Feature and Architecture Selection on Deep Feedforward Network for Roll Motion Time Series Prediction

Novri Suhermi[1](✉), Suhartono[1], Santi Puteri Rahayu[1], Fadilla Indrayuni Prastyasari[2], Baharuddin Ali[3], and Muhammad Idrus Fachruddin[4]

[1] Department of Statistics, Institut Teknologi Sepuluh Nopember, Kampus ITS Sukolilo, Surabaya 60111, Indonesia
novri@statistika.its.ac.id
[2] Department of Marine Engineering, Institut Teknologi Sepuluh Nopember, Kampus ITS Sukolilo, Surabaya 60111, Indonesia
[3] Indonesian Hydrodynamic Laboratory, Badan Pengkajian Dan Penerapan Teknologi, Surabaya 60111, Indonesia
[4] GDP Laboratory, Jakarta 11410, Indonesia

Abstract. The neural architecture and the input features are very substantial in order to build an artificial neural network (ANN) model that is able to perform a good prediction. The architecture is determined by several hyperparameters including the number of hidden layers, the number of nodes in each hidden layer, the series length, and the activation function. In this study, we present a method to perform feature selection and architecture selection of ANN model for time series prediction. Specifically, we explore a deep learning or deep neural network (DNN) model, called deep feedforward network, an ANN model with multiple hidden layers. We use two approaches for selecting the inputs, namely PACF based inputs and ARIMA based inputs. Three activation functions used are logistic sigmoid, tanh, and ReLU. The real dataset used is time series data called roll motion of a Floating Production Unit (FPU). Root mean squared error (RMSE) is used as the model selection criteria. The results show that the ARIMA based 3 hidden layers DNN model with ReLU function outperforms with remarkable prediction accuracy among other models.

Keywords: ARIMA · Deep feedforward network · PACF · Roll motion
Time series

1 Introduction

Artificial neural network (ANN) is one of nonlinear model that has been widely developed and applied in time series modeling and forecasting [1]. The major advantages of ANN models are their capability to capture any pattern, their flexibility form, and their free assumption property. ANN is considered as universal approximator such that it is able to approximate any continuous function by adding more nodes on hidden layer [2–4].

Many studies recently developed a more advanced architecture of neural network called deep learning or deep neural network (DNN). One of the basic type of DNN is a feedforward network with deeper layers, i.e. it has more than one hidden layer in its architecture. Furthermore, it has also been shown by several studies that DNN is very promising for forecasting task where it is able to significantly improve the forecast accuracy [5–10].

ANN model has been widely applied in many fields, including ship motion study. The stability of a roll motion in a ship is a critical aspect that must be kept in control to prevent the potential damage and danger of a ship such as capsizing [11]. Hence, the ship safety depends on the behavior of the roll motion. In order to understand the pattern of the roll motion, it is necessary to construct a model which is able to explain its pattern and predict the future motion. The modeling and prediction can be conducted using several approaches. One of them is time series model.

Many researches have frequently applied time series models to predict roll motion. Nicolau et al. [12] have worked on roll motion prediction in a conventional ship by applying time series model called artificial neural network (ANN). The prediction resulted in remarkable accuracy. Zhang and Ye [13] used another time series model called autoregressive integrated moving average (ARIMA) to predict roll motion. Khan et al. [14] also used both ARIMA and ANN models in order to compare the prediction performance from each model. The results showed that ANN model outperformed compared to ARIMA model in predicting the roll motion. Other researches have also shown that ANN model is powerful and very promising as its results in performing roll motion prediction [15, 16].

Another challenge in performing neural network for time series forecasting is the input or feature selection. The input used in neural network time series modeling is its significant lag variables. The significant lags can be obtained using partial autocorrelation function (PACF). We may choose the lags which have significant PACF. Beside PACF, another potential technique which is also frequently used is obtaining the inputs from ARIMA model [17]. First, we model the data using ARIMA. The predictor variables of ARIMA model is then used as the inputs of the neural network model. In this study, we will explore one of DNN model, namely deep feedforward network model, in order to predict the roll motion. We will perform feature selection and architecture selection of the DNN model to obtain the optimal architecture that is expected to be able to make better prediction. The architecture selection is done by tuning the hyperparameters, including number of hidden layers, the number of hidden nodes, and the activation function. The results of the selection and its prediction accuracy are then discussed.

