Multi-task learning using non-linear autoregressive models and recurrent neural networks for tide level forecasting

Nerfita Nikentari, Hua-Liang Wei
Department of Automatic Control and Systems Engineering, Faculty of Engineering, University of Sheffield, Sheffield, United Kingdom

Article Info

Article history:

Received May 5, 2023
Revised Aug 22, 2023
Accepted Sep 6, 2023

Keywords:

Forecasting
Multi-task learning
Nonlinear autoregressive moving average with exogenous model
Recurrent neural network
Tide level

ABSTRACT

Tide level forecasting plays an important role in environmental management and development. Current tide level forecasting methods are usually implemented for solving single task problems, that is, a model built based on the tide level data at an individual location is only used to forecast tide level of the same location but is not used for tide forecasting at another location. This study proposes a new method for tide level prediction at multiple locations simultaneously. The method combines nonlinear autoregressive moving average with exogenous inputs (NARMAX) model and recurrent neural networks (RNNs), and incorporates them into a multi-task learning (MTL) framework. Experiments are designed and performed to compare single task learning (STL) and MTL with and without using non-linear autoregressive models. Three different RNN variants, namely, long short-term memory (LSTM), gated recurrent unit (GRU) and bidirectional LSTM (BiLSTM) are employed together with non-linear autoregressive models. A case study on tide level forecasting at many different geographical locations (5 to 11 locations) is conducted. Experimental results demonstrate that the proposed architectures outperform the classical single-task prediction methods.

Corresponding Author:

Hua-Liang Wei
Department of Automatic Control and Systems Engineering, Faculty of Engineering, University of Sheffield
Amy Johnson Building, Portobello Street, Sheffield, S1 3 JD, United Kingdom
Email: w.hualiang@sheffield.ac.uk

1. INTRODUCTION

Tide level could have significant impact on human life, and the study of ocean phenomena is an essential part of coastal engineering, coastal ecosystem, and human activity. In the field of coastal engineering, tide level data are valuable for the construction of ports, offshores and cross-sea bridges [1]–[4]. For coastal ecosystems, tide level data can be crucial for sediment movements and pollutant tracing and monitoring [1], [5]. In the domain of human activity, tide level data can be used as important information in fishing, doing recreational activities [6], and potentially developing tidal energy [7], [8]. Therefore, it is important to effectively model and forecast tide levels. In doing so, tide level data are usually observed and recorded as time series. A classical way to make tide level forecasting is by implementing harmonic analysis method. Such a traditional way of forecasting can be ineffective if the data are incomplete (e.g., with some data being missing) [9]. Harmonic analysis methods usually also demand a substantial amount of parameters because such a method needs to use not only astronomical but also non-astronomical features [2], [10]. To overcome these drawbacks, an alternative forecasting method is required.

Various algorithms and models have been proposed and explored to improve the accuracy of time series forecasting results. One of the most popular approaches for forecasting is using neural networks. Forecasting using artificial neural networks (ANNs), combined with other models, has attracted extensive attention. Specifically, nonlinear autoregressive with exogenous input model (NARX) and nonlinear autoregressive moving average with exogenous input model (NARMAX) have been widely applied to complex system identification, modelling and time series forecasting [11]. For example, Aguirre *et al.* [12] employed both nonlinear autoregressive (NAR) and multi-layer perceptron models to investigate two fundamental issues which underlie periodic time series forecasting tasks (e.g., daily load forecasting): pattern mapping and dynamical prediction; the results are interesting and useful for designing more effective predictive approaches for short-term periodic time series forecasting. Wu *et al.* [13] proposed a combination of genetic algorithm, backpropagation and NARX for tide level prediction. Muñoz and Acuña [14] developed NARX and NARMAX models combined with shallow and deep neural networks (DNNs) to forecast daily demand data and air quality conditions. The performance of NARMAX and radial basis function (RBF) neural networks was examined by Omri *et al.* [15] for water flow depth forecasting. Gu *et al.* [16] proposed a neural network enhanced NARMAX model to predict the disturbance storm time (Dst) index and the model showed better performance than either NARX model and a typical neural network model alone. A novel cloud-NARX model is presented by Gu *et al.* [17] for the auroral electrojet (AE) index forecasting and prediction uncertainty analysis.

The aforementioned methods, combining ANNs and other models, demonstrate promising results for time series prediction. However, most of these methods are designed for single task learning (STL), where a model trained using a set of time series data can only be used to make prediction of the same time series process, but cannot be used for other time series predictions, and therefore cannot benefit from sharing knowledge among related tasks [18], [19]. In many real scenarios, two or more different events or processes may be closely associated with each other; therefore, it is desirable to design a new framework that can be used to deal with more than one different but similar datasets simultaneously.

To extend STL models to multi-task learning (MTL) cases, this paper proposes a new MTL framework by combining nonlinear aggressive models and recurrent neural network (RNN). The proposed MTL framework performs multiple tasks simultaneously, allowing for sharing information between related tasks. For RNN, an MTL scheme can be implemented by sharing information of the structures of the network models (e.g., numbers of layers and numbers of nodes in each layer) and their training process. Sharing information between multiple related tasks can boost learning efficiency and improve the generalization ability of the resulting models. These performance improvements are achieved through knowledge sharing between different tasks [19]–[21].

Several MTL RNN models have been proposed for forecasting purposes in the literature. In study [22], MTL and long short-term memory (LSTM) RNN models were combined for wind power forecasting; the prediction accuracy was increased by more than 23.13% in comparison with the existing STL forecasting models. In study [23], MTL and RNN were used to forecast urban traffic flow; the forecasting accuracy was improved around 10% to 15% over baseline models. In study [18], another RNN variant, gated recurrent unit (GRU), was combined with MTL for traffic flow and speed forecasting; the model does not only improve the forecasting accuracy but also solve the problems caused by enlarging the dataset. In study [24], the performance of MTL with LSTM (MTL+LSTM) was compared with that of MTL+GRU for health assessment and remaining useful life forecasting; it shows interesting results where LSTM gives smaller loss and simpler model while GRU can perform well with less training time. MTL based on bidirectional LSTM (BiLSTM) was proposed to forecast cooling, heating, and electric load [25]; the MTL+BiLSTM model is able to increase the accuracy significantly and improve the time efficiency for training the model.

