Multitask Learning Using Non Linear Model for Forecasting Task

Nerfita Nikentari

29th January 2025
Version:

Multitask Learning Using Non Linear Model for Forecasting Task

by

Nerfita Nikentari

A thesis submitted to

The University of Sheffield

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

29th January 2025

Nerfita Nikentari

Multitask Learning Using Non Linear Model for Forecasting Task

PhD Thesis, 29th January 2025

Supervisors: Dr Hua-Liang wei
Professor Michael Balikhin

The University of Sheffield

Department of Automatic Control and Systems Engineering

Amy Johnson Building

Portobello Street

Sheffield, S1 3JD

The best way to predict your future is to create it

Abraham Lincoln

To my parents, who will always be my biggest critic but also the best supporter I ever had.

Acknowledgement

First and foremost, I would like to thank Allah S.W.T. for letting me through all the difficulties. I have experienced your guidance day by day. You are the one who let me finish my degree. I will keep on trusting you for my future.

I would like to acknowledge and give my warmest thanks to my supervisor, Dr. Hua-Liang Wei, who made this work possible. I am extremely grateful for his invaluable advice, continuous support, and patience during my PhD study.

I would also like to give special thanks to my mom and dad for their continuous support and understanding when undertaking my research and writing my project. Your prayer for me sustained me this far.

Finally, I would like to thank all my friends in Room 316: Nathan, Alex, Ethan, Oswaldo, George, Mian, Jinni, Meng Xiao, George, Jingyu, and Junxi. Thank you for making my PhD journey less stressful. One of the best things about being PhD student is being your friend.

Declaration

I, Nerfita Nikentari, declare that the work presented in this thesis is my own. All material in this thesis which is not of my own work, has been properly accredited and referenced.

Sheffield, 29th January 2025

Nerfita Nikentari

Abstract

Multitask learning is a model that is able to solve several machine learning models by only using a single framework. Designed to enhance the performance of machine learning by simultaneously sharing useful information across multiple tasks, this model works by combining with other models. Neural networks are the most common model used to combine with multitask learning. By integrating multitask learning with the high computational power of neural networks, their performance can be improved.

The combination of neural networks and multitask learning can increase the effectiveness of the resulting models by augmenting the training set and by using different datasets for related tasks. Because the model has to learn shared information for multiple tasks, there is a lower possibility of overfitting. Another advantage of multitask learning is that it will cut down the number of networks that need to be trained, optimised, and evaluated. Therefore, multitask learning would be more cost effective in terms of computational resources. In the industrial world, this plays an important role in cost-saving when dealing with deep neural networks.

Even though this model offers promising results, multitask learning has two significant drawbacks. The first issue is that this model often suffers from negative transfer, or one of the tasks in the multitask learning result is worse than singletask learning. Another problem is that the loss gap between tasks is large or the loss distribution is unequal, and this happens because certain tasks are too dominant and degrade the other tasks.

This study experiments with four novel multitask learning models to find the model that can deliver better accuracy than singletask learning, avoid negative transfer, and manage to balance errors between tasks. All models employ nonlinear models and recurrent neural networks to build multitask learning frameworks.

The first model is multitask learning with hard parameter sharing and a non-deep neural network approach. This model improves forecasting accuracy and does not suffer from negative transfer and unequal loss distribution. The second and third models we created are multitasked learning with mixed and soft parameter sharing techniques. Both models are able to improve the accuracy and avoid negative transfer. The last model is gated multitask learning. This model also successfully improves forecasting accuracy and does not suffer from negative transfer.

In this work, multitask learning was also experimented without applying a nonlinear model. The same four models previously mentioned were used, but a nonlinear model was not added. The proposed model without a nonlinear model gives varying results. Some models outperformed singletask learning, but all failed to prevent negative transfer.

The results obtained from comparing multitask learning with a nonlinear model and without a nonlinear model show that the nonlinear model proved more effective than the model without applying a nonlinear model. All four proposed multitask learning models with a non-linear model solved all the multitask learning issues, while those without a non-linear model only overcame one.

Contents

1

Introduction

In this chapter, we describe the motivation and goals of this thesis and set the stage for the research presented in the following chapters. First, we present the study's background and motivation. Then, we state the research aims and objectives. Finally, we give research contributions, thesis outlines, and a list of publications.

1.1. Background and Motivation

Machine learning provides an important tool for solving real-life modeling problems. The two most popular machine learning problems to be solved are forecasting and classification. Traditional forecasting and classification models based on neural networks are usually designed for a singletask, which ignores the source of knowledge or information from other related tasks. These modeling tasks are usually solved separately by choosing a specific technique to find optimal solutions. Such a modeling approach is called singletask learning. The singletask learning method can only give one forecast or classification result using an available model; it requires fine-tuning other models in cases where there is a need to produce different results for a similar or related task. However, in certain cases, it is possible to simultaneously deal with several tasks/problems by exploiting the information and knowledge from other similar and related tasks [1, 2]. Such an approach is called Multitask Learning. The term multitask

describes a single model that can be used to conduct two or more tasks simultaneously [3]. The multitask learning model attempts to share useful information across tasks in parallel. The goal of multitask learning is to leverage information in various learning tasks to improve the model generalization performance [3, 4]. An effective multitask learning model is where the tasks are closely associated because it will make it more efficient while sharing data and knowledge[4].

From this perspective of performance generalization, the high computational capacity of deep neural networks can be combined with multitask learning to improve its performance. This approach has been widely applied in numerous areas of science and engineering [2, 5]. One of the benefits of neural networks is their capability to perform various data sets and endpoints concurrently. Caruana [6] argued that neural networks are suitable for multitask learning problems because neural networks can learn hidden information from the data shared between multiple tasks [3, 7]. The basic idea of combining neural networks with multitask learning is as follows: the neural networks hidden layer is shared by all tasks (targets/outputs); the training or learning process for all tasks is implemented in parallel; some hidden nodes may be specially designed for a specific task; add auxiliary output nodes for auxiliary tasks [8, 9].

Deep neural networks integrated with multitask learning offer different solutions by jointly sharing information from all tasks. The networks then train together in the shared layer, and the outcomes from different tasks are attained through different outputs [3, 7]. By sharing particular layers in the training process and multiple tasks simultaneously, deep neural networks can reduce latency during inference and give better performance results compared to singletask learning [10]. The combination of multitask learning and neural networks can also increase the effectiveness of the resulting models by augmenting the training set and using different datasets for related tasks. Because the model has to learn shared information for multiple tasks, there is a lower possibility of overfitting [11]. Another advantage of multitask learning is that it reduces the number of networks that need to be trained, optimized, and evaluated. Therefore, multitask learning would be more cost-effective in terms of computational resources. In the industrial world, this plays an important role in cost-saving when dealing with deep neural networks [3].

Various deep neural networks models have been proposed and explored to improve the accuracy of the multitask learning results. One of the most popular approaches is to combine neural networks with other models. Two promising models that can be mixed with deep neural networks are the nonlinear autoregressive with exogenous input model (NARX) and the nonlinear autoregressive moving average with exogenous input model (NARMAX). NARX

and NARMAX have been widely applied to complex system identification and modeling [12].

Over the past years, NARX and NARMAX models have been applied to singletask learning, and deep neural networks have been introduced to solve multitask learning problems, but relatively few works have been done to combine nonlinear model autoregressive models and deep neural networks, and applied them to multitask learning.

1.2. Challenge on Building Multitask Learning model

Despite numerous advantages and important progress, optimizing multitask learning to achieve higher performance and more efficiency than singletask learning remains challenging.

The first challenge is to design multitask learning Neural Networks Architecture [1, 2, 4]. Searching for an optimal architecture or designing a multitask neural network architecture that shares only the relevant features across the tasks and keeps the remaining ones task-specific can be exhaustive and computationally expensive. The model performance may be better on average overall tasks using the multitask learning setting, but for some particular tasks, because they are not closely related, multitask learning may produce unsatisfactory performance compared to the independently trained model or singletask learning. Such a detrimental effect on multitasking learning performance is called negative transfer.

The second challenge is the loss function [1, 2]. This problem is related to how to balance loss between tasks. Because multitask learning involves jointly minimizing a set of loss functions for various problems with different difficulty levels and characteristics. A multitask loss function should be able to treat all tasks equally without acknowledging one or more specific tasks to be more important than another.

The third challenge is implementing multitask learning in areas not previously studied in depth. Most prior study cases related to multitask learning are focused on solving specific classification tasks. The literature on multitask learning that can solve forecasting tasks is little studied.

1.3. Aim and Objectives

The aim of this thesis is to develop and formulate a novel multitask learning model based on machine learning and non-linear models to solve the forecasting problem for oceanic data.

Objectives of this thesis are summarised as follows:

a) Design, develop, and implement a novel multitask learning model using nonlinear system identification based on the nonlinear Autoregressive Moving Average Model with Exogenous Inputs (NARMAX) that can solve negative transfer and loss balanced between tasks in multitask forecasting.

b) The development of a novel multitask learning model using the Mixture of Experts model involves: (1) Identification of a Mixture of Experts using NARMAX model based on traditional neural network model (2) Advanced neural network model hybrid with a Mixture of Expert and NARMAX

c) Carry out an analysis of the effect of nonlinear system identification based on the NARMAX model in the multitask learning model. A comparison of proposed multitask learning models to forecasting problems built with and without NARMAX.

1.4. Research Contributions

The prime contributions gathered throughout the development of this project are summarised below:

a) A novel multitask learning model that can perform several forecast tasks using a single model.

b) A comparison of, and contrast between, the nonlinear multitask learning and the linear multitask learning approach for the forecasting problem. For this nonlinear multitask learning system, we observed that the multitask learning employed with NARMAX is the most effective solver.

c) Proved that a combination of NARMAX, neural networks, and multitask learning can improve forecasting performance, avoid negative transfer and unequal loss distribution.

1.5. Thesis Outline

The structure of this thesis, along with research contributions, is presented below:

Chapter 2 reviews the fundamental concept of multitask learning. A brief review of machine learning and the system identification approach to build and develop multitask learning is provided. The approaches of machine learning are summarised into three models: (i) multitask learning based on Hard Parameter Sharing, (ii) multitask learning based on Soft Parameter Sharing, and (iii) multitask learning based on a Mixture of Experts (MoE).

Chapter 3 A hard parameter sharing multitask learning using NARX and NARMAX based on RNN is proposed. The work presented in this chapter has been published in [13–15].

Chapter 4 presents a novel model of multitask learning by applying soft parameter sharing and NARMAX.

Chapter 5 presents a novel multitask learning model by applying the MoE model. The MoE with convolutional neural networks and NARMAX. A new approach to forecasting is proposed.

Chapter 6 draws the conclusion, summarising its findings and recommending some ideas for further research to improve and extend the proposed models.

1.6. Research Outputs

The materials from Chapter 3 have been adopted for the following paper (published):

Nerfita Nikentari and Hua-Liang Wei. 'Tide Level Prediction Using NARX-based Recurrent Neural Networks'. In: 2022 27th International Conference on Automation and Computing (ICAC). 2022, pp. 1–6 (cit. on p. 4).

Nerfita Nikentari and Hua-Liang Wei. 'Multitask Learning for Time Series Forecasting Using NARMAX-LSTM'. In: 2022 27th International Conference on Automation and Computing (ICAC). 2022, pp. 1–6 (cit. on p. 4).

N. Nikentari and H.-L. Wei. 'Multitask learning using non-linear autoregressive models and recurrent neural networks for tide level forecasting'. In: International Journal of Electrical and Computer Engineering (IJECE) 14.1 (Feb. 2024). Publisher: Institute of Advanced Engineering and Science, pp. 960–970 (cit. on p. 4).

2

Background and Literature Review

The aim of this chapter is to present a theoretical background on multitask learning, neural networks, mixture of experts and also give a detailed review of designing multitask learning. The chapter will primarily discuss models or frameworks that will be used to combine with multitask learning.

2.1. Multitask Learning

Multitask learning is a machine learning approach for simultaneously solving and learning multiple tasks that are related to each other by sharing knowledge among the tasks. Multitask learning, also called learning with auxiliary tasks, jointly learns and learns to learn [1]. The auxiliary tasks are expected to help the main task train or learn its tasks by supplying knowledge and representation [2]. The definition of multitask learning is as follows:

Given K learning tasks where the subset of the tasks or all the tasks are related but not exactly alike, multitask learning aims to improve the learning of a model by using the information in the m tasks.

Multitask learning can be seen as a machine learning way to mimic the activity of how humans learn. The information or knowledge that we acquire from somewhere can often be transferred to and/or used elsewhere. For instance, skills learned from a sports activity can usually applied to another sport, such as skills learned in squash and tennis can benefit each

other [1, 2, 16]. Multitask learning is useful for dealing with a data modeling problem under the following scenario: a set of tasks have commonalities between each other; therefore, the model for each task could gain advantages by sharing knowledge across the tasks than training then independently [1, 2].

Multitask learning has been implemented using many machine learning models. One of the most popular approaches used is neural networks. In earlier research on multitask learning, the neural network model used was feed-forward neural networks. Later, complex neural network models like recurrent neural variants such as Long Short Term Memory and Gated Recurrent Neural Networks are implemented. Nowadays, most research focuses on more advanced models like Convolutional Neural Networks (CNN). CNN, combined with multitask learning, proved to be effective in solving many multitask problems.

Gated Multitask Learning is another machine learning model that can be used to solve multi-task learning problems. This model ensembles several experts and employs a gate to choose the best input for each expert. Inspired by the mixture of experts, gated multitask learning implements this model by using more than one gate to accommodate the number of tasks to be solved.

2.2. Recurrent Neural Network

Recurrent Neural Network (RNN) is a neural network model that handles input as sequential data. As an improvement of the classic feedforward neural network, RNN has the ability to handle variable length data sequentially, where the output from the previous step is treated as an input for the current step. Because of this, RNN can leverage information from previous input, and this condition is referred to as a network that has memory. This memory feature in RNN is represented as a hidden state in RNN architecture [17, 18]. Another unique RNN capability is RNN in certain steps sharing the same weights while feedforward neural networks have independent weight for every input feature [18]. The architecture of the RNN is shown in Figure. 2.1.

Formally, the RNN can be expressed as follows:

$$h_t = \sigma(W_{xh}X_t + W_{hh}h_{t-1} + b_h) \tag{2.1}$$

$$y_t = (W_{hy}X_t + b_y) \tag{2.2}$$

where h_t denote the hidden state, σ is activation function, W represent weight matrix (eg. W_x is weight connecting input x to hidden layer h), b represent bias vector. Common choices for activation functions are rectified linear units (ReLU), hyperbolic tangent functions, and sigmoid

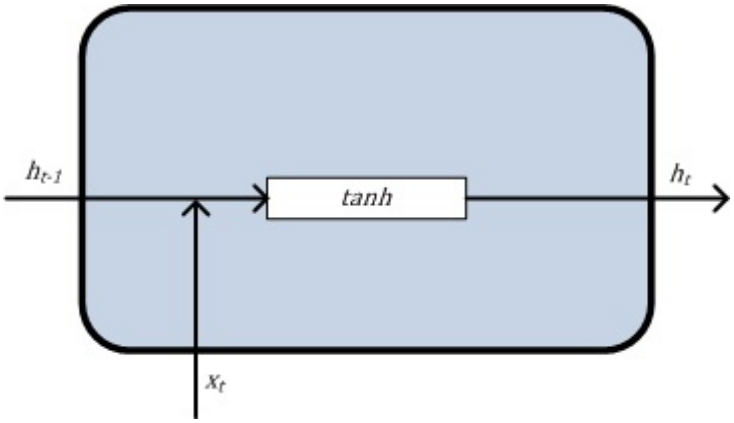


Figure 2.1.: Recurrent Neural Network Architecture

functions.

Despite having good features, RNN suffers from exploding gradients and vanishing gradients. Exploding gradient is a condition where a slight change in the norm of gradient during training causes the gradient of the loss function to become locally large or explode [17, 18]. Vanishing gradients arise when norm of gradient propagated over many layers, it likely to vanish or went to norm 0. The washout gradient makes RNN difficult to handle the long-term dependencies [17, 19].

2.2.1 Long-Short Term Memory

To address the exploding gradient and vanishing gradients problems, Hochreiter and Schmidhuber [20] created Long-Short Term Memory (LSTM) RNN. In order to enable RNN to learn or handle long-term dependencies, LSTM added a cell called the memory cell. This cell is able to store or forget the information from input [21]. This additional cell has three gates: forget gates, input gates, and output gates. The forget gates was originally composed by Gers et.al [22]. LSTM architecture is shown in Figure. 2.2.

The forget gate manages the input to be removed/forgotten or to be preserved. The equation of forgetting gate f_t can be expressed by

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_h) \tag{2.3}$$

The input gate is in charge of whether input information would be added to the memory cell or not. This gate manages the information through two layers: a sigmoid layer and a tanh layer. The equation for these two layers is as follows:

$$i_t = \sigma(W_i.[h_{t-1}, x_t] + b_i) \tag{2.4}$$

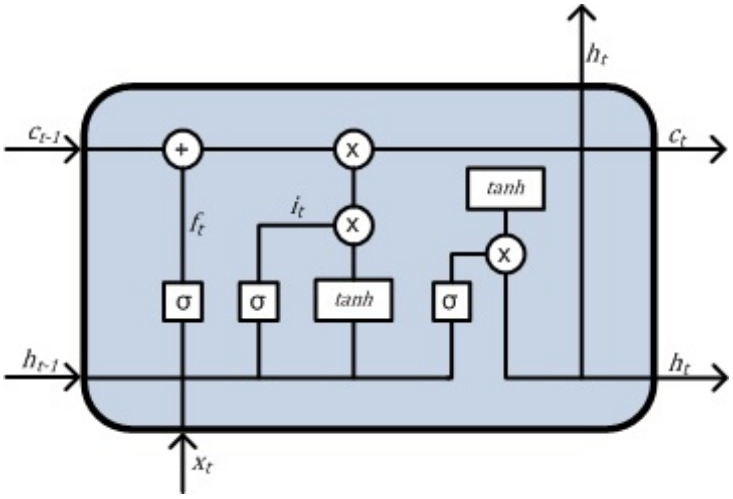


Figure 2.2.: Long-Short Term Memory Architecture

$$\tilde{C}_t = tanh(W_c.[h_{t-1}, x_t] + b_c) \tag{2.5}$$

The output from these two layers then becomes updated for network memory. The equation of updated memory can be expressed by

$$C_t = f_t * tC_{t-1} + i_t * \tilde{c}_t \tag{2.6}$$

where f_t is the result of the forget gate, which is a value between 0 and 1, where 0 indicates completely getting rid of the value, whereas one implies completely preserving the value. The output gate decides to update the output. Among the formulas, W represents the weight parameter, and the subscript of W (W_f, W_i, W_c, W_o) can be denoted as the corresponding gate, and b denotes the bias parameter. The σ represents the sigmoid function, and the $tanh$ is the hyperbolic tangent function. The formulas to compute the output can be represented by the following:

$$o_t = \sigma(W_o.[h_{t-1}, x_t] + o_i) \tag{2.7}$$

$$h_t = o_t * tanh(C_t) \tag{2.8}$$

2.2.2 Gated Recurrent Unit

One of the LSTM weaknesses is the computation time quite extensive [18]. To solve this problem, Cho et al. [23] designed a Gated Recurrent Unit (GRU) by simplifying LSTM. GRU modified the LSTM structure by combining the forget gate and input gate. Because the network has fewer gates, the network needs less training time [18, 24, 25]. This model only

uses two gates: z_t represents the update gate, and r_t represents the reset gate. The GRU structure is illustrated in Figure 2.3.

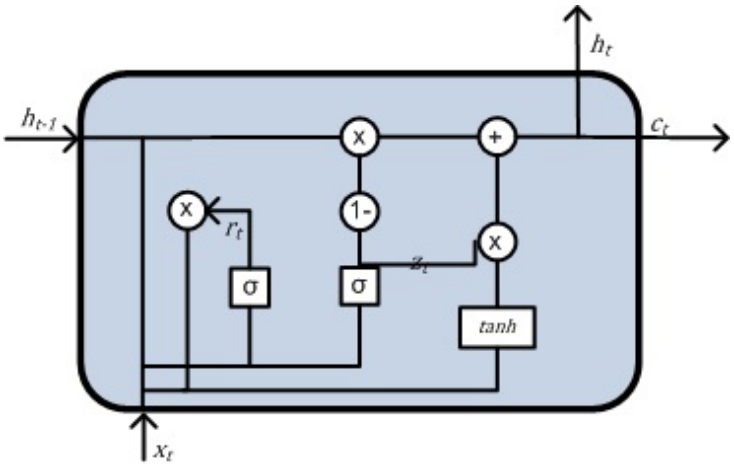


Figure 2.3.: Gated Recurrent Unit Architecture

GRU is implemented by applying the following equations:

$$z_t = \sigma(W_z.[h_{t-1}, x_t]) \tag{2.9}$$

$$r_t = \sigma(W_r.[h_{t-1}, x_t]) \tag{2.10}$$

$$\tilde{h}_t = tanh(W.[r_t * h_{t-1}, x_t]) \tag{2.11}$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t \tag{2.12}$$

The input and memory information at time t are x_t and h_t, respectively. The W denotes the weight parameter, and the subscript of W (W_z, W_r) can be denoted as the corresponding gate.

2.2.3 Bidirectional LSTM

The original concept of Bidirectional LSTM (BiLSTM) comes from bidirectional RNN [26]. Similar to LSTM, BiLSTM also uses forward and backward directions. The difference is that BiLSTM uses forward and backward at the same time or simultaneously. BiLSTM is able to do this by building two LSTM layers, the first layer for forward and the other one for backward. The backward and forward LSTM was proposed to overcome the limitation of LSTM that was only able to absorb previous information but failed to gain future information [27–29]. The

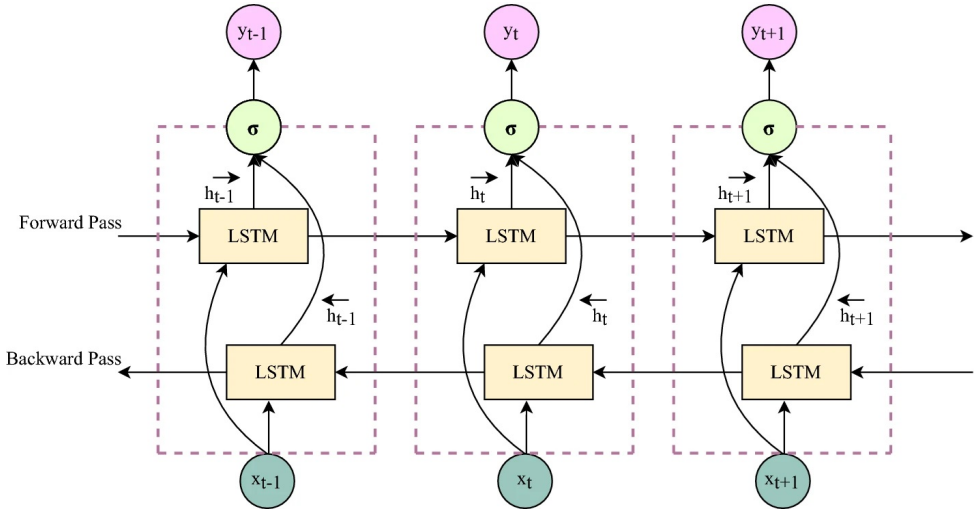


Figure 2.4.: The unfolded architecture of three layers BiLSTM. Reprinted from [31]

final output of BiLSTM is the concatenation of forward and backward layers [29, 30]. The unfolded architecture of the two-layer BiLSTM network is shown in Figure.2.4.

$$\overrightarrow{h_t} = \overrightarrow{LSTM}(h_{t-1}, x_t) \tag{2.13}$$

$$\overleftarrow{h_t} = \overleftarrow{LSTM}(h_{t+1}, x_t) \tag{2.14}$$

where $\overrightarrow{h_t}$ represent forward layer and $\overleftarrow{h_t}$ represent the backward layer.

2.3. The Nonlinear Auto Regressive with Exogenous Inputs Model

The Nonlinear Auto Regressive with Exogenous Inputs (NARX) model is a nonlinear identification model that uses lagged inputs and outputs to represent the system behavior. NARX model is a special case of the nonlinear autoregressive moving average model with exogenous inputs (NARMAX) model that does not contain lagged noise variables [12]. For a single input, singletask learning system, NARX model can be formulated as:

$$y(t) = f(y(t-1), \ldots, y(t-n_y), u(t-d), u(t-d-1), \ldots, u(t-d-n_u)) + e(t) \tag{2.15}$$

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 multitask learning systems, NARX can be formulated as follows:

$$y_i(t) = f_i(y_1(t-1), ..., y_1(t-n_y), ..., (y_m(t-1), ..., y_m(t-n_y), u_1(t-d), u_1(t-d-1), \\ ..., u_1(t-d-n_u), ..., u_r(t-d), u_r(t-d-1), ..., u_r(t-d-n_u)) + e(t) \tag{2.16}$$

where $i = 1, 2..., m$, with m being the number of outputs; r is the number of inputs.

2.4. The Nonlinear Autoregressive Moving Average Model with Exogenous Inputs Model

The Nonlinear Autoregressive Moving Average Model with Exogenous Inputs (NARMAX) model is a transparent non-linear system identification method that focuses on determining the best model structure, including selecting the most useful and important model terms based on available systems output and input data [12]. The NARMAX model for singletask and multitask learning systems can be respectively formulated as:

$$y_t = f(y(t-1), ..., y(t-n_y), u(t-d), u(t-d-1), ..., u(t-d-n_u), e(t-1), ..., e(t-n_e)) + e(t) \tag{2.17}$$

and

$$y_i(t) = f_i(y_1(t-1), ..., y_1(t-n_y), ..., y_m(t-1), ..., y_m(t-n_y), u_1(t-d), u_1(t-d-1), ..., \\ u_1(t-d-n_u), ..., u_r(t-d), u_r(t-d-1), ..., u_r(t-d-n_u), e_1(t-1), ..., e_1(t-n_e), \\ ..., e_m(t-1), e_m(t-n_e)) + e(t) \tag{2.18}$$

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 enter the model structure through output delays [32]. Both models introduce nonlinear functions to learn the system input-output relationships, but the NARMAX model also includes prediction errors to improve the modeling performance [33]. The introduction of "error variable" or "residual variable" makes NARMAX more powerful for nonlinear system identification [34]. The most commonly used basis functions in NARX and NARMAX modeling are polynomials, which usually lead to transparent, interpretable, and parsimonious models [12, 32, 35]. Another widely used nonlinear representation to apply with NARX and NARMAX is neural networks, which usually result in black box models. A modeling model combining NARX or NARMAX and neural networks may be called grey box model identification [36, 37].

