A Convolutional Neural Network Developed to Predict Speed Using Operational Data

de Geus-Moussault, S.R.A.; Buis, Mark; Koelman, H.J.

Publication date
2021

Document Version
Final published version

Published in
Proceedings of the 20th International Conference on Computer and IT Applications in the Maritime Industries, COMPIT'21

Citation (APA)
de Geus-Moussault, S. R. A., Buis, M., & Koelman, H. J. (2021). A Convolutional Neural Network Developed to Predict Speed Using Operational Data. In V. Bertram (Ed.), *Proceedings of the 20th International Conference on Computer and IT Applications in the Maritime Industries, COMPIT'21* (pp. 246-264). Technische Universität Hamburg-Harburg.

20th International Conference on

Computer and IT Applications in the Maritime Industries

COMPIT'21

Mülheim, 9-10 August 2021

Edited by Volker Bertram

A Convolutional Neural Network Developed to Predict Speed Using Operational Data

Sietske R.A. Moussault, Delft University of Technology, Delft/The Netherlands, s.r.a.moussault@tudelft.nl,
Mark Buis, Vrije Universiteit Amsterdam, Amsterdam/The Netherlands, m3.buis@student.vu.nl
Herbert J. Koelman, NHL Stenden University of Applied Sciences, Leeuwarden/The Netherlands, Herbert.Koelman@nhlstenden.com

Abstract

In the shipping industry power prediction methods are commonly used. An option is to predict the power based with a theoretical analysis. However, with a purely theoretical approach it is not possible to evaluate all operating conditions. The second, simulation methods, are able to describe all the necessary quantities in detail. Nonetheless, simulation requires relatively high computational power. Thus, the current power prediction methods used in the shipping industry are insufficiently all-encompassing or accessible. Therefore, a machine learning approach is developed to calculate the ships speed over ground using neural network and convolutional neural network techniques. For training and validation of the model operational data from a fall-pipe vessel is used. The developed method combined with ship motion could result in an optimal power usage, and thus leads to reduced fuel consumption and emissions. The method could also be used for optimised routing. Although in this case study applied to one single vessel, the developed model is generally applicable, providing ship management companies the possibility to train the model with operational data from their fleet, therewith, offering the possibility of reduced fuel consumption and thus emissions on a global level.

1. Introduction

In the shipping industry power prediction methods are commonly used. Current speed prediction methods are limited and/or expensive. A machine learning approach can improve the accuracy and decrease the cost of speed and resistance predictions, *Brandsæter and Vanem (2018)*, *Liang et al. (2019), Petersen et al. (2012), Pedersen and Larsen (2009)*. No such speed prediction model is available tailored to the shipping industry as a whole. In order to predict the ships speed and resistance, this research presents an improved power prediction method, based on operational data. This method should be able to:

- accurately predict speed and resistance of a vessel,
- for different operational conditions,
- that is easy to use and widely accessible (on board) (low computational power).

1.1 Literature Review

Based on the requirements, the need arises for a method that predicts speed and resistance of a vessel for different operational conditions that is widely accessible. Thus, a literature review is presented about, the current speed and resistance calculation methods in shipping, their mutual benefits and shortcomings and other relevant machine learning applications in the field of marine engineering.

There are several methods presented in literature and currently applied in industry. *Petersen et al. (2012)* suggests dividing the different methods in four separate groups, namely:

1. Standardised traditional methods relying mostly on describing the hull using typical parameters, *Holtrop and Mennen (1982), Holtrop (1984)*
2. Direct model testing in towing tanks, *Chuang and Steen (2011)*
3. Methods based on computational fluid dynamics, *Sadat-Hosseini et al. (2013), Ozdemir and Barlas (2017)*

4. Regression methods that use sensor measurement data, *Brandsæter and Vanem (2018), Petersen et al. (2012)*

These methods differ in their dependence on either theoretical base using physical laws, or empirical and data driven insights that base their descriptions on statistics of historical data. In the following sections the methods are described and evaluated in more detail.

1.1.1 Traditional Methods

A widely used approximate power prediction method was introduced by *Holtrop and Mennen (1982)* and later improved in *Holtrop (1984)*. *Nikolopoulos and Boulougouris (2019)* states that Holtrop-Mennen is considered to be one of the most accurate and computational efficient methods for estimation and speed prediction. The method uses equations that describe various resistance components that add up to a total resistance from which, together with the propulsion power the engine and screws, provide the speed can predicted.

The equations presented in *Holtrop (1984)*, are based on a regression analysis of the data of 334 ship models. The ship hulls were parameterized in a range of dimension ratios. With these equations, using the parameters of the hull ratios as input, resistance and speed can be predicted for a vessel.

Another method that is used to predict resistance and speed of a vessel is presented in *Hollenbach et al. (1999)*. *Matulja and Dejhalla (2007)* states that this method is at least as precise as other traditional methods. The Hollenbach method is based on the data of 433 ships and these ships are from the era between 1980 and 1995. These more recent hulls are more similar to the hulls designed nowadays compared to methods based on older hull shapes. *Feijo and De Oliveira (2020)* conclude that the mean prediction scenario of Hollenbach and the prediction scenario of Holtrop-Mennen are very similar, and no preference is given. The method proposed in *Hollenbach et al. (1999)* does give a maximum and a minimum which can be useful for different operating conditions.

Traditional methods are reasonably accurate at predicting speed and resistance. *Matulja and Dejhalla (2007), Nikolopoulos and Boulougouris (2019), Grabowska and Szczuko (2015)* all agree that Holtrop-Mennen gives a good estimate of the ship's speed and resistance. *Matulja and Dejhalla (2007), Grabowska and Szczuko (2015)* conclude that Hollenbach is also able to make accurate predictions. Another benefit of both methods is, the calculations are based on dimensions of the hull, which are determined in the earlier design phase. Therefore, these traditional methods can be applied early in the design process.

However, Holtrop-Mennen as well as Hollenbach can only accurately predict speed and resistance, if the dimensions of the hull are in between the boundaries of the method. Therefore, not for all ships predictions can be made. Also as explained by *Bertram (2012)*, these methods are outdated and underestimate the resistance of modern hulls. Furthermore, the traditional methods do not take environmental factors into account. According to *Mao et al. (2016)* environmental factors can increase ship resistance by more than 50-100%.

1.1.2 Towing Tank Direct Model Testing

These methods are based on the design of downsized ship models that are tested in towing tanks. In the research presented in *Chuang and Steen (2011)*, a scale model of a 8000 DWT tanker is used and is towed through a large towing tank (L/W/D= 260/10/5). In the research presented in *Chuang and Steen (2011)* waves were created in the towing tank to measure the influence these waves had on the speed of the ship. The force needed to tow the vessel model through the towing tank is recorded. From these measurements the resistance and speed of the real size vessel are be predicted.

Direct model tests are performed in a controlled environment, all influential factors can be tested separately, *Chuang and Steen (2011)*. However, for direct model testing a towing tank is required. Tow-

ing tanks are expensive and testing approximately costs around 10000 dollars per test, *Barczak (2020)*. Furthermore, in model test the waves brake different from the full-scale ship, resulting in scale effects, leading to a slight accuracy problem *Bertram (2014)*.

