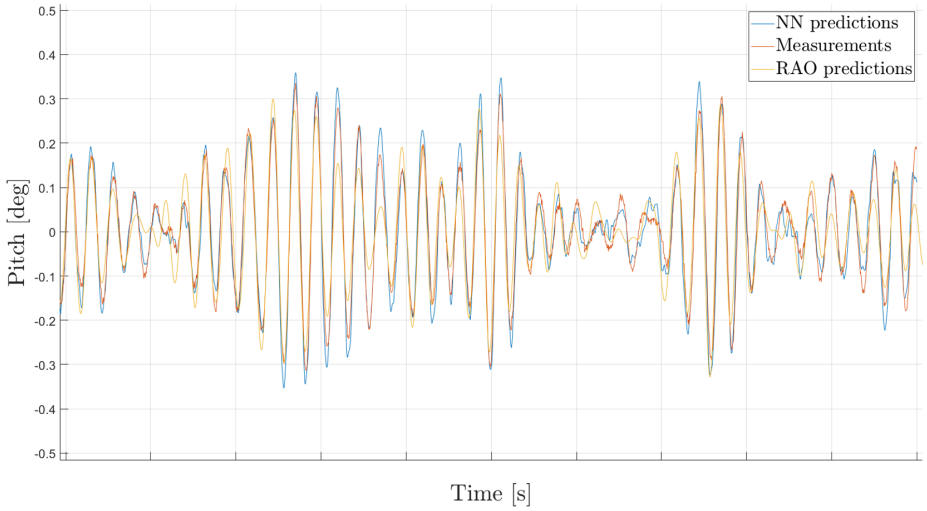


Figure D.4: Modeltest sea-state 1, neural network and diffraction predictions

Bibliography

[1] Keras: The python deep learning library. URL `https://keras.io/`.

[2] M.A. Cauchy. *M´ethode g´en´erale pour la r´esolution des syst'emes d'´equations simultan´ees*.

[3] G. Cybenko. Approximation by superpositions of a sigmoidal function. 2:303–314.

[4] M.C. Deo. Artificial neural networks in coastal and ocean engineering. 39(4):589–596.

[5] A.A. Elshafey, M.R. Haddara, and H. Marzouk. Estimation of excitation and reaction forces for offshore structures by neural networks. 1(1):1–15.

[6] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[7] HMC. *Fatigue Tool - Balder tower fatigue during transit*.

[8] S. Hochreiter and J. Schmidhuber. Long short-term memory. 9(8):1735–1780.

[9] K. Hornik. Approximation capabilities of multilayer feedforward networks. 4:251–257.

[10] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators.

[11] M. Khashei and M. Bijari. An artificial neural network (p,d,q) model for timeseries forecasting. pages 479–489.

[12] Y. Kim, H. Kim, and I. Ahn. A study on the fatigue damage model for gaussian wideband process of two peaks by an artificial neural network. 111:310–322.

[13] D.P. Kingma and J. Lei Ba. Adam: A method for stochastic optimization. In *ICRL*.

[14] J. G. Kusters, K. L. Cockrell, B. S. H. Connell, J. P. Rudzinsky, and V. J. Vinciullo. Futurewaves: a real-time ship motion forecasting system emplouing adcanced wave-sensing radar. In *OCEANS 2016 MTS/IEEE Monterey*.

[15] C.H. Lee and J.N. Newman. *Computation of wave effects using the panel method*. WIT Press.

[16] W.S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. 5(4):115–133.

[17] S. Namekar and M.C. Deo. Application of artificial neural network model in estimation of wave spectra.

[18] S.J. Nowlan and G.E. Hinton. Simplifying neural networks by soft weight-sharing. pages 473–493.

[19] G. Orr, N. Schraudolph, and F. Cummins. Neural networks.

[20] M.J.L. Orr. Introduction to radial basis function networks.

[21] I. Rentoulis. Improvement of fatigue damage prediction for the balder j-lay tower during transits.

[22] H.P. Riedel and A.P. Byrne. Random breaking waves - horizontal seabed. In editor, editor, *Coastal Engineering Proceedings*, pages 903–908.

[23] S. Ruder. An overview of gradient descent optimization algorithms.

[24] D. Skandali. Identification of response amplitude operators for ships based on full scale measurements.

[25] N. et al Srivastava. Dropout: A simple way to prevent neural networks from overfitting. (15):1929–1958.

[26] Tensorflow. Api documentation. URL `https://www.tensorflow.org/api_docs/`.

[27] H.L. Tolman, V.M. Krasnopolsky, and D.V. Chalikov. Neural network approximations for nonlinear interactions in wind wave spectra: direct mapping for wind seas in deep water. 8:253–278.

[28] WAMIT. *WAMIT user manual*.

[29] S.F. Yasseri, H. Bahai, H. Bazargab, and A. Aminzadeh. Prediction of safe sea-state using finite element method and artificial neural networks.

[30] J.H Yi, J.S. Park, and K.S. Lee. Long-term strain measurement on a jacket-type offshore structure and neural networks based prediction model.