2 Time Series Analysis: Concept and Methods

Time series analysis aims to analyze time series data in order to find the pattern and the characteristics of the data where the application is for forecasting or prediction task [18]. Forecasting or prediction is done by constructing a model based on the historical data and applying it to predict the future value. Contrast with regression model where it consists of response variable(s) Y and predictor variable(s) X, time series model uses the variable itself as the predictors. For instance, let Y_t a time series response variable at time t, then the predictor variables would be $Y_{t-1}, Y_{t-2}, Y_{t-3}, Y_{t-4}$. The variables $Y_{t-1}, Y_{t-2}, Y_{t-3}, Y_{t-4}$ are also called lag variables. There are many time series models that have been developed. In this section, we present two time series models which have been widely used for forecasting task, namely autoregressive integrated moving average (ARIMA) and artificial neural network (ANN). We also present several important and mandatory concepts in order to understand the idea of time series model. They are autocorrelation and partial autocorrelation.

2.1 Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF)

Let Y_t a time series process, the correlation coefficient between Y_t and Y_{t-k} is called autocorrelation at lag k, denoted by ρ_k, where ρ_k is a function of k only, under weakly stationarity assumption. Specifically, autocorrelation function (ACF) ρ_k is defined as follows [19]:

$$\rho_k = \frac{\mathrm{Cov}(Y_t, Y_{t-k})}{\mathrm{Var}(Y_t)}. \tag{1}$$

Hence, the sample autocorrelation is defined as follows:

$$\hat{\rho}_k = \frac{\sum_{t=k+1}^{T} (Y_t - \bar{Y})(Y_{t-k} - \bar{Y})}{\sum_{t=1}^{T} (Y_t - \bar{Y})^2}. \tag{2}$$

Then, partial autocorrelation function (PACF) is defined as the autocorrelation between Y_t and Y_{t-k} after removing their mutual linear dependency on the intervening variabels $Y_{t-1}, Y_{t-2}, \ldots, Y_{t-k+1}$. It can be expressed as $\mathrm{Corr}(Y_t, Y_{t-k} \mid Y_{t-1}, Y_{t-2}, \ldots, Y_{t-k+1})$. PACF in stationary time series is used to determine the order of autoregressive (AR) model [20]. The calculation of sample partial autocorrelation is done recursively by initializing the value of partial autocorrelation at lag 1, $\hat{\phi}_{11} = \hat{\rho}_1$. Hence, the value of sample correlation at lag k can be obtained as follows [21, 22]:

$$\hat{\phi}_{k,k} = \frac{\hat{\rho}_k - \sum_{j=1}^{k-1} \hat{\phi}_{k-1,j}\hat{\rho}_{k-j}}{1 - \sum_{j=1}^{k-1} \hat{\phi}_{k-1,j}\hat{\rho}_j}. \tag{3}$$

PACF is used to determined the order p of an autoregressive process, denoted by AR (p), one of the special case of ARIMA(p, d, q) process, where $d = 0$ and $q = 0$. It is also used to determined the lag variables which are chosen as the inputs in ANN model [23].

2.2 The ARIMA Model

Autoregressive integrated moving average (ARIMA) is the combination of autoregressive (AR), moving average (MA), and the differencing processes. General form of ARIMA(p, d, q) model is given as follows [19]:

$$(1 - B)^d Y_t = \mu + \frac{\theta_q(B)}{\phi_p(B)} a_t, \tag{4}$$

where:

- $\phi_p(B) = 1 - \phi_1 B - \phi_2 B^2 - \cdots - \phi_p B^p$
- $\theta_q(B) = 1 - \theta_1 B - \theta_2 B^2 - \cdots - \theta_q B^q$
- $BY_t = Y_{t-1}$

Y_t denotes the actual value, B denotes the backshift operator, and a_t denotes the white noise process with zero mean and constant variance, $a_t \sim \mathrm{WN}(0, \sigma^2)$. $\phi_i(i = 1, 2, .., p)$, $\theta_j(j = 1, 2, \ldots, q)$, and μ are model parameters. d denotes the differencing order. The process of building ARIMA model is done by using Box-Jenkins procedure. Box-Jenkins procedure is required in order to identify p, d, and q; the order of ARIMA model, estimate the model parameters, check the model diagnostics, select the best model, and perform the forecast [24].

2.3 Artificial Neural Network

Artificial neural network (ANN) is a process that is similar to biological neural network process. Neural network in this context is seen as a mathematical object with several assumptions, among others include information processing occurs in many simple elements called neurons, the signal is passed between the neurons above the connection links, each connection link has a weight which is multiplied by the transmitted signal, and each neuron uses an activation function which is then passed to the output signal. The characteristics of a neural network consist of the neural architecture, the training algorithm, and the activation function [25]. ANN is a universal approximators which can approximate any function with high prediction accuracy. It is not required any prior assumption in order to build the model.

There are many types of neural network architecture, including feedforward neural network (FFNN), which is one of the architecture that is frequently used for time series forecasting task. FFNN architecture consist of three layers, namely input layer, hidden layer, and output layer. In time series modeling, the input used is the lag variable of the data and the output is the actual data. An example of an FFNN model architecture consisting of p inputs, a hidden layer consisting of m nodes connecting to the output, is shown in Fig. 1 [26].