1.1.3 Computational Fluid Dynamics (CFD) Methods

Prediction methods based on CFD simulate the flow around the hull of the ship. Most CFD methods roughly take the following steps:

- Preprocessing:
  - Forming the geometry and physical bounds of the problem, to extract fluid domain and volume
  - Discretization of the fluid domain and volume
  - Defining the physical modelling, e.g. conservation laws and equations of fluid motion
  - Definition of the boundary conditions. E.g. the fluid behaviour, initial system conditions, reciprocally interaction between fluid and objects and properties of bounding surfaces
- Simulation: iterative solving the equation for each discretization step
- Processing the outcome, visually as well as analytically

Sadat-Hosseini et al. (2013) verified this technique for added resistance for the KVLCC2 ship model in short and long waves. *Ozdemir and Barlas (2017)* also tested added resistance for the KVLCC2 that was created by waves. The KVLCC2 is a ship model that is based on the design of the KVLCC model. The KVLCC model was presented in *Van et al. (2000)* to verify CFD models with the measuring of flow experimental and comparing it to the flow calculated by the CFD algorithm. This made the verification of CFD models possible and thereby confirming the model. The KVLCC2 model is presented in *Hino (2005)* and is a slight differentiation on the hull presented in *Van et al. (2000). Ozdemir and Barlas (2017)* concludes that the CFD model can calculate the total resistance with a margin of 4.1 % of the experimental value. Another conclusion of *Ozdemir and Barlas (2017)* is that since the hull profile is very complex, the motion, and thus resistance, behaves highly nonlinearly. This is a useful insight for the choice of models regarding the developed method.

With CFD models every hull type can be tested and the outcome is able to illustrate which part of the hull generates most resistance, *Sadat-Hosseini et al. (2013)*. Earlier CFD methods were based on elementary flow models, resulting in less precise results, compared to enhanced CFD models *Bertram (2014)*. However, CFD calculations are expensive and time consuming, *Cui et al. (2012), Gatin (2019).*

1.1.4 Regression Methods

Regression methods rely on the statistical interference of historical data, *Brandsæter and Vanem (2018). In Petersen et al. (2012)* two different techniques are used to predict the speed of a vessel, Artificial Neural Networks (ANN) and Gaussian Processes (GP). It is concluded that the ANN works better than the GP. *Petersen et al. (2012)* also concludes that a direct comparison with similar work, *Pedersen and Larsen (2009)*, is hard, as different data is used.

Mao et al. (2016) propose and compare a linear model, an autocorrelation method and an autoregressive model. It is concluded that linear regression, without environmental factors, gives poor results. When environmental factors are included, an autoregressive process reduces the prediction errors significantly. Also, *Mao et al. (2016)* stresses that travelling direction has an influence on the speed-resistance relation. Therefore, *Mao et al. (2016)* suggests to only include travelling direction as input value, when there is a clear leading wind and wave direction.

Brandsæter and Vanem (2018) tests various models on their ability to accurately predict ship speed. Based on data regarding ship motions, wind speed relative to the ship and draft. Other external factors are, at least partially, causal to the ship motion and thus accounted for in the data. Hence, this data is not used as input. *Brandsæter and Vanem (2018)* compare a linear regression model (LRM), a general additive model (GAM) and a projection pursuit model (PPR) with a baseline model. For these methods the shaft thrust, wind and sea conditions are used as explanatory variables. *Brandsæter and Vanem (2018)* concludes that in many cases, especially in calm water conditions, the baseline model performs well in terms of prediction accuracy. Furthermore, according to *Brandsæter and Vanem (2018)* more advanced models do not perform better than the baseline model, when only thrust is taken as explanatory variable. However, when environmental factors are taken as explanatory variables the accuracy increased dramatically. This is a valuable insight for this research, as this data is available and thus it is likely to increase the accuracy of the developed model.

Liang et al. (2019) proposes multiple regression models to predict ship speed, based on automatic identification system (AIS) sensor data and known weather data. The weather data is derived from the AIS data, as the position of the ship is known. *Liang et al. (2019)* concludes that linear regression methods do not predict accurately, due to the highly non-linear tendency of speed over ground. Other methods proposed to predict vessel speed based on data, are decision tree regressor and multiple ensemble methods. The ensemble methods used are a random forest regressor, an extra tree regressor and a gradient boost regressor. *Liang et al. (2019)* concludes that the extra tree regressor and random forest regressor are the best for this task.

Kim et al. (2020) use a support vector regression and harvests good results with this method. *Li et al. (2014)* uses varying machine learning techniques and finds that in some situations they perform just as good as RANS solvers, which are a type of CFD method.

For the use and development of regression methods no knowledge of the physical phenomena is required. The underlying mathematical equations do not have to be known to make accurate predictions. Furthermore, all influential phenomena can be taken into account and when the method is implemented it calculates outcomes very fast. However, data is needed the develop the model and regression algorithms rely heavily on the quantity of data. Also, this data needs to be pre-processed, which often is a time-consuming task. Lastly, the training of some methods, for instance convolutional neural networks, can be quite computational heavy.

1.1.5 Other Machine Learning Applications

Over recent years numerous applications of machine learning algorithms have been researched, that are not in the direct field of resistance-speed prediction. This research can provide useful insights, about what approaches are likely to work and which approaches are more likely to fail.

Abramowski (2013) develops a neural network to successfully predict the effective power for a ship. The research presented in *Abramowski (2008)* shows that neural networks are also able to predict ship manoeuvrability. *Abramowski and Zmuda (2008)* develops a method of presenting a hull in parameters using neural networks.

Liang et al. (2019) compare multiple machine learning and traditional algorithms to predict the vessel propulsion power. Concluding, that the machine learning techniques performed significantly better on ships were a lot of data was available, also they observed big drop in performance when little data was available.

1.2 Problem Definition

In the shipping industry power prediction methods are commonly used. As elaborated upon, the statistical design formulas are for a narrow operational range, across many ships, *Holtrop and Mennen (1982), Holtrop (1984), Bertram (2012)*. While simulations are for one case, but initial condition only.

CFD will be more accurate and far more expensive, *Cuiet al. (2012), Gatin (2019), Bertram (2014)*. Concluding, the current speed prediction methods are limited or expensive.

A machine learning approach can improve the accuracy and decrease the cost of speed and resistance predictions, *Brandsæter and Vanem (2018), Liang et al. (2019), Petersen et al. (2012), Pedersen and Larsen (2009)*. No such speed prediction model is available tailored to shipping. In principle, this is a statistical method, based on data from one ship, across many operational conditions, for the as-is, as-built condition. When the quantity of data is large enough with sufficient variation (in speed, draft, trim), results should be equally accurate as simulations at much lower expense.

2. Method

2.1 Scope

This research covers the development of a machine learning algorithm based on the data of one fall pipe ship. The data set consist of approximately 15000 data points. The outcome model of this research is in principle applicable to data from this ship specifically. There is a possibility that the model is also applicable to ships with comparable designs, however this will not be investigated in this research. By nature of machine learning techniques, the accuracy of the predictions of situations that are not in the original data cannot be determined. However, this does not imply that predictions of these situations are not accurate. Rather, it follows that these situations are not within the scope.

2.2 Algorithm Description

2.2.1 Linear Regression

Linear regression models have been used in different parts of science, from psychology to economics. Linear regression is used to model linear relationships between a response variable and explanatory variable(s). The model can be described using the following formula:

$$\hat{y}_i = a_0 + a_1 x_{1i} + a_2 x_{2i} + \cdots + a_m x_{mi} \quad (1)$$

$\hat{y}_i$ the predicted value, x_i the value that is known and used to predict $\hat{y}_i$ and a_i the linear coefficient that states how much a change in x_i influences $\hat{y}_i$.