Over the past years, NARX and NARMAX models have been applied to STL, and RNN has been introduced to solve multi task learning problems, but relatively few works have been done to combine nonlinear model autoregressive models and RNNs, and applied them to MTL. This study aims to develop new models for tide level forecasting. The proposed approach is as follows. It first combines three RNN variants (LSTM, GRU, and BiLSTM) with NARX and NARMAX, respectively. Then, it compares the performance of all the resulting models. Finally, it determines the best models for tide level forecasting by evaluating the accuracy of these models, with and without using the proposed MTL scheme.

The study conducts comprehensive comparisons between three types of recurrent neural networks (LSTM, GRU, and BiLSTM), paired with NARX and NARMAX models, and analyze their performances for forecasting tide level at different locations. The main contributions of the work are summarized as follow:

a. The proposal of NARMAX and NARX modelling framework, combined with an MTL scheme, for forecasting tide levels at many different locations simultaneously.
b. A comparison studies of NARMAX and NARX, paired with three RNN model structures with MTL schemes, to improve tide level prediction performances.

2. METHOD

2.1. Multi-task learning

MTL is a transfer learning method where multiple related tasks are solved in parallel by sharing information between them. The task relatedness is defined based on the recognition of how each task is related, based on which MTL models are designed, trained and implemented. The classic methods to perform MTL can be categorized into soft parameter sharing and hard parameter sharing.

MTL with hard parameter sharing concept is by using the hidden layer together for entire tasks but the output layers will be allocated separately for different tasks. Hard parameter technique shows the possibility of lower chance of overfitting and smaller loss or error because the MTL is able to hold all the knowledge and information, sharing all the parameters and training all tasks jointly [26]–[28]. In soft parameter sharing, each task has it is own specific hidden layer with independent parameters. The distance between the model parameters of different tasks is then regularized to make the parameter to be similar. Although every task has it is own model with its own setting, the distance between the model parameters of every dissimilar task is added to unite the objective functions [26], [27]. Several approaches have been proposed to explore the sharing mechanism in MTL, for example, supervised learning, unsupervised learning, semi-supervised learning, reinforcement learning, active learning, online learning, parallel and distributed learning, and multi-view learning [29].

2.2. NARX and NARMAX models

NARX and NARMAX are models that not only utilize exogenous (external) input variables, but also the lagged versions of the system's own output that enters the model structure through output delays [11]. Both models introduce nonlinear functions to learn the system input-output relationships but the NARMAX model also includes prediction errors to improve the modelling performance [17]. The introduction of "error variable" or "residual variable" makes NARMAX more powerful for nonlinear system identification [30]. The most commonly used basis functions in NARX and NARMAX modelling are polynomials, which usually lead to transparent, interpretable and parsimonious models [11], [31], [32]. Another widely used nonlinear representation to apply with NARX and NARMAX is neural networks, which usually result in black box models. A modelling framework combining NARX or NARMAX and neural networks may be called grey box model identification [11], [33]. For single input, STL systems, NARX model can be formulated as (1):

$$y(t) = f\left(y(t-1), \dots, y(t-n_y), u(t-d), \dots, u(t-d-n_u)\right) + e(t) \tag{1}$$

where $y(t)$, $u(t)$ and $e(t)$ are the system output, input, and noise, respectively; n_y, n_u, n_e are the maximum lags in the output, input and noise; $f(.)$ is a nonlinear function, and *d* is a time delay which is typically set to be *d*=0 or *d*=1. For multi task leaning systems, NARX can be formulated as (2):

$$\begin{aligned} y_i\,(t) = \; & f_i\,(y_1(t-1), \dots, y_1(t-n_y), \dots, y_m(t-1), y_m(t-n_y), u_1(t-1), \dots, u_1(t-n_u), \\ & \dots, u_r(t-1), \dots, u_r(t-n_u)\,) \end{aligned} \tag{2}$$

where $i = 1, 2 \dots, m$, with m being the number of outputs; *r* is the number of inputs. Correspondingly, for STL and MTL systems, NARMAX models can be respectively formulated as (3):

$$\begin{aligned} y\,(t) = \; & f\left(y(t-1), \dots, y(t-n_y), u(t-d), \dots, u(t-n_u), e(t-1), e(t-2) \dots, e(t - \right. \\ & \left. n_e)\right) + e(t) \end{aligned} \tag{3}$$

and

$$\begin{aligned} y_i\,(t) = \; & f_i(y_1(t-1), \dots, y_1(t-n_y), \dots, y_m(t-1), \dots, y_m(t-n_y), u_1(t-1), \dots, u_1(t-n_u), \\ & \dots, u_r(t-1), \dots, u_r(t-n_u), e_1(t-1), \dots, e_1(t-n_e), \dots, e_m(t-1), \dots, e_m(t-n_e)\,) \end{aligned} \tag{4}$$

In this work, three variants of RNN, that is, LSTM, Bi-LSTM and GRU, are used to implement the nonlinear functions *f* and f_i (*i* =1, 2, …, *m*) and adopt the hard parameter sharing MTL framework.

2.3. The NARX-RNN-MTL frameworks

The structure of the proposed NARX-RNN-MTL framework is presented in Figure 1, where $\hat{y}_i\,(t)$ $(i = 1,2, \dots, m)$ are the predicted values of the output, n and r are the number of samples and number of locations, respectively. The proposed model has an input layer that accepts the sequence data of tide level measured at r locations; the next layer is built using RNN. The RNN layer can be GRU, LSTM or BiLSTM, and the last layer, which is a fully connected dense layer with the output, provides predicted values of the tide levels.

