Modelling of an AUV with Voith-Schneider Vector Thruster

Rajat Mishra
NUS Graduate School for Integrative Sciences and Engineering, National University of Singapore, Singapore, 117456
Email: rajat.mishra@u.nus.edu

Mandar Chitre
Department of Electrical and Computer Engineering, National University of Singapore, Singapore, 119222
Email: mandar@nus.edu.sg

***Abstract*— First principles physics models are generally used in system identification of Autonomous Underwater Vehicles (AUVs). These models, through different parameters, capture the effects of hydrodynamics, inertial weight and other important characteristics. Due to the large number of parameters, which can number to hundreds, it is difficult to estimate such models. Moreover, AUV capabilities like thrust vectoring increases the non-linearity of the model. We suggest solving the problem of modelling AUVs with the help of a rectifier activated multilayer perceptron, making use of their motion data and control inputs. We also provide details on the optimisation of our model and compare its performance with that of a standard system identification technique. Although the rectifier neural network's performance was tested for a typical streamlined AUV with a Voith-Schneider thruster, the model presented here is general and can be easily extended to other systems.**

Keywords—AUV modelling; neural network; vector thruster; voith-schneider; system identification

I. INTRODUCTION

An important aspect of designing a controller for any robotic system is to develop an accurate dynamics model of it. Traditional approaches in modelling AUVs are based on physics models derived using first principles. Generally, the dynamics of an AUV are described using six degrees of freedom and their respective differential equations of motion [5]. These equations have parameters that represent the nonlinear components of AUV dynamics. However, in practice, it is quite difficult to estimate these parameters and therefore, certain assumptions are made to linearise this model for easier estimation [9]. These linearised models are actually the simplified equations of inter-dependent state variables, which are computed using regression analysis of AUV motion data. However, due to linearisation, these models usually fail to predict the dynamics accurately during complex manoeuvres. Moreover, AUV designs which have different propulsion mechanisms or have significantly different body structure as compared to industrial AUVs like Iver, Gavia and Remus cannot use the linear physics model defined in [5, 9]. Therefore, for such systems, the linearised model needs to be estimated again and tuned to compensate for the change in dynamics. One way of modelling these systems is to use a linear representation of the AUV's state variables and control inputs, also known as output equation of a state-space model. In comparison to the traditional approach, this is an easier method for estimating unknown dynamics as it utilizes the motion data directly without requiring detailed analysis of the AUV's physics model.

Other than the physics-based approach and the state-space model, some other approximation-based techniques have also been used in the past for learning the dynamics model [2, 16, 20, 23]. In particular, the feedforward neural networks have the capability of approximating any continuous function [8] and therefore, they are good candidates for such applications. Inspired by this, a linearly parameterised neural network was used to estimate dynamics of a surface vessel [21]. However, the neural network presented in [21] has no hidden neurons and its input features are directly connected to its output layer. It is shown in [18] that if a complex mapping exists between the input and output units, a large hidden layer is required in between to estimate the mapping perfectly. Therefore, in order to develop an efficient dynamics model and predict complex manoeuvres accurately, a multilayer perceptron (MLP) type of neural network appears to be a promising solution.

An MLP is a fully-connected feedforward neural network used for function approximation. In an MLP, most of its neurons have a nonlinear activation function, the standard choices of which are signum, sigmoidal or hyperbolic tangent. Reference [10], along with [15], showed that the use of rectifiers as activation functions in different neural networks improved their discriminative performance. Following this approach, in [6] the performance of different activation functions was compared, and the results demonstrated that rectifier neurons were better at finding minima during training for classification tasks, as well as for contextual analysis. These promising results led to recent advancements in the area of Deep Learning. Rectifier neural networks have demonstrated state-of-the-art performance in natural language processing [14], image understanding [11] and speech processing [13, 22]. Motivated by these results, rectifier neural networks were also used to learn the dynamics model of a helicopter [17]. In [17], a rectifier neural network coupled with a quadratic function of the helicopter's state variables, accurately predicted its acceleration during complex manoeuvres.