2.2.2 Artificial Neural Network

The concept of neural networks was first described in *McCulloch and Pitts (1943)*. In *Hebb (1949)* it was proposed that these networks could be trained using a mechanism known as neuroplasticity. The ability of neural networks to be trained by recognition. Current day neural networks are based around the idea of the perceptron, presented first by *Rosenblatt (1958)*. *Aggarwal (2018)* states that perceptrons are able to classify linear data points in the same way as least squares methods. With the difference that perceptrons guarantee to find complete separation, on the condition that the underlying data is linearly separable. Fig.1 shows the conceptual model of a perceptron. X_{ij} with $i\ \varepsilon\ 1,2,\dots,n$ and $j\ \varepsilon\ 1,2,\dots,m$ are the input signals, w_{ij} with $i\ \varepsilon\ 1,2,\dots,n$ and $j\ \varepsilon\ 1,2,\dots,m$ the weights which the inputs are multiplied by before summing in the neuron, φ the activation function, and o_j the output. n represents the number of input signals and *m* the number of alike networks.

Outcome values of a perceptron are calculated using a two-step process. First, the product of weights and input is taken as described in Eq.(3). After this the sum is taken of all these inputs and the output of the sum is fed into the activation function. This transforms the input to output $\hat{y}$ as can be seen in Eq.(2). In the perceptron described by *Rosenblatt (1958)* this activation φ was the step function described in Eq.(4). *Rosenblatt (1958)* stated that this design of the perceptron imitated the working of neurons in animal brains. From Eq.(4), it is evident that the neuron activates, if and only if, the

product sum of the inputs and weights is bigger or equal than the critical, qj in 1, in this case 0, value.

$$z = \sum_{i=1}^{n} x_i \, w_i \tag{2}$$

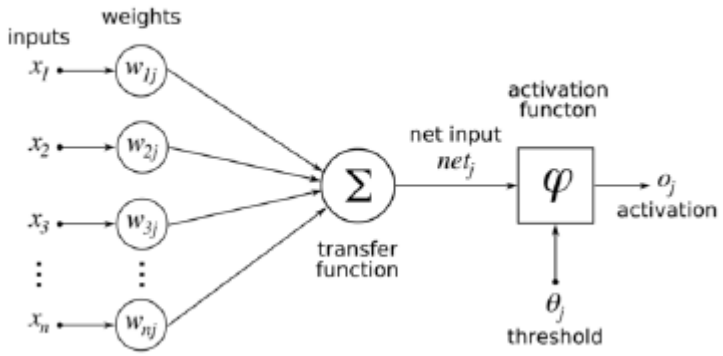

Fig.1: Perceptron model, https://nl.wikipedia.org/wiki/Perceptron

$$\hat{y} = \varphi(z) \tag{3}$$

$$\varphi(z) = \begin{cases} 0 \; if \; z < 0 \\ 1 \; if \; z \geq 0 \end{cases} \tag{4}$$

While, as illustrated by *Aggarwal (2018)*, the perceptron model guarantees linear separable decision boundaries. Due to its architecture the perceptron model is not able to model nonlinear data, *Minsky and Papert (2017)*. This can be solved when coupling multiple perceptrons together. This way, activation functions can be used, and thus nonlinear boundaries and relations can be mapped.

To map even more complicated relations slightly more complicated models are needed. These models have an input layer, one or more fully connected hidden layers and an output layer. A schematic overview of this model is given in Fig.2. The model presented in Fig.2 can be extended unlimited, keeping the accuracy computation time ratio in eyesight, by varying the number of hidden layers and the size of these hidden layers. The last model that is described is the beginning of the neural networks that are used nowadays. Here the main components are described.

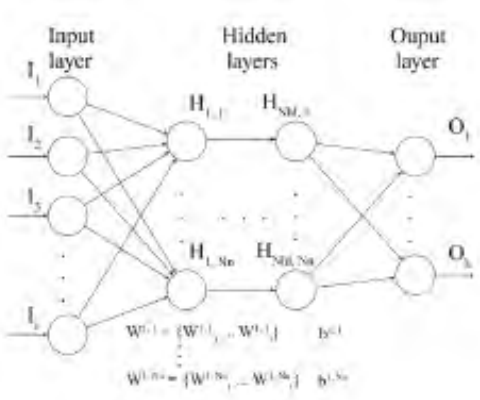

Fig.2: General neural network, https://www.researchgate.net/figure/General-neural-network architecture-used-in-this-study-33-43-44-The-input-layer fig1_341200069

Artificial neural networks consist of three components. Neurons, weights and forward propagation. Neurons are the intersection of the network. They have input(s) and a single output that can be send to multiple other neurons. In the neuron the weighted sum of the inputs is taken as described in Eq.(2) and this sum is either directly send as output or put in into an activation function and passed on as output. Weights connect neurons to each other and determine how 'heavy' the output of the sending

neuron should weigh as input of next neuron. At the initiation of the network the weights can either be set at a fixed value equal for all weights, or assigned randomly. Forward propagation (FP) is the process that calculates the output of a neural network. The mathematical process follows the same principles as described in Eqs.(2) and(3).

As shown in the Perceptron model, Fig.1, neurons in layers can apply activation functions to their input to generate their output. Where the Perceptron model use the step function, neural networks are capable of using a whole lot of other activation functions. Activation function can vary between different layers but in the vast majority all neurons in the same layer apply the same activation function. The choice of activation functions is mostly dependent on the type of relationship of the underlying data and the form of the outcome variable. For example, to map a value that can be everything to a probability the Sigmoid function can be used. By using non-linear activation functions the network can be trained to understand nonlinear relations between input and output variables *Leshno and Schocken (1991)*. Before the Rectifier Linear Unit (ReLU), *Rumelhart et al. (1986)*, the sigmoid function was the most widely used activation function, *Pedamonti (2018)*. Functions that are like the Sigmoid function, like the Hyperbolic Tangent (TanH) are also commonly applied. *Pattanayak (2017)* states that because the ReLU functions is faster in training, it has overtaken Sigmoid and the other activation in usage. Another benefit of the ReLU is that, unlike the Sigmoid and TanH, it is not subject to vanishing gradients *Pattanayak (2017)*.

2.2.3 Convolutional Neural Network

Convolutional neural networks have proven to be very useful in various fields of science, from image recognition *Sharma et al. (2018)*, to regression analysis for mass-spectrometry analysis, *Visin et al. (2015)*. The foundation of the idea of modern CNNs lies in the LeNet introduced by *LeCun et al. (1989)*. LeNet stands for LeNet-5, a simple convolutional neural network. LeNet-5 was one of the first convolutional neural networks and promoted the development of deep learning. The original purpose of LeNet was to identify handwritten numbers. Nowadays CNNs are still mainly used for image recognition tasks but also in other applications as audio recognition CNNs are applied. One of the great benefits of CNNs, is that CNNs don't take single input vectors but take the surrounding data into account. It is clear that this will aid the research, since it's evident that data in near past is more likely to have a predictive value to speed at this point than data that has a bigger distance on the timeline. Another key benefit of CNNs, is that they are able to share weights across layers which can significantly reduce the number of parameters that the model needs. This makes it possible for CNNs to model very complex structures with limited amounts of parameters. Another benefit is that dimensional information can be stored in the network.

Just like ANNs, CNNs are built of multiple layers. The main part where CNN layers differ from ANN is the dimensional structure that is preserved. Where ANNs mainly use fully connected layers (FCL), where each neuron of layer $l - 1$ is connected to all neurons in layer l, CNNs use a great variety of layer types. All layers have different function which can vary from mapping multidimensional to flattening the outputs of these layers. These flattening layers are mainly used in the final layers of CNNs to map the multidimensional output to a single output vector in a regression problem. If the architecture of a CNNs into considerations one can see that the input of CNNs can thus be, but does not necessarily has to be, of matrix form. Because whole matrices are very big and thus hard to handle a solution was found to still be able to handle matrices as input. Three commonly used sorts of CNN layers are described in the rest of this paragraph.