2.5. Convolutional Neural Networks

As many real-world applications nowadays deal with computer vision, standard neural networks are no longer powerful enough to solve this task. Convolutional Neural Networks (CNN)

were then created as an alternative solution related to computer vision. The first concept of CNN was proposed by Kunihiko Fukushima [36], but the CNN model that became the foundation of the modern approach for computer vision was introduced by Yann LeCun et al. [38]. CNN consists of three layers: convolution layers, pooling layers, and fully connected layers. Convolution layers are responsible for filtering the input and mapping it, while the pooling layer is in charge of downsampling or reducing the input spatial dimension [39, 40]. These two layers became CNN's prominent features that cannot be found in standard neural networks.

2.5.1 Extreme Inception

Extreme Inception (XCeption), composed by Francois Chollet [41], is one variant of convolutional neural networks improvement version of Inception convolutional neural networks. XCeption has 36 convolutional layers located in 14 different blocks. This model uses a depth-separable convolution, responsible for two processes: depth-wise convolution and point-wise convolution. Combining both processes proved to improve the accuracy of image classification [37]. The main purpose of depth separable convolution is to reduce memory usage and decrease the execution time during the training process [42, 43]. The XCeption model architecture can be seen in Figure. 2.5.

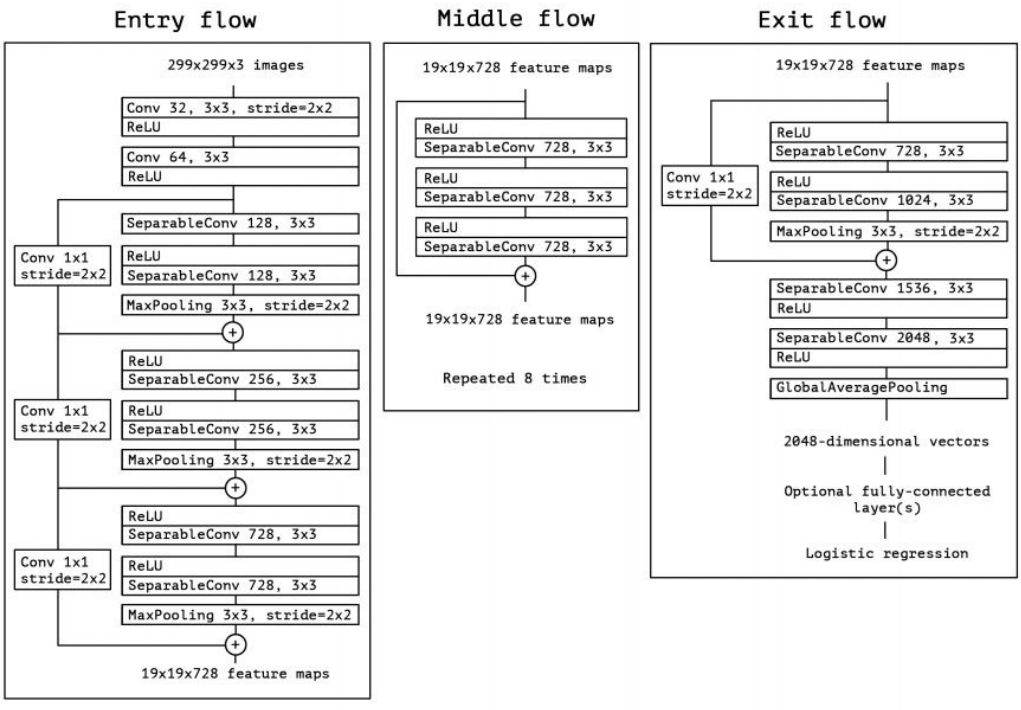


Figure 2.5.: Xception Architecture. Reprinted from [41]

2.6. Mixture of Experts

Ensemble learning is one of the popular techniques that combine two or more models to solve a single machine learning problem. This merging approach is in the hope that by assembling the model, it can also combine the strength from every model so that it can enhance and gain better performance generalization [44–46]. A mixture of Experts (MoE) is a neural network-based ensemble model that introduces the concept of having different experts deal with one modeling problem. Each expert in MoE is in charge of a different region of input space with the possibility that the region between experts will be overlapping because of the soft partition done by gating network [47, 48].

The original MoE concept introduced in [49, 50] consists of a set of m expert networks and a single gating network. For each expert, MoE is typically built using a single layer network and applied softmax function for gating network [51]. A mixture of Experts basic architecture can be seen in Fig 2.6.

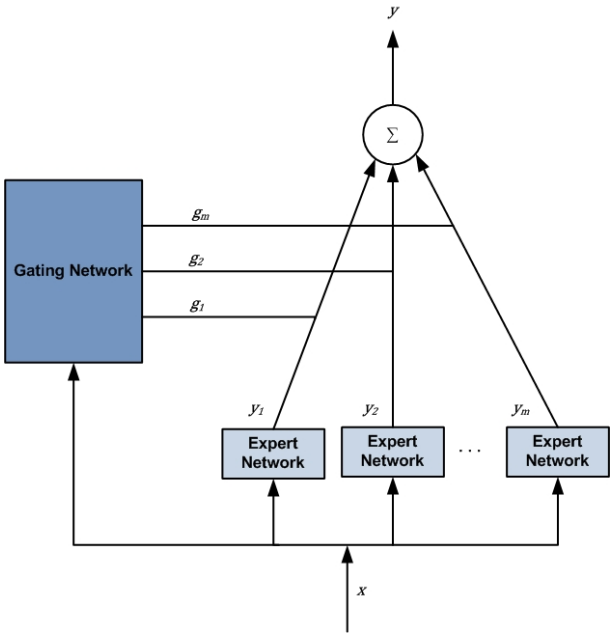


Figure 2.6.: Mixture Of Experts Basic Architecture

The expert networks are typically single-layer linear networks, while gating networks also use single-layer linear networks but with softmax functions.

Given a training sample

$$(x, y) \in D = \{(x_t, y_t)\ t = 1...T\} \tag{2.19}$$

MoE is expressed by following form

$$y(x) = \sum_{j=1}^{m} g_j(x) y_j(x) \tag{2.20}$$

Where $y_j(.)$ is the output function for expert j, m is the number of experts, and $g(.)$ is the gating network, which gives a weight for each expert, given an input x. The gating network is usually a layer network that applies the softmax function, where g_j satisfies

$$0 \leq g_j \leq 1, j = 1, 2, ..., m \tag{2.21}$$

$$\sum_{j=1}^{m} g_j(x) = 1 \tag{2.22}$$

The gating function with softmax function is given by

$$g_j(x) = \frac{\exp(u_j)}{\sum_j \exp(u_j)}, j = 1, ...K \tag{2.23}$$

Where $u_j = (x, v_j)$ and v_j is the gating network weight vector.

For the training or learning, MoE uses the Expectation-Maximization (EM) Algorithm [50]. This iterative method is used to find the maximum likelihood for the experts and gating networks parameters of the MoE architecture [52, 53].

2.6.1 Experts

There are no specific rules on how many experts should be used in the MoE model. In practice, the appropriate number of experts should be chosen manually [54]. By using a large number of experts, there is a big possibility the experts will be more focused on small subspace data, but this will lead to the model will suffer overfitting issues, so decreasing the number of experts will be a better option [48, 55].

2.7. Time Series

A time series is a historical data sequence in which each data point x_t is taped at a time stamp t. A time series of length t can be represented as a sequence $X = [x_1, x_2, \ldots, x_t]$. For the convenience of the description, the notation X_{t-p}^{t} to denote a segment of time series $X = [x_1, x_2, \ldots, x_t]$ [56]. The purpose of studying time series is to build a model that

can forecast the future behavior of the associated process based on past observations. An important element in predicting a task is the size of the horizon [57, 58].

In terms of the horizon, forecasting can be classified into two types: one step ahead and multistep ahead. One step ahead forecasting is forecasting using previous observations, e.g. $= [x_1, x_2, \ldots, x_t]$, to predict the value of x_{t+1}. The architecture for ones tep ahead forecasting is given in Figure 2.7. The forecast for one step ahead performs the following function mapping [59]:

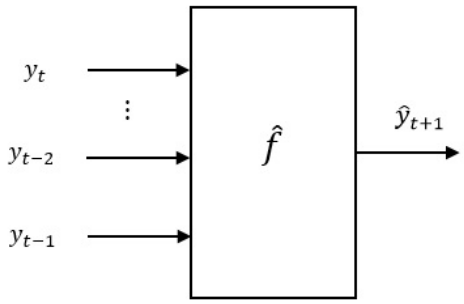


Figure 2.7.: One-step forecasting architecture

$$y_{t+1} = f(X_t, X_{t-1}, \ldots, X_{t-p}) \tag{2.24}$$

where X_t is the observation at time t

Multistep ahead forecasting is a way to predict a sequence of H future values based on observed data [56, 60]. The task of multistep ahead (also called multi-stage or long-term prediction) time series forecasting task consists of predicting the following H values $[y_{N+1}, \ldots, y_{N+H}]$ of historical time series $[y_1, \ldots, y_N]$ comprising of N observations, where $H > 1$ denotes the forecasting horizon. This study will use two notations where f and F, to denote the functional dependency between the past and future observations respectively [60]. The architecture for multi-step-ahead forecasting is given in Figure 2.8.

Compared to one-step-ahead prediction, forecasting multiple steps ahead is a more challenging problem. The reason behind this is that the approach has to experience a higher dose of uncertainty from several aspects, including error accumulation, needs a higher functional complexity, and requires an extensive computational time as a result of the number of models to learn is equivalent to the size of the horizon [58–60]. There are several strategies in multistep forecasting; among them, the Direct Multi Step Strategy is considered in the study here.

The direct strategy consists of forecasting each horizon individually from the other. In other

words, H model f_h is learned (one for each horizon) from time series $[y_1, \ldots, y_N]$ where

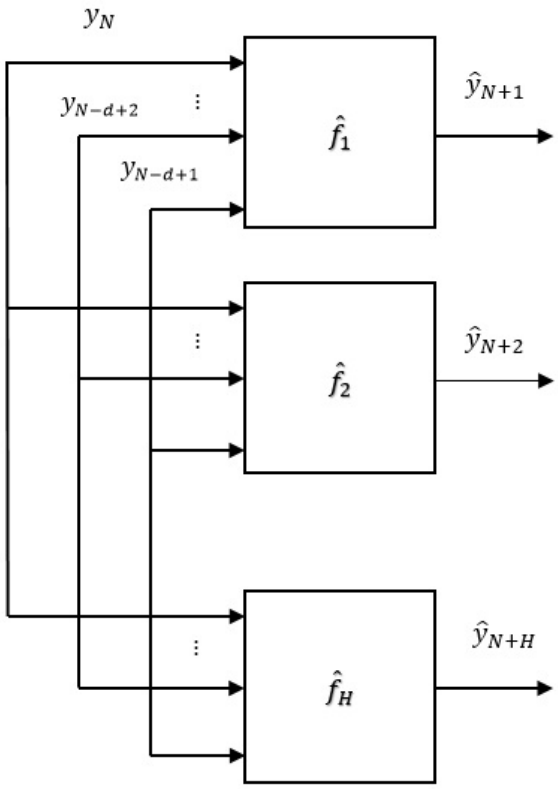


Figure 2.8.: Multi-step-ahead forecasting architecture using direct strategy

$$y_{t+h} = f_h(y_1, \ldots, y_{t-d+1}) + w \tag{2.25}$$

with d refers to the embedding dimension of the time series, $t \in \{d, \ldots, N - H\}$ and $h \in [1, \ldots, H]$

2.8. Review on Designing Multitask Learning

Designing neural network architecture is one of the most crucial parts of building a multitask learning model. Hard and soft parameter sharing are the core concepts in designing multitask learning architecture, but several references proposed different approaches. In the following subsections, we will describe and review more details about hard and soft parameter sharing. Alternative techniques can also be used to create multitask learning model.

2.8.1 Hard Parameter Sharing

The fundamental idea of hard parameter sharing is to share the hidden layer among all tasks and have output separately or allocate several specific output layers for specific tasks [1, 3,

4]. This approach dramatically affects the risk of overfitting and minimizing the loss function of multiple tasks [1, 2]. Baxter [5] demonstrated the possibility of smaller overfitting by using the shared parameters compared to overfitting in task-specific parameters. The rationale of this claim is that the more tasks learn or train jointly, the more the multitask learning model has to capture all representation or information from all tasks, and the less the model has the opportunity to overfitting [5]. The hard parameter sharing architecture is shown in Figure. 2.9.

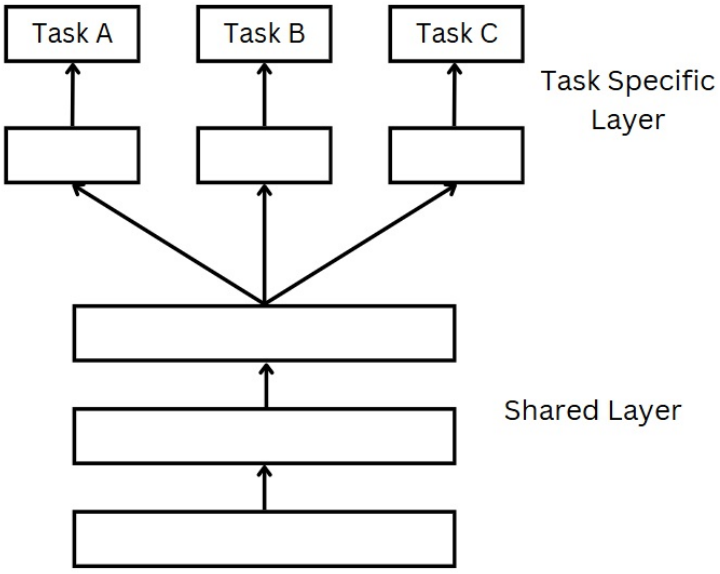


Figure 2.9.: Hard parameter sharing architecture.
Reprinted from[3, 6]

One of the earliest works of multitask learning using hard parameter sharing was proposed by Caruana [6]. The proposed work builds three multitask learnings based on Backpropagation Neural Networks (only using one fully connected hidden layer and shared for all the tasks), K-nearest neighbor (KNN), kernel regression, and Decision Tree Induction. Caruana proved that multitask learning model can be built not only by using neural networks but also by using different machine learning approaches. In [5] multitask learning acts like probabilistic model by using Gaussian likelihood and applying weight in multi loss function. The proposed model uses a shared encoder in the first network and then a specific decoder for each task. To demonstrate the efficacy of the proposed method, the authors implement a classification study case with three tasks. Mahmoud et al. [8] created multitask deep learning by implementing four Convolutional Neural Networks, a single dense layer, and a single Gated Recurrent Unit (GRU) as shared networks and single GRU and softmax layers as an individual network for each task. This work evaluated their methods by implementing the

classification task of human activity recognition. The structure of the Convolutional Neural Networks with a single dense layer and single GRU model is shown in Figure 2.10.

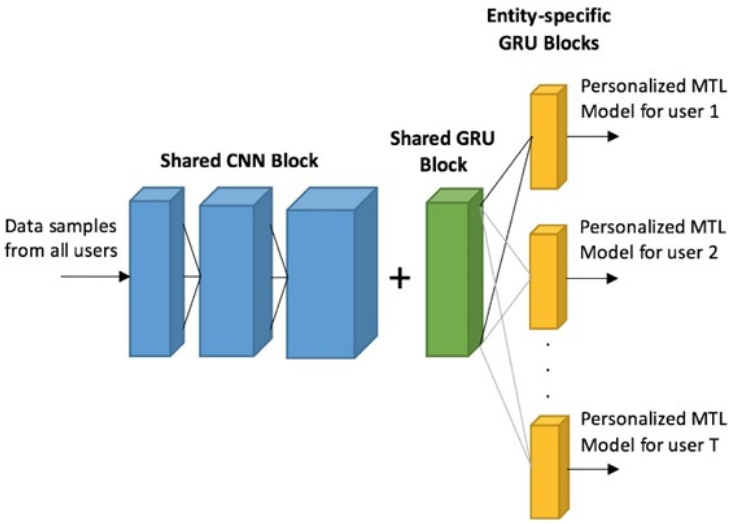


Figure 2.10.: Convolutional Neural Networks with single dense layer and single GRU architecture. Reprinted from [8]

To improve accuracy and minimize overfitting effects in multitask learning, loss balancing by normalization of the gradient is conducted [9]. The loss balance for the multitask is gained by directly calibrating gradient magnitudes. This approach used a symmetric Visual Geometry Group (VGG) with 16 layers and Semantic Segmentation Networks (SegNet) as a shared network. This work trained their model using video input data to solve clustering task problems. The study in [61] carried out different hard parameter sharing approaches by using not only one but also two shared networks. The first shared networks are applied in the first layer networks, followed by task-specific networks, and the final network is shared with multiple outputs. Specific networks in multitasking learning are usually used for specific tasks, but the variants of the networks or the layers are the same for all tasks. The proposed model efficiency is trained in forecasting tasks.

Ilias et al. [62] designed a multitask learning model that used different networks for the prediction tasks. The proposed network used Bidirectional Long-Short Term Memory (BiLSTM) as a shared layer and applied a multihead self-attention layer with global average pooling for specific layers, but for the primary task, the networks added BiLSTM layer before the multihead self-attention layer. This difference makes the primary task has a bigger network than the auxiliary task.

2.8.2 Soft Parameter Sharing

In soft parameter sharing, each task has its specific hidden layer with independent parameters. The distance between the model parameters of different tasks is then regularised to make the parameters similar. Although every task has its model with its setting, the distance between the model parameters of every dissimilar task is added to unite the objective functions [1, 3]. The soft parameter sharing architecture is shown in Figure. 2.11.

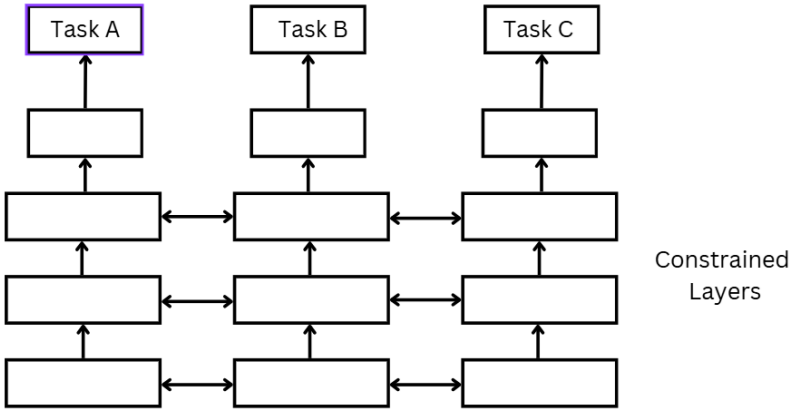


Figure 2.11.: Soft parameter sharing architecture. Reprinted from [1, 3]

Cross-stitch networks, as seen in Fig 2.12, are one of the well-known soft parameter sharing proposed by Misra et al. [63]. The architecture consists of two separate AlexNets that share information between tasks using a linear combination. The combination itself is parameterized using α. The combination in the networks only happens in the pooling layer and fully connected layer. Because AlexNet has three pooling layers and two fully connected layers, there will be five combinations in the multitask learning architecture. The conducted model is implemented in two tasks: (1) semantic segmentation and surface normal prediction and (2) object detection and attribute prediction.

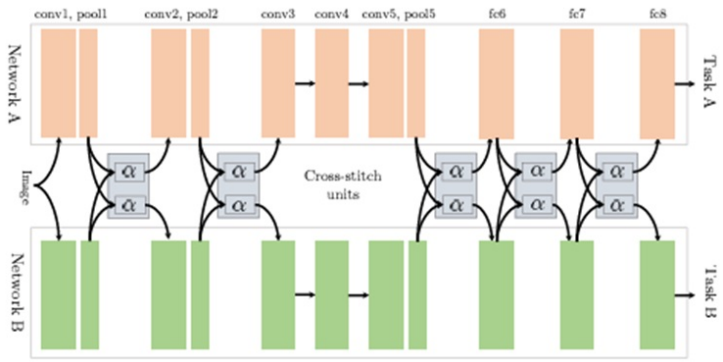


Figure 2.12.: Cross-stitch networks. Reprinted from [63]

One of the most challenging things in designing multitasking learning is how to control the amount of shared information. A sluice network [64] is suggested to address this problem. The network is built based on two Recurrent Neural Networks (RNN), where each RNN represents one task. Each RNN network consists of three hidden layers, and then each hidden layer is split into two subspaces. Between RNN layers consist of α parameters control to combine subspace between tasks. In the output layer, it also provides β parameters control, which is used for prediction. Sluice network performance is evaluated for the chunking tasks, named entity recognition tasks, semantic role labeling tasks, and part-of-speech tagging tasks. The Sluice network is illustrated in Figure 2.13.

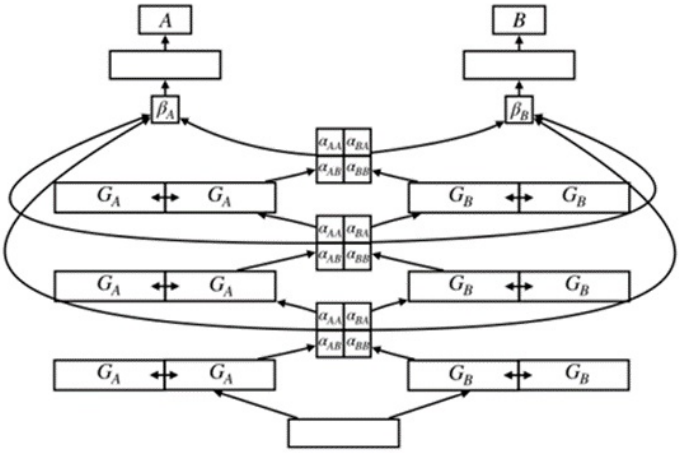


Figure 2.13.: Sluice network. Reprinted from[64]

Deep learning with attention was designed by Liu et al. to allow specific features to learn from global features [65]; each task is not directly learning from a shared network but from a soft attention mask implemented in every convolutional block in the shared network. The proposed architecture consists of single shared networks followed by K number of specific networks. The number of K equals the number of tasks in the multitask learning. The proposed method is evaluated on image-to-image regression tasks and image classification tasks.

If cross-stitch networks combine the output from certain layers to share information, Xioa et al. [66] used output from the first networks as input to other specific networks for their sharing mechanism. The proposed multitask learning is assembled with three networks, as three represent the task number. The output from the first layer in each network will be the input for the second layer for other networks, so in this way, the information from a specific network can be used by other networks. Another different approach is also delivered by this work, which uses not only one but also two LSTM networks as the second layer. The

output from these two LSTMs concatenates and becomes the output for a specific task. The proposed model is applied to the electricity load prediction study case to investigate the efficiency of the performance.

2.8.3 Non-Deep Multitask Learning Model

Earlier works involving multitask learning are mostly based on non-deep neural networks or without using neural networks. The main ideas of this approach are still the same with multitask learning using neural networks, which are trying to gain benefit from the relatedness between tasks as an attempt to improve the generalization performance of multitask learning [4]. Non-deep multitask learning usually consists of an input layer, one middle layer, and an output layer with multiple outputs. The middle layer is a part where information is shared between tasks. The structure itself is similar to regular neural networks with multiple inputs and multiple outputs (MIMO). What makes multitask learning different from the typical MIMO neural networks is that multitask learning has a specific output layer for a specific output or task.

Several works [67, 68] applied a Backpropagation neural network for their multitask learning model. The works followed the basic neural network architecture with a single input, a hidden output layer, and multiple outputs to accommodate multiple tasks. The multitask learning architecture for non-deep neural networks using Backpropagation is shown in Figure. 2.14
Not all multitask learning is implement based on neural networks, some works using others learning methods to build its model. In [69], semi-supervised multitask learning with Gaussian Processes for classification tasks is proposed and applied hyperparameter sharing of the co-variance functions as a sharing procedure. Another work that implemented semi-supervised multitask learning for classification tasks with sharing parameters is employed by Dai and Li [70]. The proposed multitask learning model is built using dynamic fuzzy to solve the next task by using the information from preceding related tasks.

As discussed earlier, the non-neural network approach focuses on hyperparameter sharing as its information sharing. Tian et al. [71] conducted different ways to learn task relationships by a shared set of eigenfunctions. The model contains a common eigenfunction shared by different tasks and a unique eigenfunction for individual tasks.

2.8.4 Gated Multitask Learning

Inspired by a Mixture of Experts (MoE), which comprises several expert networks or base learners and one gating network, Gated Multitask Learning is a multitask learning model that adopts experts and gating as its main network structure. Gated multitasking learning is a

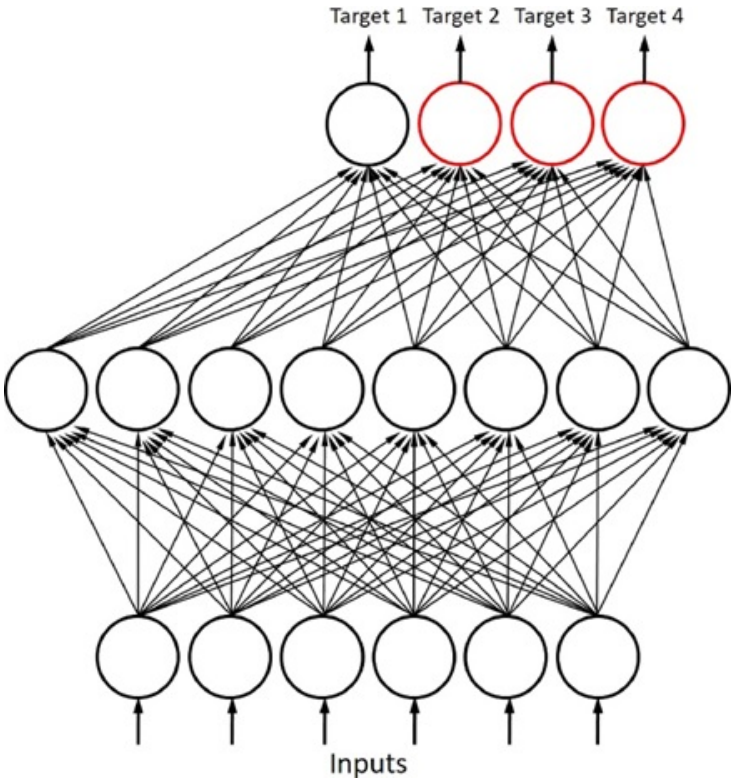


Figure 2.14.: Multitask learning with non-deep neural networks Reprinted from [67]

type of multitasking learning that is dissimilar to regular multitasking learning in terms of input distribution.