The mathematical expression of the FFNN is defined as follows [27]:

$$f(\mathbf{x}_t, \mathbf{v}, \mathbf{w}) = g_2\left\{\sum\nolimits_{j=1}^{m} v_j g_1\left[\sum\nolimits_{i=1}^{n} w_{ji} x_{it}\right]\right\}, \tag{5}$$

where $\mathbf{w}$ is the connection weight between the input layer and the hidden layer, $\mathbf{v}$ is the connection weight between the hidden layer and the output layer, $g_1(.)$ and $g_2(.)$ are the

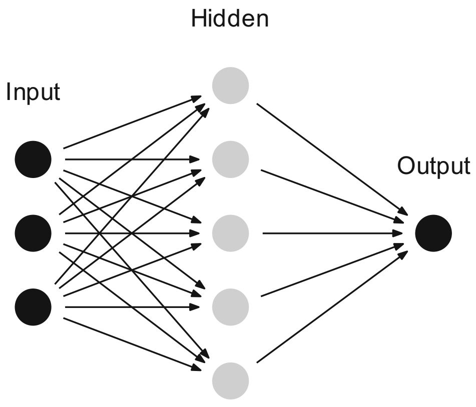

Fig. 1. The example of FFNN architecture.

activation functions. There are three activation functions that are commonly used, among others include logistic function, hyperbolic tangent (tanh) function, and rectified linear units (ReLU) function [28–30] The activations functions are given respectively as follows:

$$g(x) = \frac{1}{1+e^{-x}}, \tag{6}$$

$$g(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}, \tag{7}$$

$$g(x) = \max(0, x). \tag{8}$$

2.4 Deep Feedforward Network

Deep feedforward network is a feedforward neural network model with deeper layer, i.e. it has more than one hidden layer in its architecture. It is one of the basic deep neural network (DNN) model which is also called deep learning model [31]. The DNN aims to approximate a function f^*. It finds the best function approximation by learning the value of the parameters θ from a mapping $y = f(x;\theta)$. One of the algorithm which is most widely used to learn the DNN model is stochastic gradient descent (SGD) [32]. The DNN architecture is presented in Fig. 2. In terms of time series model, the relationship between the output Y_t and the inputs $Y_{t-1}, Y_{t-2}, \ldots, Y_{t-p}$ in a DNN model with 3 hidden layers is presented as follows:

$$Y_t = \sum_{i=1}^{s} \alpha_i g\left(\sum_{j=1}^{r} \beta_{ij} g\left(\sum_{k=1}^{q} \gamma_{jk} g\left(\sum_{l=1}^{p} \theta_{kl} Y_{t-l}\right)\right)\right) + \varepsilon_t, \tag{9}$$

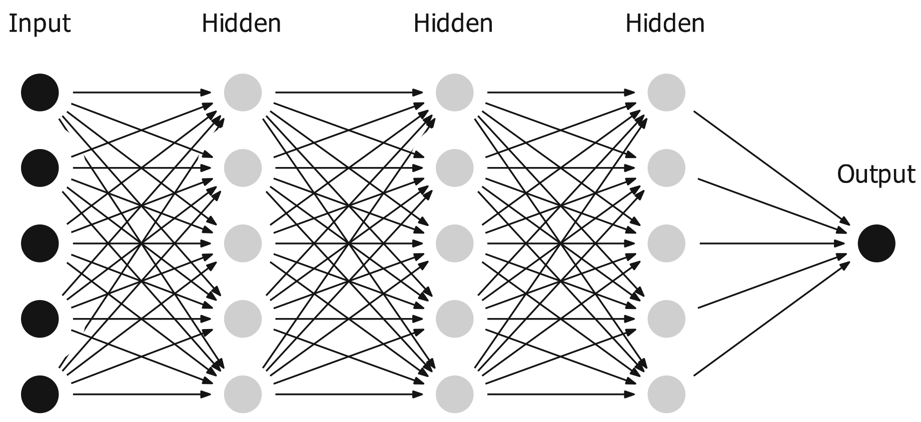

Fig. 2. The Example of DNN architecture.

where ε_t is the error term, $\alpha_i(i = 1, 2, \ldots, s)$, $\beta_{ij}(i = 1, 2, \ldots, s; j = 1, 2, \ldots, r)$, $\gamma_{jk}(j = 1, 2, \ldots, r; k = 1, 2, \ldots, q)$, and $\theta_{kl}(k = 1, 2, \ldots, q; l = 1, 2, \ldots, p)$ are the model parameters called the connection weights, p is the number of input nodes, and q, r, s are the number of nodes in the first, second, and third hidden layers, respectively. Function $g(.)$ denotes the hidden layer activation function.