The convolutional layer is the main component of interest of CNN. The concept of convolutions follows from the assumption that data points share more information with data points that are directly surrounding them than data points that are further away. Since it is computational heavy to calculate with whole matrices. Sub-matrices of the convolutional layer are multiplied with a so-called filter to send this information from layer l to layer $l+1$ *Gulli and Pal (2017)*. By applying the same filter over all the different sub-matrices the convolution is calculated. Also, this way the amount of weights is reduced, since very large input vectors can be mapped as a smaller number of sub-matrices. There are

a couple of hyperparameters that define how and which filters are applied to the matrices and what the outputs dimension will be. These hyperparameters and their meaning are:

- Filter size *(F)*: The dimension size of the filters that will be used. Width, Height and if applicable more dimensional measures
- Number of filter (N_f): The number of applied filters
- Stride *(S)*: The size of shift on the axis that is shifted on. For example, a stride of 1 means that the centre of the sub matrix shifts one step on the axis that is iterated
- Padding *(P)*: Padding increases the matrix size with a border width and height of i with zeroes. Padding can be used to preserve matrix input size after the convolution is applied

These parameters are shown in Fig.3. (N_f) is not shown but can be seen as the number of green filters there are in the convolutional layer or layers.

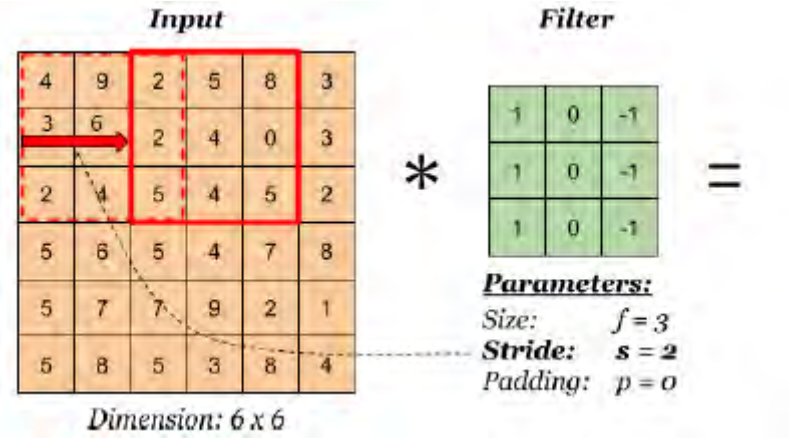

Fig.3: Parameters of CNN, https://stackoverflow.com/questions/51930312/how-to-include-a-custom-filter-in-a-keras-based-cnn

Pooling layers are used to gain dimension reduction (e.g. decrease required memory), by performing an operation on a sub-sample. This way the spatial volume of a matrix can be reduced with huge matrix operations. Max pooling, where the element with the maximum value in a sub matrix is returned, is one of the most common used forms of pooling layers. Other pooling techniques are normalisation pooling and average pooling, *Brownlee (2019)*. Pooling is a robust way of downsizing matrix sizes, *Brownlee (2019). Desphande (2016)* states that there are arguments to remove pooling layers from CNN since the gains that are made by matrix size reduction are nullified by the extra computational costs of the operation performed by the pooling layer. The hyperparameters used by pooling layers are mainly the same as with convolution layers with the important note that with padding the addition of extra zeroes or other numbers problems can occur with minimum and maximum pooling.

A flattening pooling layer is used to transform the matrix representation into a one-dimensional array. For instance, in the last step of image recognition to map to a single word. A flattening layer is a FCL that thus maps all the matrix inputs to a one-dimensional vector. It is important to note that this layer generates a lot of weights, because it is fully connected to all the matrix elements which can be very much.

2.3 Training of ANNs and CNNs

Once the framework is complete, the network is trained. There is a specific training routine that works for ANNs and CNNs. This process is described in this paragraph. In this training process a couple factors are important to know, these so called hyperparameters are described. Furthermore, there is supervised learning, unsupervised learning and reinforcement learning among others. Supervised

learning is specifically appropriate for problems where the form of the input data is known and the form of the required output is known, therefore supervised learning is applied in this research.

2.3.1 Backward Propagation

The concept of Backward Propagation (BP) is first described in *Werbos and John (1974)*. Later the method was made available for practical use by *Rumelhart et al. (1986).* The main goal of BP is to describe derivative of the loss function with regards to the weights using an efficient numerical iterative approach. When accomplished other methods can use this information to update the weights of network to lower the loss function value. Since the loss function often is a very difficult function, and thus is not easy to solve analytically the backward propagation method a combination of analytically derived partial derivatives and numerical approaches. A common problem with numerical iterative approaches is that they do no guarantee to find a global minimum, because the method might get stuck in a local minimum. The BP method thus calculate the derivative of the loss function with regard to the weights, to allow other algorithms to use this information to update the weights accordingly. The notation for neural network components is as follows:

- L the amount of layers
- N_l amount of neurons in layer l
- m_{ln} is the output of neuron n in layer l
- $w_{(l+1)j}^{l(i)}$ is the weight from node *i* in layer l to node j in layer *l + 1* and w_{ln} denotes all weights into node *n* in layer l
- m_{0n} is input *n*
- $m_{(L+1)n}$ is the output of the neural network
- z_{lm} is the sum product of all the inputs from layer *l - 1* to node *m* in layer l

The starting point of the algorithm is to calculate the derivative of the loss function notated as E. After this the weights from layer L+1 are updated, after this the weights of layer L are updated and so on until layer 1 is reached. Important to note is that the formulations will first be described on the level of the nodes. The matrix notation will be derived from this representation per node. Since the goal is to minimise the loss function while updating the weights it is important to express the derivative of loss in terms of the weights and their derivatives. To accomplish this also the derivative of the loss function in terms of m_{ln} and z_l are needed, because these are needed to propagate either forward or backward through the network. Using BP the error function and it's derivative are described. This is used to update the weights to train a better network. Following, hyper parameters are described that play a crucial role in updating of the weights.

2.3.2 Hyper Parameter Tuning

Weights are not the only parameters present in neural networks. There are numerous, so called, hyper parameters that are of influence to the networks behaviour and training process. The number of layers and number of neurons or filters are also hyper parameters. The most commonly used hyper parameters are:

- Learning rate (η): How heavily should the weights be updated after each iteration
- Batch size: How much time point should be put in per iteration
- Specific optimiser parameters: some optimiser algorithms have specific hyper parameters

3. Experimental Setup

This paragraph describes the experimental setup, which is used to train, test and verify the models. Also methods that can speed up training and prevent overfitting are described. These experiments are conducted to create the most accurate possible model for this data set and compare the results.

3.1 System Specifics

The models in this research are implemented using Python (version 3.8). The packages that are used for the model implementations are SKlearn, Keras, Tensorflow and Pytorch. The experiments are all conducted on a laptop running on Mac OS catalina with a 3 GHz i7 processor and 8 GB RAM storage. When considering computation times, these factors have to be taken into account.

3.2 Model Implementation

3.2.1 Data Preparation

For linear regression no data preparation needed is required. For the ANNs and CNNs the input data is normalised. This can improve predictions and reduce computation time *Sola and Sevilla (1997).* Normalisation can also prevent fading gradients with certain activation function such as the sigmoid function. With normalisation the data is scaled to certain characteristics of the data set. For instance, with min-max scaling all data is scaled between 0 and 1 or -1 and 1 with the max value of the set represented by 1 and the minimum represented with 0 or 1-. All the other data points are scaled within that range. In this research both non-regularised data and min-max-regularised data is used with the ANN and CNN approach.