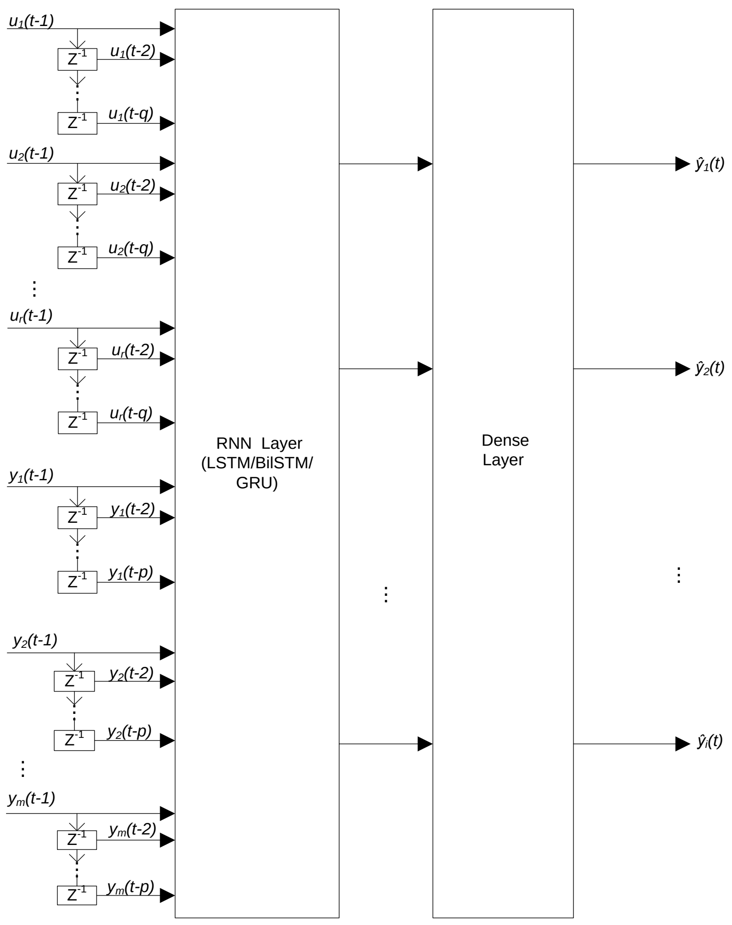

Figure 1. NARX RNN architecture

2.4. The NARMAX-RNN-MTL frameworks

Similar to the NARX-RNN-MTL model, the NARMAX-RNN-MTL model is also implemented by using GRU, LSTM or BiLSTM and the forecasting results are produced by the dense layer. The difference between these two models is that the NARMAX architecture includes 'noise' as an input. Note that noise cannot be measured but can be estimated using model prediction error. The architecture of the NARMAX-RNN-MTL model is presented in Figure 2.

To identify an NARMAX model, we use the prediction error from NARX-RNN-MTL as additional inputs. To differentiate between noise and prediction error, we use $e(t)$ and $\varepsilon(t)$ to represent noise and model prediction error, respectively. The model prediction error of the NARX-RNN-MTL model is:

$$\varepsilon_i(t) = y_i(t) - \hat{y}_i\,(t) \tag{5}$$

where $\hat{y}_i\,(t)$ is the one-step ahead prediction calculated from NARX-RNN-MTL model and $y_i(t)$ is the corresponding actual signal. For an MTL system with r inputs and m outputs, the i^{th} output of the NARMAX-RNN-MTL model calculated based on the associated NARX-RNN-MTL model is:

$$y_i\,(t) = f_i\big(y_1(t-1), \dots, y_1(t-n_y), \dots, y_m(t-1), \dots, y_m(t-n_y), u_1(t-1), \dots, u_1(t-n_u), \dots, u_r(t-1), \dots, u_r(t-n_u), \varepsilon_1(t-1), \dots, \varepsilon_1(t-n_e), \dots, \varepsilon_i(t-1), \dots, \varepsilon_i(t-n_e)\big)\; i = 1, \dots, m \tag{6}$$

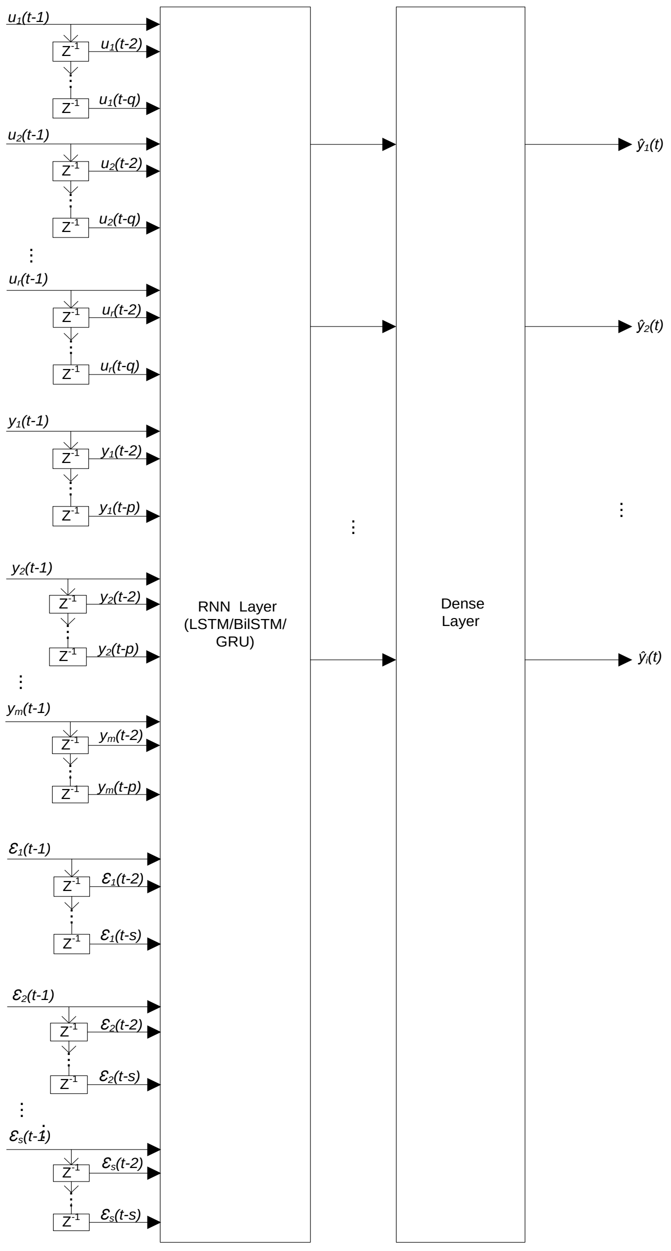

Figure 2. NARMAX RNN architecture

3. EXPERIMENTS

3.1. The task

This study is concerned with a multi-task problem, that is, to forecast tide level in five stations (5 tasks) at the same time, namely, Harwich, Lerwick, Millport, Portrush and Weymouth, using the proposed NARX-RNN-MTL and NARMAX-RNN-MTL models. The MTL model is designed for one-step ahead prediction of tide level; here one-step is equal to 15 minutes. The datasets measured for the five stations, from January 1, 2022 to May 31, 2022, with a sampling period of 15 minutes, are used for model building. In doing so, the data were split into three parts: 60% for training, 20% for validation and 20% for testing.