We propose a rectifier activated MLP to approximate an AUV's dynamics. In addition, we also describe the techniques required for optimising our MLP's learning process. We discuss about a state-space model of our AUV in Section II. For the purpose of experimentation, we chose a cylindrical-shape AUV as it is hydro-dynamically efficient, and because it is a standard design used in many industrial AUVs. Our AUV was also

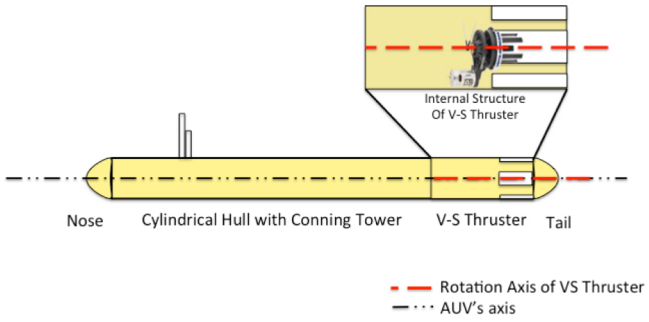
equipped with a thrust vectoring module described in Section II. In Section III, we introduce our MLP network and provide a possible interpretation of why our model works. The optimisation process, details about our experiments and performance comparison of our models are given in Section IV. Finally, Section V summarises the overall performance of our model.

II. Linearised Model of AUV with Vector Thruster

A. Voith-Schneider Propeller as Vector Thruster

Typical design of an AUV consists of a cylindrical hull with a horizontal thruster and four protruding control fins. The integrated motion of these four fins control the AUV's roll, pitch, yaw and depth. However, as the motion is dependent on the control surfaces around the AUV, this design has limited maneuverability. For example, such AUVs under nominal speeds generally have large turning radii. In addition, the protruding fins increase the AUV's drag and are mechanically the weakest part of its body. An alternative to this fin design is the vector thrusters.

In sea vessels like ships or ferries, complex maneuvering such as on-the-spot turning is made possible by the use of Voith-Schneider (V-S) propulsion mechanism [24]. In general, this propulsion system consists of a cycloidal rotor that provides thrust in the direction perpendicular to its rotation axis. This thruster mechanism can be used to control the AUV's yaw and pitch, replacing the function of the fins. In order to substitute the function efficiently, we align the rotation axis of the V-S propeller to coincide with the AUV's roll axis. Furthermore, this thruster is positioned close to the AUV's tail to replicate the effect of the four fins (Fig. 1.a). This design makes the vehicle's motion independent of control surfaces and adds capabilities like



(a)

(b)

Fig. 1. (a) Sketch showing the alignment of V-S thruster with respect to AUV and the possible thrust directions. (b) Fabricated and assembled V-S propeller being tested in a tank.

thrust vectoring and on-the-spot turning. However, by adding this thruster module, the non-linear behavior of the AUV increases and the system dynamics becomes more complex.

B. State-Space Equation Model

Generally, the dynamics of an AUV is described using six degrees of freedom differential equations of motion [5]. The model equations are defined in two different coordinate frames: earth-fixed or North-East-Down (NED) frame and body-fixed frame. Six velocity components, $v = [u, v, w, p, q, r]^T$ (surge, sway, heave, roll, pitch, yaw) are used to define dynamics in the body-fixed frame, while $\eta = [\eta_r, \eta_e] = [x, y, z, \phi, \theta, \psi]^T$ defines Euler angles ($\eta_e = \phi, \theta, \psi$) and distance ($\eta_r = x, y, z$) between NED and body-fixed frame in NED coordinate system. The notation used in this paper is in accordance with SNAME [19]. These two vectors are related through Euler Angle transformation:

$$\dot{\eta} = Jv \tag{1}$$

The nonlinear vehicle dynamics of any robotic system operating in fluids can be expressed in a compact form as [5]:

$$M\dot{v} + C(v)v + D(v)v + g(\eta) = B(v)u \tag{2}$$

where M represents the total inertia, $C(v)$, the coriolis and centripetal forces, $D(v)$, the hydrodynamic damping, $g(\eta)$, the vector of restoring forces and moments, $B(v)$, the control matrix and u, which accounts for all the control inputs to the system. However, it is not practical to use (2) for estimating the dynamics of an AUV as each component of the equation is highly nonlinear and difficult to estimate via field experiments. For details on this, interested readers are recommended to review literature in [5]. Therefore, an alternative solution is to separate the model into non-interacting or lightly interacting subsystems. The widely accepted solution is to divide the model into the following three categories: speed subsystem, steering subsystem and diving subsystem [9]. However, in cases where pitch and yaw dynamics are coupled, for example in thrust vectoring setups, it is not feasible to divide the AUV into these subsystems. Therefore, for such AUV designs, the output equation of a state-space representation can be used to obtain a simplified dynamics model. This model can be represented as:

$$f = \boldsymbol{A}\boldsymbol{x}_{v,\eta_e} + \boldsymbol{B}\boldsymbol{u} \tag{3}$$

where f is a system parameter that is being modelled and $\boldsymbol{x}_{v,\eta_e}$ is the vector containing the AUV's state variables. The state-space equation model is capable of capturing all possible linear dependencies on state variables and control inputs. Interestingly, the three subsystems mentioned in [9] can also be represented using (3). For example, the yaw acceleration in steering subsystem is dependent on yaw rate and rudder input. Therefore, in such cases, all the array elements of matrices **A** and **B** not corresponding to yaw rate and rudder input will be zero. This implies that the matrices **A** and **B** contain elements that have physical significance like components of hydrodynamic added mass or the effects of stern input. Therefore, even though a linearised physics model is not derived for our AUV, it can be concluded that the state-space representation is very similar to a linear physics model and it can be considered as a good baseline for performance comparison.

In a state-space equation system, the coefficient matrices **A** and **B** define the accuracy of the model and generally, they are estimated using simple least square analysis on input data. As the state-space model is a linear approximation of the AUV's dynamics rather than a theoretical derivation, its accuracy will depend mostly on the quality of motion data. If the data used to estimate the coefficient matrices do not capture sufficient dynamics, it will be difficult to obtain an accurate dynamics model. Therefore, the state-space equation model offers a good linear representation of the AUV but it does not guarantee an accurate dynamics model as it lacks non-linear dependency and is also affected by the quality of motion data.

III. Multilayer Perceptron as Function Approximators

A multilayer perceptron is a feedforward artificial neural network that maps the input data onto a set of relevant outputs and is known for its function approximation capability. The conventional architecture of an MLP is a fully connected network with three layers: an input layer having all the input features, a hidden layer activated by an activation function and an output layer giving the final result. We have adopted the same network architecture for our model. The output f_{neural} of our MLP can be expressed as follows:

$$f_{neural} = \boldsymbol{w}_{hidden} \times \varphi\left(\boldsymbol{w}_{input} \times \boldsymbol{y}_{v,\eta_e,u} + \boldsymbol{b}_{input}\right) + \boldsymbol{b}_{hidden} \quad (4)$$

where $\boldsymbol{w}_{hidden}$ and $\boldsymbol{b}_{hidden}$ are the weights and biases for the hidden layer, $\boldsymbol{w}_{input}$ and $\boldsymbol{b}_{input}$ are the weights and biases for input layer, $\varphi(\cdot)$ is the activation function and $\boldsymbol{y}_{v,\eta_e,u}$ is the input feature vector containing the AUV's state variables and control input.

MLPs have the capability of universal function approximation because of the nonlinear transformation, $\varphi(\cdot)$, in their hidden layer. This transformation of the scaled and shifted input features can either activate or deactivate each hidden unit. Therefore, some of the hidden units get activated for certain regions of the input data and learn their representation in the final output. This results in projecting the entire input dataset into a space where it becomes linearly separable with respect to each hidden unit. Therefore, the estimated output is obtained using only the hidden units activated by a particular input. We believe that such characteristics would be very powerful in learning an accurate dynamics model for AUVs. For example, we know that in a linear physics model, yaw acceleration is dependent on yaw velocity and rudder deflection [9]. However, this representation is a simplified version of (2), which shows that yaw acceleration is dependent on other state variables as well. Therefore, we opted to design an MLP to estimate the dynamics model as it is capable of learning such underlying representations through the nonlinear transformations.