3.2.2 Train Test Split of Data

To address the true performance of models it is preferred that the model performance is tested on unseen data. This also validates the model before it is used. This unseen data is created by splitting all the data in a train set and a test set. The train set will consist of 80% of the original data set and the test set will consist of the other 20%. First, random sample of the size of 80% of the size of the original data. All the other points are appointed to the test set. While *Flach (2012)* proposes a test set size of 10%, in this research there is chosen for a bigger test set. Because the training of the models is computationally very heavy and thus takes great time, there is no cross validation. To compensate for this fact a bigger test size is chosen. Furthermore, there is plenty of data, so the training will not be compromised. And this way the test set gets twice as big, which makes the results of the test set more reliable.

3.2.3 Set Hyper Parameters

ANNs and CNNs have multiple hyperparameters. Linear regression does not have hyperparameters other than the model itself and the loss function. The hyper parameters must be set, this is a delicate process. By varying the parameters, the model performance can increase or decrease. Since, the parameters can be varied in unlimited ways, it is good to map how changes in hyperparameters affect the performance of the model. This can be done by grid search *Pontes et al. (2016)*, random search, or varying only one hyper parameter at the time.

In grid search a *n*-dimensional, with *n* the amount of tuneable hyperparameters, grid is created. The model is trained with the setting corresponding to the place in the grid. Every next step the settings are changed in the direction of one of the parameters. The setting of hyperparameters that yields the best performance is chosen. Random search makes random jumps to other places in the parameters space. Varying the neural network hyperparameters by hand also yields a good result, since natural persons have the ability to make ‘educated guesses’. In this research there is chosen to vary the hyperparameters by hand, since the parameter space is so big and the data set large, the computation time would exceed the research time.

3.2.4 Training and Validating the Model

Training the model is a simple process. Since the data is known, the model is defined and the hyperparameters are set. The trained model yields an outcome model. This model then has to be verified.

This verification process can take place in many ways as described in subsection 3.2.2. In this research it chosen to predict the test set and calculate the MAE in comparison to the true values. The model with the lowest MAE compared to the true data is the best performing model. MAE is chosen because it is very explainable. In short MAE states that the predictions will, on average, be off by margin of the value of the MAE.

Another verification step is to check if the model is not 'overfitting'. Overfitting is when the model fits itself too much to the train data and thereby yields a worse performance on unseen test data. This can be checked by comparing the performance on the test set and the training set. If the performance on the train data is significantly better than the performance on the training set, the model is likely to be overfitting. Since in this research interpolated data is used, the model is also tested versus a subset of the true data. To make this possible some time points in the data that are looked for where all sensors have measurements within an acceptable time span. This is only done for the best performing methods, since these models will be used in further applications.

3.2.5 Compare Performance of Different Models

After all models and architectures are tested the performances of the models are compared to each other. The results of this are presented in paragraph 4. There is no standardised data set or standardised algorithm the model can be compared too. This means that one of the models that will be tested will act as a baseline model. The architecture of the baseline model will be described in section 3.3. It is important to note that comparisons between models are only sensible if the same kind of measure is used for all models. In this research this is the MAE.

3.3 Tested Model Set-ups

3.3.1 Baseline Model

The baseline model is the simplest and most naive model. It uses a linear regression method, with only engine-power as input. Intuitively this method should work, since power output should be the most influential factor on speed. For this linear regression to work the features that have collinearity between each other should be omitted from the analysis. The features that are omitted, because of collinearity, for this linear model are thruster 2 and 3 since they are strongly co-linear with thruster 1 and each other. The ordinary least squares (OLS) method is used for fitting the linear regression.

3.3.2 Linear Regression

In the architecture of linear regression models is not much room for changes to improve performance, only the data that is given as input can be altered. In this research the linear regression method that is tested, is trained on all input features but the derivatives of the ship motions. Further the fitting method that is used is the OLS method.

3.3.3 Artificial Neural Networks

As described in section 2.3 there is a big diversity of hyperparameters that can be altered. Also the features of the data that are given as training data can be varied to see if the derivatives of the ship motion have any predicting value. Further the data is given in normalised state and in original state. Following, there is a multitude of activation functions that are tested as well as amounts on neurons in particular hidden layers. In first instance 1 hidden layer neural networks is tested, after deeper neural networks are tested. As explained the hyperparameters are varied by hand.

For computation time purposes, first all models are trained for 50 epochs. An epoch is one complete pass through the training data. The model with the best performances will be trained for 250 epochs to obtain a better performance. The number of neurons in the first layer is originally set at 23 of the input variables. Later greater amounts are tested. When models do not converge in 15 epochs the training is

cancelled and the performance is described as non-converging, and the value of the loss functions of the last epoch is given. A list of the tested architectures is provided in table 1. In this list not all varieties of the tested architecture are presented; the best of a particular architecture are. When there is a significant performance difference between for example 17 or 18 neurons in a particular hidden layer, the different architectures will be both be presented.

3.3.4 Convolutional Neural Networks

CNNs share a lot of the hyperparameters with regular ANNs. The extra hyperparameters, needed for CNNs are described in section 2.2.3. The list of CNN architectures is given in Table I. The architecture of the best performing ANNs is used as a skeleton for the initial design of the CNNs. Again here not all architectures are described.

Table I: Description of Tested Models

Model code	Model	Normalisation	Derivatives	Hidden layers	Neurons/layer	Activation functions	Optimiser	Batch size	Kernel size
NN_N_18x_1024_RMSprop_ExDer	NN	No	No	1	18	x	RMSprop	1024	x
NN_MM_18x_1024_RMSprop_ExDer	NN	Min Max	No	1	18	x	RMSprop	1024	x
NN_MM_18x_1R_1024_RMSprop_ExDer	NN	Min Max	No	2	18 1	x relu	RMSprop	1024	x
NN_MM_22x_1R_2048_RMSprop_Der	NN	Min Max	Yes	2	22 1	x relu	RMSprop	2048	x
NN_MM_26S_1R_2048_RMSProp_Der	NN	Min Max	Yes	2	26 1	Sigmoid Relu	RMSprop	2048	x
NN_MM_26S_1R_2048_Adam_Der	NN	Min Max	Yes	3	26 1	Sigmoid Relu	ADAM	2048	x
NN_MM_28S_1R_512_Adam_Der	NN	Min Max	Yes	3	28 1	Sigmoid Relu	ADAM	512	x
NN_MM_26_14S_1R_2048_RMSProp_Der	NN	Min Max	Yes	3	26 14 1	Sigmoid x Relu	RMSprop	2048	x
NN_MM_26_14S_1R_2048_ADAM_Der	NN	Min Max	Yes	3	26 14 1	Sigmoid x Relu	ADAM	2048	x
NN_MM_26_14S_5S_1R_2048_ADAM_Der	NN	Min Max	Yes	4	26 14 5 1	x Sigmoid Sigmoid Relu	ADAM	2048	x
CNN_MM_26S_F_1R_2048_ADAM_Der_3	CNN	Min Max	Yes	2	26 F 1	Sigmoid Relu	ADAM	2048	3
CNN_MM_26T_F_1R_2048_ADAM_Der_3	CNN	Min Max	Yes	2	26 F 1	TanH Relu	ADAM	2048	3
CNN_MM_26R_F_1R_2048_ADAM_Der_3	CNN	Min Max	Yes	2	26 F 1	Relu Relu	ADAM	2048	3
CNN_MM_26R_F_1R_2048_ADAM_Der_5	CNN	Min Max	Yes	2	26 F 1	Relu Relu	ADAM	2048	5
CNN_MM_26_14R_F_1R_2048_ADAM_Der_5	CNN	Min Max	Yes	2	26 14 F 1	x Relu Relu	ADAM	2048	5
CNN_MM_28_15T_F_1R_2048_ADAM_Der_5	CNN	Min Max	Yes	2	28 15 F 1	x Tanh Relu	ADAM	2048	5