Following the basic rules of MoE, data to be fed into the gating network is based on training data samples, but for the experts, the input is constructed by resampling the training data samples. The generated process of resampling the dataset itself allows the same sample data to appear again and again within the same dataset [50, 72]. The purpose of repeating the appearance of sampling data is to create diversity in the training set in order to find a good group of datasets [73]. There are two main differences between Gated Multitask Learning and the regular multitask learning model. First, Gated Multitask Learning has multiple networks (gate and experts) to be fed with input data, while regular multitask learning only has one. Second, whereas regular multitask learning receives the same dataset for the input, Gated Multitask Learning resamples the input data for expert networks.

The sharing mechanism in Gated Multitask Learning takes place in the input where the gating networks and expert networks receive the same sampling data but are distributed differently for gating networks and expert networks. The architecture of Gated Multitask Learning itself is the same as any other multitask learning model with input networks, single middle networks, and output networks. The output network in Gated Multitask Learning is called a tower, and

the number of towers is always the same as the number of tasks.

One of the first works to implement MoE in multitasking learning was by Aljundi et al. [74]. The proposed model applies an encoder network with a single encoder/decoder layer as a gating network and an undercomplete autoencoder network for the expert model. This model also created a mechanism where the autoencoder gate can be regarded as the task recognizer by identifying task relatedness. To validate the performance of the proposed gated multitask learning, the author implements the model with an image classification task. The structure of this model is shown in Figure. 2.15.

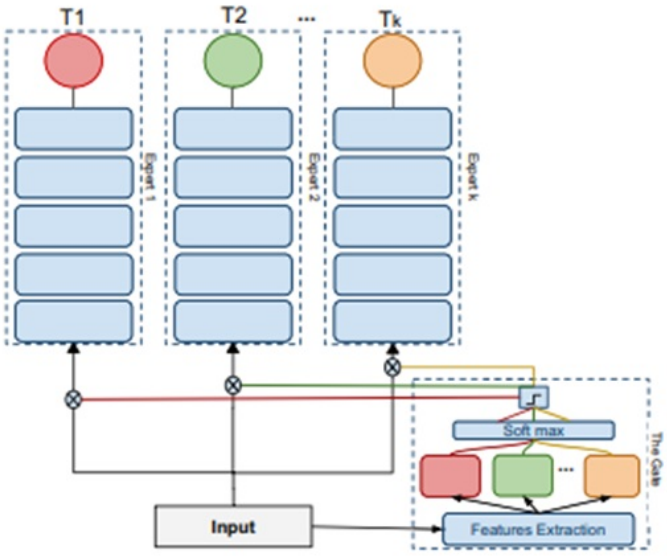


Figure 2.15.: Single Gate Multitask Learning. Reprinted from [74]

Instead of using one gating network, some models [75–77] introduce multi gate network. A multigate network can be interpreted as a soft parameter sharing method where each task has its own gate, and because of this, the number of gates is identical to the number of tasks. In [75], shows that Gated Multitask Learning with three gating networks and training using three prediction tasks can work better in handling less related tasks compared to the single gate model.

Adding more networks was also proposed by Wang et al. [78], but instead of adding at the beginning of the model, the model inserts a network between expert networks. This additional network is able to be inserted because the proposed network has not only one level of expert network but two.

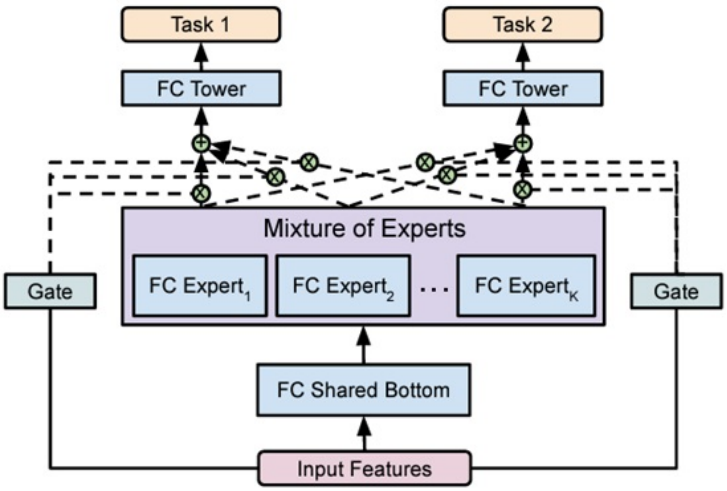


Figure 2.16.: Multi gate Mixture Of Experts. Reprinted from [77]

2.9. Summary

Chapter 2 provided background knowledge on multitask learning. We began with four methods to build multitask learning: hard parameter sharing, soft parameter sharing, non-deep multitask learning, and gated multitask learning. Then, several multitask learning models were reviewed based on these four methods. A literature review was conducted to investigate the development of multitask learning, the variants of neural networks used, the sharing mechanisms, and the study cases used to verify the effectiveness of the multitask learning model. After a technical review of the state-of-art, we identified several issues:

- Loss equality and negative transfer as one of multitask learning main problems rarely discussed
- To validate the proposed multitask learning model, most studies focus on classification (to predict if the observed data belongs to a distinct group) data task rather than forecasting.

To bridge the above practical gaps, we built nondeep multitask learning in the following chapters to avoid negative transfer. Chapter 3 will present multitask learning using hard parameter sharing with nondeep and standard neural networks. Chapter 4 of this thesis will address negative transfer and error balance in multitask learning by implementing soft parameter sharing methods using more advanced neural networks. Finally, in Chapter 5, we developed multitask learning based on a mixture of experts to reduce the negative transfer effect. All proposed models applied NARMAX on their model as an approach to increase information sharing between tasks. To answer the challenge of implementing multitasking learning in areas that have not been previously studied in depth, the study cases used in

chapters 3, 4, and 5 are based on the regression model.

3

Hard Parameter Sharing Multitask Learning using Non-Linear Autoregressive Models and Recurrent Neural Networks

3.1. Introduction

The previous chapter gives an overview of several techniques for building multitask learning. This chapter proposes two multitask learning with non-deep neural networks and a hard parameter sharing technique. Both models are built based on a combination of multitask learning, recurrent neural networks, and nonlinear autoregressive models. Two nonlinear autoregressive models, namely NARX and NARMAX, are used, aimed at comparing and finding the best models in forecasting tasks using the MTL mechanism. Over the past years, NARX and NARMAX models have been applied to singletask learning, and RNN has been introduced to solve multitask learning problems, but relatively little work has been done to combine nonlinear model autoregressive models and RNNs and apply them to multitask learning. Model performance was evaluated by applying tide level forecasting as a study case.

This chapter conducts comprehensive comparisons between three types of recurrent neural networks (LSTM, GRU, and BiLSTM), paired with NARX and NARMAX models, and analyzes their performances for forecasting tide levels at different locations. Therefore, a total of six models are developed: NARX-LSTM, NARX-biLSTM, NARX-GRU, NARMAX-LSTM, NARMAX-

biLSTM and NARMAX-GRU. Comparative studies are performed to evaluate the two types of models based on the same real data and for tide level prediction with three different time horizons. All six models are used to forecast the tide levels of four different locations in the UK simultaneously to reduce the complex process of designing the neural network models and improving the forecast accuracy using models that avoid or overcome experience negative transfer and overfitting. The main contributions of the work are summarized as follows:

- The proposal of a novel NARMAX and NARX modeling framework, combined with an multitask learning scheme addresses the challenges of overfitting and negative transfer problems.
- The design and scheme of a single model structure that can be used for multiple tasks/cases (e.g. forecasting) simultaneously.
- The analysis of the performance of NARMAX and NARX, paired with three RNN model structures with MTL schemes
- A comparison studies with singletask baselines
- A comparison studies of multitask learning without NARX/NARMAX

3.2. Tide Level Forecasting

Tides are a natural ocean phenomenon that is mainly influenced by the gravitational forces of the rotation of the earth and the alignment of the sun and moon. The rising and falling of the sea levels can also be induced by meteorological conditions and shallow water. Tide levels could have a significant impact on human life, and the study of ocean phenomena is an essential part of coastal engineering, coastal ecosystems, and human activity. In the field of coastal engineering, tide level data are valuable for the construction of ports, offshore and cross-sea bridges [79–82]. For coastal ecosystems, tide level data can be crucial for sediment movements and pollutant tracing and monitoring [19, 83]. In the domain of human activity, tide level data can be used as important information in fishing, doing recreational activities [84] and potentially developing tidal energy [85, 86]. Therefore, it is important to effectively model and forecast tide levels.

Forecasting future tide levels plays a vital part in keeping people and the environment safe because tide level fluctuations play a substantial role in marine meteorology, the operation of oceanography, and coastal zone management. In the fields of marine meteorology and oceanography, future tide level data will be helpful for the disposal and movements of sediments, tracers, and pollutants. Besides that, tide level data is crucial for coastal structure

harbor projects such as constructing cross-sea bridges or navigating large engineering. For coastal zone management, the tide level prediction can be used as early warning systems of occurrences of Tsunamis and prevent flooding caused by sea rise [82, 83, 87, 88]. A classic or conventional method to make tide level forecasting is the harmonic analyzing method. Such a traditional way of forecasting can be ineffective if the data are incomplete (e.g., with some data being missing) [88]. 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 [80, 89]. 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, modeling, and time series forecasting [12]. For example, Aguirre et al. 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 [90]; the results are interesting and useful for designing more effective predictive approaches for short-term periodic time series forecasting.

Wu et al. [91] proposed a combination of Genetic Algorithm, Backpropagation and NARX for tide level prediction. Muñoz and Acuña [92] developed NARX and NARMAX models combined with shallow and deep neural networks to forecast daily demand data and air quality conditions. The performances of NARMAX and Radial Basis Function neural networks were examined by Omri et al. [93] 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 the NARX model or a typical neural network model alone. Gu et al. [33] present a novel cloud-NARX model 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 singletask learning (STL), where a model trained using a set of time series data can only be used to predict the same time series process but cannot be used for other time series predictions and therefore cannot benefit from sharing knowledge among related tasks [94, 95]. 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 dataset simultaneously.

To extend STL models to MTL cases, this work proposes a new MTL framework by combining nonlinear aggressive models and RNNs. The proposed MTL framework performs multiple tasks simultaneously, allowing for information sharing between related tasks. For RNN, an MTL scheme can be implemented by sharing information on the structures of the network models (e.g., the number of layers and number 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 [37, 95, 96].

Several MTL RNN models have been proposed for forecasting purposes in the literature. In [97], MTL and 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 [98], MTL and RNN were used to forecast urban traffic flow; the forecasting accuracy was improved by around 10% to 15% over baseline models. In [94], another RNN variant, GRU, was combined with MTL for traffic flow and speed forecasting; the model does not only improve the forecasting accuracy but also solves the problems caused by enlarging the dataset.

In [99], 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 BiLSTM was proposed to forecast cooling, heating, and electric load [100]; the MTL+ BiLSTM model is able to increase the accuracy significantly and improve the time efficiency for training the model.

3.3. Methodology

The designed architecture consists of multitask learning from several datasets (measured at different locations). The tasks in our model are h steps ahead of forecast results in multiple locations. The model only delivers single h steps ahead prediction, but data are from multiple locations. The model is able to forecast the tide level at different locations one time by using the results from a few single models. The inputs are tide levels in four locations at the present time instant t, and the output is the predicted data of tide level for h step/s ahead. This study implements direct forecasting models where there are h different models for h different time step/s.

3.3.1 Multitask RNN model

As mentioned earlier, LSTM, biLSTM, and GRU will be considered to implement multitask forecasting. The general architecture of the multitask RNN is shown in Figure 3.1, where $u_k(t)(k = 1, 2, \ldots, r)$ and $y_j(t)(j = 1, 2, \ldots, i)$ are the system inputs and outputs, n, r, i are the numbers of lags, stations/locations and output variables, respectively. The outputs in this study are assumed to be the same as the number of inputs. For example, if the inputs are 4 locations $(r = 4)$, the output is the h steps ahead forecasting in four locations $(i = 4)$. The proposed network model's input layer consists of sequences of tide data in r locations followed by an RNN layer (LSTM or biLSTM or GRU), a dense layer, or a fully connected layer. The outputs layer consists of tide level data in i location at time t. The multitask RNN architecture can be seen in Fig 3.1.

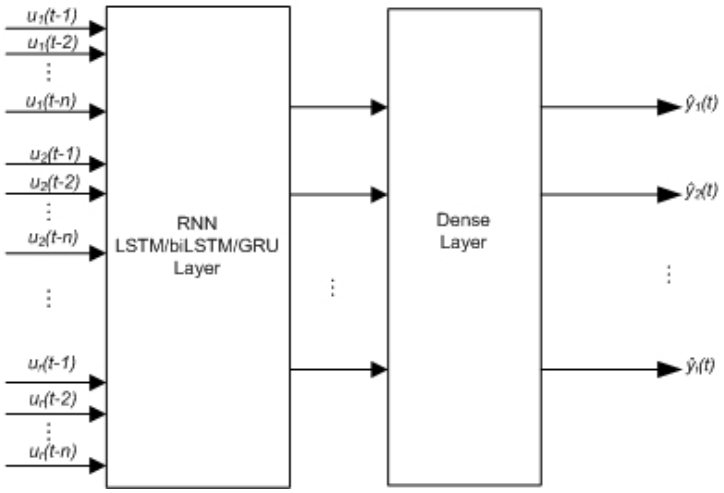


Figure 3.1.: Multitask RNN Architecture

3.3.2 Multitask NARX RNN model

This study employs a direct approach using RNN to predict tide levels. The basic idea behind the direct approach is that forecasting for h time horizons (h steps ahead forecasting) is realized using different models [34, 101, 102] . The prediction of tide level in the h steps can be obtained using NARX LSTM or NARX biLSTM or NARX GRU. The model for h steps ahead prediction can be expressed as [2, 3, 100]:

$$y(t) = f(y(t-h), y(t-h-1), \ldots, y(t-h-n_y), u(t-1), u(t-2) \ldots, u(t-n_u)) \quad (3.1)$$

where $y(t)$ and $u(t)$ and $e(t)$ are the system output and input respectively;n_y and n_u are the maximum lags of the output and input; $f(.)$ is a nonlinear function or mapping. MTL NARX

RNN is a multitask forecasting model that employs NARX by incorporating LSTM, biLSTM, or GRU. The multitask model based on these three neural networks for the direct forecasting approach can be defined as [34, 99]:

$$\begin{aligned} y_i(t) =& f_i(y_i(t-h), y_i(t-h-1), \ldots, y_i(t-h-n_y), \ldots, y_m(t-h), y_m(t-h-1), \ldots, \\ & y_m(t-h-n_y), u_1(t-1), u_1(t-2), \ldots, u_1(t-n_u), \ldots, u_r(t-1), u_r(t-2), \ldots, \\ & u_r(t-n_u)), i = 1, .., m \end{aligned} \tag{3.2}$$

The general architecture of MTL NARX RNN is shown in Figure 3.2. The architecture of MTL NARX RNN is basically similar to that of MTL RNN in that both have four layers: input layer, RNN layer, dense layer, and output layer. The difference is in the inputs: the MTL NARX RNN model needs previous output values. This study focuses on the direct approach, and the observational values of the output are assumed to be known [98].

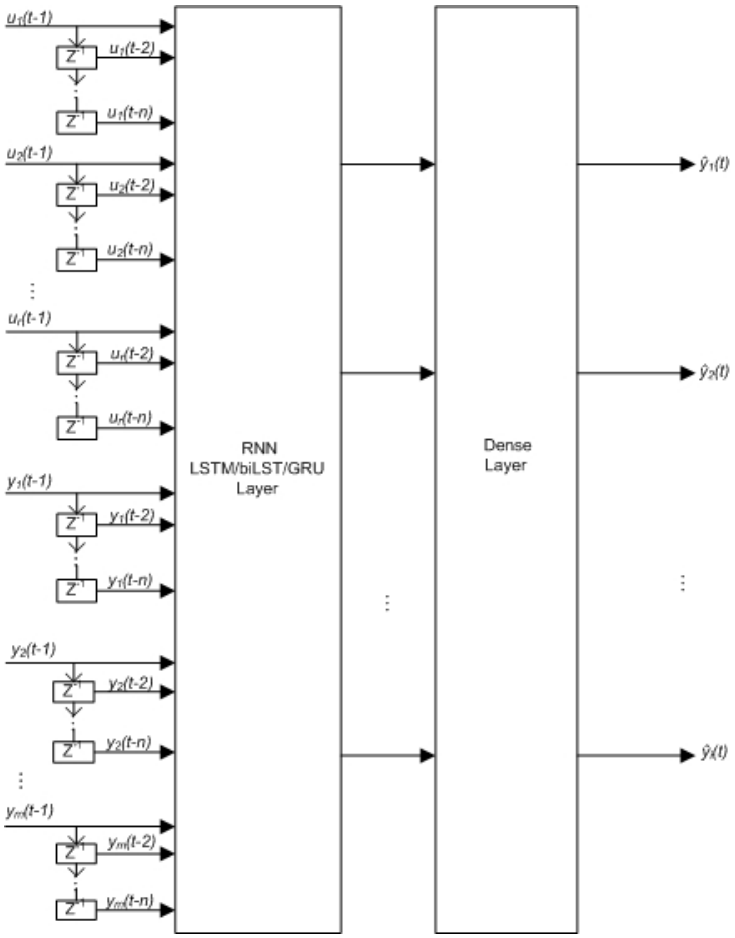


Figure 3.2.: Multitask RNN NARX Architecture

In (3.2), $f_i(i = 1, 2, \ldots, m)$ are approximated by using LSTM or biLSTM and GRU. The NARX-LSTM and NARX-biLSTM models can be represented as:

Forget Gate:

$$\begin{aligned} f(t) =&\sigma(W_f[ht-1,(y_i(t-h),y_i(t-h-1),...,y_i(t-h-n_y),...,y_m(t-h),\\ &y_m(t-h-1),...,y_m(t-h-n_y),u_1(t-1),u_1(t-2),...,u_1(t-n_u),...,\\ &u_r(t-1),u_r(t-2),...,u_r(t-n_u))]+b_f) \end{aligned} \tag{3.3}$$

Input Gate:

$$\begin{aligned} i(t) =&\sigma(W_i[ht-1,(y_i(t-h),y_i(t-h-1),...,y_i(t-h-n_y),...,y_m(t-h),\\ &y_m(t-h-1),...,y_m(t-h-n_y),u_1(t-1),u_1(t-2),...,u_1(t-n_u),...,\\ &u_r(t-1),u_r(t-2),...,u_r(t-n_u))]+b_i) \end{aligned} \tag{3.4}$$

$$\begin{aligned} \tilde{c}_t =&tanh(W_c[ht-1,(y_i(t-h),y_i(t-h-1),...,y_i(t-h-n_y),...,y_m(t-h),\\ &y_m(t-h-1),...,y_m(t-h-n_y),u_1(t-1),u_1(t-2),...,u_1(t-n_u),...,\\ &u_r(t-1),u_r(t-2),...,u_r(t-n_u))]+b_c) \end{aligned} \tag{3.5}$$

$$c(t) = f(t).c(t-1) + i(t).\tilde{c}_t \tag{3.6}$$

Output Gate

$$\begin{aligned} o(t) =&\sigma(W_o[ht-1,(y_i(t-h),y_i(t-h-1),...,y_i(t-h-n_y),...,y_m(t-h),\\ &y_m(t-h-1),...,y_m(t-h-n_y),u_1(t-1),u_1(t-2),...,u_1(t-n_u),...,\\ &u_r(t-1),u_r(t-2),...,u_r(t-n_u))]+b_o) \end{aligned} \tag{3.7}$$

$$h(t) = o(t)tanh(c(t)) \tag{3.8}$$

GRU for NARX model can be represented as:
Hidden Gate:

$$h(t) = (1-z(t)).h(t-1) + z(t).h(t) \tag{3.9}$$

Update Gate:

$$\begin{aligned} z(t) =&\sigma(W_z[ht-1,(y_i(t-h),y_i(t-h-1),...,y_i(t-h-n_y),...,y_m(t-h),\\ &y_m(t-h-1),...,y_m(t-h-n_y),u_1(t-1),u_1(t-2),...,u_1(t-n_u),...,\\ &u_r(t-1),u_r(t-2),...,u_r(t-n_u))]) \end{aligned} \tag{3.10}$$

Reset Gate:

$$\begin{aligned} r(t) =&\sigma(W_r[ht-1,(y_i(t-h),y_i(t-h-1),...,y_i(t-h-n_y),...,y_m(t-h),\\ &y_m(t-h-1),...,y_m(t-h-n_y),u_1(t-1),u_1(t-2),...,u_1(t-n_u),...,\\ &u_r(t-1),u_r(t-2),...,u_r(t-n_u))]) \end{aligned} \tag{3.11}$$

3.3.3 Multitask NARMAX RNN

The h steps prediction of tide level forecasting using NARMAX as seen in [34] can be defined as:

$$\begin{aligned} y(t) =& f(y(t-h), y(t-h-1), \ldots, y(t-h-n_y), u(t-1), u(t-2) \ldots, u(t-n_u), \\ & e(t-1), e(t-2) \ldots, e(t-n_e)) + e(t) \end{aligned} \tag{3.12}$$

The multitask model bas NARMAX for direct forecasting approach can be represented as:

$$\begin{aligned} y_i(t) =& f_i(y_i(t-h), y_i(t-h-1), ..., y_i(t-h-n_y), ..., y_m(t-h), y_m(t-h-1), ..., \\ & y_m(t-h-n_y), u_1(t-1), u_1(t-2), ..., u_1(t-n_u), ..., u_r(t-1), u_r(t-2), .., \\ & u_r(t-n_u), e_1(t-1), e_1(t-2), ..., e_1(t-n_e), ..., e_m(t-1), e_m(t-2), ..., e_m(t-n_e)), \\ & i = 1, .., m \end{aligned} \tag{3.13}$$

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

$$\varepsilon_i(t) = \hat{y}_i(t) - y_i(t) \tag{3.14}$$

where $\hat{y}_i(t)$ is the one-step-ahead prediction calculated from the MTL NARX RNN 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:

$$\begin{aligned} y_i(t) =& f_i(y_i(t-h), y_i(t-h-1), ..., y_i(t-h-n_y), ..., y_m(t-h), y_m(t-h-1), ..., \\ & y_m(t-h-n_y), u_1(t-1), u_1(t-2), ..., u_1(t-n_u), ..., u_r(t-1), u_r(t-2), ..., \\ & u_r(t-n_u), \varepsilon_1(t-1), \varepsilon_1(t-2), ..., \varepsilon_1(t-n_e), ..., \varepsilon_m(t-1), \varepsilon_m(t-2), ..., \\ & \varepsilon_m(t-n_e)), i = 1, .., m \end{aligned} \tag{3.15}$$

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. The architecture of the NARMAX-RNN-MTL model is presented in Figure 3.3.