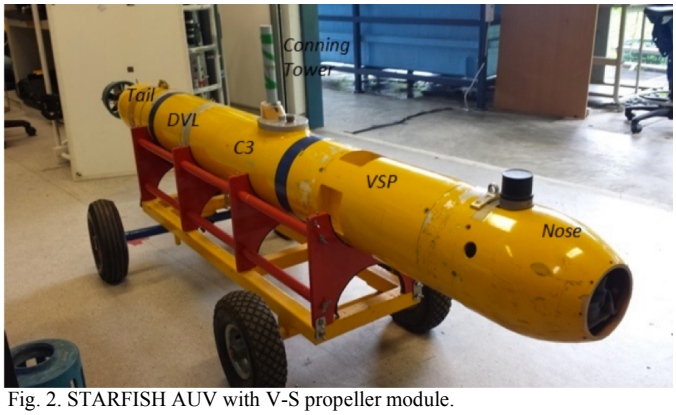


Fig. 2. STARFISH AUV with V-S propeller module.

The activation function, $\varphi(\cdot)$, is one of the key features of a neural network. The standard options for an activation function of MLP are sigmoid and tanh. However, recent advancements in Deep Learning have been driven by the use of rectifiers as activation function for neural networks. A simple rectifier activation function, $\varphi(\cdot)$, can be represented as:

$$\varphi(\cdot) = max(0, \cdot) \quad (5)$$

An important characteristic of such activation function is that its output is proportional to its input for all positive input values, and zero for all negative input values. This gives rectifiers a much larger active region than the tanh function. Also, such activation function helps in efficient gradient propagation during training, and does not suffer from the vanishing gradient problem [7].

IV. Experiments

A. AUV's Motion Data

In order to test the performance of our network against the baseline model, we collected the motion data using our modular AUV called STARFISH (See Fig. 2). It has a thruster for horizontal propulsion and a V-S propeller for thrust vectoring. The horizontal propulsion and V-S propeller together control the yaw, pitch and depth of STARFISH. In order to suppress its roll dynamics, STARFISH has an internal rolling compensator mechanism [4]. For data collection, it was taken to a reservoir and commanded to execute a compact set of manoeuvres, which included 360 ° on-the-spot turning. This was done under different thrust values to excite the AUV's dynamics and a total of 12 minutes of motion data was recorded. This data has 13 features: orientation (ϕ, θ, ψ), linear and angular velocities (u,v,w,p,q,r), servo positions of V-S propeller's control rod (δ_{s1}, δ_{s2}), V-S propeller's rotational rpm ($n_{V\text{-}S}$) and horizontal thruster's rpm (n). As some of the sensor units had low sampling frequency, the data was interpolated to obtain a sampling rate of 10 Hz (f_s) using a cubic polynomial fit on each subset of 10 adjacent points. Sensor units having sampling frequency greater than (f_s) were down sampled to achieve a fixed sampling rate across the entire dataset.

The total dataset consists of 7,314 data points out of which 4,994 data points ($n_{training}$) were randomly selected for training, 1,070 (n_{val}) for validation and an equal number of data points for testing ($n_{test} = n_{val}$).

B. Optimisation

The coefficient matrices of our baseline state-space model were obtained using regression analysis. This analysis was done on the training dataset and Levenberg-Marquardt algorithm was used for its optimisation. For each regression, a system parameter was selected and it was put as the output variable f in (3). After this, the least square analysis was performed and the best fit coefficients were obtained. This step was repeated to

obtain coefficient matrices for system parameters commonly used in controller design. Usually, the output variable f is a system parameter that cannot be directly measured using any sensor but has an important role in the design of a controller for AUVs.