4. Results

4.1 Baseline and Linear Regression Model

As described, for linear regression models to work first the values of vector a must be calculated using the model. After the base model is fitted the parameters of the base model are as follows: a = [2.024, -4.581e-04, -1.5099e-03, -1.206e-03, 1.131e-03, 1.866e-04]. The mean absolute error (MAE) of this model is 2.198 kn. This is not very precise given the range of the true values lying between 0 and 30. While the values of the predicted values roughly lie between 0 and 5. This states that the model does not fit well, but it does provide a good starting point from where the optimisation models can start. The linear regression model with all data (except the derivatives of ship motions) as input was fitted, Table II. When observing the predictions of this model over the time span, Fig.4, one thing stands out. There are negative speeds predicted, which is not good. Although there are no linear correlations found between the independent and dependent variables this model performs well with an MAE of 0.5512. This is a significant increase in performance from the baseline model. Also the influence of the ship motions has negative coefficients, i.e. when ship motions get more extreme the predicted speed of the ship goes down. This seems logical, as extreme ship motions imply extreme weather. As explained, this affect the resistance of the ship, decreasing the speed. The expectation is that this model cannot be improved significantly anymore; for further performance increase, other models and methods should be used.

Table II: Parameters of Fitted Model

Intercept	0.34556	KMT_thruster_1_pitch	-0.00171	KMT_thruster_4_power	0.00059
draught fore	0.05085	KMT_thruster_4_pitch	-0.00952	KMT_thruster_5_power	0.00038
draught aft	-0.01709	KMT_thruster_5_pitch	-0.00481	KMT_thruster_6_power	0.00009
trim	-0.51743	KMT_thruster_6_pitch	0.02035	KMT_thruster_7_power	-0.00028
list	0.13808	KMT_thruster_7_pitch	0.01552	heave	-0.3261
KMT_thruster_4_azimuth	-0.00092	KMT_thruster_1_power	-0.00079	pitch	-0.03288
KMT_thruster_5_azimuth	-0.00065			roll	-0.02212

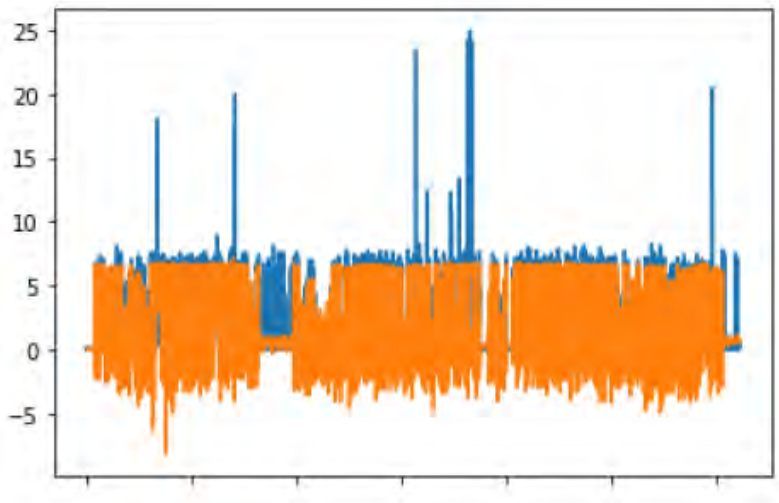

Fig.4: Results of Linear Regression over Interval

4.2 Results of ANNs and CNNs

The best performance reached with linear regression is a MAE of 0.55. The expectation is that the machine learning methods will outperform these linear regression models. The training errors and ultimate test set errors are presented in table 3. The increase of performance once the sigmoid activation function is deployed is remarkable. This can be explained by the non-linearity of speed and resistance. As can be seen the tanh activation function is also tested, but yielded similar to worse results. Furthermore, it occurs the derivatives of ship motions do not have predictive value on their own. They need a non-linear activation function in the base of the network to exploit their full potential. Also the optimiser seems to be of great influence when training for 50 epochs.

Table 3: Results of Machine Learning Approaches

Model code	Training error	Test error
NN_N_18x_1024_RMSprop_ExDer	0.4892	0.4862
NN_MM_18x_1024_RMSprop_ExDer	0.4834	0.4831
NN_MM_18x_1R_1024_RMSprop_ExDer	0.3835	0.3838
NN_MM_22x_1R_2048_RMSprop_Der	0.3835	0.3868
NN_MM_26S_1R_2048_RMSProp_Der	0.2719	0.2746
NN_MM_26S_1R_2048_Adam_Der	NC (1.74)	NC
NN_MM_28S_1R_512_Adam_Der	0.2450	0.2413
NN_MM_26S_1R_2048_RMSProp_Der Trained for 250 epochs	0.2354	0.2422
NN_MM_26_14S_1R_2048_RMSProp_Der	0.2372	0.2357
NN_MM_26_14S_1R_2048_ADAM_Der	0.2260	0.2230
NN_MM_26_14S_5S_1R_2048_ADAM_Der	0.2223	0.2227
CNN_MM_26S_F_1R_2048_ADAM_Der_3	NC (1.74)	NC
CNN_MM_26T_F_1R_2048_ADAM_Der_3	0.2871	0.2853
CNN_MM_26R_F_1R_2048_ADAM_Der_3	0.2635	0.2626
CNN_MM_26R_F_1R_2048_ADAM_Der_5	0.2537	0.2548
CNN_MM_26_14R_F_1R_2048_ADAM_Der_5	0.2465	0.2482
CNN_MM_28_15T_F_1R_2048_ADAM_Der_5	0.2366	0.2365

In the ANN model with 2 hidden layers with sigmoid activation function on the first layer the model does converge with RMSprop but does not converge with the ADAM optimiser. The batch size had to be altered and the architecture of the network had to be changed slightly to get a converging model. Also the increase of performance can be up to and even more than 4% when only changing the optimiser. This is clearly visible in the models with 2 layers with sigmoid activation on the first layer. Adding more layers will not work. The increase from the model NN MM 26 14S 1R 2048 ADAM Der to model NN MM 26 14S 5S 1R 2048 ADAM Der is roughly 0.14% the number of parameters and thus the required computation times goes up.

Analysing the results of both models NN MM 26S 1R 2048 RMSProp Der, illustrates training a simpler model for greater time can cause overfitting. The performance on the training set is better than the performance on the test set. Also, the CNN model with sigmoid activation function does not converge. This could be because, the sigmoid function maps all input to a value between 0 and 1 while the tanh function maps between -1 and 1. Furthermore, the computation time of the CNN is longer than that of their ANN counterparts. This increase is up to a factor six. Increasing the kernel size, while not significantly increasing the computation time, does increase the performance of the model. Overall, more ANN models converged than CNN models. Small changes in the number of filters per layer could induce non convergence. In CNN models a significant improvement is found when the model is made more complex, with different layers.

The NN MM 26S 1R 2048 RMSProp Der model behaves as expected and converges. The same pattern is found in all ANN methods that converge. All converging CNN methods converge likewise. What can be seen is the difference in convergence speed and the shape of the convergence plot between the Adam optimiser and the RMSprop optimiser. The Adam optimiser converges faster at the beginning. The loss function derivative goes to zero over time, while the RMSprop loss function behaves as a straight line from epoch 10 on.