Experiment on different numbers of more tasks will be further performed. Specifically, the proposed models are used to forecast six more stations, namely, Aberdeen, Devonport, Fishguard, Holyhead, St Mary and Stornoway, to further assess the model's generalization performances for solving many tasks, ranging from 6 to 11. The datasets for these six stations were measured in the same period and with the same sampling period of 15 minutes. These datasets were extracted from the website of the British Oceanographic Data Centre (BODC) [34].

3.2. Experimental settings and metrics performance

To achieve good model performances, the network hyper-parameters were determined through simulations by testing a set of parameters shown in Table 1. For a single task, for example, the prediction of a single individual time series $y_i(t)$ (i =1,2,…, m), without using shared information from any other signals, the loss function can be defined as (7):

$$L_i = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(\hat{y}_i(t) - y_i(t))^2} \tag{7}$$

where $\hat{y}_i$ is the predicted value, y_i is actual value and n is the number of observations.

Note that this study is concerned with dealing with multiple tasks simultaneously; the loss function for model training should accommodate the losses of all the tasks. Keeping this in mind, we use the averaged root mean square error (aRMSE) as a joint loss that is defined as (8):

$$L = \frac{1}{m}\sum_{i=1}^{m} L_i \tag{8}$$

Table 1. The parameter setting

Parameter	Values
Number of hidden layer	1
Number of nodes	25,50, 100, 150,200,250,300,350
Optimizer	Adam, SGDM
Number of iterations	300
Maximum lags for NARX	n_y = 1,2,3,4 n_u = 1,2,3,4
Maximum lags for NARMAX	n_y = 1,2,3,4 n_u = 1,2,3,4 n_e = 1

3.3. Experimental results

3.3.1. NARX and NARMAX combine with GRU, LSTM, and BiLSTM

We compare NARX and NARMAX using three different RNN models, with a variety of time lags. Two optimizers, namely stochastic gradient descent with a momentum (SGDM) and adaptive moment estimation (ADAM), are used to optimize the associated models. Joint loss value or average value of root mean square error (RMSE), based on the combinations of the candidate parameters presented in Table 1, was simultaneously carried out on the datasets for the five stations (i.e., Harwich, Lerwick, Millport, Portrush and Weymouth) for finding the optimal network model parameters and determined the best model structure. The performance of the NARX-RNN-MTL and NARMAX-RNN-MTL models were then compared based on the values of the RMSE of five tasks. Three RNN structures, namely, LSTM, BiLSTM and GRU were used for building the NARX-RNN-MTL and NARMAX-RNN-MTL models.

We compare the performance of the three NARX-RNN-MTL models, with 16 specific lags and trained with SGDM and ADAM, for the five tasks. The comparison is based on joint loss or average RMSE value which indicates the performance of each model. The details of these experimental settings are shown in Table 2. From the comparison we have the following findings:

a. The two lowest average RMSE values are 0.10598 and 0.15848, which are produced by the model of NARX GRU using SGDM and ADAM, respectively.
b. The distribution of average RMSE for NARX GRU and NARX LSTM using SGDM are relatively small compared to NARX GRU and NARX LSTM using ADAM. From Table 2, it can be seen that the minimum and maximum average values of RMSE for NARX GRU and LSTM GRU using SGDM are 0.10598, 0.15848, 0.15774 and 0.21189 respectively, while for NARX LSTM and NARX LSTM using ADAM, the values are 0.15848, 0.25016, 0.17758 and 0.35830, respectively.
c. NARX GRU using SGDM outperforms the following models for all the settings of time lags (ranging from 1 to 4): i) GRU using ADAM, ii) LSTM using SGDM and ADAM, and iii) BiLSTM using SGDM and ADAM.
d. For NARX BiLSTM, the relatively lower RMSE distribution is gained by using the ADAM optimizer.

Table 3 shows details of the 16 resulting models using three NARMAX-RNN-MTL models, with 16 specific lags and trained with the two optimizers, SGDM and ADAM, for the five tasks. Similar to the NARX-RNN-MTL results, the RMSE range distribution of the NARMAX-GRU and NARMAX LSTM models using SGDM is smaller than using ADAM, but the range becomes significantly wider for BiLSTM implemented with SGDM. Furthermore, NARMAX GRU trained with SGDM outperforms other models. From Table 3, we have the following observations: SGDM works better than ADAM for both NARMAX-GRU and NARMAX-LSMT models, whereas ADAM outperforms SGDM for the NARMAX-BiLSMT model.

Table 2. Performances of identified 16 NARX-RNN-MTL models

No	Lag delay	Joint loss (Average RMSE)					
		SGDM			ADAM		
		GRU	LSTM	BiLSTM	GRU	LSTM	BiLSTM
1	$(n_y = 1, n_u = 1)$	**0.10598**	0.17322	0.28982	0.16231	0.19635	0.18830
2	$(n_y = 1, n_u = 2)$	0.10649	0.18470	0.28459	0.21822	0.17758	0.28459
3	$(n_y = 1, n_u = 3)$	0.14729	0.16285	0.30918	0.22208	0.27188	0.19872
4	$(n_y = 1, n_u = 4)$	0.12999	0.16768	0.30962	0.23365	0.27584	0.17331
5	$(n_y = 2, n_u = 1)$	0.11040	0.15774	0.34582	0.19442	0.21047	0.15283
6	$(n_y = 2, n_u = 2)$	0.11865	0.15775	0.31986	0.19442	0.24187	0.20332
7	$(n_y = 2, n_u = 3)$	0.13333	0.17340	0.29100	0.22796	0.26231	0.17221
8	$(n_y = 2, n_u = 4)$	0.13932	0.19142	0.28496	0.24223	0.34013	0.20563
9	$(n_y = 3, n_u = 1)$	0.12164	0.16396	0.28633	0.18606	0.29282	0.17917
10	$(n_y = 3, n_u = 2)$	0.13305	0.18853	0.28329	0.23771	0.25633	0.22868
11	$(n_y = 3, n_u = 3)$	0.11662	0.17910	0.30352	0.21835	0.24891	0.25139
12	$(n_y = 3, n_u = 4)$	0.15127	0.19995	0.31023	0.25016	0.24891	0.20282
13	$(n_y = 4, n_u = 1)$	0.14329	0.18544	0.30231	0.21711	0.28983	0.20301
14	$(n_y = 4, n_u = 2)$	0.13971	0.18291	0.32236	**0.15848**	0.35830	0.23086
15	$(n_y = 4, n_u = 3)$	0.15848	0.16859	0.30973	0.22282	0.20589	0.21365
16	$(n_y = 4, n_u = 4)$	0.14329	0.21189	0.28626	0.21075	0.35164	0.25752