Note that in this study, the NARMAX model is realized based on LSTM, and the functions

$f_i(i = 1, 2, \ldots, m)$ are derived from the mathematical model of LSTM. LSTM for NARMAX model can be established as: The NARX-LSTM model can be represented as:

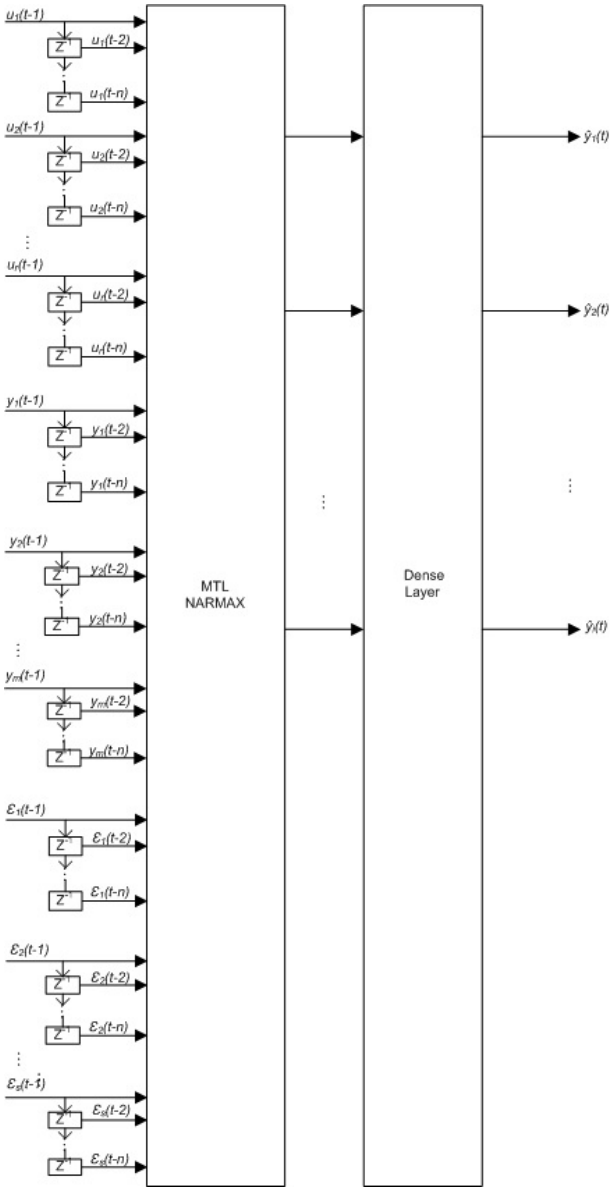


Figure 3.3.: Multitask RNN NARMAX Architecture

Forget Gate:

$$\begin{aligned} f(t) =& \sigma(W_f[ht-1, (y_i(t-h), y_i(t-h-1), \ldots, y_i(t-h-n_y), \ldots, y_m(t-h), \\ & y_m(t-h-1), \ldots, y_m(t-h-n_y), u_1(t-1), u_1(t-2), \ldots, u_1(t-n_u), \ldots, \\ & u_r(t-1), u_r(t-2), \ldots, u_r(t-n_u), \varepsilon_1(t-1), \varepsilon_1(t-2), \ldots, \varepsilon_1(t-n_e), \ldots, \\ & \varepsilon_m(t-1), \varepsilon_m(t-2), \ldots, \varepsilon_m(t-n_e))] + b_f) \end{aligned} \tag{3.16}$$

Input Gate:

$$\begin{aligned} i(t) =&\sigma(W_i[ht-1,(y_i(t-h),y_i(t-h-1),...,y_i(t-h-n_y),...,y_m(t-h),\\ &y_m(t-h-1),...,y_m(t-h-n_y),u_1(t-1),u_1(t-2),...,u_1(t-n_u),...,\\ &u_r(t-1),u_r(t-2),...,u_r(t-n_u),\varepsilon_1(t-1),\varepsilon_1(t-2),...,\varepsilon_1(t-n_e),...,\\ &\varepsilon_m(t-1),\varepsilon_m(t-2),...,\varepsilon_m(t-n_e))]+b_i) \end{aligned} \tag{3.17}$$

$$\begin{aligned} \tilde{c}_t =&tanh(W_c[ht-1,(y_i(t-h),y_i(t-h-1),...,y_i(t-h-n_y),...,y_m(t-h),\\ &y_m(t-h-1),...,y_m(t-h-n_y),u_1(t-1),u_1(t-2),...,u_1(t-n_u),...,\\ &u_r(t-1),u_r(t-2),...,u_r(t-n_u),\varepsilon_1(t-1),\varepsilon_1(t-2),...,\varepsilon_1(t-n_e),...,\\ &\varepsilon_m(t-1),\varepsilon_m(t-2),...,\varepsilon_m(t-n_e))]+b_c) \end{aligned} \tag{3.18}$$

$$c(t) = f(t).c(t-1) + i(t).\tilde{c}_t \tag{3.19}$$

Output Gate

$$\begin{aligned} o(t) =&\sigma(W_o[ht-1,(y_i(t-h),y_i(t-h-1),...,y_i(t-h-n_y),...,y_m(t-h),\\ &y_m(t-h-1),...,y_m(t-h-n_y),u_1(t-1),u_1(t-2),...,u_1(t-n_u),...,\\ &u_r(t-1),u_r(t-2),...,u_r(t-n_u),\varepsilon_1(t-1),\varepsilon_1(t-2),...,\varepsilon_1(t-n_e),...,\\ &\varepsilon_m(t-1),\varepsilon_m(t-2),...,\varepsilon_m(t-n_e))]+b_o) \end{aligned} \tag{3.20}$$

$$h(t) = o(t)tanh(c(t)) \tag{3.21}$$

3.3.4 Metrics Performance

For a singletask, for example, the prediction of a single individual time series $y_i(t)(i = 1,2,\ldots,m)$, without using shared information from any other signals, the loss function can be defined as:

$$yL_i = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(\hat{y}_i(t) - y_i(t))^2} \tag{3.22}$$

where $\hat{y}_i(t)$ is the predicted value, $y_i(t)$ is the 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, the averaged root mean square error (aRMSE) is used as a joint loss, which is defined as follows:

$$L = \frac{1}{K}\sum_{1=1}^{K} L_i \tag{3.23}$$

where L is the joint loss, and K is the number of tasks.

3.4. Experiments

The experiment section is split into two parts. The first part is for Dataset 1, and the second is for Dataset 2. The differences between Dataset 1 and Dataset 2 are as follows :

a) The first dataset forecasts four tasks, while the second forecasts five to eleven tasks.

b) First dataset trained with three different forecast horizons: one step ahead, two steps ahead, and three steps ahead, while the other one was only trained for a one step ahead forecast.

c) Data set one covers the tide levels in four stations/locations in the UK (Wick, Workington, Ullapol, and New Heaven), while dataset 2 concerns the tide level forecast for five stations (five tasks) simultaneously: Harwich, Lerwick, Millport, Portrush, and Weymouth. Six more stations—Aberdeen, Devonport, Fishguard, Holyhead, St Mary, and Stornoway—are also forecasted for dataset 2.

d) Dataset 1 and Dataset 2 are not using the same parameter setting for training the models.

Each part will explain more details about parameter setting, results and discussion in every dataset. Both data sets are implemented using the methodology mentioned in sub-chapter 3.3. The datasets were extracted from the British Oceanographic Data Centre (BODC) website. https://www.bodc.ac.uk/data/hosted_data_systems/sea_level/uk_tide_gauge_network/.

3.4.1 Data Set 1

Data set one in this study is the tide levels in four stations/locations of the UK (Wick, Workington, Ullapol, and New Heaven). This dataset is used to forecast values (of h steps ahead) of tide levels in the four locations, each using its own h steps ahead prediction model. The data are time series of tide levels recorded every 15 minutes. This study considered three different forecast horizons: one step ahead, two steps ahead, and three steps ahead, corresponding to 15 minutes, 30 minutes, and 45 minutes ahead of predictions. In other words, the model will predict the next interval of 15 minutes, 30 minutes, and 45 minutes.

The model's input is tide level data from four stations, and the output is also tide level from four stations. The output for one step ahead, two steps ahead, and three steps ahead forecast is the tide level for the next 15 minutes, 30 minutes, and 45 minutes, respectively.

3.4.1.1. Parameters Setting

In order to fairly compare the six models, all RNN models are trained, validated, and tested using the same parameter setting. The setting can be seen in Table 3.1.

Parameter	Values
Number of hidden layers	1,2,3,4,5
Number of nodes	25,32,64,128,256
Activation function	SGDM
Number of iterations	200
Time Lag for NARX model	n_y =1,2,3,4 n_u=1,2,3,4

Table 3.1.: Parameter Setting for multitask RNN and NARX RNN

3.4.1.2. Results and Discussion NARX Model

This subsection discusses NARX models and neural networks with multitask learning to forecast tide levels in several areas simultaneously. It aims to demonstrate the feasibility of using multitask learning models based on two types of neural networks: a) solely using recurrent neural network models and b) recurrent neural networks combined with NARX models. Three RNN models, namely, LSTM, biLSTM, and Gated Recurrent Unit (GRU), were built into this study. These three neural networks are then integrated with the NARX model respectively. Therefore, a total of six models were developed: LSTM, biLSTM, GRU, NARX-LSTM, NARX-biLSTM, and NARX-GRU.

For the single RNN models, several different neural network settings were implemented. The optimal settings obtained for the three RNN network models are shown in Table 3.2. These results were achieved after fine-tuning the RNN parameters. The best results for the LSTM model are by applying and testing a large number of hidden layers and using large time horizons. GRU delivers the best accuracy for forecasting in all time horizons, while biLSTM failed to gain good overall performance.

RNN Model	Forecast Horizon	Number of Hidden Layer	Number of Hidden Node	Average RMSE
LSTM	1	1	256	0.2781
	2	4	128	0.4894
	3	5	256	0.4841
BiLSTM	1	1	32	0.5653
	2	1	128	0.7131
	3	1	32	0.6285
GRU	1	2	256	0.1383
	2	1	64	0.1082
	3	5	25	0.4447

Table 3.2.: Performance comparison and optimal setting of single RNN using GRU, LSTM and BiLSTM

The optimal structure of the multitask NARX RNN model is determined using the lowest aver-

age RMSE values obtained after performing an empirical analysis, as shown in Table 3.3. As seen from the table, for one step ahead and two steps ahead forecasting, the GRU models, with time lags $n_y = 4$ and $n_u = 2$, perform best among the group of multitask learning NARX RNN. The hidden layer of the optimal NARX-GRU model for one step and two steps ahead includes 128 and 64 nodes, respectively. For the three steps ahead case, the best model is still NARX-GRU, with the following setting: time lags $n_y = 1$ and $n_u = 4$; there are 128 nodes in the hidden layer. The details of the network parameter setting for the best NARX RNN model are shown in Table 3.4.

Network lags	Average RMSE								
	One Step Ahead			Two Steps Ahead			Three Steps Ahead		
	LSTM	biLSTM	GRU	LSTM	biLSTM	GRU	LSTM	biLSTM	GRU
$(n_y = 1, n_u = 1)$	0.1576	0.5364	0.1089	0.1990	0.5070	0.134	0.2840	0.7233	0.1412
$(n_y = 1, n_u = 2)$	0.1717	0.4244	0.1358	0.6959	0.6899	0.1708	0.3054	0.6735	0.1638
$(n_y = 1, n_u = 3)$	0.2483	0.5483	0.1238	0.2372	0.5179	0.1209	0.1787	0.6728	0.1493
$(n_y = 1, n_u = 4)$	0.2041	0.4591	0.1427	0.1999	0.5390	0.1514	0.1755	0.6401	**0.1379**
$(n_y = 2, n_u = 1)$	0.1875	0.4650	0.1037	0.2742	0.4633	0.1462	0.4760	0.7084	0.1545
$(n_y = 2, n_u = 2)$	0.1120	0.5493	0.1053	0.1913	0.7097	0.2333	0.2881	0.7025	0.2111
$(n_y = 2, n_u = 3)$	0.2504	0.5148	0.1531	0.2443	0.6501	0.1309	0.2185	0.6991	0.1532
$(n_y = 2, n_u = 4)$	0.154	0.3841	0.1263	0.2847	0.5355	0.1537	0.1464	0.4114	0.1732
$(n_y = 3, n_u = 1)$	0.3640	0.5163	0.1541	0.2491	0.5312	0.2721	0.4883	0.5957	0.1703
$(n_y = 3, n_u = 2)$	0.1735	0.5553	0.1428	0.2635	0.5132	0.1751	0.2006	0.6928	0.1964
$(n_y = 3, n_u = 3)$	0.1948	0.4520	0.1214	0.2361	0.6943	0.2000	0.2788	0.7154	0.1533
$(n_y = 3, n_u = 4)$	0.1909	0.4619	0.1094	0.2199	0.2969	0.1619	0.2317	0.6299	0.1553
$(n_y = 4, n_u = 1)$	0.1822	0.3736	**0.0955**	0.2167	0.4606	**0.1265**	0.1591	0.5741	0.1455
$(n_y = 4, n_u = 2)$	0.1098	0.4163	0.1704	0.1385	0.5287	0.1284	0.2341	0.7481	0.1699
$(n_y = 4, n_u = 3)$	0.1660	0.4733	0.1216	0.2099	0.432	0.1454	0.1962	0.6849	0.1698
$(n_y = 4, n_u = 4)$	0.1341	0.5130	0.1216	0.1689	0.5524	0.2707	0.2346	0.7377	0.1796

Table 3.3.: Performance of Multitask RNN with different time lag setting

The models identified by the NARX RNN based on the optimal results in Table 3.3 for each horizon are listed below:

One Step Ahead

$$\begin{aligned}\hat{y}_i(t) =& f_i(y_1(t-1), y_1(t-2), y_1(t-3), y_1(t-4), y_1(t-5), ..., y_4(t-1), y_4(t-2),\\ & y_4(t-3), y_4(t-4), y_4(t-5), u_1(t-1), u_1(t-2), ..., u_4(t-1), u_4(t-2))\end{aligned} \tag{3.24}$$

Two Step Ahead

$$\begin{aligned}\hat{y}_i(t) =& f_i(y_1(t-2), y_1(t-3), y_1(t-4), y_1(t-5), y_1(t-6)..., y_4(t-2), y_4(t-3),\\ & y_4(t-4), y_4(t-5), y_4(t-6), u_1(t-1), u_1(t-2), ..., u_4(t-1), u_4(t-2))\end{aligned} \tag{3.25}$$

Three Step Ahead

$$\hat{y}_i(t) = f_i(y_1(t-3), y_1(t-4), ..., y_4(t-3), y_4(t-4), u_1(t-1), u_1(t-2), u_1(t-3), u_1(t-4), ..., u_4(t-1), u_4(t-2), u_4(t-3), u_4(t-4)) \quad (3.26)$$

Model	Forecast Horizon	Number of Hidden Layer	Number of Hidden Node
NARX- LSTM	1	1	256
	2	1	256
	3	1	128
NARX- biLSTM	1	1	25
	2	1	25
	3	1	25
NARX- GRU	1	1	128
	2	1	64
	3	1	128

Table 3.4.: Naural Networks Setting for Optimal RNN NARX Models

The comparison of RNN and NARX RNN models was performed based on two metrics: the average RMSE and individual RMSE for tide level prediction for each location. Table 3.5 shows the comparison results based on average RMSE for one step ahead, two steps ahead and three steps ahead forecasting. It can be seen that in both GRU and NARX GRU groups, GRU gives the best overall performance in comparison with LSTM and biLSTM for one step and three steps ahead. The performance of the biLSTM is the worst among the three models: the average RMSE values are larger than 0.5. The biLSTM forecast results are slightly improved when the NARX model is deployed in the RNN network, where the average RMSE values in the three different time horizons are below 0.45.

Model	Time Horizon		
	1	2	3
LSTM	0.2781	0.4894	0.4841
biLSTM	0.5653	0.7131	0.6285
GRU	0.1383	**0.1082**	0.4447
NARX-LSTM	0.1098	0.1385	0.1464
NARX-biLSTM	0.3736	0.2969	0.4114
NARX-GRU	**0.0955**	0.1209	**0.1379**

Table 3.5.: Comparison Performance of Multitask RNN and Multitask RNN in Terms of Average RMSE

The RMSE values of the individual models for tide level prediction at different locations are now compared. The RMSE values for three prediction time horizons across the four locations are shown in Figure 3.4. The following findings are presented:

1. For one step ahead prediction, the best model for Wick and Workington is NARX GRU with RMSE values of 0.0508 and 0.0888, respectively, while the optimal model for Ullapol and New Heaven is NARX LSTM.

2. The optimal model for two steps ahead prediction of the tide level in Wick, Workington, and New Heaven is GRU, while NARX-GRU gives the best results for Ullapol.

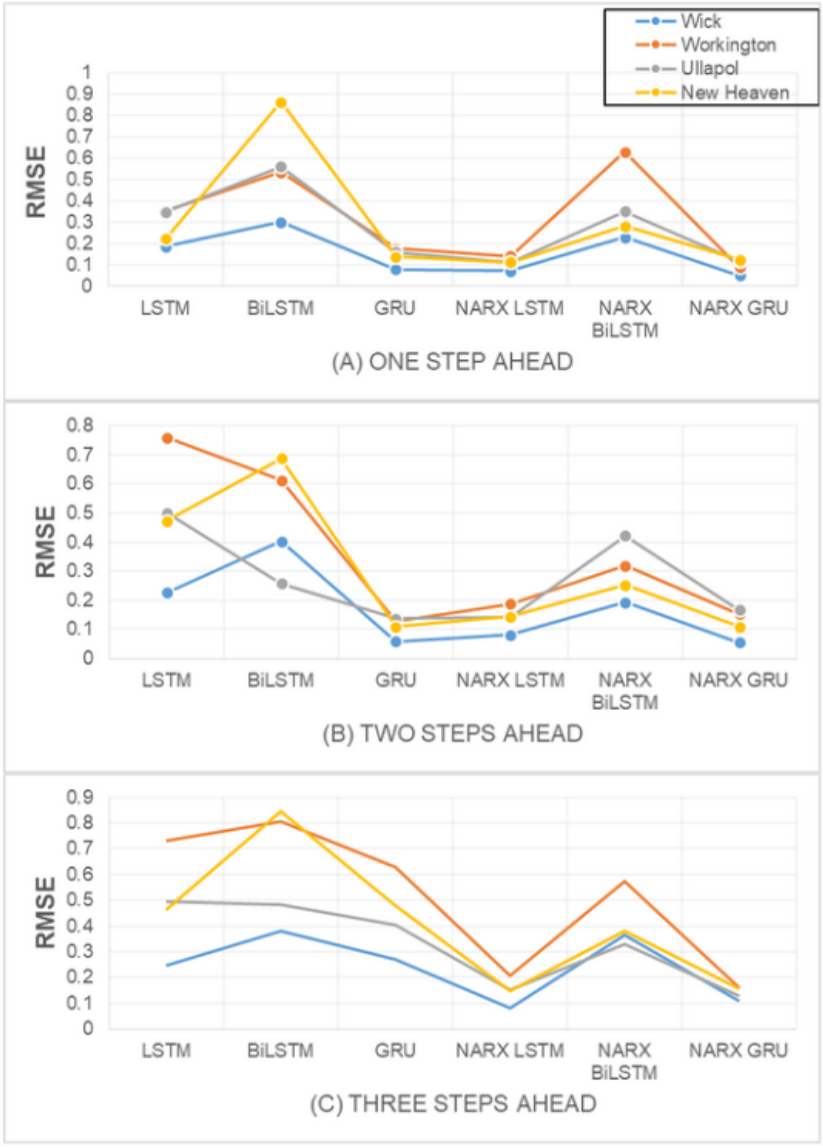


Figure 3.4.: Comparison performance of Multitask RNN and Multitask NARX for every location

According to the results shown in Figure 3.4, it can be noticed that the NARX LSTM model performs best for three steps ahead prediction in Wick and New Heaven, whereas NARX GRU performs best for the data at Workington and Ullapol. Similar to the case of using average RMSE, the biLSTM model demonstrates poor results in single models for almost all locations and time horizons for each location. The model performs good only in Wick for one step

ahead and two steps ahead forecasting.

Based on the RMSE values for all three different horizons and all four stations, it also can be seen that the RMSE values are not at the same level for the four locations. The location that has a relatively low and stable RMSE is Wick, while other locals have slightly higher RMSE values. This may suggest that one of the inputs or features can be detrimental to the performance of other features while using a multitask learning model.

The proposed models show the positive benefits of using several similar data to forecast the tide levels at different locations or stations simultaneously. However, combining NARX networks and the multitask learning models requires high computational costs. The multitask learning models need to be well designed; in doing so, it does not only need to find the best number of delays (for multitask learning NARX) but also the optimal number of hidden layers and hidden nodes.

3.4.1.3. Results and Discussion NARMAX LSTM Model

This subsection discusses a novel MTL framework based on NARMAX-LSTM. The optimal time lag in the multitask NARMAX –LSTM structure for each network was determined by using the lowest joint loss value obtained after performing empirical analysis. To find the best neural networks architecture, the parameter settings for all the network models are as follows: the iteration number for each time horizon case is set to be 200; the number of hidden layers, hidden nodes, and time lags for input and output are set to be from 1 to 5, 32 to 256, and 1to 4, respectively.

Table 3.6 summarizes the details of joint loss results for three different time horizons based on the best parameter setting. It was observed that for one step ahead prediction, the optimal setting for the multitask NARMAX-LSTM is as follows: the time lag for input is 2 (n_u), the time lag for output is 1 (n_y), and the time lag for noise is 1 (n_e). The hidden layer has a total of 128 nodes.

The one step ahead forecast model based on the optimal model in Table 3.6 can be written as follows:

$$\begin{aligned}\hat{y}_i(t) = & f_i(y_1(t-1), y_1(t-2), \ldots, y_4(t-1), y_4(t-2), u_1(t-1), \ldots, u_4(t-1), \\ & e_1(t-1), \ldots, e_4(t-1))\end{aligned} \tag{3.27}$$

For two step ahead case, the best model setting is as follows: the time lag for input is 4 (n_u), the time lag for output is 1 (n_y), the time lag for noise is 1 n_e, and there are 128 nodes in the

hidden.

$$\hat{y}_i(t) = f_i(y_1(t-2), y_1(t-3), ..., y_4(t-2), y_4(t-3), u_1(t-1), u_1(t-2), u_1(t-3), u_1(t-4), ..., u_4(t-1), u_4(t-2), u_4(t-3), u_4(t-4), e_1(t-1), ..., u_4(t-1)) \tag{3.28}$$

The best model for the three steps ahead horizon case is as follows: two lagged input variables, two lagged output variables, one lagged noise variable, and 64 nodes in the hidden layer. For three steps ahead prediction, the model is:

$$\hat{y}_i(t) = f_i(y_1(t-3), y_1(t-4), y_1(t-5), ..., y_4(t-3), y_4(t-4), y_4(t-5), u_1(t-1), u_1(t-2), ..., u_4(t-1), u_4(t-2), e_1(t-1), ..., u_4(t-1)) \tag{3.29}$$

Networks Lag	One Step Ahead			Two Step Ahead			Three Step Ahead		
	LSTM Setting		Joint Loss	LSTM Setting		Joint Loss	LSTM Setting		Joint Loss
	Hidden Node	Hidden Layer		Hidden Node	Hidden Layer		Hidden Node	Hidden Layer	
$(n_y = 1, n_u = 1, n_e = 1)$	4	64	0.3088	1	64	0.1287	1	64	0.1739
$(n_y = 1, n_u = 2, n_e = 1)$	**1**	**128**	**0.1542**	1	32	0.1607	2	32	0.2317
$(n_y = 1, n_u = 3, n_e = 1)$	3	32	0.3185	3	128	0.2273	2	32	0.2252
$(n_y = 1, n_u = 4, n_e = 1)$	1	32	0.2578	**1**	**128**	**0.1262**	2	128	0.1929
$(n_y = 2, n_u = 1, n_e = 1)$	2	64	0.2416	1	64	0.1607	5	64	0.2767
$(n_y = 2, n_u = 2, n_e = 1)$	1	32	0.1931	3	256	0.2340	**1**	**64**	**0.1546**
$(n_y = 2, n_u = 3, n_e = 1)$	1	64	0.1774	1	32	0.1798	3	128	0.2427
$(n_y = 2, n_u = 4, n_e = 1)$	2	64	0.2126	4	64	0.2326	1	256	0.1566
$(n_y = 3, n_u = 1, n_e = 1)$	2	64	0.2248	3	32	0.2697	3	64	0.2723
$(n_y = 3, n_u = 2, n_e = 1)$	2	32	0.2276	1	32	0.1633	1	32	0.2037
$(n_y = 3, n_u = 3, n_e = 1)$	1	64	0.1854	1	64	0.1693	2	256	0.1802
$(n_y = 3, n_u = 4, n_e = 1)$	1	64	0.1953	1	64	0.1621	2	128	0.2324
$(n_y = 4, n_u = 1, n_e = 1)$	3	32	0.2988	4	64	0.5914	1	32	0.2733
$(n_y = 4, n_u = 2, n_e = 1)$	1	32	0.2101	2	64	0.2340	1	128	0.1723
$(n_y = 4, n_u = 3, n_e = 1)$	2	32	0.2380	2	128	0.2083	3	64	0.2985
$(n_y = 4, n_u = 4, n_e = 1)$	3	64	0.2683	1	128	0.1731	3	64	0.2596

Table 3.6.: Networks Lag Performance For One Step Ahead, Two Steps Ahead and Three Steps Ahead Forecasting

By analyzing the time lag variation and the LSTM setting from three different time horizons in Table 3.6, it can be seen that the numbers of input time lag, output time lag, hidden layers, and hidden nodes have no correlations at all.

The increase of the network complexity, namely, the increase of hidden layers, number of nodes, and time lag, does not have a significant effect on the accuracy or joint loss value of

the models. In some cases, it needs a relatively larger number of hidden layers to generate the lowest joint loss value, while in other cases, it only needs a single hidden layer with a minimal number of hidden nodes. This means that manual intervention is still needed in the fine-tuning process of the neural network models.

The models for the above three different time horizons were tested using four new tasks, their performance were analyzed and summarized as follows. As can be seen in Table 3.7, individual loss values of the tasks (old and new) in the same time horizon are slightly different from each other. No task has an obviously larger loss value than other tasks. This means that the performance of some specific tasks does not significantly impact the performance of the models for other tasks. The joint loss value results show that both new and old tasks do not encounter overfitting.

Multitask LSTM-NARMAX with Old Tasks					
Horizon	Task 1	Task 2	Task 3	Task 4	Joint Loss
1	0.0981	0.2167	0.1492	0.1527	0.1542
2	0.1228	0.1356	0.1136	0.1331	0.1262
3	0.0973	0.2026	0.1716	0.1472	0.1546
Multitask LSTM-NARMAX with New Tasks					
Horizon	Task 1	Task 2	Task 3	Task 4	Joint Loss
1	0.1201	0.1717	0.1959	0.1608	0.1621
2	0.1504	0.1058	0.1462	0.103	0.1263
3	0.0896	0.1415	0.2817	0.129	0.16045

Table 3.7.: Comparison of Individual Loss Value of Old and New Tasks

To further assess the capability of the proposed multitask NARMAX-LSTM model, its performance was compared with LSTM, ARIMA, and Backpropagation models. The comparison results are shown in Figure 3.5, which clearly shows that in terms of joint loss, the multitask NARMAX LSTM outperforms all the other models in all tasks in three different time horizons.

From Figure 3.5, it can be seen that ARIMA severely suffers negative transfer: its RMSE values for Tasks 1 and 4 are much higher than those of the other models. The ARIMA model also shows the worst performance for all tasks and for all time horizons. Backpropagation also suffers negative transfer in all time horizons. The negative transfer occurs in Task 2 and Task 4 for the Backpropagation model. The LSTM model performs well in one step ahead, especially in tasks 1, 2, and 3, but it has poor results in other time horizons.