Our MLP model has a single hidden layer and a rectifier as an activation function. It used Gradient Descent (GD) optimiser to minimise the residual sum of squares (RSS) between the model's prediction f_{neural} and the observed dynamics. We also used RSS as a metric to determine MLP's performance on the validation and test set. In addition, the optimiser used a decaying learning rate α, given by the following equation:

$$\alpha = \alpha_0 e^{-kn} \tag{6}$$

where, α_0 is the initial learning rate and n is the current iteration number. Also, all the input features $\boldsymbol{y}_{v,\eta_e,u}$ were scaled between 0 and 1 to maintain consistency across all the features for the entire dataset. For effective termination of training process, the idea of using patience interval is discussed in [3]. This parameter is basically the number of iterations to go further in training before stopping it and look for a better performance. In our training, whenever a low RSS value is recorded, the number of iterations is extended by adding this parameter.

Other than number of iterations, the initialisation of variables is an important consideration in optimising neural networks. The standard method for this step is to initialise weights and biases randomly from a zero-mean Gaussian distribution [12]. With small learning rate α and random initialisation, it is possible for the training performance to get stuck at a local minima. On the other hand, a higher learning rate may result in completely missing the global optima. Therefore, to solve this problem we used a two-phase initialisation process. In the first phase, the weights and biases are initialised using a zero-mean Gaussian and trained with a relatively average learning rate α_{01}. The α_{01} value is obtained from some preliminary tests on the training dataset and takes a value between very low and very high learning rate relevant to training dataset. The training process is allowed to run using the decaying learning rate given by (6) until it is terminated by exceeding the patience interval. Before entering into the second phase, the weights and biases corresponding to the best performance during the initial training are restored. This restoration is required because the first phase termination happens only when the network is not able to perform beter within the patience interval and therefore, the last iteration's weights and biases do not correspond to the best performance or the lowest RSS score. At the beginning of the second phase, the weights and biases are initialised using restored values. After this second initialisation, the training is performed again using a decaying learning rate with its initial value α_{02} lower than the previous learning rate α_{01}. This type of two-phase learning process ensures that the initial weights and biases come close enough to a global minima and then critical updates are made during the second run to find the best performance. For the implementation of our network, we have used Google's open source library, TensorFlow [1], for numerical computations.

TABLE I. RESIDUAL SUM OF SQUARES FOR ACCELERATION PARAMETERS

S. No.	RSS Score on Normalized Test Dataset		
	System parameter	*MLP*	*State Space*
1.	Yaw Acceleration	4.2	19.2
2.	Pitch Acceleration	5.2	11.0
3.	Roll Acceleration	1.8	6.5