Table 3. Performance of 16 identified NARMAX-RNN-MTL models

No	Lag delay	Joint loss (Average RMSE)					
		SGDM			ADAM		
		GRU	LSTM	BiLSTM	GRU	LSTM	BiLSTM
1	$(n_y = 1, n_u = 1, n_e = 1)$	0.10191	0.18091	0.28408	**0.15780**	0.20377	0.17585
2	$(n_y = 1, n_u = 2, n_e = 1)$	**0.09961**	0.17588	0.30377	0.16879	0.20888	0.19176
3	$(n_y = 1, n_u = 3, n_e = 1)$	0.13875	0.15918	0.29990	0.18699	0.22137	0.18809
4	$(n_y = 1, n_u = 4, n_e = 1)$	0.12616	0.16064	0.32019	0.20136	0.29417	0.22080
5	$(n_y = 2, n_u = 1, n_e = 1)$	0.10972	0.16576	0.29414	0.19492	0.22167	0.20305
6	$(n_y = 2, n_u = 2, n_e = 1)$	0.10980	0.16958	0.29454	0.19750	0.24746	0.16862
7	$(n_y = 2, n_u = 3, n_e = 1)$	0.12900	0.16516	0.29947	0.21943	0.31842	0.20583
8	$(n_y = 2, n_u = 4, n_e = 1)$	0.13144	0.20250	0.26188	0.24519	0.33824	0.20687
9	$(n_y = 3, n_u = 1, n_e = 1)$	0.12864	0.15395	0.32849	0.20649	0.30592	0.22373
10	$(n_y = 3, n_u = 2, n_e = 1)$	0.12250	0.17948	0.32849	0.23217	0.30592	0.22373
11	$(n_y = 3, n_u = 3, n_e = 1)$	0.11954	0.16456	0.30606	0.23067	0.27288	0.22879
12	$(n_y = 3, n_u = 4, n_e = 1)$	0.13561	0.19274	0.28792	0.23114	0.30840	0.21352
13	$(n_y = 4, n_u = 1, n_e = 1)$	0.13431	0.21346	0.28896	0.22387	0.25997	0.19986
14	$(n_y = 4, n_u = 2, n_e = 1)$	0.14343	0.16216	0.31024	0.23185	0.29529	0.20957
15	$(n_y = 4, n_u = 3, n_e = 1)$	0.13754	0.18699	0.28547	0.23874	0.24995	0.21745
16	$(n_y = 4, n_u = 4, n_e = 1)$	0.14401	0.19065	0.28606	0.23418	0.28934	0.24220

Table 4 illustrates the optimal settings for the NARX-RNN and NARMAX-RNN model and their performances. It can be observed that the NARX-GRU and NARMAX-GRU models with smaller lags, trained with SGDM have better performances than other models. The best NARX-RNN-MTL model reported in Table 2 can be written as (9):

$$\hat{y}_i(t) = f_i\big(y_1(t-1), \dots, y_4(t-1), u_1(t-1), \dots, u_4(t-1)\big) \tag{9}$$

Similarly, the best NARMAX-RNN-MTL model reported in Table 3 can be written as (10):

$$\hat{y}_i(t) = f_i(y_1(t-1), \dots, y_4(t-1), u_1(t-1), u_1(t-2), \dots, u_4(t-1), u_4(t-2), \varepsilon_1(t-1), \dots, \varepsilon_4(t-1)) \tag{10}$$

3.3.2. Baselines

In order to validate the overall performance and effectiveness, we tested and compared the proposed method with the following baseline methods, including STL and MTL without using NARX or NARMAX model: i) MTL implemented with GRU, LSTM and BiLSTM using SGDM and ADAM optimizer and ii) STL implemented with GRU, LSTM, and BiLSTM using SGDM optimizer and ADAM. The comparison results of the individual RMSE and the lowest joint loss RMSE are tabulated in Table 5, where the lowest average RMSE value is for MTL using NARMAX and GRU with SGDM optimizer. Based on average

RMSE value this model outperforms all the baseline models. It can be noticed that this lowest average value does not guarantee that the individual RMSE of each task is also has smaller value. For example, for Tasks 1 and 5, the NARMAX-GRU-MTL model shows the best performance with the lowest individual RMSE values, while for Tasks 2, 3 and 4, GRU-STL, GRU-MTL and NARX-GRU-MTL show the best performance, respectively. Table 5 shows that all the baseline models using SGDM deliver better results than using ADAM; the SGDM optimizer has best performance measured in either average RMSE or individual RMSE. Figure 3 provides a graphical illustration of joint loss, measured as the average RMSE of different models. The bar plots show that GRU networks display better performance than the other two RNN variants (LSTM and BiLSTM). To further evaluate the performances of NARMAX-GRU models for more tasks, we conducted experiments, where six more tasks of predicting tide levels at stations 6-11 were included and performed simultaneously together with the other five tasks (for stations 1-5). The joint losses of the NARMAX-GRU models for these six tasks are shown in Figure 4, where it can be noted that the prediction error increases with the increase of the number of tasks but still maintains the errors at a stable level.

Table 4. The optimal settings of the NARX-RNN-MTL and NARMAX-RNN-MTL models and their performances

Optimizer	Model	Lag delay	Number of hidden layer	Joint loss (Average RMSE)
		NARX-RNN-MTL		
SGDM	**GRU**	$(n_y = 1, n_u = 1)$	**300**	**0.10598**
	LSTM	$(n_y = 2, n_u = 1)$	50	0.15774
	BiLSTM	$(n_y = 1, n_u = 2)$	50	0.28459
ADAM	GRU	$(n_y = 4, n_u = 2)$	25	0.15848
	LSTM	$(n_y = 1, n_u = 2)$	25	0.17758
	BiLSTM	$(n_y = 2, n_u = 1)$	25	0.15283
		NARMAX-RNN-MTL		
SGDM	**GRU**	$(n_y = 1, n_u = 2, n_e = 1)$	**100**	**0.09961**
	LSTM	$(n_y = 3, n_u = 1, n_e = 1)$	25	0.15395
	BiLSTM	$(n_y = 2, n_u = 4, n_e = 1)$	25	0.26188
ADAM	GRU	$(n_y = 1, n_u = 1, n_e = 1)$	100	0.15780
	LSTM	$(n_y = 1, n_u = 1, n_e = 1)$	25	0.20377
	BiLSTM	$(n_y = 2, n_u = 2, n_e = 1)$	25	0.16862