3 Dataset and Methodology

3.1 Dataset

In this study, we aim to model and predict the roll motion. Roll motion is one of ship motions, where ship motions consist of six types of motion, namely roll, yaw, pitch, sway, surge, and heave, which are also called as 6 degrees of freedom (6DoF). Roll is categorized as a rotational motion. The dataset used in this study is a roll motion time series data of a ship called floating production unit (FPU). It is generated from a simulation study conducted in Indonesian Hydrodynamic Laboratory. The machine recorded 15 data points in every one second. The total dataset contains 3150 data points. Time series plot of the data set is presented in Fig. 3.

3.2 Methodology

In order to obtain the model and predict the dataset, we split the data into there parts, namely training set, validation set, and test set. The training set which consists of 2700 data points is used to train the DNN model. The next 300 data points is set as validation set which is used for hyperparameter tuning. The remaining dataset is set as test set which is used to find the best model with the highest prediction accuracy (Fig. 4). In

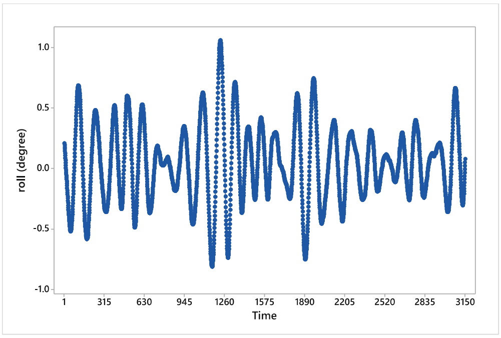

Fig. 3. Time series plot of roll motion.

order to calculate the prediction accuracy, we use root mean squared error (RMSE) as the criteria [33]. RMSE formula is given as follows:

$$\mathrm{RMSE} = \sqrt{\frac{1}{L}\sum_{l=1}^{L}\left(Y_{n+l} - \hat{Y}_n(l)\right)^2}, \tag{10}$$

where L denotes the out-of-sample size, Y_{n+l} denotes the l-th actual value of out-of-sample data, and $\hat{Y}_n(l)$ denotes the l-th forecast.

The steps of feature selection and architecture selection is given as follows:

$Y_1, Y_2, \ldots, Y_{2700}$	$Y_{2071}, Y_{2702}, \ldots, Y_{3000}$	$Y_{3001}, \ldots, Y_{3150}$
Training Set	Validation Set	Test Set

Fig. 4. The structure of dataset.

1. Feature selection based on PACF and ARIMA model, the significant lags of PACF and ARIMA model are used as the inputs in DNN model.
2. Hyperparameter tuning using grid search, where we use all combinations of the hyperparameters, including number of hidden layer: {1, 2, 3}, number of hidden nodes: [1,200], activation function: {logistic sigmoid, tanh, reLU}. The evaluation uses the RMSE of validation set.
3. Predict the test set using the optimal models which are obtained from the hyperparameters tuning.
4. Select the best model based on RMSE criteria.

4 PACF Based Deep Neural Network Implementation

4.1 Preliminary Analysis

Our first approach is using PACF of the data for choosing the lag variables that will be set as the input on the neural network model. At first, we have to guarantee that the series satisfies stationarity assumption. We conduct several unit root tests, namely Augmented Dickey-Fuller (ADF) test [34], Phillips-Perron (PP) test [35], and Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test [36, 37], which are presented in Table 1. By using significant level $\alpha = 0.05$, ADF test and PP test resulted in the p-values are below 0.01, which conclude that the series is significantly stationary. KPSS test also resulted in the same conclusion.

Table 1. Stationarity test using ADF test, PP test, and KPSS test.

Test	Test statistic	P-value	Result
ADF	−9.490	0.01	Stationary
PP	−35.477	0.01	Stationary
KPSS	0.030	0.10	Stationary

4.2 Feature Selection

Based on the PACF on Fig. 5, it shows the plot between the lag and the PACF value. We can see that the PACFs of lag 1 until lag 12 are significant, where the values are beyond the confidence limit. Hence, we will use the lag 1, lag 2, …, lag 12 variables as the input of the model.

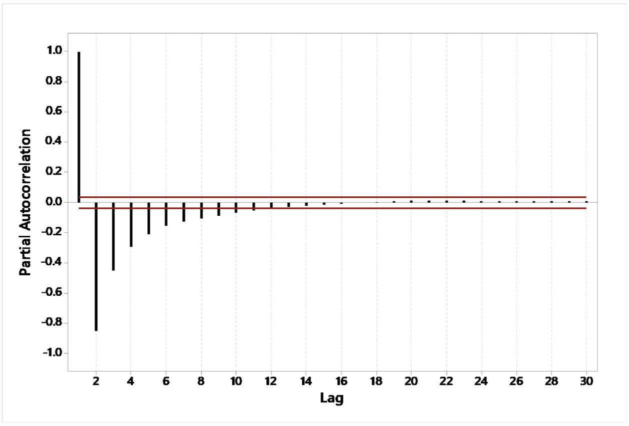

Fig. 5. PACF plot of roll motion series.