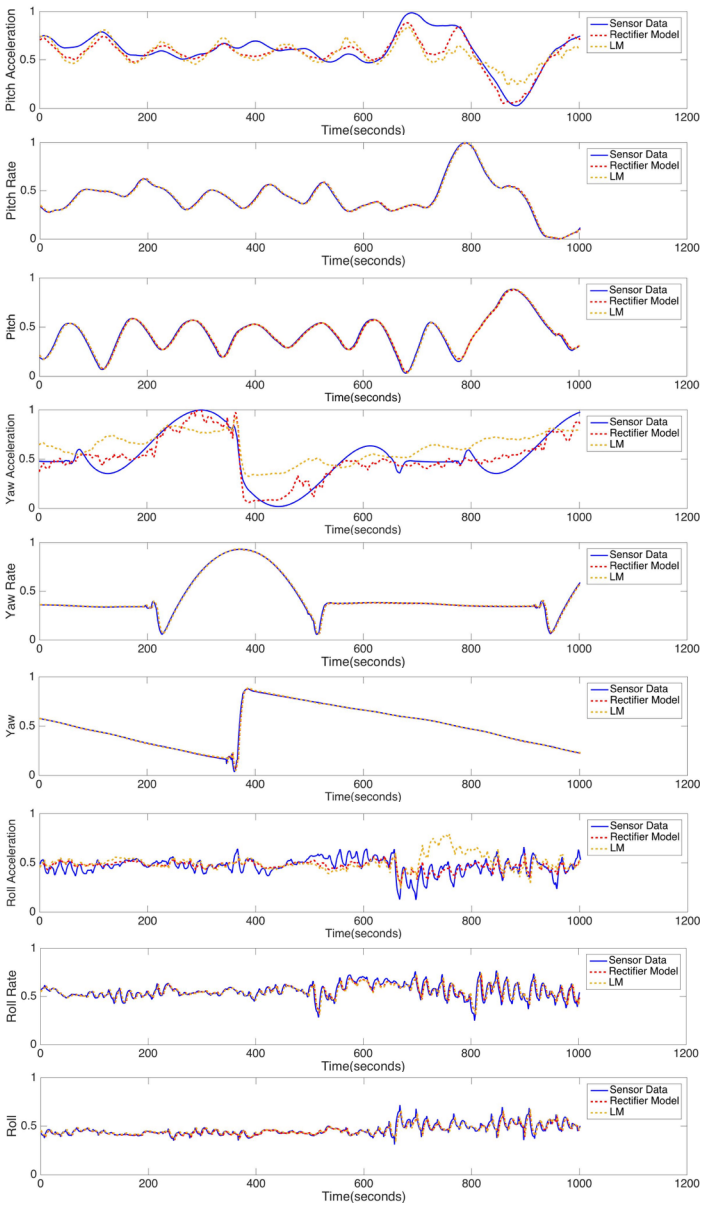


Fig. 3: Performance comparison of our MLP network with a state-space model in predicting normalized pitch, roll and yaw dynamics on test dataset. Except for the acceleration values, all other parameters were present as input features for both the models and therefore, their values were easy to predict.

C. Performance

We trained the baseline model and our rectifier network using the training set and used RSS as a performance metric. For our MLP models, we used 2,500 hidden units (N) and a mini-batch size of 1,000 samples for training. The learning rate α_{01} was set to 1×10^{-4} and α_{02} to 1×10^{-6} with the constant k as 0.96 (See Section IV.B). These learning rate values were decided

after some initial experimental runs using the training dataset. The patience interval was set to 10,000 iterations. The rectifier MLP model has 13 input features as mentioned in Section IV-A. Both the state-space and MLP models, are trained to predict the system parameter one time-step ahead. In real time, this one time-step ahead prediction is equivalent to estimating the value 0.2s ahead of the current timestamp. The test dataset performance for both the models is presented in Fig. 3 and Table 1 gives the details about the RSS value for those system parameters which were not present as input state variables.

As can be clearly observed from Fig. 3, our MLP network and state-space model gave promising performance in predicting orientation and angular velocities. However, MLPs significantly outperformed state-space models in predicting acceleration state variables. Intriguingly, these are also the parameters which were not present as input features to our models. Therefore, no prior information was available about their current state when both models were trying to predict the variable's future value. This difference in performance is strong evidence that MLPs are capable of learning the underlying non-linear dynamics whereas, state-space models are only able to provide rough estimates. On further inspection using Table 1, the performance of both models can be compared using RSS value obtained from the test dataset. This again demonstrates that the MLP model learned a better representation of acceleration variables as compared to the state-space model. Therefore, it can be concluded from these results that state-space models can only perform well if sufficient information about their output variable is already present as a state input. However, the MLP model is able to predict unknown as well as known dynamics accurately and therefore, can be considered ideal for modelling complex AUV dynamics.

V. Summary

In this work, we defined a rectifier activated MLP network for learning AUV dynamics. We also developed a linear dynamics system using the state-space equation and used it as a baseline model. These models were tested for predicting system parameters for an AUV with a vector thruster. From our results, it can be easily observed that MLP's performance is either better or at least as good as our baseline model's. Interestingly, the baseline model gives a decent performance only when the output variable is present as a state input. Whereas, the MLP model accurately predicts all the system parameters irrespective of their presence as an input feature. Therefore, the MLP model is shown to be a better choice over a linear dynamics model, and capable of modelling complex AUV dynamics. Lastly, we also discussed briefly the methods for optimising our model's learning process.

Acknowledgment

The authors would like to thank the STARFISH AUV team, especially Eng You Hong, Ashish Raste and Hari Vishnu from the Acoustic Research Laboratory (TMSI), National University of Singapore for their help in this work.

References

[1] M. Abadi et al., "TensorFlow: LargeScale Machine Learning on Heterogeneous Distributed Systems," arXiv:1603.04467, 2015.