Table 5. Comparison of RMSE values of of different models

Model	Hidden Node	Task 1	Task 2	Task 3	Task 4	Task 5	Average
		SGDM					
NARMAX-GRU-MTL	100	**0.16140**	0.09458	0.12771	0.05048	**0.06388**	**0.09961**
NARMAX-LSTM-MTL	25	0.24487	0.12499	0.14582	0.10795	0.14610	0.15395
NARMAX-BiLSTM-MTL	25	0.42726	0.18621	0.31956	0.18871	0.18764	0.26188
NARX-GRU-MTL	300	0.21347	0.10470	0.10519	**0.04220**	0.06432	0.10598
NARX-LSTM-MTL	50	0.29863	0.13280	0.14630	0.09633	0.11466	0.15774
NARX-BiLSTM-MTL	25	0.48090	0.22515	0.36826	0.17236	0.17627	0.28459
GRU-MTL	150	0.25929	0.15322	**0.07548**	0.04405	0.10889	0.12819
LSTM-MTL	50	0.28030	0.16359	0.17418	0.10057	0.08497	0.16072
BiLSTM-MTL	25	0.52474	0.22105	0.40890	0.21039	0.20774	0.31456
GRU-STL	250	0.29145	**0.07239**	0.18870	0.09555	0.34522	0.19866
LSTM-STL	250	0.24690	0.09911	0.12101	0.10604	0.20814	0.15624
BiLSTM-STL	300	0.56596	0.28664	0.47933	0.24418	0.29645	0.37451
		ADAM					
NARMAX-GRU-MTL	100	0.19824	0.09073	0.16851	0.12965	0.20186	0.15780
NARMAX-LSTM-MTL	25	0.32049	0.09641	0.24447	0.10655	0.25092	0.20377
NARMAX-BiLSTM-MTL	25	0.25218	0.14970	0.17170	0.10981	0.15970	0.16862
NARX-GRU-MTL	25	0.19162	0.15533	0.22263	0.11317	0.10961	0.15848
NARX-LSTM-MTL	25	0.34495	0.09603	0.21576	0.09469	0.13649	0.17758
NARX-BiLSTM-MTL	25	0.23515	0.10303	0.17927	0.10553	0.14120	0.15283
GRU-MTL	50	0.26933	0.15941	0.19025	0.09281	0.15661	0.17368
LSTM-MTL	25	0.49935	0.15125	0.15912	0.10630	0.19532	0.22227
BiLSTM-MTL	50	0.30707	0.16566	0.16943	0.09655	0.15826	0.17939
GRU-STL	50	0.45873	0.19463	0.15439	0.14449	0.21260	0.23297
LSTM-STL	25	0.39998	0.17163	0.18295	0.10671	0.29542	0.23134
BiLSTM-STL	50	0.33139	0.27435	0.34500	0.24185	0.14102	0.26672

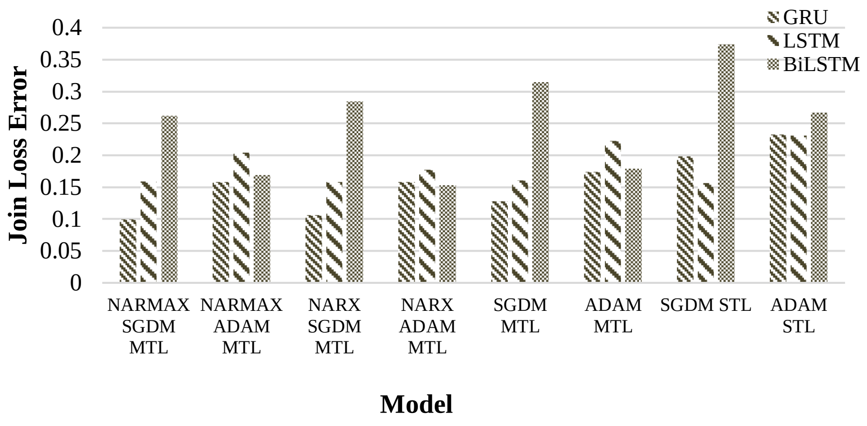

Figure 3. Comparison of joint loss error of different model structures builts based on three RNNs, namely, GRU, LSTM, and BiLSTM

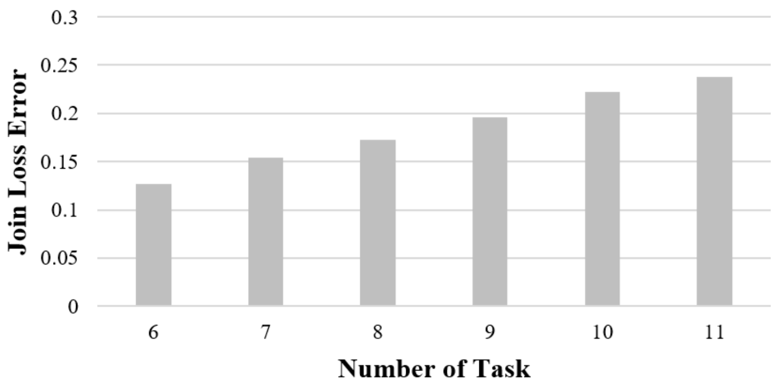

Figure 4. Joint loss error of MTL NARMAX with many tasks

4. DISCUSSION

From the results and comparisons presented in section 4, it can be noticed that NARX/NARMAX models, paired with GRU, showed better performance than paired with LSTM and BiLSTM for all the experimented scenarios. Another noticeable observation is that the SGDM optimizer was proved to be more effective than ADAM (in terms of RMSE) for both single task and multi-tasks. The only occasion where SGDM showed a poor performance was when BiLSTM was applied to solve a single task problem as shown in Table 5. The maximum lags for both NARX and NARMAX models were limited to the range from 1 to 4. It appeared that models with smaller lags usually produced slightly better prediction performances. However, it was noted that neural network models with a relatively smaller lag usually needed more hidden nodes, meaning that the training of the models needed more time. It is also worth mentioning that the network models that include NARX or NARMAX as a sub-model have lower average RMSE values in comparison with models that do not include NARX or NARMAX as a sub-model.