4.3 Optimal Neural Architecture: Hyperparameters Tuning

We perform grid search algorithm in order to find the optimal architecture by tuning the neural network hyperparameters, including number of hidden layers, number of hidden nodes, and activation functions [38]. Figure 6 shows the pattern of RMSE with respect to the number of hidden nodes from each architecture. From the chart, it is apparent that there is a significant decay as a result of increasing the number of hidden nodes. We can also see that the increase of number of hidden layers affects the stability of the RMSE decrease. For instance, it can be seen that the RMSEs of 1 hidden layer model with logistic function is not sufficiently stable. When we add more hidden layers, it significantly minimize the volatility of the RMSEs. Hence, we obtain the best architectures with minimal RMSE which is presented in Table 2.

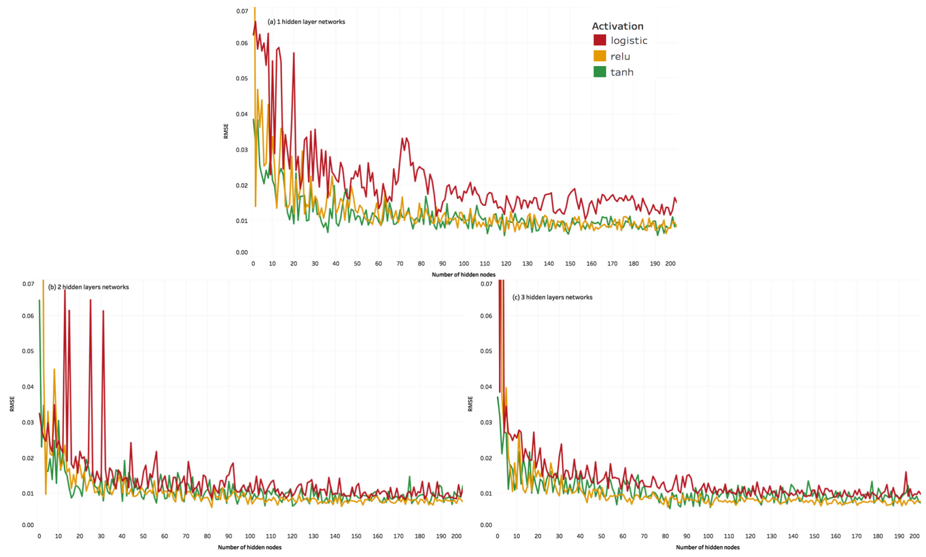

Fig. 6. The effect of number of hidden nodes to RMSE.

4.4 Test Set Prediction

Based on the results of finding the optimal architectures, we then apply theses architectures in order to predict the test set. We conduct 150-step ahead prediction which can be seen in Fig. 7 and we calculate the performance of the models based on the RMSE criteria. The RMSEs are presented in Table 3. The results show that 2 hidden layers model with ReLU function outperforms among other models. Unfortunately, the models with logistic function are unable to follow the actual data pattern such that the RMSEs are the lowest. Furthermore, the performance of the models with tanh function are also promising for the predictions are able to follow the actual data pattern although ReLU models are still the best.

Table 2. Optimal architectures of PACF based DNN.

Activation function	Number of hidden layers	Number of hidden nodes
Logistic	1	156
Logistic	2	172
Logistic	3	179
Tanh	1	194
Tanh	2	81
Tanh	3	80
ReLU	1	118
ReLU	2	119
ReLU	3	81