4.3 Model Performance Comparison

There are a couple of important trends that can be picked up from the comparison. More complex models perform better than the less complex models. The CNN do not perform significantly better than the ANN, in some instances they perform worse. The ADAM optimiser the convergences quicker and further than the RMSprop optimiser, as can be seen in Fig.8. When comparing models that use standardised data with models that used non-standardised data, it can be concluded that the models using standardised data converge quicker, but do not perform better.

Table IV: Approximate Computation Time for Certain Models

Model code	Approximation of computation time
Linear regression models	<1 min
NN_N_18x_1024_RMSprop_ExDer	5 min
NN_MM_18x_1024_RMSprop_ExDer	5 min
NN_MM_18x_1R_1024_RMSprop_ExDer	5 min
NN_MM_22x_1R_2048_RMSprop_Der	5 min
NN_MM_26S_1R_2048_RMSProp_Der	10 min
NN_MM_26_14S_1R_2048_ADAM_Der	15 min
NN_MM_28S_1R_512_Adam_Der	25 min
CNN_MM_26S_F_1R_2048_ADAM_Der_3	30 min
CNN_MM_26R_F_1R_2048_ADAM_Der_3	30 min
CNN_MM_26_14R_F_1R_2048_ADAM_Der_5	2.5 hours
CNN_MM_28_15T_F_1R_2048_ADAM_Der_5	2.5 hours
NN_MM_26_14S_1R_2048_ADAM_Der 250 epochs	80 min
CNN_MM_28_15T_F_1R_2048_ADAM_Der_5 250 epochs	4 hours

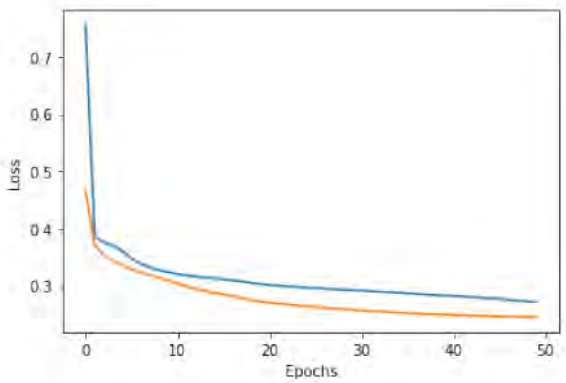

Fig. 8. Convergence per Epoch for RMSprop (Blue) and Adam (Orange) Optimisers

In Table IV the rough estimates of training time per model is presented. Taken these computation times into consideration the artificial neural network model with code NN MM 26 14S 1R 2048 ADAM Der is the best model in this research.

4.4 Final Validation and Improvement of Best Models

To unlock the full potential of the models, the models NN MM 26 14S 1R 2048 ADAM Der and CNN MM 28 15T F 1R 2048 ADAM Der 5 were trained for 250 epochs. The performance of the CNN model increased to 0.2274 on the training set and 0.2270 on the test set. The ANN model performance increased to 0.2151 on the training set and 0.2150 on the test set. This is very good in comparison to the baseline model and also the first linear model. The MAE of best linear regression method is approximately 2.5 bigger than the MAE of these two methods. As described in section 4.3 the 250 epoch version of the best performing models are also tested.

The performance of the NN MM 28S 1R 512 Adam Der model trained for 50 epochs is 0.1845. This is a better performance than on the test set of the interpolated data. This is a very promising result. Especially since the model will be used on real life data. The CNN MM 28 15T F 1R 2048 ADAM Der 5 trained for 250 epochs had an performance of 0.2059 on the true test set. Also this model shows an increase in performance which is a positive sign and is another implication that the models are not overfitted to the interpolated data. This implies the interpolated data are a good representation of the true data. The performance of the NN MM 26 14S 5S 1R 2048 ADAM Der on the true data set is 0.2003. This is worse than the NN MM 28S 1R 512 Adam Der model but still performing better on the true data set than on the training and test set from the interpolated data. The fact that all these models perform good on the true data set implies that no overfitted models were created.

5. Discussion of Results

5.1 Novel Approach

This research opens the door for a new way of estimating ships speed during different operational profiles with data from on-board sensors. The developed method is industry wide applicable, more accurate, less expensive and therewith more encompassing and accessible then other methods. This model could be combined with a model that predicts ship motions, based on environmental phenomena and the ships position. Furthermore, the developed model could be combined with an optimiser algorithm to find the most optimal power output for the engines, given predicted weather forecasts and related ship motions. This combination could be used for route optimisation from point to point since speed can be predicted. So optimal power and speed couple could be predicted for the route. This is a very complex problem combining routing problems with ship motion, and speed prediction. This model could also serve as the start of a prediction model, if data of more ships could be used, where hull form parameters would also be given as an input. If this succeeds, the model could be turned into a more generic model that could be applied to predict the speed of a series of ships.

5.2 Model Usability

The model is suited to be reproduced and used by any shipping company. The training and calculations are based on vessel specific variables. This vessel specific dataset can be altered into any other similar dataset, when training the model for other vessels. Furthermore, the required computational power is very low, the development and training was performed on a normal laptop. Meaning the model could be used on the bridge of a vessel for live operational decision making. Concluding the developed model is generally available and applicable.

5.3 Discussion

During the research there were several assumptions and limitations to the developed model. Critical points of discussion with regards to the assumptions and limitations include:

- Due to the nature of the original data, interpolated data was used. Since there is always a chance of information loss with interpolation this is ideally prevented. Although the models have a decent performance on the 'true' data, this true data is also an interpretation of the real data. Further research might prevent this by actively managing the data gathering procedure.
- During this research there was a lack of certain context data. Historical wind and wave data was not available at the exact position of the ship. When evaluating the outcomes of the models these features could have increased the model performance.
- It is important to note that this model by itself is not capable of predicting the speed of a vessel of a future trip. This because the model takes the ship motion as input, and ship motion is not known in advance.
- In this research field it is very hard to relate the outcome of this study to outcomes of other studies. It is difficult to compare the results of this study to other studies that assess comparable topics. This complicates this particular study because it is hard to map progress on an absolute scale. A standardised validation set would improve this research and the maritime research field as a whole.

6. Conclusion

The goal of this research is to present a way to improve the speed and resistance prediction of ships. This research presents a method, a machine learning approach. The speed over ground of a vessel can be predicted with numerous methods. The best predictions on training and testing data were made by an artificial neural network containing three hidden layers with sigmoid and relu activation functions. The best performance on the 'true' data was reached with a model with one hidden layer with sigmoid activation.

This novel approach for speed and resistance prediction, offers a quick and accurate speed over ground based on operational data. For ship design and ship management companies this is important information, that can be used in ship design or operation. The method developed in this research is inexpensive, generally applicable and proves the concept.

To summarise the findings of this study, it is indeed possible to predict the speed of the ship in operational condition with data from on-board sensors. Next to being an improved speed and resistance prediction method, the successful implementation and validation of machine leaning to predict speed over ground also provided valuable novel insights in machine learning in marine engineering in general. Therewith, contributing to the body of knowledge of data-science in marine engineering.

Recommendations for further research comprise the development of a benchmark model and standardised data set, development of a ship motion prediction model and further development and generalisation of the developed model.

Acknowledgements

This research has been performed within the TODDIS project, partially funded by the Dutch Research Council (NWO) under grant agreement Raak-Pro program 2018, n° 03.023.