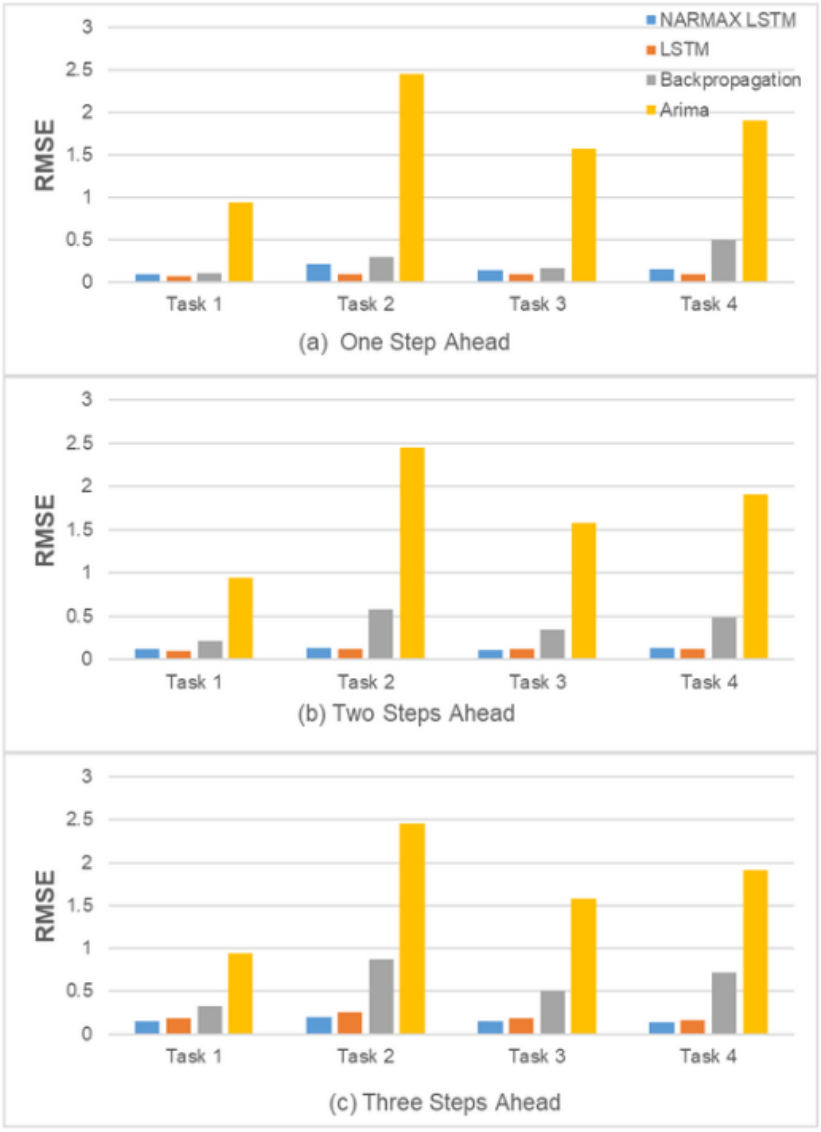


Figure 3.5.: Comparison of the joint loss of the NARMAX-LSTM, LSTM, Backpropagation, and ARIMA. (a) One Step Ahead (b) Two Steps Ahead (c) Three Steps Ahead

3.4.1.4. Comparison Between NARX LSTM and NARMAX LSTM

3.4.1.5. Summary

NARX RNN Model

A total of six multitask learning RNN models, namely, LSTM, bilSTM, GRU, NARX-LSTM, NARX-biLSTM, and NARX-GRU, have been investigated for tide level prediction from one to three steps ahead. All models were initialized with the same neural network setting, e.g. the number of hidden layers and the number of nodes in each hidden layer. It turned out that GRU in both individual RNN models and combined NARX-RNN models outperformed other mod-

els in terms of average RMSE. It was also observed that the prediction performances of the three individual RNN models, namely, LSTM, biLSTM and GRU, were improved when they were combined with the NARX model. In other words, the three hybrid models NARX-LATM, NARX-biLATM, and NARX-GRU perform better than LSTM, biLSTM, and GRU, respectively, for all the cases of the three-time horizons (i.e., 1-, 2-, and 3-steps ahead prediction). The study also found that the biLSTM and NARX biLSTM may not be suitable for predicting tide level with multitask learning models due to their poor performance for all three different prediction cases (three-time horizons).

NARMAX LSTM Model

This work proposed a novel multitask learning approach for time series forecasting by effectively integrating the NARMAX and LSTM models. Unlike traditional time series forecasting methods which follow a single model for a singletask practice, the proposed MLT-NARMAX-LSTM method deals with time series modeling and forecasting from a multitask viewpoint: it does not treat a single time series modeling and forecasting as an isolated task, but one of a set of tasks; the individual tasks are not the same but closely associated to each other (e.g., common features, similar change patterns, sharable knowledge, and information).

The proposed method has several advantages. For example, it generates robust models by using information shared by all the tasks; it avoids overfitting that can easily occur when using traditional time series modeling methods; and the combination of the two different types of modeling methods, namely, NARMAX and LSTM, makes the proposed modeling framework immune to negative transfer. The predictive capability of MTL-NARMAX-LSTM network models was tested and evaluated on several real datasets of tide levels. Experimental results confirm the good performance of the proposed method.

By analyzing the NARX LSTM and NARMAX LSTM forecast results in three different forecast horizons, it can be seen that models with nonlinear performance are better than those without NARX or NARMAX. Nonlinear models show better results in one-step-ahead, two-step-ahead, and three-step-ahead forecasts because these approaches not only include past data but also output from the previous forecast. In NARMAX, the input for the model also includes noise, which is assumed to be able to read additional information for the model. The source of input from the previous output and noise gives the model more information while forecasting the tide level. More information means the model has additional knowledge to deliver more accurate results.

3.4.2 Data Set 2

Data set 2 concerns a multitask problem: forecasting the tide level in five stations (five tasks) simultaneously, namely, Harwich, Lerwick, Millport, Portrush, and Weymouth. Experimentation on different numbers of 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 15-minute sampling period.

3.4.2.1. Parameters Setting

The network hyper-parameters were determined through simulations by testing a set of parameters shown in Table 3.8.

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	$n_y = 1,2,3,4, n_u = 1,2,3,4, n_e = 1$

Table 3.8.: Parameter Setting

3.4.2.2. Results and Discussion

NARX and NARMAX are compared using three different RNN models with a variety of time lags. Two optimizers, namely stochastic gradient descent with momentum (SGDM) and adaptive moment estimation (ADAM), are used to optimize the associated models. The joint loss value or average value of RMSE, based on the combinations of the candidate parameters presented in Table 3.8, 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. Both NARX-RNN-MTL and NARMAX-RNN-MTL models were then established based on the values of the root mean square error (RMSE) of five tasks. Three RNN structures, namely, LSTM, BiLSTM, and GRU, were used to build the NARX-RNN-MTL and NARMAX-RNN-MTL models.

The performance of three NARX-RNN-MTL models with 16 specific lags, trained with SGDM and ADAM for the five tasks, was compared. The comparison is based on joint loss, or aver-

age RMSE value, indicating the performance of each model. The details of these experimental settings are shown in Table 3.9. The findings from the comparison are as follows:

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 3.9.: Performances of Identified 16 NARX-RNN-MTL Models

- 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.

- The distribution of average RMSE for NARX GRU and LSTM GRU using SGDM are relatively small compared to NARX GRU and NARX LSTM using ADAM. From Table 3.9, 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 GRU and NARX LSTM using ADAM, the values are 0.15848, 0.25016, 0.17758 and 0.35830, respectively.

- 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 (ii) BiLSTM using SGDM and ADAM.

- For NARX BiLSTM, the relatively lower RMSE distribution is gained by using the ADAM optimizer.

Table 3.10 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 3.10 it can bee seen that SGDM works better than ADAM for the NARMAX-GRU and NARMAX-LSMT models, whereas ADAM outperforms SGDM for the NARMAX-BiLSMT model.

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 3.10.: Performances of Identified 16 NARMAX-RNN-MTL Models

Table 3.11 illustrates the optimal settings for the NARX-RNN and NARMAX-RNN models and their performances. The NARX-GRU and NARMAX-GRU models with smaller lags, trained with SGDM, perform better than other models.

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 3.11.: The optimal settings of the NARX-RNN-MTL and NARMAX-RNN-MTL models and their performances

3.4.2.3. Baselines

In order to validate the overall performance and effectiveness, the proposed method was tested and compared with the following baseline methods, including STL and MTL, without using the NARX or NARMAX model:

- MTL implemented with GRU, LSTM and BiLSTM using SGDM and ADAM Optimizer
- 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 3.12, where the lowest average RMSE value is for MTL using NARMAX and GRU with SGDM optimizer. Based on the 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 also has a 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 3.12. shows that all the baseline models using SGDM deliver better results than using ADAM; the SGDM optimizer has the best performance measured in either average RMSE or individual RMSE.

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

Table 3.12.: Comparison of RMSE values of of different models

Figure 3.6 provides a graphical illustration of joint loss, measured as the average RMSE of different models. The bar plots show that GRU networks perform better than the other two RNN variants (LSTM and BiLSTM).

To further evaluate the performances of NARMAX-GRU models for more tasks, experiments were conducted in which six additional tasks of predicting tide levels at stations 6-11 were included and performed simultaneously 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 3.7, 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.

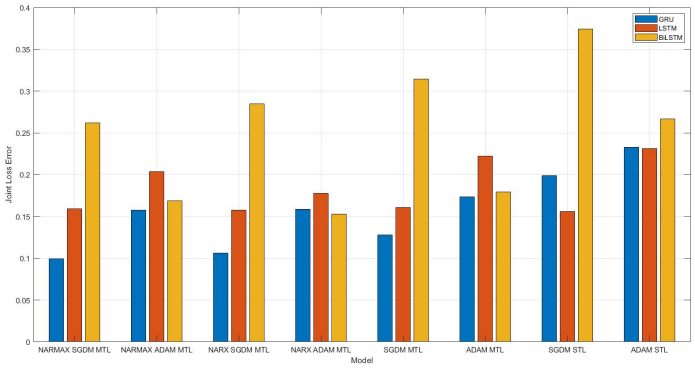


Figure 3.6.: Comparison of joint loss error of NARMAX MTL, NARX MTL, MTL and STL

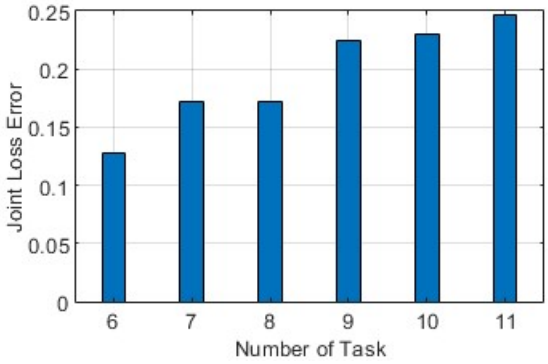


Figure 3.7.: Joint loss error of MTL NARMAX with many tasks

3.4.2.4. Summary

This study proposes a new class of NARMAX-RNN models, namely, NARMAX-LSTM, -BiLSTM, and –GRU, combined with MTL learning for simultaneous multiple tide level forecasting. 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 value 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 value is 0.09961. The results showed that NARMAX-GRU outperformed its counterpart NARX-GRU.

The model performances were also compared 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 the multitask 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 model variable and

the number of hidden nodes, to build the best models. In the future, MTL models will be designed to better fine-tune the training process using transfer learning. In addition, the data used model for model training and forecasting are not only univariate but also multivariate and multidimensional.

4

Attention Soft Parameter Sharing Multitask Learning using Nonlinear Autoregressive Moving Average Model with Exogenous Inputs

4.1. Introduction

This chapter conducted two multitask learning with a soft parameter sharing framework. The first model is pure soft parameter sharing, where the sharing mechanism is through input and parameter. The second model is mixed between hard and soft parameter sharing. The sharing mechanism in the second model is not only via input and parameters but also via a specialized sharing network. The soft and mixed parameter sharing models are built using XCeption Convolution Neural Network, BiLSTM, and NARMAX. The performance of the two proposed models was further evaluated on two forecasting datasets. Although convolution neural networks and recurrent neural networks have been widely employed in significant wave height forecasting, these applications mainly focus on singletask forecasting.
In summary, the main contributions are listed as follows:

- A novel significant wave height forecasting system based on neural networks and a nonlinear model, which improves the forecasting accuracy and avoids negative transfer

effect and unequal loss distribution.

- Extensive comparative experiments were conducted on our proposed model, confirming the effectiveness of the combination of NARMAX and multitask learning.

4.2. Significant Wave Height Forecasting

As one of the main challenges for human and environmental sustainability, climate change become world attention. One major solution to this issue is transformation in energy sources. The world no longer depends on fossil fuels, and it has been transformed into renewable energy. This new approach offers clean, sustainable, and affordable solutions that can reduce the impact of climate change [103].

Wave energy, which has high power density [104, 105], is a potential renewable energy that can be an alternative to solve the world energy crisis. One of the most vital parameters of energy waves is significant wave height. Therefore, past, present, and future information about significant wave height is important to be managed. Future information about significant wave height can be delivered through the forecasting process. The machine learning approach has plenty of choices to do this forecasting task. The use of machine learning, especially in the context of forecasting tasks, has already been investigated and studied.

In [106], Wei forecasts significant wave height, average wave period, and mean wave period using eight input variables by applying 2D convolutional neural networks and LSTM. This work uses multiple inputs and multiple outputs methods in their models. The extreme Learning model with multiple inputs and multiple outputs is employed for forecasting significant wave height and energy flux prediction [107] and uses ten variables as their inputs. Another multiple input and multiple output study was conducted by Bento et al. [108]. This model uses feedforward neural networks to forecast significant wave height, energy flux, and energy period.

Bai et al. [109] develop 2D convolutional neural networks with the random search algorithm to optimize the hyper-parameters of the CNN neural networks to forecast significant wave height. The proposed model uses two components as their input variables and aims to forecast data in nine different buoys by using singletask learning model. One popular model for forecasting is LSTM; in [110, 111], LSTM is used to forecast significant wave height in several stations using historically significant wave height as their input variables. Both proposed model uses a single-learning concept in their work.

Yevnin et al. [112] test their using LSTM and GRU neural network to forecast significant wave

height by using six variables as inputs. The LSTM and GRU models use a singletask learning model to forecast three buoy locations. In [113], Yang et al. combine seasonal-trend decomposition procedures based on loess (STL) using 1D convolutional neural networks and positional encoding (PE) to forecast significant wave height. The proposed model uses historical data and a singletask learning approach to forecast significant wave height in three stations. Numerous studies have been done to forecast significant wave height, but most were by applying singletask learning or multiple inputs and multiple outputs methods, none of them, up to the authors' knowledge, considers multitask learning models combined with NARMAX.

4.3. Methodology

4.3.1 Soft Parameter Sharing Network Architecture

The proposed soft parameter sharing multitask learning consists of three layers with the following components :

- The first layer is task-specific input layer. This layer manages specific input into a specific network. The number of networks in this layer equals the number of tasks. This study applied three tasks to the network so that this layer would have three networks, each based on Xcpetion and BiLSTM neural networks. Each network will have three outputs, and this output will be inputted into the next layer.
- The second layer is also a specific layer called the middle layer, and the number of networks is equal to the number of networks in the previous layer. The middle layer is built using XCeption, BiLSTM, and the NARMAX model. The purpose of adding NARMAX is to increase the number of past information entered into the networks. The input for the middle layer is one output from every previous network. For the output, the middle layer only has one output for each network, and this input will be fed into the last layer or output layer.
- The final or output layer is built based on Xcpetion, BiLSTM and Attention Neural Networks. This layer only has one input and one output for each specific network.

Architecture for soft parameter sharing is shown in Figure 4.1

4.3.2 Mixed Paramter Sharing Network Architecture

The proposed mixed parameter sharing multitask learning consists of three layers with the following components :

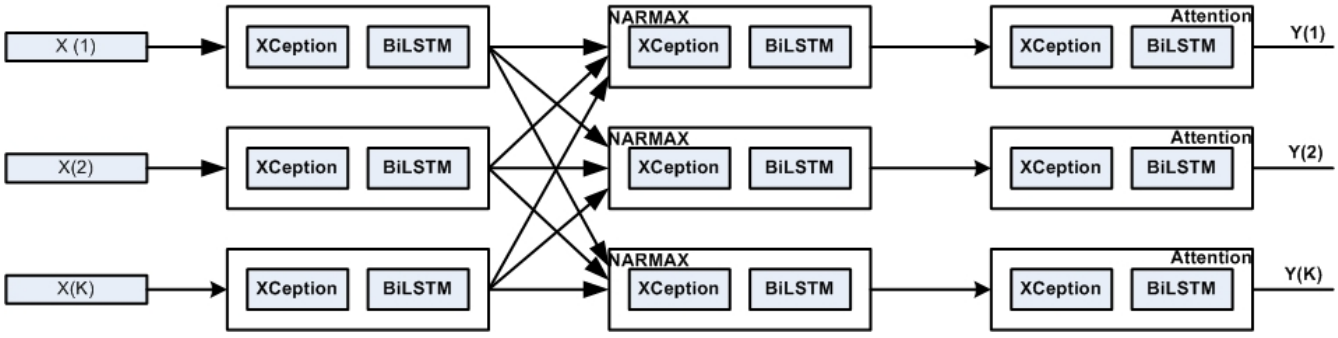


Figure 4.1.: Soft Parameter Sharing Multitask Learning Architecture

- The specification for the first layer is similar to the soft parameter input layer. The layer also has a task-specific input layer, and the number of networks equals the number of tasks. The input layer in the mixed parameter sharing approach used Xception and BiLSTM in each network. All output for this layer will be the input for the next layer or networks.
- The middle layer for the mixed parameter sharing model only consists of one network and is built based on XCeption, BiLSTM, and NARMAX. The output from this network is equal to the number of tasks.
- Similar to soft parameter sharing, the output layer for this approach is also based on Xcpetion, BiLSTM, and Attention Neural Networks. This layer consists of an output-specific layer to accommodate each task.

Architecture for mixed parameter sharing in Figure 4.2.

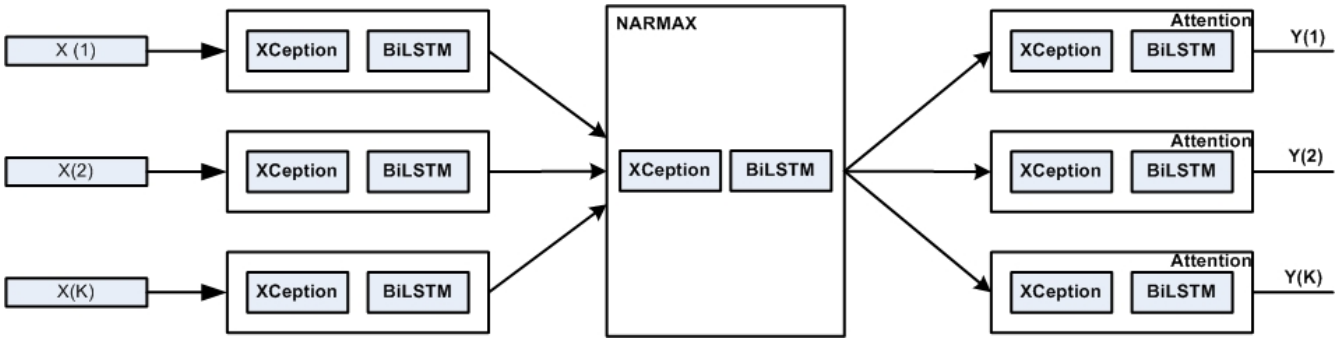


Figure 4.2.: Mixed Parameter Sharing Multitask Learning Architecture

4.3.3 Data Normalization

Normalization is a process of adjusting or transforming data into a specific or common scale. The purpose of data normalization is to increase comparability or balance between the datasets and avoid any specific variable or feature from dominating the analysis. In

this study, observation data was transformed using z-score normalization. The formula for z-score normalization for a feature x is defined as:

$$z = \frac{(x - \mu)}{\sigma} \tag{4.1}$$

where

$$\mu = \frac{1}{n} \sum_{i=i}^{n} x_i \tag{4.2}$$

where

$$\sigma = \sqrt{\frac{\sum_{i=i}^{n} (x_i - \mu)^2}{N}}. \tag{4.3}$$

where x is observation, μ is mean, σ is sigma, x_i is data points, and N is the total number of data points

4.4. Experiments

This section presents the experiment scenario for mixed and soft parameter sharing models. The detailed information on the datasets used in this chapter is introduced first. The input network selection process is then introduced. Finally, details of the training setting for the proposed models are shown.

4.4.1 Dataset

Experiments are conducted on two datasets:

The first dataset is used to forecast three standard meteorological data sets in Grays Reef, and nine variables are used for their inputs. This data set consists of 12.000 datasets. The variables are as follows:

1. Average wave period (seconds).
2. Air temperature (celsius).
3. Dew point temperature (celsius).
4. Mean Wave Direction (degrees).
5. Sea level pressure (hPa).
6. Wind direction (m/s)

7. Wind speed (m/s)
8. Sea surface temperature (celsius).
9. Significant wave height (meters)

The output of the systems or tasks to be solved using the first dataset is to forecast (1) significant wave height (WVHT), (2) average wave period (APD), and (3) mean wave direction (MWD).

The second data set is used to forecast three stations/locations simultaneously and uses ten variables for their inputs. The second dataset consists of 26.304 data. The variables are as follows:

1. Average wave period (seconds).
2. Air temperature (celsius).
3. Wind Gust (m/s).
4. Mean Wave Direction (degrees).
5. Sea level pressure (hPa).
6. Wind direction (m/s)
7. Wind speed (m/s)
8. Sea surface temperature (celsius).
9. Significant wave height (meters)
10. Dominant Wave Period (second)

The output of the systems or tasks to be solved using the second dataset is to forecast significant wave height in three different locations (Grays Reef, Canaveral, and Frying Pan Shoals).

Both datasets are time series of standard meteorological data recorded every 15 minutes. This study focuses on one step ahead of forecasting so that the model will predict the next interval of 15 minutes. The datasets were extracted from the National Data Buoy Center (NDBC) website https://www.ndbc.noaa.gov/to_station.shtml.

4.4.2 Input Network Selection

The input network is the network that receives input and is located in the first layer of the network. This network plays an important role because if it delivers poor output, it will affect the next layer results. Five convolutional and two recurrent neural network models are investigated to find the best input network for mixed and soft parameter sharing networks. Input networks with a combination of convolutional neural network models and recurrent neural networks are as follows:

1. AlexNet with GRU or BiLSTM
2. LeNet with GRU or BiLSTM
3. GoogleNet with GRU or BiLSTM
4. MobileNet with GRU or BiLSTM
5. XCeption with GRU or BiLSTM

4.4.3 Training Setting

All experiments utilize 32 hidden nodes, a learning rate of 0.05, and a total of 300 epochs. The whole network is trained through stochastic gradient descent with momentum (SGDM). The output lag n_y, input lag n_u, and noise lag n_e for both models are set to 3.

4.5. Results and Discussions

This section first evaluates four convolutional and two recurrent neural network models to identify the best network for input. Then, the performance of mixed and soft parameter sharing with NARMAX is compared. The comparison includes singletask learning and mixed and soft parameter sharing without NARMAX. The performance comparison of both proposed models is reflected by the average and individual RMSE metrics.

4.5.1 Input Network Selection

For input network selection, combinations of convolutional neural networks with GRU or BiLSTM are tested and compared. Figures 4.3 and 4.4 show the performance comparison of the input network based on average RMSE and individual RMSE, respectively.

From Figure 4.3, it can be observed that XCeption using GRU or BiLSTM is the best model. XCeption using GRU and BiLSTM has an average RMSE of 0.210308 and 0.206751, while

single RNN (GRU and BiLSTM), AlexNet RNN, LeNet RNN, GoogleNet RNN, and MobileNet RNN produce an average RMSE of more than 0.28.

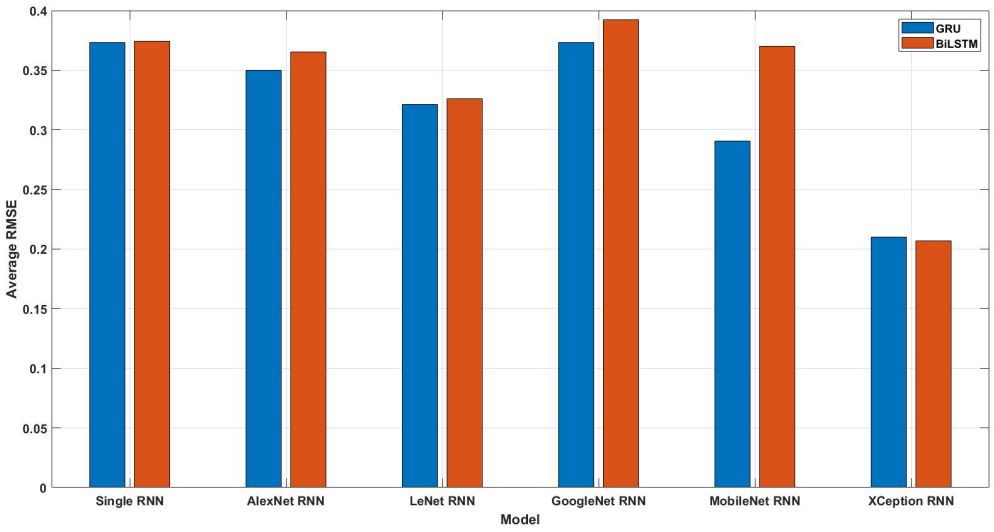


Figure 4.3.: Performance Comparison of Input Network Based on Average RMSE

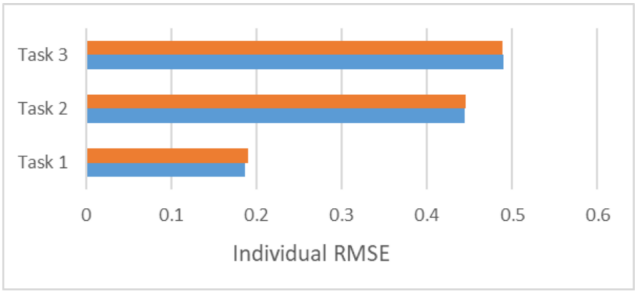
As shown in Figure 4.4 (a), GRU and BiLSTM performance for Task 1 is better compared with RMSE under 0.2, while Task 2 and Task 3 have RMSE more than 0.4. AlexNet with GRU or BiLSTM performance for Task 1 is worse than all models with an RMSE of more than 0.22. The RMSE results for Task 2 and Task 3 with this model are also no better than Task 1, with an RMSE of more than 0.39.

The performance of LeNet with RGU or BiLSTM is slightly better than AlexNet with GRU or BiLSTM. The RMSE for Task 1 is 0.213 for LeNet with GRU and 0.217 for LeNet with BiLSTM. LeNet with GRU for Task 2 and Task 3 is 0.365 and 0.365071, respectively. For Task 2 and 3, LeNet with BiLSTM delivers RMSE equal to 0.385 and 0.403, respectively.