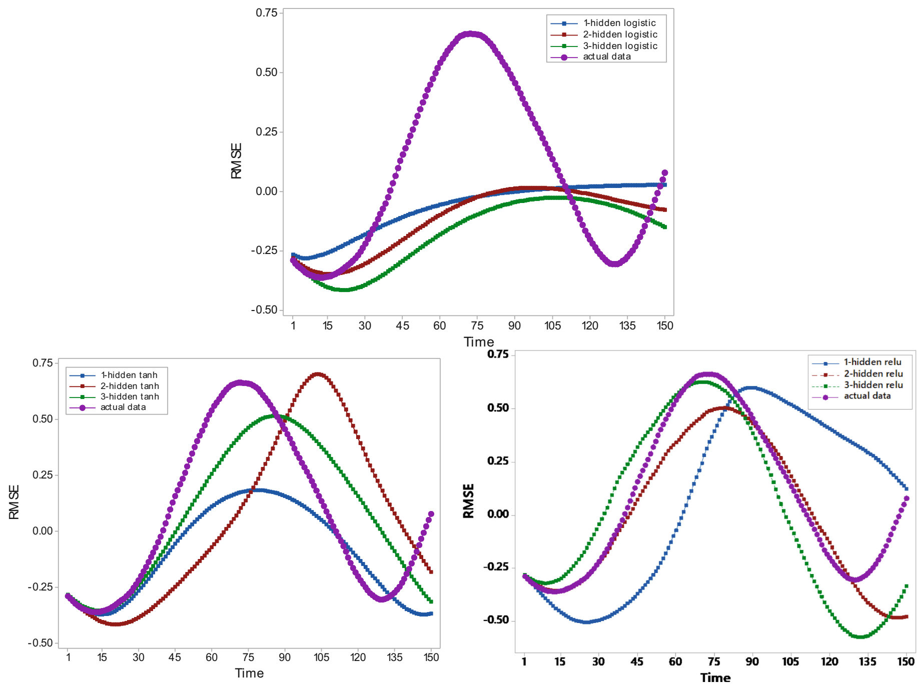

Fig. 7. Test set prediction of PACF based DNN models.

5 ARIMA Based Deep Neural Network

5.1 Procedure Implementation

We also use ARIMA model as another approach to choose the input for the model. The model is obtained by applying Box-Jenkins procedure. We also perform backward elimination procedure in order to select the best model where all model parameters are

Table 3. The RMSE of test set prediction of PACF based DNN models.

Architecture	RMSE
1-hidden layer logistic	0.354
2-hidden layers logistic	0.360
3-hidden layers logistic	0.407
1-hidden layer tanh	0.252
2-hidden layers tanh	0.406
3-hidden layers tanh	0.201
1-hidden layer ReLU	0.408
2-hidden layers ReLU	0.150
3-hidden layers ReLU	0.186

significant and it satisfies ARIMA model assumption which is white noise residual. Then, the final model we obtain is ARIMA ([1–4, 9, 19, 20], 0, [1, 9]) with zero mean. Thus, we set our DNN inputs based on the AR components of the model, namely $\{Y_{t-1}, Y_{t-2}, Y_{t-3}, Y_{t-4}, Y_{t-9}, Y_{t-19}, Y_{t-20}\}$.

We then conduct the same procedure as we have done in Sect. 4. The results are presented in Table 4 and Fig. 8.

Table 4. The RMSE of test set prediction of ARIMA based DNN models.

Architecture	RMSE
1-hidden layer logistic	0.218
2-hidden layers logistic	0.222
3-hidden layers logistic	0.512
1-hidden layer tanh	0.204
2-hidden layers tanh	0.274
3-hidden layers tanh	0.195
1-hidden layer ReLU	0.178
2-hidden layers ReLU	0.167
3-hidden layers ReLU	0.125

6 Discussion and Future Works

In Sects. 4 and 5, we see how the input features, the hidden layers, and the activation function affect the prediction performance of DNN model. In general, it can be seen that the DNN models are able to predict with good performance such that the prediction still follow the data pattern, except for the DNN model with logistic sigmoid function. In Fig. 7, it is shown that PACF based DNN models with logistic sigmoid function failed to follow the test set pattern. It also occured to the 3-hidden layers ARIMA based DNN model with logistic sigmoid function, as we can see in Fig. 8. Surprisingly, the models are significantly improved when we only used 1 or 2 hidden layers. The model