References

ABRAMOWSKI, T. (2008), *Application of artificial neural networks to assessment of ship manoeuvrability qualities,* Polish Maritime Research 15(2), pp.15-21

ABRAMOWSKI, T. (1999), *Application of Artificial Intelligence Methods to Preliminary Design of Ships and Ship Performance Optimization*, Naval Engineers Journal 125(3), pp.101-112

ABRAMOWSKI, T.; ZMUDA, A. (2008), *Generalization of container ship design by means of neural networks,* Polish J. Env. Studies, pp.111-115

AGGARWAL, C.C. (2018), *Neural Networks and Deep Learning: A Textbook*, Springer

BARCZAK, N. (2020), *The ship towing tank*, https://dmsonline.us/the-ship-towing-tank/

BERTRAM, V. (2012), *Practical Ship Hydrodynamics*, Butterworth-Heinemann

BERTRAM, V. (2014), *Trim optimization - Don't blind me with science,* The Naval Architect, pp.66-68

BRANSAETER, A.; VANEM, E. (2017), *Ship speed prediction based on full scale sensor measurements of shaft thrust and environmental conditions*, Ocean Eng. 162, pp.316-330

BROWNLEE, J. (2019), *A gentle introduction to Pooling layers for Convolutional Neural Networks*, https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks/

CHUANG, Z.; STEEN, S. (2011), *Prediction of Speed Loss of a Ship in Waves,* 2nd Int. Symp. Marine Propulsors (smp'11), Hamburg

CUI, H.; TURAN, O.; SAYER, P. (2012), *Learning-based ship design optimization approach,* CAD Computer Aided Design 44(3), pp.186-195

DESPHANDE, A. (2016), *A Beginner's Guide To Understanding Convolutional Neural Networks*, https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks/

FEIJO, H.; DE OLIVEIRA, J. (2020), *Comparison between empirical forecast methods of resistance to advance and propulsive design of a container ship,* CILAMCE, Foz do Iguaçu

FLACH, P. (2012), *Machine Learning: The Art and Science of Algorithms That Make Sense of Data,* Cambridge University Press

GATIN, I. (2019), *CFD in the marine industry: Today and Tomorrow*, https://thenavalarch.com/cfd-in-the-marine-industry-today-and-tomorrow/

GRABOWSKA, K.; SZCZUKO, P. (2015), *Ship resistance prediction with Artificial Neural Networks. Signal Processing - Algorithms*, Architectures, Arrangements, and Applications Conf., pp.168-173

GULLI, A.; PAL, S. (2017), *Deep Learning with Keras*, Packt Publ.

HEBB, O. (1949), *The Organization of Behavior: A Neuropsychological Theory,* Taylor & Francis

HINO, T. (2005), *Proceedings of CFD Workshop Tokyo,* Univ. Tokyo

HOLLENBACH, U.; CHRYSSOSTOMIDIS, C.; JOHANSSON, K. (1999), *Estimating Resistance and Propulsion for Single-Screw and Twin-Screw Ships in the Preliminary Design,* Int. Conf. Computer Applications in Shipbuilding, Vol. 2, pp.237-250

HOLTROP, J. (1984), *Statistical re-analysis of resistance and propulsion data*, Int. Shipb. Progress, pp.272-276

HOLTROP, J.; MENNEN, G. (1982), *An Approximate Power Prediction Method,* Int. Shipb. Progress 29, pp.166-170

KIM, D.; LEE, S.; LEE, J. (2020), *Data-driven prediction of vessel propulsion power using support vector regression with onboard measurement and ocean data,* Sensors (Switzerland) 20(6)

LECUN, Y.; BOSER, B.; DENKER, J.; HENDERSON, D.; HOWARD, R.; HUBBART, W.; JACKEL, L. (1989), *Backpropagation Applied to Handwritten Zip Code Recognition,* Neural Computation 1(4), pp.541-551

LESHO, M.; SCHOCKEN, S. (1991), *Multilayer Feedforward Networks with Non-Polynomial Activation Functions Can Approximate Any Function,* NYU Stern School of Business Res. Paper Series

LI, D.; GUAN, Y.; PHILIP, A.; WILSON; ZHAO, X. (2014), *An effective approximation modeling method for ship resistance in multidisciplinary ship design optimization,* Int. Conf. Offshore Mechanics and Arctic Eng. (OMAE)

LIANG, Q.; TVETE, H.; BRINKS, H. (2019), *Prediction of vessel propulsion power using machine learning on AIS data, ship performance measurements and weather data,* J. Physics: Conference Series 1357(1)

MAO, W.; RYCHLIK, I.; WALLIN, J.; STORHAUG, G. (2016), *Statistical models for the speed prediction of a container ship.* Ocean Engineering 126, pp.152-162

MATULJA, D.; DEJHALLA, R. (2007), *A Comparison of a Ship Hull Resistance,* Eng. Rev, 27, pp.:13-24

McCULLOUGH, W; PITTS, W. (1943) A *logical calculus of the ideas immanent in nervous activity,* Bulletin of Mathematical Biophysics 5(4), pp.115-133

MINSKY, M.; PAPERT, S. (2017), *Perceptrons: An Introduction to Computational Geometry,* The MIT Press

NIKOPOULOS, L.; BOULOUGOURIS, E. (2019), *A study on the statistical calibration of the Holtrop and Mennen approximate power prediction method for full hull form, low Froude number vessels*, J. Ship Production and Design 35(1), pp.41-68

OZDEMIR, Y.; BARLAS, B. (2017), *Numerical study of ship motions and added resistance in regular incident waves of KVLCC2 model*, Int. J. Naval Architecture and Ocean Eng. 9(2), pp.149-159

PATTANAYAK, S. (2017), *Pro Deep Learning with TensorFlow: A Mathematical Approach to Advanced Artificial Intelligence in Python,* Apress

PEDAMONTI, D. (2018), *Comparison of non-linear activation functions for deep neural networks on MNIST classification task*, arXiv (3)

PEDERSEN, B.; LARSEN, J. (2009), *Prediction of full-scale propulsion power using artificial neural networks,* 8th COMPIT Conf., Budapest, pp.537-550

PEDERSEN, B.; JACOBSEN, D.; WINTHER, O. (2012), *Statistical modelling for ship propulsion efficiency,* J. Marine Science and Technology 17(1), pp.:30-39

PONTES, F.; AMORIM, G.; BALESRASSI, P.; PAIVA, A.; FERREIRA, J. (2016), *Design of experiments and focused grid search for neural network parameter optimization,* Neurocomputing 186, pp.:22-34

ROSENBLATT, F. (1958), *The perceptron: A probabilistic model for information storage and organization in the brain*, Psychological Review 65(6)

RUMELHART, D.; HINTON, G; WILLIAMS, R. (1986), *Learning representations by back propagating errors*, Nature 323(6088), pp.533-536

SADAT-HOSSEINI, H.; WU, P.; CARRICA, P.; KIM, H.; TODA, Y.; STERN, F. (2013), *CFD verification and validation of added resistance and motions of KVLCC2 with fixed and free surge in short and long head waves.* Ocean Engineering 59, pp.240-273

SHARMA, N.; JAIN, V.; MISHRA, A. (2018), *An analysis of convolutional neural networks for image classification,* Procedia Computer Science 132, pp.377-384

SOLA, J.; SEVILLA, J. (1997), *Importance of input data normalization for the application of neural networks to complex industrial problems,* IEEE Trans. Nuclear Science 44(3 PART 3), pp.1464-1468

VAN, S.; KIM, W.; KIM, D. (2000), *Experimental investigation of local flow around KRISO 3600TEU container ship model in towing tank*, J. Society of Naval Architects of Korea 37 (01)

VISIN, F.; KASTNER, K.; COURVILLE, A.; BENGIO, Y.; MATTEUCCI, M.; CHO, K. (2015), *ReSeg: A Recurrent Neural Network for Object Segmentation*, Computer Vision and Pattern Recognition

WERBOS, P.; JOHN, P. (1974), *Beyond regression: new tools for prediction and analysis in the behavioral sciences*, Harvard University