5. CONCLUSION

This study proposes a new class of NARMAX-RNN models, namely, NARMAX-LSTM, -BiLSTM and -GRU, combined with MTL learning for multiple tide level forecasting simultaneously. Experimental results revealed that NARMAX-GRU trained with SGDM outperformed the other two RNN variants; the NARMAX-GRU model requires relatively small lags but may need a relatively larger number of hidden nodes. The optimal NARX-GRU structure involves 300 hidden nodes, the maximum lag for input is $n_u = 1$, for output is $n_y = 1$, and the RMSE values is 0.10598. For the NARMAX-GRU model, the best setting is as follows: 100 hidden nodes, the maximum lag for input is $n_u = 2$ and for output is $n_y = 1$, and the RMSE values is 0.09961. The results showed that NARMAX-GRU outperformed its counterpart NARX-GRU.

We also compared the model performances with and without using the MTL scheme. It turned out that NARMAX-GRU has the lowest joint loss values. The three RNN models without using the MTL scheme displayed poor performance compared to NARX and NARMAX with MTL the scheme. One limitation of this work is that the proposed model still needs manual fine-tuning to find the best hyper-parameters, e.g., time lags for each of the model variables and the number of hidden nodes, to build the best models. In future, we will design MTL models that can better fine-tune the training process by using transfer learning. In addition, the data used model for model training and forecasting are not only univariate but also multivariate and multidimensional.

ACKNOWLEDGEMENTS

The authors acknowledge that this work was supported in part by NERC (Ref. NE/V001787/1), NERC (Ref. NE/V002511/1), EPSRC (Ref. EP/H00453X/1), and EPSRC (Ref. EP/I011056/1). The author gratefully acknowledges a scholarship from the Indonesia Endowment Fund for Education/*Lembaga Pengelola Dana Pendidikan* (LPDP). For the purpose of open access, the author has applied a Creative Commons Attribution (CC BY) license to any Author Accepted Manuscript version arising from this submission.

REFERENCES

[1] A. Riazi, "Accurate tide level estimation: A deep learning approach," *Ocean Engineering*, vol. 198, Feb. 2020, doi: 10.1016/j.oceaneng.2020.107013.
[2] W. Wang, H. Yuan, and M. Tan, "Application of BP neural network in monitoring of ocean tide level," in *2015 International Conference on Computational Intelligence and Communication Networks (CICN)*, Dec. 2015, pp. 1238–1240, doi: 10.1109/CICN.2015.238.
[3] C.-H. Yang, C.-H. Wu, and C.-M. Hsieh, "Long short-term memory recurrent neural network for tidal level forecasting," *IEEE Access*, vol. 8, pp. 159389–159401, 2020, doi: 10.1109/ACCESS.2020.3017089.
[4] W. Bao and W. Bin, "Real-time tide prediction based on an hybrid HA-WANN model using wind information," in *2018 14th IEEE International Conference on Signal Processing (ICSP)*, Aug. 2018, pp. 604–608, doi: 10.1109/ICSP.2018.8652339.
[5] M. A. Rizkina, D. Adytia, and N. Subasita, "Nonlinear autoregressive neural network models for sea level prediction, study case: in Semarang, Indonesia," in *2019 7th International Conference on Information and Communication Technology (ICoICT)*, Jul. 2019, pp. 1–5, doi: 10.1109/ICoICT.2019.8835307.
[6] F. Granata and F. Di Nunno, "Artificial intelligence models for prediction of the tide level in Venice," *Stochastic Environmental Research and Risk Assessment*, vol. 35, no. 12, pp. 2537–2548, Dec. 2021, doi: 10.1007/s00477-021-02018-9.
[7] M. S. Chowdhury *et al.*, "Current trends and prospects of tidal energy technology," *Environment, Development and Sustainability*, vol. 23, no. 6, pp. 8179–8194, Jun. 2021, doi: 10.1007/s10668-020-01013-4.
[8] F. O Rourke, F. Boyle, and A. Reynolds, "Tidal energy update 2009," *Applied Energy*, vol. 87, no. 2, pp. 398–409, Feb. 2010, doi: 10.1016/j.apenergy.2009.08.014.
[9] C.-H. Yang, C.-H. Wu, C.-M. Hsieh, Y.-C. Wang, I.-F. Tsen, and S.-H. Tseng, "Deep learning for imputation and forecasting tidal level," *IEEE Journal of Oceanic Engineering*, vol. 46, no. 4, pp. 1261–1271, Oct. 2021, doi: 10.1109/JOE.2021.3073931.
[10] S. Consoli, D. R. Recupero, and V. Zavarella, "A survey on tidal analysis and forecasting methods for Tsunami detection," *arXiv preprint arXiv:1403.0135*, Mar. 2014.
[11] S. A. Billings, *Nonlinear system identification: NARMAX methods in the time, frequency, and spatio-temporal domains*. John Wiley and Sons, 2013.
[12] L. A. Aguirre, D. D. Rodrigues, S. T. Lima, and C. B. Martinez, "Dynamical prediction and pattern mapping in short-term load forecasting," *International Journal of Electrical Power and Energy Systems*, vol. 30, no. 1, pp. 73–82, Jan. 2008, doi: 10.1016/j.ijepes.2007.11.001.
[13] W. Wu, L. Li, J. Yin, W. Lyu, and W. Zhang, "A modular tide level prediction method based on a NARX neural network," *IEEE Access*, vol. 9, pp. 147416–147429, 2021, doi: 10.1109/ACCESS.2021.3124250.
[14] F. Munoz and G. Acuna, "Time series forecasting using NARX and NARMAX models with shallow and deep neural networks," in *2021 IEEE Latin American Conference on Computational Intelligence (LA-CCI)*, Nov. 2021, pp. 1–6, doi: 10.1109/LA-CCI48322.2021.9769832.
[15] T. Omri, A. Karoui, D. Georges, and M. Ayadi, "Prediction of water flow depth with kinematic wave equations and NARMAX approach based on neural networks in overland flow model," in *2020 4th International Conference on Advanced Systems and Emergent Technologies (IC_ASET)*, Dec. 2020, pp. 193–198, doi: 10.1109/IC_ASET49463.2020.9318321.
[16] Y. Gu, H.-L. Wei, M. A. Balikhin, R. J. Boynton, and S. N. Walker, "Machine learning enhanced NARMAX model for dst index forecasting," in *2019 25th International Conference on Automation and Computing (ICAC)*, Sep. 2019, pp. 1–6, doi: 10.23919/IConAC.2019.8895027.
[17] Y. Gu, H.-L. Wei, R. J. Boynton, S. N. Walker, and M. A. Balikhin, "System identification and data-driven forecasting of AE index and prediction uncertainty analysis using a new Cloud-NARX model," *Journal of Geophysical Research: Space Physics*, vol. 124, no. 1, pp. 248–263, Jan. 2019, doi: 10.1029/2018JA025957.
[18] K. Zhang, L. Wu, Z. Zhu, and J. Deng, "A multitask learning model for traffic flow and speed forecasting," *IEEE Access*, vol. 8, pp. 80707–80715, 2020, doi: 10.1109/ACCESS.2020.2990958.
[19] S. Pentyala, M. Liu, and M. Dreyer, "Multi-task networks with universe, group, and task feature learning," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 820–830, doi: 10.18653/v1/P19-1079.
[20] K. G. Dizaji, W. Chen, and H. Huang, "Deep large-scale multitask learning network for gene expression inference," *Journal of Computational Biology*, vol. 28, no. 5, pp. 485–500, May 2021, doi: 10.1089/cmb.2020.0438.
[21] R. Caruana, "Multitask learning," *Machine Learning*, vol. 28, no. 1, pp. 41–75, 1997.
[22] S. Ruder, "An overview of multi-task learning in deep neural networks," *arXiv preprint arXiv:1706.05098*, Jun. 2017.
[23] J. Wei, X. Wu, T. Yang, and R. Jiao, "Ultra-short-term forecasting of wind power based on multi-task learning and LSTM,"