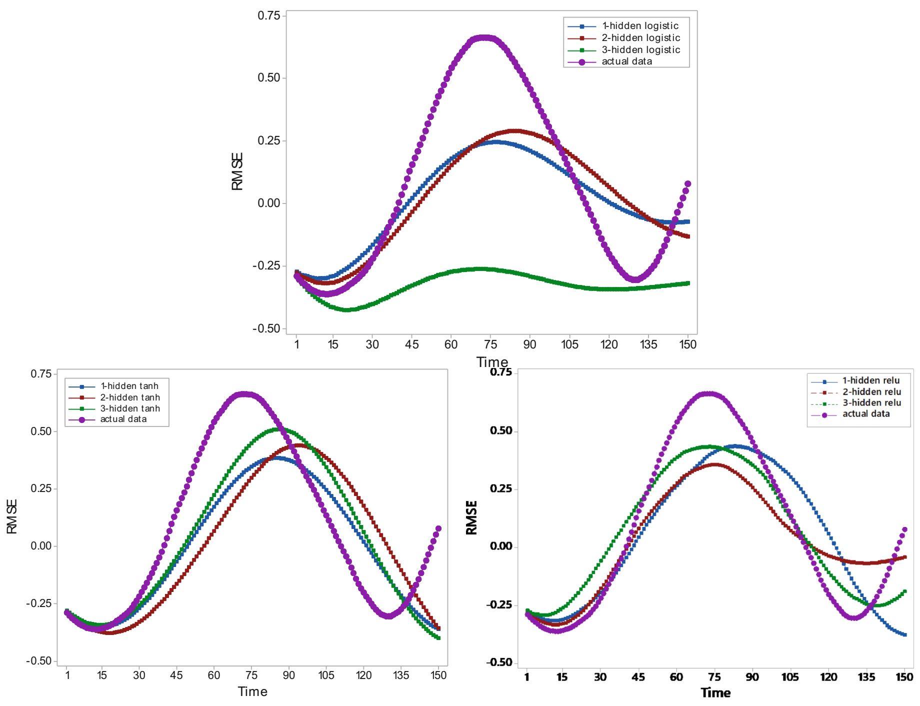

Fig. 8. Test set prediction of ARIMA based DNN.

suffers from overfitting when we used 3 hidden layers. In contrast, the other models with tanh function and ReLU function tend to outperform when we use more layers, as we can see in Tables 3 and 4. It is also shown that in average, ReLU function shows better performance, compared to other activation functions. Gensler et al. [39] also showed the same results where ReLU function outperformed than tanh function for forecasting using deep learning. Ryu et al. [40] also obtained the results that DNN with multiple hidden layers can be easily trained using ReLU function because of its simplicity; and performs better than simple neural network with one hidden layer.

Based on the input features, our study shows that the input from ARIMA model performs better than PACF based inputs. In fact, ARIMA based model has less features than the PACF based model. It is considered that adding more features in the neural input does not necessarily increase the prediction performance. Hence, it is required to choose correct inputs to obtain the best model. In time series data, using inputs based on ARIMA model is an effecctive approach to build the DNN architecture.

Based on the results of our study, it is considered that deep learning model is a promising model in order to handle time series forecasting or prediction task. In the future works, we suggest to apply feature and architecture selection for other advanced deep learning models such as long short term memory (LSTM) network. In order to prevent overfitting, it is also suggested to conduct regularization technique in the DNN architecture, such as dropout, L1, and L2.

Acknowledgements. This research was supported by ITS under the scheme of "Penelitian Pemula" No. 1354/PKS/ITS/2018. The authors thank to the Head of LPPTM ITS for funding and to the referees for the useful suggestions.

References

1. Zhang, G., Patuwo, B.E., Hu, M.Y.: Forecasting with artificial neural networks. Int. J. Forecast. **14**, 35–62 (1998)
2. Cybenko, G.: Approximation by superpositions of a sigmoidal function. Math. Control Sig. Syst. **2**, 303–314 (1989)
3. Funahashi, K.I.: On the approximate realization of continuous mappings by neural networks. Neural Netw. **2**, 183–192 (1989)
4. Hornik, K., Stinchcombe, M., White, H.: Multilayer feedforward networks are universal approximators. Neural Netw. **2**, 359–366 (1989)
5. Chen, Y., He, K., Tso, G.K.F.: Forecasting crude oil prices: a deep learning based model. Proced. Comput. Sci. **122**, 300–307 (2017)
6. Liu, L., Chen, R.C.: A novel passenger flow prediction model using deep learning methods. Transp. Res. Part C: Emerg. Technol. **84**, 74–91 (2017)
7. Qin, M., Li, Z., Du, Z.: Red tide time series forecasting by combining ARIMA and deep belief network. Knowl.-Based Syst. **125**, 39–52 (2017)
8. Qiu, X., Ren, Y., Suganthan, P.N., Amaratunga, G.A.J.: Empirical mode decomposition based ensemble deep learning for load demand time series forecasting. Appl. Soft Comput. **54**, 246–255 (2017)
9. Voyant, C., et al.: Machine learning methods for solar radiation forecasting: a review. Renew. Energy. **105**, 569–582 (2017)
10. Zhao, Y., Li, J., Yu, L.: A deep learning ensemble approach for crude oil price forecasting. Energy Econ. **66**, 9–16 (2017)
11. Hui, L.H., Fong, P.Y.: A numerical study of ship's rolling motion. In: Proceedings of the 6th IMT-GT Conference on Mathematics, Statistics and its Applications, pp. 843–851 (2010)
12. Nicolau, V., Palade, V., Aiordachioaie, D., Miholca, C.: Neural network prediction of the roll motion of a ship for intelligent course control. In: Apolloni, B., Howlett, Robert J., Jain, L. (eds.) KES 2007. LNCS (LNAI), vol. 4694, pp. 284–291. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74829-8_35
13. Zhang, X.L., Ye, J.W.: An experimental study on the prediction of the ship motions using time-series analysis. In: The Nineteenth International Offshore and Polar Engineering Conference (2009)
14. Khan, A., Bil, C., Marion, K., Crozier, M.: Real time prediction of ship motions and attitudes using advanced prediction techniques. In: Congress of the International Council of the Aeronautical Sciences, pp. 1–10 (2004)
15. Wang, Y., Chai, S., Khan, F., Nguyen, H.D.: Unscented Kalman Filter trained neural networks based rudder roll stabilization system for ship in waves. Appl. Ocean Res. **68**, 26–38 (2017)
16. Yin, J.C., Zou, Z.J., Xu, F.: On-line prediction of ship roll motion during maneuvering using sequential learning RBF neural networks. Ocean Eng. **61**, 139–147 (2013)
17. Zhang, G.P.: Time series forecasting using a hybrid ARIMA and neural network model. Neurocomputing. **50**, 159–175 (2003)