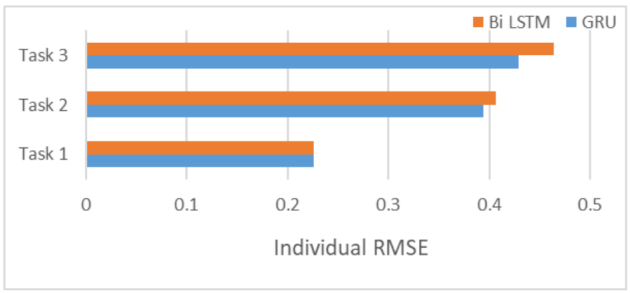
For Task 1, GoogleNet with GRU or BiLSTM performs better than AlexNet, with RMSE 0.205 for GRU and 0.209 for BiLSTM. In contrast to their Task 1 result, Task 2 and Task 3 using either GRU or BiLSTM performed worse than other models, with RMSE 0.443 and 0.4714 for GoogleNet with GRU and 0.4725 and 0.494 for GoogleNet with BiLSTM.

MobileNet, when applied with GRU, delivers better results than BiLSTM. XCeption with GRU or BiLSTM achieved the best RMSE results compared to all models. For Task 1, Task 2, and Task 3, the RMSE value for XCeption with GRU is 0.0991, 0.285, and 0.2461, respectively. XCeption with BiLSTM, compared with XCeption with GRU, achieved better performance in all tasks with RMSE is 0.0971, 0.268, and 0.255, respectively. Based on these results, XCeption

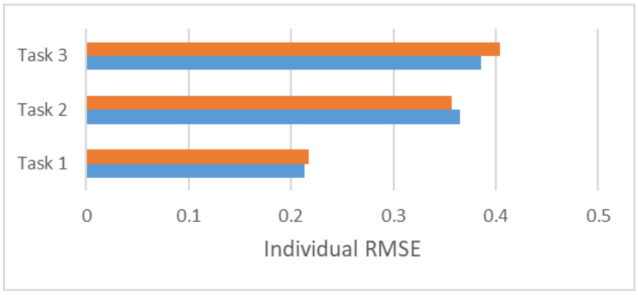
with BiLSTM was used as our input network.



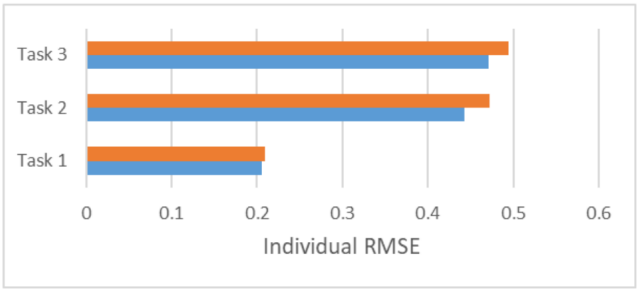
(a) Single RNN Individual RMSE



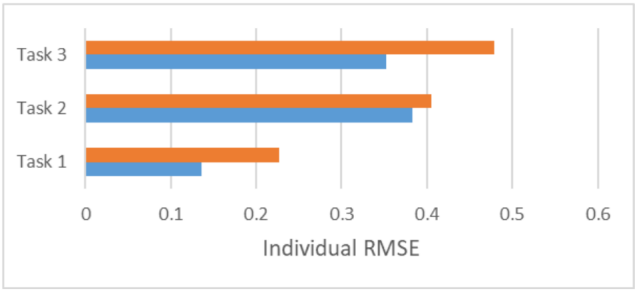
(b) AlexNet RNN Individual RMSE



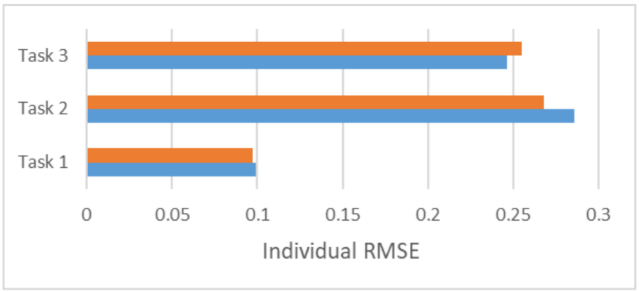
(c) LeNet RNN Individual RMSE



(d) GoogleNet RNN Individual RMSE



(e) MobileNet RNN Individual RMSE



(f) XCeption RNN Individual RMSE

Figure 4.4.: Performance Comparison of Input Networks Model Based on Average RMS

4.5.2 Mixed and Soft Parameter Sharing Performance

As shown in Table 4.1, two proposed models using NARMAX outperform singletask learning and multitask learning models without NARMAX in all datasets. For dataset 1, the optimal model is Mixed Parameter Sharing Multitask Learning NARMAX with a loss equal to 0.179272, and for dataset 2, Soft Parameter Sharing Multitask Learning NARMAX delivers the best result with RMSE equal to 0.088589. All models using mixed and soft parameter sharing show better results than singletask learning except for mixed parameter sharing without NARMAX in dataset 1.

Next, based on individual loss, soft and mixed parameter sharing are compared with singletask and multitask approaches without NARMAX. The individual task loss for dataset 1 is shown in Table 4.2. Table 4.2 shows that mixed parameter sharing is the only model in

Model	Dataset 1	Dataset 2
Singletask Learning	0.206751	0.102468
Mixed Parameter Sharing Multitask Learning	0.212074	0.098046
Mixed Parameter Sharing Multitask Learning NARMAX	**0.179272**	0.094526
Soft Parameter Sharing Multitask Learning	0.203452	0.090384
Soft Parameter Sharing Multitask Learning NARMAX	0.194847	**0.088589**

Table 4.1.: Comparison of Mixed and Soft Parameter Sharing Multitask Learning with singletask Learning and

Model	Dataset 1		
	Task 1	Task 2	Task 3
Singletask Learning	0.097119763	0.2680003	0.255134
Mixed Parameter Sharing Multitask Learning	0.082386769	0.27475628	0.27908
Mixed Parameter Sharing Multitask Learning NARMAX	**0.072709903**	**0.24291776**	**0.222189**
Soft Parameter Sharing Multitask Learning	0.080267876	0.25686762	0.273221
Soft Parameter Sharing Multitask Learning NARMAX	0.069742292	0.25509876	0.259702

Table 4.2.: Individual Loss of Three Task for Dataset 1

which all individual loss performance outperforms singletask learning. Compared to the singletask model, the soft parameter model with and without NARMAX only delivers better individual loss for Task 1 and Task 2. The RMSE results for Task 3 using soft parameter sharing multitask learning NARMAX are slightly worse than those for the singletask model.

The performance of individual loss of dataset 2, as shown in Table 4.3, using mixed parameter sharing multitask learning NARMAX, soft parameter sharing multitask learning, and soft parameter sharing multitask learning NARMAX in all tasks is better than singletask learning. The loss gap between tasks is also relatively small, which indicates that these three models are able to avoid negative transfer and maintain the loss balance between tasks.

Model	Dataset 2		
	Task 1	Task 2	Task 3
Singletask Learning	0.169046	0.067288	0.071069
Mixed Parameter Sharing Multitask Learning	0.169602	0.0604575	0.06408
Mixed Parameter Sharing Multitask Learning NARMAX	0.152556	0.0638525	0.067171
Soft Parameter Sharing Multitask Learning	0.145516	0.060706	0.064931
Soft Parameter Sharing Multitask Learning NARMAX	**0.143373**	**0.061243**	**0.061153**

Table 4.3.: Individual Loss of Three Task for Dataset 2

By comparing the results from Dataset 1 and Dataset 2, it can be seen that Dataset 2 delivers lower RMSE results in individual loss and joint loss compared to Dataset 2. These results not only appear in soft parameter sharing but also mixed parameter sharing and singlet ask learning models. The significant difference in results occurs because the expected output from Dataset 1 is more complex than that of Dataset 2. The output for Dataset 1 is 1) significant wave height (WVHT), (2) average wave period (APD), and (3) mean wave direction (MWD). In contrast, the output for Dataset 2 is only one variable (significant wave height). Output for Dataset 1 consists of three different variables with three different measurement units: meters, seconds and degrees. Different in unit mean, each variable can have a different range number of output, which can cause the model to work harder to narrow the distance between tasks. On the other hand, the Dataset 2 event though has three outputs, but all of them have the same unit, which causes the model to work less to find the similarity between task or output.

4.6. Summary

Two multitask learning approaches were proposed by implementing soft parameter sharing and mixed parameter sharing approaches. The first model is soft parameter sharing NARMAX, and the other is mixed parameter sharing. While the soft parameter approach has a fully independent network, the mixed model has an independent network and a sharing network in its network. Both models are performed in two datasets. The mixed parameter sharing multitask learning with NARMAX outperforms all models for dataset 1, while soft parameter sharing multitask learning with NARMAX outperforms all models for dataset 2.

From the results and discussion,it was found that multitask learning with mixed or soft parameter-sharing with NARMAX techniques not only improves forecasting accuracy but is also capable of avoiding a negative transfer. For mixed and soft parameter sharing, multitask learning without NARMAX using dataset 1 and dataset 2 shows better average results than singletask learning but fails to avoid negative transfer because one of the tasks gains worse results than singletask learning.

5

Gated Multitask Learning with Nonlinear Autoregressive Moving Average Model with Exogenous Inputs

5.1. Introduction

This chapter introduces Gated Multitask Learning using NARMAX and applies a hard parallel mixture of expert [73]. The proposed model is an extended model of soft parameter sharing presented in Chapter 4. Like soft parameter sharing, the gated model concept only shares the input and parameter. Although numerous studies have been conducted on significant wave height forecasting, none of them, up to the authors' knowledge, considers the application of a mixture of expert models combined with NARMAX. The goal is to apply gated and NARMAX models to enhance the accuracy of significant wave height forecasting, prevent negative transfer, and equally distribute the loss value.

In summary, the main contributions are listed as follows:

- Proposed a novel gated multitask learning with NARMAX and compared it with a singletask learning model and gated multitask learning without NARMAX.
- The performance of Gated Multitask Learning is demonstrated by using a nonlinear model and convolutional neural network to forecast significant wave height.

- A comparative study is done to evaluate the effectiveness of NARMAX in the gated multitask learning model

The forecasting case in this chapter refers to dataset 1, mentioned in Chapter 4.

5.2. Methodology

5.2.1 Experts Networks Input Distribution

In a mixture of experts, the input to be fed to the expert is randomly sampled from training data and the class. This study adopted a hard mixture of experts [73], where each expert trains individually without involving a gate network. Because of this, the input sampling or data distribution to experts has to be done independently.

One of the sampling modeling methods is bootstrapping. Bootstrapping or bootstrap sampling is a sampling method that randomly samples the data with replacement. Replacement means the sample chosen from the dataset can be chosen again [31]. Bootstrapping has difficulties dealing with time series data because it has to manage the non-stationarity and autocorrelation in time series [114]. To solve this issue, Liu and Singh [115] and Künsch [116] proposed a new sampling method called the moving block bootstrap unlike bootstrapping, which creates a new group of samples by taking random data observations from the original sample, moving block bootstrap only sampling within a row of formed groups or blocks. The purpose of this process is to preserve the original structure of the time series, which is where the observations have to be formed in specific time intervals. Moving block bootstrapping of a time series can be seen in Fig 5.1.

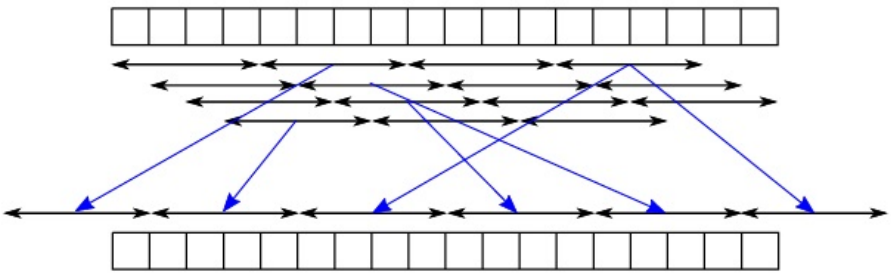


Figure 5.1.: Moving Block Bootstrapping of A Time Series Reprinted from [114]

5.2.2 Gated Multitask Learning with NARMAX Model

In this section, a mixture of experts with NARMAX is proposed for multitask modeling to forecast significant wave height. Figure.5.2 illustrates the graphical architecture of the proposed

model.

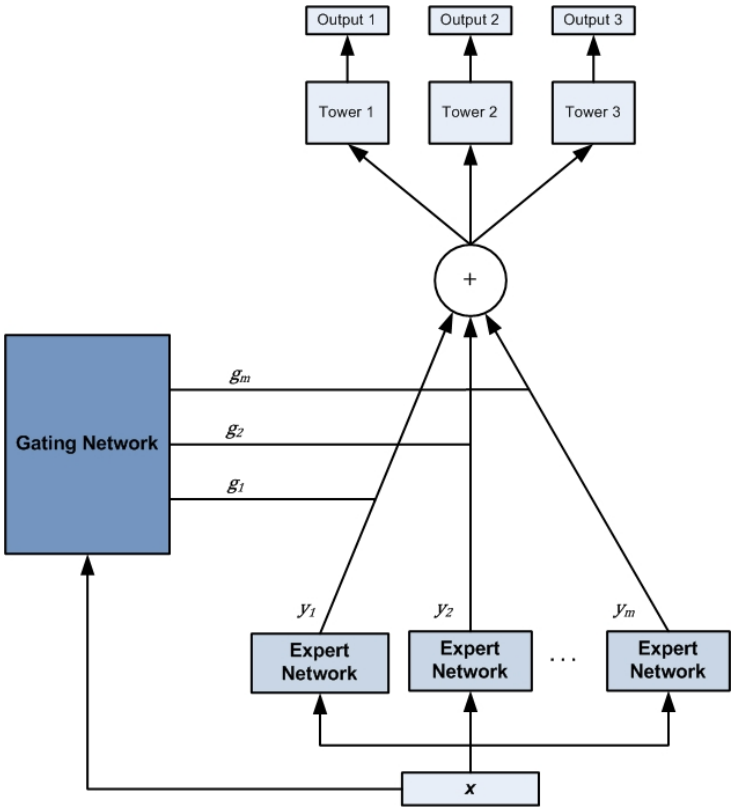


Figure 5.2.: Gated Multitask Learning

Gated Multitask Learning with NARMAX is composed of the following components.

- Experts Networks
  This network consists of seven experts. Unlike the classic expert network approach, which uses the same model or neural network for each expert, this method employs four different networks to create seven experts. Our proposed experts consist of two Xception CNN and GRU networks, two Xception CNN and LSTM networks, two Xception CNN and BiLSTM networks, and a single Xception CNN and LSTM Projected Layers network. The input for the expert is manage by using the moving block bootstrap method explained in subsection 5.1

- Gating Networks
  The gating network is built using Xception CNN followed by BiLSTM. In this network, NARMAX is implemented to enhance its sharing capability and all the data is used as input for this network.

- Tower Networks
  As the last network in the architecture, the tower network is specialized for each task. Three Backpropagation neural networks are used for the three specific tower networks.

Mathematically, given a data sequence with t time steps $x = (x_1, x_2, ..., x_t)$. For a given time step t for input (x_t), :

$$y^K_{(t+1)} = h^K_{Backpropagation}(f^K(x_t)) \tag{5.1}$$

where

$$f^K(x_t) = \sum_{i=1}^{m} (g^K x)_i f_{XCeptionRNNi}(x_t) \tag{5.2}$$

where $h^K_{Backpropagation}$ is the tower network for task K, f^K is the output of the gated mixture of experts layer, $f_{XCeptionRNNi}$ is the $i-th$ is the expert (there are m experts in total), g^K is gating network. RNN in XCeptionRNN are GRU, LSTM, BiLSTM, and LSTM Projected layer. The gating network define as

$$\begin{aligned} g^K(x) =& softmax(W_{XCeptionBiLSTM}(y_1(t-1), ..., y_1(t-n_y), ..., (y_m(t-1), \\ & ..., y_m(t-n_y), u_1(t-1), ..., u_1(t-n_u), ..., u_r(t-1), ..., u_r(t-n_u), \\ & e_1(t-1), ..., e_1(t-n_e), e_m(t-1), ..., e_m(t-n_e)) \end{aligned} \tag{5.3}$$

where

$W_{XCeptionBiLSTM}(y_1(t-1), ..., y_1(t-n_y), ..., (y_m(t-1), ..., y_m(t-n_y), u_1(t-1), ..., u_1(t-n_u), ..., u_r(t-1), ..., u_r(t-n_u), e_1(t-1), ..., e_1(t-n_e), ..., e_m(t-1), ..., e_m(t-n_e)$

is gating function using XCeption followed by BiLSTM applied with NARMAX model, $y(t), u(t)$ 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.
In this work, the gated mixture model is trained to minimize the cost function, which is the sum of squared losses:

5.2.3 Model Training

A classic mixture of experts models trains the experts alongside the gating, but in [73], Colobert et al. proposed a hard nonprobabilistic mixture where each expert is trained independently on a random subsets dataset. This train mechanism is used on large datasets that implement complex models. This training approach will be adopted to train our proposed model. The training process is as follows :

a) Sampling training data using moving block bootstrap

b) Independently train each expert $f_{XCeptionRNNi}$ over one of sampling subset data

c) Train the fixed experts $f_{XCeptionRNNi}$ with the gater g^K and tower $h^K_{Backpropagation}$ to minimize L on the whole training set.

L is a joint loss function, which refers to equation 3.24 in chapter 3.

5.3. Experiments

To find the ideal expert for the gated mixture of experts model is found through experimentation with six convolutional neural network models and four recurrent neural network models. The experiments include an expert model with only one neural network and an expert architecture with two neural networks.

The expert's model with only one neural model uses GRU, LSTM, BiLSTM, or LSTM Projector Layer (LSTM P). Expert with two neural networks is a combination of convolutional neural network models and recurrent neural networks is as follows:

1. AlexNet with GRU, LSTM, BiLSTM or LSTM P
2. LeNet with GRU, LSTM, BiLSTM or LSTM P
3. ResNet50 with GRU, LSTM, BiLSTM or LSTM P
4. VGG 16 with GRU, LSTM, BiLSTM or LSTM P
5. VGG 19 with GRU, LSTM, BiLSTM or LSTM P
6. XCeption with GRU, LSTM, BiLSTM or LSTM P

As previously mentioned, the data was sampled with the moving block bootstrap method, so it is important to choose the block's size. Determining the block's size can be challenging because a shorter block length can deliver more precise predictions, while a longer block data has more complete information. This study conducted seven variations [96, 72, 48,24, 13, 6, 3] of block length to validate our expert network.

The experts and gating are trained with SGDM using 32 hidden nodes; the learning rate is set to 0.05 with 32 hidden nodes. The Tower network is a build-based Backpropagation neural network with 25 hidden nodes, and the learning rate is set to 0.01. The best combination of input, output and noise lags is found by experimenting with varying lags [1, 2, 3]. The optimal number of experts in the Gated Mixture model is determined by experimenting with the number of experts, starting from 4 to 8.

5.4. Results and Discussion

In this section, several framework settings for our model are evaluated to find the ideal gated multitask learning architecture. First, a detailed experiment on several expert models is introduced. The average RMSE and individual RMSE values from each expert are used to determine which experts to include in our model. Following that, the optimal block size is

discussed. The average RMSE value from each expert is used to decide the optimal block length for sampling the observation data. Next, several lag networks are compared based on the average RMSE on joint loss to determine the ideal output lag (n_y), input lag (n_u) and noise lag (n_e). Finally, to validate our proposed models, comparisons are made between (1) the singletask learning model, (2) the Gated multitask learning model without NARMAX, and (3) the Gated multitask learning model with NARMAX. All comparisons are based on average RMSE and individual RMSE (RMSE for each task) and use the best settings discovered in the previous experiment.

5.4.1 Experts Selection

Four single expert models using recurrent neural networks (RNN) were compared with six convolutional neural networks (CNN) combined with expert models of recurrent neural networks. Figure 5.3 shows the performance comparison of experts based on average RMSE, and Figure 5.4 shows the performance comparison of experts based on individual RMSE.

As shown in Figure 5.3, the performance of all models, with Xeception, using GRU or LSTM or BilSTM or LSTM Projected Layer, outperforms other models. Experts with worse performance are VGG 19 combined with all RNN models and VGG 16 combined with GRU or BiLSTM, with an average RMSE of more than 4. Figure 5.4 shows that Single RNN, AlexNet RNN, LeNet RNN,

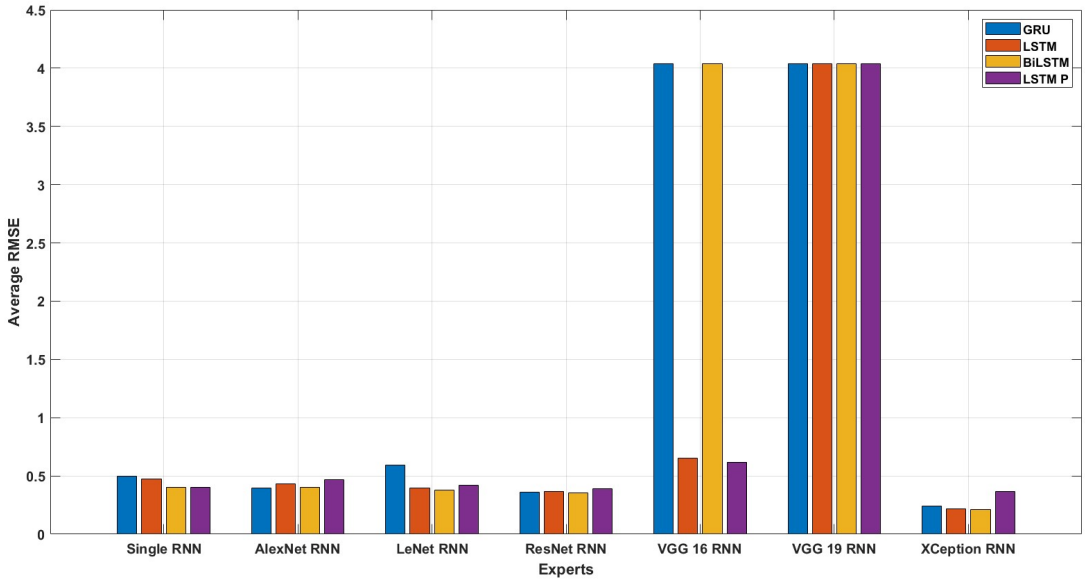


Figure 5.3.: Performance Comparison of Experts Based on Average RMSE

ResNe50 RNN, and XCeption have individual RMSE under 0.3 for Task 1 and under 0.7 for Task 2 and Task 3. In contrast, VGG 16 RNN and VGG 19 perform the worst, with RMSE for

Task 1 reaching 0.9, except for VGG 16, which uses LSTM and LSTM Projector Layer. For Task 2 and Task 3, VGG 16 and RNN 16 reached RMSE values of more than 0.9, except for VGG 16, which used the LSTM and LSTM Projector Layer.

In a model that only uses RNN (without combination with CNN), BiLSTM has the lower RMSE (0.179942) for Task 1, while LSTM Projected Layer is the best model for Task 2 and Task 3, with RMSEs of 0.471383 and 0.486482, respectively. AlexNet combined with RNN performs well, with RMSEs for all RNNs under 0.23 for Task 1 but ranges from 0.4 to 0.59 for Task 2 and Task 3.

Like single RNN and AlexNet RNN, LeNet RNN and Resnet RNN also perform well in Task 1 with RMSE ranges from 0.16 to 2.3 but fail to present good results for Task 2 and Task 3 with RMSE ranges from 0.4 to 0.8. XCeption RNN performance is the best compared to other models. For Task 1, the RMSE values range from 0.09 to 1.2, and the RMSE range from 0.29 to 0.44 for Task 2. For Task 3, Xception RNN ranges from 0.25 to 0.55.

Based on these results, XCeption GRU, XCeption LSTM, XCeption BiLSTM, and XCeption LSTM Projected Layer were used as our experts. As stated in subchapter 5.3, the number of experts is experimented with several numbers of experts, starting from 4 to 8. For the next experiment, the first expert will be XCeption GRU, the second expert will be XCeption LSTM, the third expert will be XCeption BiLSTM, and the fourth expert will be XCeption LSTM Projected. Expert number five will use XCeption GRU and so on.

5.4.2 Optimal Block Size

As given in Table 5.1, for a model with a total of experts 5, 6, 7, and 8, the ideal block size is 96. A model with four experts has the lowest loss value in all block sizes compared with a model with 5, 6, 7, and 8 experts. In detail, the average RMSE values for 96, 72, 48, 12, and 6,3 are 0.208668, 0.207816, 0.232025, 0.231136, 0.282639, 0.405442, and 0.343812, respectively. Since the loss difference between block sizes 72 and 96 is relatively small, block size 96 was used to train all models.

Number of Experts	Average RMSE						
	96	72	48	24	12	6	3
4	0.208668	**0.207816**	0.232025	0.231137	0.282639	0.405443	0.343812
5	**0.354722**	0.369535	0.371042	0.36353	0.364682	0.355724	0.364427
6	**0.354721**	0.374638	0.373696	0.380752	0.359274	0.362248	0.364098
7	**0.355947**	0.371228	0.371924	0.372777	0.37123697	0.3603	0.365572
8	**0.36584**	0.376588	0.376075	0.374488	0.362634	0.362703	0.365015

Table 5.1.: Comparison of Block Size Length with Five Different Experts Number

(a) Single RNN Individual RMSE

(b) AlexNet 50 RNN Individual RMSE

(c) LeNet RNN Individual RMSE

(d) ResNet 50 RNN Individual RMSE

(e) VGG 16 RNN Individual RMSE

(f)VGG 19 RNN Individual RMSE

Figure 5.4.: Performance Comparison of Experts Based on Average RMSE

5.4.3 Gated Multitask Learning Performance

The increasing number of experts has a significant impact on model accuracy. The model with a bigger number of experts performs better than the model with a small number of experts. Model with 4,5 and 6 experts has joint loss more than 0.2, in contrast model with 7 and 8 experts gain loss less than 0.18. From Table 5.2, it can also be observed that the proposed gated mixture of expert models needs a small output lag $n_y = 1$ and input lag $n_u = 1$, , meaning the model needs less time to be trained.