[2] P. Abbeel, V. Ganapathi and A. Y. Ng, "Learning vehicular dynamics with application to modelling helicopters," 18th Advancements in Neural Information Processing Symposium, 2006.

[3] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," K.-R. Muller, G. Montavon, and G. B. Orr, editors, Neural Networks: Tricks of the Trade, Springer, 2013.

[4] Y. H. Eng and M. Chitre, "Roll Control of an Autonomous Underwater Vehicle Using an Internal Rolling Mass," Springer Tracts in Advanced Robotics, vol. 105, pp. 229-242, 2015.

[5] T. Fossen, Guidance and Control of Ocean Vehicles, New York: Wiley, 1994

[6] X. Glorot, A. Bordes, and Y. Bengio, "Deep Sparse Rectifier Networks," Artificial Intelligence and Statistics Conference, pp. 315-323, 2011.

[7] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber, "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies," S. C. Kremer and J. F. Kolen, editors, A Field Guide to Dynamical Recurrent Neural Networks, IEEE Press, 2001.

[8] K. Hornik, M. Stinchcombe and H. White, "Multilayer feedforward networks are universal approximators," Neural Networks, vol. 2, pp. 359-366, 1989.

[9] B. Jalving, "The NDRE-AUV flight control system," IEEE Journal of Ocean Engineering, vol. 19, no. 4, pp. 497-501, 1994.

[10] K. Jarrett, M. Kavukcuoglu, M. Ranzato and Y. LeCun, "What is the best multi-stage architecture for object recognition?," International Conference on Computer Vision, 2009.

[11] A. Krizhevsky, I. Sutskever and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," Advances in neural information processing systems, pp. 1097-1105, 2012.

[12] Y. LeCun, L. Bottou, G. B. Orr, and K. R. Muller, "Efficient BackProp," Neural Networks: Tricks of the Trade, (G. B. Orr and K.-R. Muller, eds.), pp. 9-50, Springer,1998.

[13] A. L. Maas, A. Y. Hannun and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," International conference on machine learning, 2013.

[14] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado and J. Dean, "Distributed representations of words and phrases and their compositionality," Advances in Neural Information Processing Systems, pp. 3111-3119, 2013.

[15] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," Proc. 27th International Conference on Machine Learning, 2010.

[16] M. N. Polkinghorne, G. N. Roberts, R. S. Burns and D. Winwood, "The implementation of fixed rulebase fuzzy logic to the control of small surface ships," Control Engineering Practice, vol.3, pp. 321-328, 1994.

[17] A. Punjani and P. Abbeel, "Deep Learning Helicopter Dynamics Models," Proc. IEEE International Conference on Robotics and Automation (ICRA), 2015.

[18] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Internal representation by backpropating errors," Nature, vol. 323, pp. 533-536, 1986

[19] SNAME, "Nomenclature for treating the motion of a submerged body through a fluid," The Society of Naval Architects and Marine Engineers, Technical and Research Bulletin, 1950.

[20] R. Sutton, G. N. Roberts and S. R. Dearden, "Design study of a fuzzy controller for ship roll stabilization," Electronics and Communications Engineering Journal, vol. 1, pp. 159-166, 1989.

[21] K. P. Tee and S. S. Ge, "Control of Fully Actuated Ocean Surface Vessels Using a Class of Feedforward Approximators," IEEE Transactions on Control Systems Technology, vol. 14, no. 4, pp. 750-756, 2006.

[22] M. D. Zeiler, M. Ranzato, R. Monga, M. Mao, K. Yang, Q. V. Le, P. Nguyen, A. Senior, V. Vanhoucke, J. Dean, and G. E. Hinton, "On rectified linear units for speech processing," International Conference on Acoustics, Speech and Signal Processing, 2013.

[23] Y. S. Yang, C. Zhou and J. S. Ren, "Model reference adaptive robust fuzzy control for ship steering autopilot with uncertain nonlinear systems," Applied Soft Computing, vol. 3, pp. 305-316, 2003.

[24] J. E. Bartels & D. Jürgens, The Voith Schneider Propeller - Current Application and New Developments, Heidenheim: Voith Publication, 2006