18. Makridakis, S., Wheelwright, S.C., Hyndman, R.J.: Forecasting: Methods and Applications. Wiley, Hoboken (2008)
19. Wei, W.W.S.: Time Series Analysis: Univariate and Multivariate Methods. Pearson Addison Wesley, Boston (2006)
20. Tsay, R.S.: Analysis of Financial Time Series. Wiley, Hoboken (2002)
21. Durbin, J.: The fitting of time-series models. Revue de l'Institut Int. de Statistique/Rev. Int. Stat. Inst. **28**, 233 (1960)
22. Levinson, N.: The wiener (root mean square) error criterion in filter design and prediction. J. Math. Phys. **25**, 261–278 (1946)
23. Liang, F.: Bayesian neural networks for nonlinear time series forecasting. Stat. Comput. **15**, 13–29 (2005)
24. Box, G.E.P., Jenkins, G.M., Reinsel, G.C., Ljung, G.M.: Time Series Analysis: Forecasting and Control. Wiley, Hoboken (2015)
25. Fausett, L.: Fundamentals of Neural Networks: Architectures, Algorithms, and Applications. Prentice-Hall, Inc., Upper Saddle River (1994)
26. El-Telbany, M.E.: What quantile regression neural networks tell us about prediction of drug activities. In: 2014 10th International Computer Engineering Conference (ICENCO), pp. 76–80. IEEE (2014)
27. Taylor, J.W.: A quantile regression neural network approach to estimating the conditional density of multiperiod returns. J. Forecast. **19**, 299–311 (2000)
28. Han, J., Moraga, C.: The influence of the sigmoid function parameters on the speed of backpropagation learning. In: Mira, J., Sandoval, F. (eds.) IWANN 1995. LNCS, vol. 930, pp. 195–201. Springer, Heidelberg (1995). https://doi.org/10.1007/3-540-59497-3_175
29. Karlik, B., Olgac, A.V.: Performance analysis of various activation functions in generalized MLP architectures of neural networks. Int. J. Artif. Intell. Expert Syst. **1**, 111–122 (2011)
30. Nair, V., Hinton, G.E.: Rectified linear units improve restricted Boltzmann machines. In: Proceedings of the 27th International Conference on International Conference on Machine Learning. pp. 807–814. Omnipress, Haifa (2010)
31. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press, Cambridge (2016)
32. LeCun, Y.A., Bottou, L., Orr, G.B., Müller, K.-R.: Efficient BackProp. In: Montavon, G., Orr, G.B., Müller, K.-R. (eds.) Neural Networks: Tricks of the Trade. LNCS, vol. 7700, pp. 9–48. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-35289-8_3
33. De Gooijer, J.G., Hyndman, R.J.: 25 years of time series forecasting. Int. J. Forecast. **22**, 443–473 (2006)
34. Fuller, W.A.: Introduction to Statistical Time Series. Wiley, Hoboken (2009)
35. Phillips, P.C.B., Perron, P.: Testing for a Unit Root in Time Series Regression. Biometrika **75**, 335 (1988)
36. Hobijn, B., Franses, P.H., Ooms, M.: Generalizations of the KPSS-test for stationarity. Stat. Neerl. **58**, 483–502 (2004)
37. Kwiatkowski, D., Phillips, P.C.B., Schmidt, P., Shin, Y.: Testing the null hypothesis of stationarity against the alternative of a unit root. J. Econ. **54**, 159–178 (1992)
38. Hsu, C.W., Chang, C.C., Lin, C.J.: A practical guide to support vector classification. Presented at the (2003)
39. Gensler, A., Henze, J., Sick, B., Raabe, N.: Deep Learning for solar power forecasting — an approach using autoencoder and LSTM neural networks. In: 2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp. 002858–002865. IEEE (2016)
40. Ryu, S., Noh, J., Kim, H.: Deep neural network based demand side short term load forecasting. In: 2016 IEEE International Conference on Smart Grid Communications (SmartGridComm), pp. 308–313. IEEE (2016)