International Journal of Electrical Power and Energy Systems, vol. 149, Jul. 2023, doi: 10.1016/j.ijepes.2023.109073.
[24] M. Karimzadeh, S. M. Schwegler, Z. Zhao, T. Braun, and S. Sargento, "MTL-LSTM: multi-task learning-based LSTM for urban traffic flow forecasting," in *2021 International Wireless Communications and Mobile Computing (IWCMC)*, Jun. 2021, pp. 564–569, doi: 10.1109/IWCMC51323.2021.9498905.
[25] T. Wu and T. Chen, "A gated multiscale multitask learning model using time-frequency representation for health assessment and remaining useful life prediction," *Sensors*, vol. 23, no. 4, 2023, doi: 10.3390/s23041922.
[26] Y. Guo *et al.*, "BiLSTM multitask learning-based combined load forecasting considering the loads coupling relationship for multienergy system," *IEEE Transactions on Smart Grid*, vol. 13, no. 5, pp. 3481–3492, Sep. 2022, doi: 10.1109/TSG.2022.3173964.
[27] M. Crawshaw, "Multi-task learning with deep neural networks: a survey," *arXiv preprint arXiv:2009.09796*, Sep. 2020.
[28] P. Vafaeikia, K. Namdar, and F. Khalvati, "A brief review of deep multi-task learning and auxiliary task learning," *arXiv preprint arXiv:2007.01126*, Jul. 2020.
[29] Y. Zhang and Q. Yang, "An overview of multi-task learning," *National Science Review*, vol. 5, no. 1, pp. 30–43, Jan. 2018, doi: 10.1093/nsr/nwx105.
[30] J. R. A. Solares, H.-L. Wei, R. J. Boynton, S. N. Walker, and S. A. Billings, "Modeling and prediction of global magnetic disturbance in near-Earth space: A case study for K p index using NARX models," *Space Weather*, vol. 14, no. 10, pp. 899–916, Oct. 2016, doi: 10.1002/2016SW001463.
[31] R. J. Hall, H.-L. Wei, and E. Hanna, "Complex systems modelling for statistical forecasting of winter North Atlantic atmospheric variability: A new approach to North Atlantic seasonal forecasting," *Quarterly Journal of the Royal Meteorological Society*, vol. 145, no. 723, pp. 2568–2585, Jul. 2019, doi: 10.1002/qj.3579.
[32] S. Billings and H.-L. Wei, "NARMAX model as a sparse, interpretable and transparent machine learning approach for big medical and healthcare data analysis," in *2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, Aug. 2019, pp. 2743–2750, doi: 10.1109/HPCC/SmartCity/DSS.2019.00385.
[33] E. A. Jaleel and K. Aparna, "Identification of realistic distillation column using hybrid particle swarm optimization and NARX based artificial neural network," *Evolving Systems*, vol. 10, no. 2, pp. 149–166, Jun. 2019, doi: 10.1007/s12530-018-9220-5.
[34] BODC, "UK tide gauge network," *British Oceanographic Data Centre, National Oceanography Centre*. https://www.bodc.ac.uk/data/hosted_data_systems/sea_level/uk_tide_gauge_network/ (accessed Aug. 09, 2022).

BIOGRAPHIES OF AUTHORS

Nerfita Nikentari received the B.Eng. degree in informatics engineering from UPN "Veteran", Yogyakarta, Indonesia and master's degree in computer science from Universitas Gadjah Mada, Yogyakarta, Indonesia. Currently, she is a Ph.D. student in the Department of Automatic Control and Systems Engineering (ACSE), the University of Sheffield, UK. Her research interests include neural networks, deep learning, genetic algorithm, particle swarm optimization, fuzzy logic, forecasting, multi-task learning, machine learning. She can be contacted at email: nnikentari1@sheffield.ac.uk.

Hua-Liang Wei received the Ph.D. degree in the Department of Automatic Control and Systems Engineering (ACSE), the University of Sheffield, United Kingdom, in 2004. Dr. Wei is currently a Senior Lecturer (Associate Professor) with ACSE at the University of Sheffield. He is head of the Digital Medicine and Computational Neuroscience (DMCN), head of the laboratory of dynamic modelling, data mining and decision making (3DM), and Deputy Head of the Solar Physics and Space Plasma Research Centre (RP2RC). He has been awarded grants (as PI and Co-I) from a variety of funding bodies including EPSRC, NERC, EU H2020, the Royal Society and so on. He has published more than 140 peer-reviewed papers. His main research interests are in signal processing, nonlinear system identification, big data, neural networks, interpretable machine learning, deep learning, computational intelligence, data-driven modelling, data mining, pattern recognition, data based and data informed predictive modelling, fault detection and diagnosis, AI and its applications. He has been devoting to these areas for over 20 years, tackling challenging issues at the interface of data science, computer science and control and systems engineering, to develop new and fundamental methods that can be adapted to solve challenging, complex systems modelling problems in multi- and inter-disciplinary areas including engineering and industry, weather and climate, space weather, environment bio-medical and neuroscience. These are reflected in his research outputs and publications. He has a strong interest in initializing and developing new research directions that can help solve emerging challenging problems in multi-disciplinary domains. He can be contacted at w.hualiang@sheffield.ac.uk.