Number of Expert	Lag Networks	Joint Loss
4	$(n_y = 1, n_u = 2, n_e = 2)$	0.21578923
	$(n_y = 2, n_u = 3, n_e = 2)$	0.21359232
5	$(n_y = 1, n_u = 1, n_e = 1)$	0.21259185
	$(n_y = 1, n_u = 2, n_e = 1)$	0.20313972
6	$(n_y = 1, n_u = 1, n_e = 1)$	0.22113117
	$(n_y = 2, n_u = 1, n_e = 2)$	0.21468935
7	$(n_y = 1, n_u = 1, n_e = 1)$	0.17757598
	$(n_y = 1, n_u = 2, n_e = 1)$	0.17906236
8	$(n_y = 1, n_u = 2, n_e = 1)$	0.17636764
	$(n_y = 1, n_u = 1, n_e = 2)$	**0.17220143**

Table 5.2.: Optimal Lags Number for Experts

Table 5.3 shows the comparison performance of these models. The NARMAX Gated Multitask Learning with 7 and 8 experts outperformed all models regarding joint loss. The forecasting performance of the proposed model is demonstrated and compared to the singletask model and gated multitask learning without applying NARMAX. It also shows that the Gated Multitask Model with NARMAX delivers a lower joint loss value than the Gated Multitask Model without NARMAX.

To further evaluate the performance of the proposed model, the individual tasks of each model are compared. The comparison results are shown in Table 5.4. The prediction accuracy of NARMAX Gated Multitask Learning, which had 8 experts in every singletask, is strictly better than NARMAX Gated Multitask Learning with smaller experts and models without applying NARMAX. Based on joint loss and individual RMSE, it can be found that adding lags to the model improves its performance.

From the experiments, it was found that multitask learning does not always perform better than singletask learning. In Tables 5.2 and 5.3, it can be seen that singletask learning has lower joint loss and individual RMSE in each task compared to Gated Multitask Learning without NARMAX. Individual RMSE in three tasks shows that there is a small gap between Task 2 and Task 3, but the RMSE gap between Task 1 and Task 2 or Task 3 is slightly large. This

Model	Number of Experts	Joint Loss
Singletask	-	0.206751
Gated Multi Task Learning	4	0.229665
	5	0.204761
	6	0.216079
	7	0.234716
	8	0.230243
NARMAX Gated Multi Task Learning	4	0.213592
	5	0.20314
	6	0.214689
	7	0.177576
	8	**0.172201**

Table 5.3.: Performance Comparison of NARMAX Gated Multitask Learning with Singletask Learning and Gated Multitask Learning

Model	Number of Experts	Task 1	Task 2	Task 3
Singletask	-	0.09712	0.268	0.255134
Gated Multi Task Learning	4	0.139454	0.270454	0.279087
	5	0.097241	0.260541	0.2565
	6	0.099584	0.277225	0.271427
	7	0.10082	0.291135	0.312193
	8	0.101817	0.298616	0.290296
NARMAX Gated Multi Task Learning	4	0.097633	0.265173	0.277971
	5	0.096775	0.257587	0.255058
	6	0.100283	0.285246	0.277865
	7	0.090764	0.221489	0.220475
	8	**0.08928**	**0.21283**	**0.214495**

Table 5.4.: Perfomance Comparison of Individual Task RMSE between NARMAX Gated Multitask Learning, Singletask Learning, and Gated Multitask Learning

means the loss balance between tasks only happens to Task 2 and Task 3.

NARMAX works better than a singletask model and Gated Multitask Learning because it brings more information to the model. For the input, NARMAX uses past data, output from previous forecast results, and noise, while singletask learning and gated multitask learning only use past data. The large amount of input by NARMAX makes the model more likely to have more information or knowledge to deliver more accurate results.

5.5. Summary

In this chapter, two Gated Multitask frameworks are built. The first framework is based on NARMAX, while the other does not use NARMAX. The proposed model using NARMAX is proven to increase the accuracy of significant wave height forecasting compared to the singletask learning model. The optimal model is gained using eight experts, $n_y = 1, n_u = 1$, and $n_e = 2$. The loss value between the three tasks is small between Task 2 and Task 3 but became slightly large for Task 1. Based on these results, the proposed model is able to avoid the negative transfer effect by outperforming singletask model but failed to gain a loss of balance between all tasks.

6

Conclusions and future work

The motivations of this project are to build a multitask learning framework that is able to improve forecasting accuracy, avoid negative transfer, and distribute loss value equally. This thesis introduces three novel model contributions and a case study to show the effectiveness of the proposed multitask model for forecasting

6.1. Conclusions

The main purpose of multitasking learning is to solve several tasks simultaneously by using a sharing mechanism. Like any other model, this approach still has flaws. The first issue creating multitask learning is that the accuracy of multitask learning is worse than singletask learning. By multiplying the task, the model fails to grab the information from multiple sources, causing the model to have low accuracy compared to singletask learning. The second problem is unequal loss distribution between tasks or failure to distribute loss value equally. This indicated one or more tasks degrading other task performance. This thesis contributes to solving these issues by designing, developing, and implementing novel multitask learning by employing neural networks and nonlinear models.

Chapter 3 proposed two multitask learning methods, combining the Recurrent Neural Networks Model (GRU, LSTM, and BiLSTM) and the nonlinear model (NARX and NARMAX).Hard parameter sharing and non-deep neural networks are employed for both models. Furthermore, the model is evaluated using tide level forecasting. The input variable for this model

is historical tide level data, and the output is to forecast tide level data in multiple locations simultaneously. The results show that the models are able to outperform singletask learning; this means the model successfully avoids negative transfer. The model is also able to distribute loss value equally.

In Chapter 4, two novels multitask learning models are developed based on a soft parameter sharing technique to forecast significant wave height. The proposed models are built based on BiLSTM, XCeption, and NARMAX for specialization and sharing layers. For the output layer, the attention model is added to the networks. Two datasets related to significant wave height forecasts are implemented into the model. The experiments demonstrated that both models could avoid negative transfer in the first dataset, but the gap of loss value between tasks is slightly large. For the second dataset, both models are not only capable of avoiding negative transfer but also manage to distribute loss between tasks equally.

Chapter 5 focuses on developing novel multitask learning by borrowing a mixture of experts' concepts. A novel gated multitask learning is designed using eight experts and a single gate with NARMAX. The dataset used in this chapter is similar to the one used in Chapter 4, but only Datset 1 is applied. The results of the experiment show that gated multitask learning improves the forecasting results and does not suffer from negative transfer.

By analyzing the results from Chapter 3, the time lag variation and the neural networks setting show that the numbers of input time lag, output time lag, hidden layers, and hidden nodes have no correlations. The increase of the network complexity, namely, the increase of hidden layers, number of nodes, and time lag, does not significantly affect the accuracy or joint loss value of the models. In some cases, it needs a relatively more significant number of hidden layers to generate the lowest joint loss value, while in other cases, it only needs a single hidden layer with a minimal number of hidden nodes. This means that manual intervention is still needed in the fine-tune process of the neural network models.

As seen from chapters 4 and 5, the choice of which neural networks to pair with NARMAX to build the best multitask learning model plays an important role. Choosing the right pair requires quite a bit of time and can be an exhaustive process.

6.2. Future work

The work conducted in the study can be potentially extended to address limitations and further improvements. Some ideas and future extensions are listed below.

- One of the most difficult issues to solve in multitask learning is unequal loss distribution. Creating a model that can share information and distribute loss equally seems to be a complex job for a multitasking learning framework. Curriculum learning, which has a concept of training from simple to complex problems, may be able to solve this issue. Extending multitask learning by combining it with another learning method, such as curriculum learning and reinforcement learning, can be an alternative solution to solve the unequal loss distribution problems.
- Implementation of multitask learning into less related tasks or unrelated tasks. Multitask learning works well in highly related tasks, but problems are not always related in the real world.
- Gated multitask learning has big potential to solve multitasking learning issues. This model can be developed into a flexible model. Flexibility in terms of the number of experts used in the model can be based on the task difficulties. The number of experts is not decided randomly or based on trial and error but on the task to solve.

References

[1] Sebastian Ruder. 'An Overview of Multi-Task Learning in Deep Neural Networks'. In: *CoRR* abs/1706.05098 (2017). arXiv: 1706.05098 (cit. on pp. 1, 3, 7, 8, 18, 19, 21).

[2] Michael Crawshaw. 'Multi-Task Learning with Deep Neural Networks: A Survey'. In: *CoRR* abs/2009.09796 (2020). arXiv: 2009.09796 (cit. on pp. 1–3, 7, 8, 19, 33).

[3] Partoo Vafaeikia, Khashayar Namdar and Farzad Khalvati. 'A Brief Review of Deep Multi-task Learning and Auxiliary Task Learning'. In: *CoRR* abs/2007.01126 (2020). arXiv: 2007.01126 (cit. on pp. 2, 18, 19, 21, 33).

[4] Shanfeng Wang, Qixiang Wang and Maoguo Gong. 'Multi-Task Learning Based Network Embedding'. In: *Frontiers in Neuroscience* 13 (2020) (cit. on pp. 2, 3, 18, 23).

[5] Jonathan Baxter. 'A Bayesian/Information Theoretic Model of Learning to Learn via Multiple Task Sampling'. In: *Machine Learning* 28.1 (July 1997), pp. 7–39 (cit. on pp. 2, 19).

[6] Rich Caruana. 'Multitask Learning'. In: *Machine Learning* 28.1 (July 1997), pp. 41–75 (cit. on pp. 2, 19).

[7] Alex Kendall, Yarin Gal and Roberto Cipolla. 'Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics'. In: *CoRR* abs/1705.07115 (2017). arXiv: 1705.07115 (cit. on p. 2).

[8] Reem A. Mahmoud, Hazem Hajj and Fadi N. Karameh. 'A systematic approach to multi-task learning from time-series data'. In: *Applied Soft Computing* 96 (2020), p. 106586 (cit. on pp. 2, 19, 20).

[9] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee and Andrew Rabinovich. 'GradNorm: Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks'. In: *CoRR* abs/1711.02257 (2017). arXiv: 1711.02257 (cit. on pp. 2, 20).

[10] Pengsheng Guo, Chen-Yu Lee and Daniel Ulbricht. 'Learning to Branch for Multi-Task Learning'. In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, July 2020, pp. 3854–3863 (cit. on p. 2).

[11] Elena Kochkina, Maria Liakata and Arkaitz Zubiaga. 'All-in-one: Multi-task Learning for Rumour Verification'. In: *Proceedings of the 27th International Conference on Computational Linguistics*. Ed. by Emily M. Bender, Leon Derczynski and Pierre Isabelle. Santa Fe, New Mexico, USA: Association for Computational Linguistics, Aug. 2018, pp. 3402–3413 (cit. on p. 2).

[12] Stephen Billings. 'Models for Linear and Nonlinear Systems'. In: *Nonlinear System Identification*. John Wiley & Sons, Ltd, 2013, pp. 17–59 (cit. on pp. 3, 12, 13, 31).

[13] Nerfita Nikentari and Hua-Liang Wei. 'Tide Level Prediction Using NARX-based Recurrent Neural Networks'. In: *2022 27th International Conference on Automation and Computing (ICAC)*. 2022, pp. 1–6 (cit. on p. 5).

[14] Nerfita Nikentari and Hua-Liang Wei. 'Multi-Task Learning for Time Series Forecasting Using NARMAX-LSTM'. In: *2022 27th International Conference on Automation and Computing (ICAC)*. 2022, pp. 1–6 (cit. on p. 5).

[15] N. Nikentari and H.-L. Wei. 'Multi-task learning using non-linear autoregressive models and recurrent neural networks for tide level forecasting'. In: *International Journal of Electrical and Computer Engineering (IJECE)* 14.1 (Feb. 2024). Publisher: Institute of Advanced Engineering and Science, pp. 960–970 (cit. on p. 5).

[16] Yu Zhang and Qiang Yang. 'An overview of multi-task learning'. In: *National Science Review* 5.1 (Sept. 2017), pp. 30–43 (cit. on p. 8).

[17] Sima Siami-Namini, Neda Tavakoli and Akbar Siami Namin. 'The Performance of LSTM and BiLSTM in Forecasting Time Series'. In: *2019 IEEE International Conference on Big Data (Big Data)*. 2019, pp. 3285–3292 (cit. on pp. 8, 9).

[18] Susmita Das, Amara Tariq, Thiago Santos, Sai Sandeep Kantareddy and Imon Banerjee. 'Recurrent Neural Networks (RNNs): Architectures, Training Tricks, and Introduction to Influential Research'. In: *Machine Learning for Brain Disorders*. Ed. by Olivier Colliot. New York, NY: Springer US, 2023, pp. 117–138 (cit. on pp. 8–10).

[19] Alexander Rehmer and Andreas Kroll. 'On the vanishing and exploding gradient problem in Gated Recurrent Units'. In: *IFAC-PapersOnLine* 53.2 (2020), pp. 1243–1248 (cit. on pp. 9, 30).

[20] Sepp Hochreiter and Jürgen Schmidhuber. 'Long Short-Term Memory'. In: *Neural Comput.* 9.8 (Nov. 1997). Place: Cambridge, MA, USA Publisher: MIT Press, pp. 1735–1780 (cit. on p. 9).

[21] Pardeep Singla, Manoj Duhan and Sumit Saroha. 'An ensemble method to forecast 24-h ahead solar irradiance using wavelet decomposition and BiLSTM deep learning network'. In: *Earth Science Informatics* 15.1 (Mar. 2022), pp. 291–306 (cit. on p. 9).

[22] F.A. Gers, J. Schmidhuber and F. Cummins. 'Learning to forget: continual prediction with LSTM'. In: *1999 Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470)*. Vol. 2. 1999, 850–855 vol.2 (cit. on p. 9).

[23] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre et al. 'Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation'. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Ed. by Alessandro Moschitti, Bo Pang and Walter Daelemans. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1724–1734 (cit. on p. 10).

[24] K. E. ArunKumar, Dinesh V. Kalaga, Ch Mohan Sai Kumar, Masahiro Kawaji and Timothy M. Brenza. 'Comparative analysis of Gated Recurrent Units (GRU), long Short-Term memory (LSTM) cells, autoregressive Integrated moving average (ARIMA), seasonal autoregressive Integrated moving average (SARIMA) for forecasting COVID-19 trends'. In: *Alexandria Engineering Journal* 61.10 (2022), pp. 7585–7603 (cit. on p. 10).

[25] Taozheng Zhang and Rui Xu. 'Performance Comparisons of Bi-LSTM and Bi-GRU Networks in Chinese Word Segmentation'. In: *Proceedings of the 2021 5th International Conference on Deep Learning Technologies*. ICDLT '21. event-place: Qingdao, China. New York, NY, USA: Association for Computing Machinery, 2021, pp. 73–80 (cit. on p. 10).

[26] M. Schuster and K.K. Paliwal. 'Bidirectional recurrent neural networks'. In: *IEEE Transactions on Signal Processing* 45.11 (1997), pp. 2673–2681 (cit. on p. 11).

[27] Peng Lu, Ting Bai and Philippe Langlais. 'SC-LSTM: Learning Task-Specific Representations in Multi-Task Learning for Sequence Labeling'. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Ed. by Jill Burstein, Christy Doran and Thamar Solorio. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 2396–2406 (cit. on p. 11).

[28] Md. Mostafizer Rahman, Yutaka Watanobe and Keita Nakamura. 'A Bidirectional LSTM Language Model for Code Evaluation and Repair'. In: *Symmetry* 13.2 (2021) (cit. on p. 11).

[29] Luay Alawneh, Belal Mohsen, Mohammad Al-Zinati, Ahmed Shatnawi and Mahmoud Al-Ayyoub. 'A Comparison of Unidirectional and Bidirectional LSTM Networks for Human Activity Recognition'. In: *2020 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*. 2020, pp. 1–6 (cit. on pp. 11, 12).

[30] Zhiyong Cui, Ruimin Ke, Ziyuan Pu and Yinhai Wang. 'Stacked bidirectional and unidirectional LSTM recurrent neural network for forecasting network-wide traffic state with missing values'. In: *Transportation Research Part C: Emerging Technologies* 118 (2020), p. 102674 (cit. on p. 12).

[31] Dinesh Naik and C. D. Jaidhar. 'A novel Multi-Layer Attention Framework for visual description prediction using bidirectional LSTM'. In: *Journal of Big Data* 9.1 (Nov. 2022), p. 104 (cit. on pp. 12, 70).

[32] Stephen Billings and Hua-Liang 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)*. 2019, pp. 2743–2750 (cit. on p. 13).

[33] Y. Gu, Hua-Liang Wei, Richard J. Boynton, Simon N. Walker and Michael A. Balikhin. 'System Identification and Data-Driven Forecasting of AE Index and Prediction Uncertainty Analysis Using a New Cloud-NARX Model'. In: *Journal of Geophysical Research: Space Physics* 124.1 (2019), pp. 248–263 (cit. on pp. 13, 31).

[34] Jose Roberto Ayala Solares, Hua-Liang Wei, R. J. Boynton, Simon N. Walker and Stephen A. Billings. 'Modeling and prediction of global magnetic disturbance in near-Earth space: A case study for Kp index using NARX models'. In: *Space Weather* 14.10 (2016), pp. 899–916 (cit. on pp. 13, 33, 34, 36).

[35] Richard J. Hall, Hua-Liang Wei and Edward Hanna. 'Complex systems modelling for statistical forecasting of winter North Atlantic atmospheric variability: A new approach to North Atlantic seasonal forecasting'. In: *Quarterly Journal of the Royal Meteorological Society* 145.723 (2019), pp. 2568–2585 (cit. on p. 13).

[36] Kunihiko Fukushima. 'Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position'. In: *Biological Cybernetics* 36.4 (Apr. 1980), pp. 193–202 (cit. on pp. 13, 14).

[37] Jeremy Onesimus Carnagie, Aditya Rio Prabowo, Eko Prasetya Budiana and Ivan Kristianto Singgih. 'Essential Oil Plants Image Classification Using Xception Model'. In: *Procedia Computer Science* 204 (2022), pp. 395–402 (cit. on pp. 13, 14, 32).

[38] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner. 'Gradient-based learning applied to document recognition'. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324 (cit. on p. 14).

[39] I. Gogul and V. Sathiesh Kumar. 'Flower species recognition system using convolution neural networks and transfer learning'. In: *2017 Fourth International Conference on Signal Processing, Communication and Networking (ICSCN)*. 2017, pp. 1–6 (cit. on p. 14).

[40] Rikiya Yamashita, Mizuho Nishio, Richard Kinh Gian Do and Kaori Togashi. 'Convolutional neural networks: an overview and application in radiology'. In: *Insights into Imaging* 9.4 (Aug. 2018), pp. 611–629 (cit. on p. 14).

[41] F. Chollet. 'Xception: Deep Learning with Depthwise Separable Convolutions'. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. ISSN: 1063-6919. Los Alamitos, CA, USA: IEEE Computer Society, July 2017, pp. 1800–1807 (cit. on p. 14).

[42] Yaolin Zhu, Huang JiaYI, Yunhong Li and Wenya Li. 'Image identification of cashmere and wool fibers based on the improved Xception network'. In: *Journal of King Saud University - Computer and Information Sciences* 34.10, Part B (2022), pp. 9301–9310 (cit. on p. 14).

[43] Kashif Shaheed, Aihua Mao, Imran Qureshi et al. 'DS-CNN: A pre-trained Xception model based on depth-wise separable convolutional neural network for finger vein recognition'. In: *Expert Systems with Applications* 191 (2022), p. 116288 (cit. on p. 14).

[44] Omer Sagi and Lior Rokach. 'Ensemble learning: A survey'. In: *WIREs Data Mining and Knowledge Discovery* 8.4 (2018), e1249 (cit. on p. 15).

[45] M. A. Ganaie, Minghui Hu, A. K. Malik, M. Tanveer and P. N. Suganthan. 'Ensemble deep learning: A review'. In: *Engineering Applications of Artificial Intelligence* 115 (2022), p. 105151 (cit. on p. 15).

[46] Ibomoiye Domor Mienye and Yanxia Sun. 'A Survey of Ensemble Learning: Concepts, Algorithms, Applications, and Prospects'. In: *IEEE Access* 10 (2022), pp. 99129–99149 (cit. on p. 15).

[47] Rajarshi Chattopadhyay and Chen-Khong Tham. 'Mixture of Experts based Model Integration for Traffic State Prediction'. In: *2022 IEEE 95th Vehicular Technology Conference: (VTC2022-Spring)*. 2022, pp. 1–7 (cit. on p. 15).

[48] W.J. Puma-Villanueva, C.A.M. Lima, E.P. dos Santos and F.J. Von Zuben. 'Mixture of heterogeneous experts applied to time series: a comparative study'. In: *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.* Vol. 2. 2005, 1160–1165 vol. 2 (cit. on pp. 15, 16).

[49] Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan and Geoffrey E. Hinton. 'Adaptive mixtures of local experts.' In: *Neural Computation* 3.1 (1991). Place: US Publisher: MIT Press, pp. 79–87 (cit. on p. 15).

[50] M.I. Jordan and R.A. Jacobs. 'Hierarchical mixtures of experts and the EM algorithm'. In: *Proceedings of 1993 International Conference on Neural Networks (IJCNN-93-Nagoya, Japan).* Vol. 2. 1993, 1339–1344 vol.2 (cit. on pp. 15, 16, 24).

[51] V. Ramamurti and J. Ghosh. 'Structural adaptation in mixture of experts'. In: *Proceedings of 13th International Conference on Pattern Recognition*. Vol. 4. 1996, 704–708 vol.4 (cit. on p. 15).

[52] Yan Yang and Jinwen Ma. 'A Single Loop EM Algorithm for the Mixture of Experts Architecture'. In: *Advances in Neural Networks – ISNN 2009*. Ed. by Wen Yu, Haibo He and Nian Zhang. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 959–968 (cit. on p. 16).

[53] Ping Guo and Lei Xu. 'Relationship between mixture of experts and ensemble neural networks'. In: *ICONIP'99. ANZIIS'99 & ANNES'99 & ACNN'99. 6th International Conference on Neural Information Processing. Proceedings (Cat. No.99EX378)*. Vol. 1. 1999, 246–250 vol.1 (cit. on p. 16).

[54] Faïcel Chamroukhi, Florian Lecocq and Hien D. Nguyen. 'Regularized Estimation and Feature Selection in Mixtures of Gaussian-Gated Experts Models'. In: *Statistics and Data Science*. Ed. by Hien Nguyen. Singapore: Springer Singapore, 2019, pp. 42–56 (cit. on p. 16).

[55] C.Ap.M. Lima, A.L.V. Coelho and F.J. Von Zuben. 'Mixture of experts applied to nonlinear dynamic systems identification: a comparative study'. In: *VII Brazilian Symposium on Neural Networks, 2002. SBRN 2002. Proceedings.* 2002, pp. 162–167 (cit. on p. 16).

[56] Haibin Cheng, Pang-Ning Tan, Jing Gao and Jerry Scripps. 'Multistep-Ahead Time Series Prediction'. In: *Advances in Knowledge Discovery and Data Mining*. Ed. by Wee-Keong Ng, Masaru Kitsuregawa, Jianzhong Li and Kuiyu Chang. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 765–774 (cit. on pp. 16, 17).

[57] Tao Ma and Ying Tan. 'Multiple Stock Time Series Jointly Forecasting with Multi-Task Learning'. In: *2020 International Joint Conference on Neural Networks (IJCNN)*. 2020, pp. 1–8 (cit. on p. 17).

[58] Gianluca Bontempi, Souhaib Ben Taieb and Yann-Aël Le Borgne. 'Machine Learning Strategies for Time Series Forecasting'. In: *Business Intelligence: Second European Summer School, eBISS 2012, Brussels, Belgium, July 15-21, 2012, Tutorial Lectures*. Ed. by Marie-Aude Aufaure and Esteban Zimányi. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 62–77 (cit. on p. 17).

[59] Nguyen Hoang An and Duong Tuan Anh. 'Comparison of Strategies for Multi-step-Ahead Prediction of Time Series Using Neural Network'. In: *2015 International Conference on Advanced Computing and Applications (ACOMP)*. 2015, pp. 142–149 (cit. on p. 17).

[60] Rui Ye and Qun Dai. 'MultiTL-KELM: A multi-task learning algorithm for multi-step-ahead time series prediction'. In: *Applied Soft Computing* 79 (2019), pp. 227–253 (cit. on p. 17).

[61] Han Wang, Jie Yan, Jiawei Zhang et al. 'Short-term integrated forecasting method for wind power, solar power, and system load based on variable attention mechanism and multi-task learning'. In: *Energy* 304 (2024), p. 132188 (cit. on p. 20).

[62] Loukas Ilias, Panagiotis Kapsalis, Spiros Mouzakitis and Dimitris Askounis. 'A Multi-task Learning Framework for Predicting Ship Fuel Oil Consumption'. In: *IEEE Access* 11 (2023), pp. 132576–132589 (cit. on p. 20).

[63] I. Misra, A. Shrivastava, A. Gupta and M. Hebert. 'Cross-Stitch Networks for Multi-task Learning'. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. ISSN: 1063-6919. Los Alamitos, CA, USA: IEEE Computer Society, June 2016, pp. 3994–4003 (cit. on p. 21).

[64] Sebastian Ruder, Joachim Bingel, Isabelle Augenstein and Anders Søgaard. 'Sluice networks: Learning what to share between loosely related tasks'. In: *CoRR* abs/1705.08142 (2017). arXiv: 1705.08142 (cit. on p. 22).

[65] S. Liu, E. Johns and A. J. Davison. 'End-To-End Multi-Task Learning With Attention'. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE Computer Society, June 2019, pp. 1871–1880 (cit. on p. 22).

[66] Jiang-Wen Xiao, Minghui Cao, Hongliang Fang, Jinsong Wang and Yan-Wu Wang. 'Joint load prediction of multiple buildings using multi-task learning with selected-shared-private mechanism'. In: *Energy and Buildings* 293 (2023), p. 113178 (cit. on p. 22).

[67] Manuel Nunes, Enrico Gerding, Frank McGroarty and Mahesan Niranjan. 'A comparison of multitask and single task learning with artificial neural networks for yield curve forecasting'. In: *Expert Systems with Applications* 119 (2019), pp. 362–375 (cit. on pp. 23, 24).

[68] Shiliang Sun. 'Multitask learning for EEG-based biometrics'. In: *2008 19th International Conference on Pattern Recognition*. 2008, pp. 1–4 (cit. on p. 23).

[69] Grigorios Skolidis and Guido Sanguinetti. 'Semisupervised Multitask Learning With Gaussian Processes'. In: *IEEE Transactions on Neural Networks and Learning Systems* 24.12 (2013), pp. 2101–2112 (cit. on p. 23).

[70] Meiyin Dai and FanZhang Li. 'Dynamic Fuzzy Semisupervised Multitask Learning'. In: *2011 Seventh International Conference on Computational Intelligence and Security*. 2011, pp. 450–454 (cit. on p. 23).

[71] Xinmei Tian, Ya Li, Tongliang Liu, Xinchao Wang and Dacheng Tao. 'Eigenfunction-Based Multitask Learning in a Reproducing Kernel Hilbert Space'. In: *IEEE Transactions on Neural Networks and Learning Systems* 30.6 (2019), pp. 1818–1830 (cit. on p. 23).

[72] Michael I. Jordan and Robert A. Jacobs. 'Hierarchies of adaptive experts'. In: *Proceedings of the 4th International Conference on Neural Information Processing Systems*. NIPS'91. event-place: Denver, Colorado. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1991, pp. 985–992 (cit. on p. 24).

[73] Ronan Collobert, Yoshua Bengio and Samy Bengio. 'Scaling Large Learning Problems with Hard Parallel Mixtures'. In: *Pattern Recognition with Support Vector Machines*. Ed. by Seong-Whan Lee and Alessandro Verri. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 8–23 (cit. on pp. 24, 69, 70, 72).

[74] R. Aljundi, P. Chakravarty and T. Tuytelaars. 'Expert Gate: Lifelong Learning with a Network of Experts'. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. ISSN: 1063-6919. Los Alamitos, CA, USA: IEEE Computer Society, July 2017, pp. 7120–7129 (cit. on p. 25).

[75] Jiaqi Ma, Zhe Zhao, Xinyang Yi et al. 'Modeling Task Relationships in Multi-task Learning with Multi-gate Mixture-of-Experts'. In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. KDD '18. event-place: London, United Kingdom. New York, NY, USA: Association for Computing Machinery, 2018, pp. 1930–1939 (cit. on p. 25).

[76] Keyao Li and Jungang Xu. 'AC-MMOE: A Multi-gate Mixture-of-experts Model Based on Attention and Convolution'. In: *Procedia Computer Science* 222 (2023), pp. 187–196 (cit. on p. 25).

[77] Zhen Qin, Yicheng Cheng, Zhe Zhao et al. 'Multitask Mixture of Sequential Experts for User Activity Streams'. In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. KDD '20. event-place: Virtual Event, CA,

USA. New York, NY, USA: Association for Computing Machinery, 2020, pp. 3083–3091 (cit. on pp. 25, 26).

[78] Sinan Wang, Yumeng Li, Hongyan Li et al. 'Multi-Task Learning with Calibrated Mixture of Insightful Experts'. In: *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. 2022, pp. 3307–3319 (cit. on p. 25).

[79] Amin Riazi. 'Accurate tide level estimation: A deep learning approach'. In: *Ocean Engineering* 198 (2020), p. 107013 (cit. on p. 30).

[80] Wenjuan Wang, Hongchun Yuan and Mingxing Tan. 'Application of BP Neural Network in Monitoring of Ocean Tide Level'. In: *2015 International Conference on Computational Intelligence and Communication Networks (CICN)*. 2015, pp. 1238–1240 (cit. on pp. 30, 31).

[81] Cheng-Hong Yang, Chih-Hsien Wu and Chih-Min Hsieh. 'Long Short-Term Memory Recurrent Neural Network for Tidal Level Forecasting'. In: *IEEE Access* 8 (2020), pp. 159389–159401 (cit. on p. 30).

[82] WANG Bao and WANG 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)*. 2018, pp. 604–608 (cit. on pp. 30, 31).

[83] Miftahul Awali Rizkina, Didit Adytia and Nugrahinggil 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)*. 2019, pp. 1–5 (cit. on pp. 30, 31).

[84] Francesco Granata and Fabio Di Nunno. 'Artificial Intelligence models for prediction of the tide level in Venice'. In: *Stochastic Environmental Research and Risk Assessment* 35.12 (Dec. 2021), pp. 2537–2548 (cit. on p. 30).

[85] M. S. Chowdhury, Kazi Sajedur Rahman, Vidhya Selvanathan et al. 'Current trends and prospects of tidal energy technology'. In: *Environment, Development and Sustainability* 23.6 (June 2021), pp. 8179–8194 (cit. on p. 30).

[86] Fergal O. Rourke, Fergal Boyle and Anthony Reynolds. 'Tidal energy update 2009'. In: *Applied Energy* 87.2 (2010), pp. 398–409 (cit. on p. 30).

[87] Avelisa Yoelma Winona and Didit Adytia. 'Short Term Forecasting of Sea Level by Using LSTM with Limited Historical Data'. In: *2020 International Conference on Data Science and Its Applications (ICoDSA)*. 2020, pp. 1–5 (cit. on p. 31).

[88] Cheng-Hong Yang, Chih-Hsien Wu, Chih-Min Hsieh et al. 'Deep Learning for Imputation and Forecasting Tidal Level'. In: *IEEE Journal of Oceanic Engineering* 46.4 (2021), pp. 1261–1271 (cit. on p. 31).

[89] Sergio Consoli, Diego Reforgiato Recupero and Vanni Zavarella. 'A survey on tidal analysis and forecasting methods for tsunami detection'. In: *Science of Tsunami Hazard* 33.1 (2014), pp. 1–54 (cit. on p. 31).

[90] Luis Antonio Aguirre, Daniela D. Rodrigues, Silvio T. Lima and Carlos Barreira Martinez. 'Dynamical prediction and pattern mapping in short-term load forecasting'. In: *International Journal of Electrical Power & Energy Systems* 30.1 (2008), pp. 73–82 (cit. on p. 31).

[91] Wenhao Wu, Lianbo Li, Jianchuan Yin, Wenyu Lyu and Wenjun Zhang. 'A Modular Tide Level Prediction Method Based on a NARX Neural Network'. In: *IEEE Access* 9 (2021), pp. 147416–147429 (cit. on p. 31).

[92] Francisco Muñoz and Gonzalo Acuña. '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)*. 2021, pp. 1–6 (cit. on p. 31).

[93] Takwa Omri, Asma Karoui, Didier Georges and Mounir 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)*. 2020, pp. 193–198 (cit. on p. 31).

[94] Kunpeng Zhang, Lan Wu, Zhaoju Zhu and Jiang Deng. 'A Multitask Learning Model for Traffic Flow and Speed Forecasting'. In: *IEEE Access* 8 (2020), pp. 80707–80715 (cit. on pp. 31, 32).

[95] Shiva Pentyala, Mengwen Liu and Markus Dreyer. 'Multi-Task Networks with Universe, Group, and Task Feature Learning'. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Ed. by Anna Korhonen, David Traum and Lluís Màrquez. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 820–830 (cit. on pp. 31, 32).

[96] Kamran Ghasedi Dizaji, Wei Chen and Heng Huang. 'Deep Large-Scale Multitask Learning Network for Gene Expression Inference.' eng. In: *Journal of computational biology : a journal of computational molecular cell biology* 28.5 (May 2021). Place: United States, pp. 485–500 (cit. on p. 32).

[97] Junqiang Wei, Xuejie Wu, Tianming Yang and Runhai Jiao. 'Ultra-short-term forecasting of wind power based on multi-task learning and LSTM'. In: *International Journal of Electrical Power & Energy Systems* 149 (2023), p. 109073 (cit. on p. 32).

[98] Mostafa Karimzadeh, Samuel Martin Schwegler, Zhongliang Zhao, Torsten Braun and Susana Sargento. 'MTL-LSTM: Multi-Task Learning-based LSTM for Urban Traffic Flow Forecasting'. In: *2021 International Wireless Communications and Mobile Computing (IWCMC)*. 2021, pp. 564–569 (cit. on pp. 32, 34).

[99] Tong Wu and Tengpeng Chen. 'A Gated Multiscale Multitask Learning Model Using Time-Frequency Representation for Health Assessment and Remaining Useful Life Prediction.' eng. In: *Sensors (Basel, Switzerland)* 23.4 (Feb. 2023). Place: Switzerland (cit. on pp. 32, 34).

[100] Yixiu Guo, Yong Li, Xuebo Qiao et al. 'BiLSTM Multitask Learning-Based Combined Load Forecasting Considering the Loads Coupling Relationship for Multienergy System'. In: *IEEE Transactions on Smart Grid* 13.5 (2022), pp. 3481–3492 (cit. on pp. 32, 33).

[101] H. L. Wei and S. A. Billings. 'An efficient nonlinear cardinal B-spline model for high tide forecasts at the Venice Lagoon'. In: *Nonlinear Processes in Geophysics* 13 (Oct. 2006), pp. 577–584 (cit. on p. 33).

[102] H. L. Wei and S. A. Billings. 'Long term prediction of non-linear time series using multiresolution wavelet models'. In: *International Journal of Control* 79.6 (2006). Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/00207170600621447, pp. 569–580 (cit. on p. 33).

[103] Lan Xu and Jun Yang. 'Carbon pricing policies and renewable energy development: Analysis based on cross-country panel data'. In: *Journal of Environmental Management* 366 (2024), p. 121784 (cit. on p. 58).

[104] Junheng Pang and Sheng Dong. 'A novel multivariable hybrid model to improve short and long-term significant wave height prediction'. In: *Applied Energy* 351 (2023), p. 121813 (cit. on p. 58).

[105] Ruobin Gao, Ruilin Li, Minghui Hu, Ponnuthurai Nagaratnam Suganthan and Kum Fai Yuen. 'Dynamic ensemble deep echo state network for significant wave height forecasting'. In: *Applied Energy* 329 (2023), p. 120261 (cit. on p. 58).

[106] Zhangping Wei. 'Forecasting wind waves in the US Atlantic Coast using an artificial neural network model: Towards an AI-based storm forecast system'. In: *Ocean Engineering* 237 (2021), p. 109646 (cit. on p. 58).

[107] L. Cornejo-Bueno, J. C. Nieto-Borge, P. García-Díaz, G. Rodríguez and S. Salcedo-Sanz. 'Significant wave height and energy flux prediction for marine energy applications: A grouping genetic algorithm – Extreme Learning Machine approach'. In: *Renewable Energy* 97 (2016), pp. 380–389 (cit. on p. 58).

[108] P. M. R. Bento, J. A. N. Pombo, R. P. G. Mendes, M. R. A. Calado and S. J. P. S. Mariano. 'Ocean wave energy forecasting using optimised deep learning neural networks'. In: *Ocean Engineering* 219 (2021), p. 108372 (cit. on p. 58).

[109] Gen Bai, Zhifeng Wang, Xianye Zhu and Yanqing Feng. 'Development of a 2-D deep learning regional wave field forecast model based on convolutional neural network and the application in South China Sea'. In: *Applied Ocean Research* 118 (2022), p. 103012 (cit. on p. 58).

[110] Shuntao Fan, Nianhao Xiao and Sheng Dong. 'A novel model to predict significant wave height based on long short-term memory network'. In: *Ocean Engineering* 205 (2020), p. 107298 (cit. on p. 58).

[111] Felipe C. Minuzzi and Leandro Farina. 'A deep learning approach to predict significant wave height using long short-term memory'. In: *Ocean Modelling* 181 (2023), p. 102151 (cit. on p. 58).

[112] Yuval Yevnin, Shir Chorev, Ilan Dukan and Yaron Toledo. 'Short-term wave forecasts using gated recurrent unit model'. In: *Ocean Engineering* 268 (2023), p. 113389 (cit. on p. 58).

[113] Shaobo Yang, Zegui Deng, Xingfei Li et al. 'A novel hybrid model based on STL decomposition and one-dimensional convolutional neural networks with positional encoding for significant wave height forecast'. In: *Renewable Energy* 173 (2021), pp. 531–543 (cit. on p. 59).

[114] Fotios Petropoulos, Rob J. Hyndman and Christoph Bergmeir. 'Exploring the sources of uncertainty: Why does bagging for time series forecasting work?' In: *European Journal of Operational Research* 268.2 (2018), pp. 545–554 (cit. on p. 70).

[115] Regina Y. Liu and Kesar Singh. 'Moving Blocks Jackknife and Bootstrap Capture Weak Dependence'. In: *Journal of the Royal Statistical Society Series D: The Statistician* 43.1 (1994) (cit. on p. 70).

[116] Hans R. Kunsch. 'The Jackknife and the Bootstrap for General Stationary Observations'. In: *The Annals of Statistics* 17.3 (1989). Publisher: Institute of Mathematical Statistics, pp. 1217–1241 (cit. on p. 70).

Appendices

A

Paper 1

Proceedings of the 27th International Conference on Automation & Computing, University of The West England, Bristol, UK, 1-3 September 2022

Multi-Task Learning for Time Series Forecasting Using NARMAX-LSTM

Nerfita Nikentari, Hua-Liang Wei
Department of Automatic Control and Systems Engineering
University of Sheffield, Sheffield
nnikentari1@sheffield.ac.uk,w.hualiang@sheffield.ac.uk (*corresponding author: H.-L. Wei)

2022 27th International Conference on Automation and Computing (ICAC) | 978-1-6654-9807-4/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICAC55051.2022.9911071

Abstract—**Traditional time series forecasting models based on neural networks are usually designed for a single task, which ignores the source of knowledge or information from other related tasks. On the other hand, multitask learning (MTL) offers a new perspective where one model is not only for one forecasting case or task but several tasks at the same time. The MTL model proposed in this study is based on the Nonlinear AutoRegressive Moving Average with eXogenous input (NARMAX) model and Long-Short Term Memory (LSTM) Neural Networks. The NARMAX-LSTM model is tested and validated using three different time horizons and different cases/tasks. The model forecasting performance is compared with that of three other linear and nonlinear models. The comparison results show that the multi-task NARMAX-LSTM model does not only have better prediction performance but also avoid suffering from negative transfer and overfitting.**

Keywords-multitask learning, NARMAXx, LSTM, forecasting;

I. INTRODUCTION

Machine learning provides an important tool for solving real-life data based modelling problems. Traditionally, different modelling tasks are usually solved separately by choosing a specific technique for the task to find optimal solutions. Such a modelling approach is called single task learning (STL). The STL method can only give one forecast result using an available model; it requires to fine-tune other models in cases where there is a need to produce other forecast results for a similar or related task.

However, in certain cases, it is possible to simultaneously deal with several tasks/problems by exploiting the information and knowledge from other similar and related tasks [1, 2]. Such an approach is called Multi-Task Learning (MTL). The term Multi Task is used to describe a single model that can be used to conduct two or more tasks at the same times [3]. The MTL framework attempt to share useful information across tasks in parallel. The goal of MTL is to leverage information in various learning tasks to improve the model generalisation performance [4, 5]. From this perspective of performance generalisation, the high computational capacity of deep neural networks (DNN) can be combined with MTL to improve its performance. This approach has been widely applied in numerous area of science and engineering [2,5].

One of the benefits of neural networks (NN) is their capability to perform various data sets and endpoints concurrently. Caruana [6] argued that neural networks are suitable for MTL problems because NN can learn hidden information of the data shared between multiple tasks [3, 7]. The basic idea of combining NN with MTL is as follows: the NN hidden layer are shared by all tasks (targets/outputs); the training or learning process for all task is implemented in parallel; some hidden nodes may be specially designed for a specific task; add auxiliary output nodes for auxiliary tasks [8, 9].

NARMAX is a class of non-linear model that can be treated as a neural network. This study uses LSTM neural network as non-linear framework for NARMAX [10].

The combination of MTL can increase the effectiveness of the resulting models by augmenting the training set and by using different datasets for related tasks. Because the model has to learn shared information for multiple tasks, there is a lower possibility of overfitting [10]. Another advantage of MTL is that it will cut down the number of networks that need to be trained, optimised, and evaluated. Therefore, MTL would be more cost-effective in terms of computational resources. In the industrial world, this plays an important role in cost-saving when dealing with DNN [3]. Although MTL has the aforementioned advantages, it has drawbacks and one of them is the negative transfer [11]. The model performance may be better on average overall tasks using the MTL setting, but for some particular tasks, MTL may produce unsatisfactory performance compare to independently trained framework or STL. Such a detrimental effect in MTL performance is called negative transfer. Negative transfer can also be described as the situation where accurate forecasting for less difficult tasks are negatively affected by inaccurate forecasting results for harder tasks [2, 12, 13].

This study proposes a novel MTL framework based on NARMAX-LSTM that can simultaneously forecast several tasks to reduce the complex process of designing the neural network models and improving the forecast accuracy using models that avoid or overcome experience negative transfer and overfitting.

The main contributions of the work are summarized as follows:

1) The proposal of a novel MTL framework using NARMAX-LSTM that addresses the challenges of overfitting and negative transfer problems.
2) The design and scheme of a single model structure that can be used for multiple tasks/cases (e.g. forecasting) simultaneously.
3) The analysis of the performance of MTL NARMAX – LSTM, and the comparison with single-task baselines for three different time horizons.

II. Theoretical Background for Multitask Learning and Narmax

A. Multitask Learning

MTL is a machine learning approach for simultaneously solving and learning multiple tasks that related to each other by shared knowledge among the tasks [5, 14]. MTL is also called learning with auxiliary tasks, jointly learns and learns to learn [15]. The auxiliary tasks are expected to help the main task train or learn its tasks by supplying knowledge and representation [16]. The definition of MTL is as follows:

Given m learning tasks $\{T_i\}_{i=1}^{m}$ where the subset of the tasks or all the tasks are related but not exactly alike, multitask learning aims to improve the learning of a model for T_i by using the information contained in the m tasks.

MTL can be seen as a machine learning way to mimic the activity of how human learns. The information or knowledge that we acquire from somewhere can often be transferred to and/or used somewhere else. For instance, skills learned from a sport activity can usually applied to another sport, such as skills learned in squash and tennis can benefit each other [4,15,16]. MTL is useful for dealing with a data modelling problem under the following scenario: a set of tasks have commonalities between each other; therefore, the model for each task could gain advantages by sharing knowledge across the tasks than training then independently [5, 16].

MTL methods can be roughly categorized into two classes soft parameter sharing and hard parameter sharing [15, 16, 17]. The fundamental idea of hard parameter sharing is sharing the hidden layer among all tasks and having output separately or allocated several specific output layers for specific tasks [15,17, 18]. This approach dramatically impacts the risk of overfitting and minimizing the loss function of multiple tasks [15, 16]. The rationale of this claim is as follows: the more tasks learn or train jointly, the more the MTL model has to capture all representation or information from all tasks and the less the model has the opportunity of overfitting [15]. Baxter [19] demonstrated the possibility of smaller overfitting by using the shared parameters in comparison that in task-specific parameters.

MTL is useful for dealing with a data modelling problem under the following scenario: a set of tasks have commonalities between each other; therefore, the model for each task could gain advantages by sharing knowledge across the tasks than training then independently [5, 16].

Compared to soft parameter sharing, hard parameter sharing is more likely to use in MTL architecture while soft parameter sharing extent their used in multi task optimisation. These two approaches are not ideal enough to accurately describe the multi task expertise [7, 15]. The hard parameter sharing architecture is shown in Figure. 1.

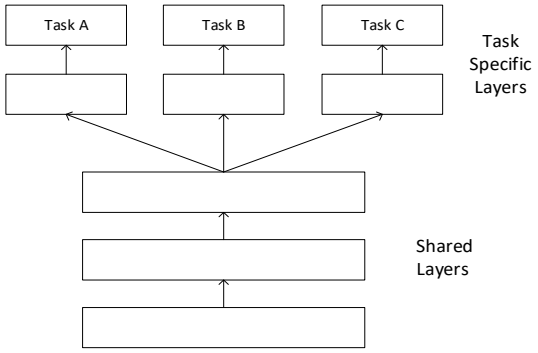


Figure 2. Hard parameter sharing architecture
Source: adapted from [15, 18]

B. NARMAX

NARMAX is transparent non-linear system identification method which focuses on determining the best model structure including selecting most useful and important model terms based on available systems output and output data [10]. The NARMAX model is expressed as follows:

$$y(t) = f\Big(y(t-1), y(t-1), \dots, y(t-n_y), u(t-d), u(t-d-1) \dots, u(t-d-n_u),\ e(t-1), e(t-2) \dots, e(t-n_e)\Big) + e(t) \quad (1)$$

where $y(t)$, $u(t)$ and $e(t)$ are the system output, input, and noise sequences, respectively; n_y, n_u , n_e are the maximum delays of the output, input and noise; $f(.)$ is nonlinear function or mapping, and d is a time delay typically set to d=1. The problems or tasks that is relevant to nonlinear systems identification are also suitable to neural networks.

The nonlinear mapping used in this study is based on LSTM neural networks. LSTM as nonlinear neural networks, which can be used to deal with nonlinear time series modeling tasks.

III. The Proposed Model

A. Model

The proposed NARMAX-LSTM model is designed based on the characteristics of the problem to be solved. If the input data pattern changes over time, like in a time series modelling problem, the feed-forward neural network may not perform well. This underperformance results from that the preceding output data are not being

used as input to derived the relationship between the input and output.

This study uses a direct approach to generate forecast for time series data. The direct approach uses a separate model for each fixed horizon h of interest [21,22,23]. The h steps prediction of tide level forecasting using NARMAX-LSTM as seen in [21] can be define as:

$$y(t) = f\Big(y(t-h), y(t-h-1), \dots, y\big(t-h-n_y\big), u(t-1), u(t-2) \dots, u(t-n_u),\ e(t-1), e(t-2) \dots, e(t-n_e)\Big) + e(t) \quad (2)$$

The multi-task model bas NARMAX-LSTM for direct forecasting approach can be represented as:

$$y_i(t) = f_i\ \big(y_1(t-h), y_1(t-h-1), y_1\big(t-h-n_y\big), \dots, y_m(t-h), y_m(t-h-1), y_m\big(t-h-n_y\big), u_1(t-1), u_1(t-2), \dots, u_1(t-n_u), \dots, u_r(t-1), u_r(t-2), \dots, u_r(t-n_u), e_1(t-1), e_1(t-2), \dots, e_1(t-n_e), \dots, e_m(t-1), e_m(t-2), \dots, e_m(t-n_e)\big),\ i = 1, \dots, m \quad (3)$$

Note that in this study the NARMAX model is realized based on LSTM, the functions f_i (i=1,2,…,m) are derived from the mathematical model of LSTM. LSTM for NARMAX model can be established as:

Forget Gate:

$$f(t) = \sigma\,(W_f[y(t-d), u(t-d), e(t-d)] + b_f) \quad (4)$$

Input Gate:

$$i(t) = \sigma\,(W_i[y(t-d), u(t-d), e(t-d)] + b_i) \quad (5)$$

$$\hat{c}(t) = tanh\,(W_c[y(t-d), u(t-d), e(t-d)]) + b_c \quad (6)$$

$$c(t) = f(t).c(t-1) + i(t).\hat{c}(t) \quad (7)$$

Output Gate:

$$o(t) = \sigma\,(W_o[y(t-d), u(t-d), e(t-d)] + b_o) \quad (8)$$

$$h(t) = o(t)\,tanh(c(t)) \quad (9)$$

where σ is the sigmoid function; tanh is the tangent function; i is the input gate; f is the forget gate; o is the output gate; c is the memory cell; h is the hidden layer output; x_t , h_t is the input vector and hidden vector at time step t; W denotes the weight matrix and b denotes the bias.

The sharing layer is located in the hidden layer. The remaining layers are divided into different specific tasks, as illustrated in Fig. 1. The LSTM layer acts as hidden layer and is shared all their properties with all four tasks. The lower layer is task-specific layer which is spilt according to their specific task. Note that in this study multi-task NARMAX-LSTM model is designed for tide level forecasting, more details are given in the next section.

IV. Experiments

A. Data Preparation

This study is concerned with tide level prediction. The experimental dataset used consist of about 29,000 samples. The data are time series of tide level recorded every 15 minutes. This study considered three different forecast horizons: one step, two steps ahead and three steps, corresponding to 15 minutes, 30 minutes and 45 minutes ahead of predictions. In other words, the model will predict on the next interval of 15 minutes, 30 minutes and 45. The data are divided into three parts: training, test and validation.

B. Multitask LSTM-NARK Architecture

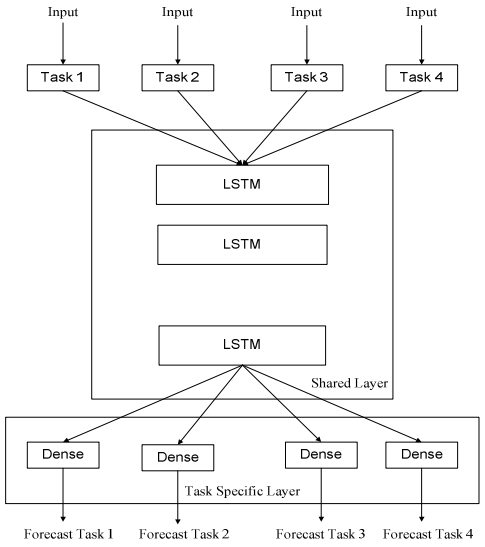


Figure 1. Architecture of Multitask NARMAX-LSTM

The multi task problems in this study are to predict tide level in four stations (locations) simultaneously. The first task or main task is to forecast the tidal level in Wick, and the other tasks (as auxiliary tasks) are to predict tidal in Workington (task 2), Ullapool (task 3) and New Heaven (task 4).

The prediction capability of the models was tested not only tested on validation and test data but also on other new tasks, including the tide level data at Barmouth, Cromer, Milford Haven and Tobermory.

C. Performance Metrics

Multiple loss functions were defined based on error values of the outputs of multiple tasks and the predicted results. STL may yield multiple local minimum values and the possibility that gradient vanishing appears in the training process is high. Because the tasks in MTL are executed simultaneously, the learning process can be prevented from falling into local minimal through sharing information between tasks [24].

An overall loss of the prediction model was obtained by summing losses of different tasks. The loss of task n is defined in the form of the root mean square error:

$$L_{RMSE} = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}} \quad (10)$$

where $\hat{y}_i$ is the predicted value, y_i is actual value and n is the number of observations. The various task-specific loss functions must be combined into a single aggregated loss function by the joint loss function, which is defined as:

$$L = \frac{1}{m}\sum_{m=1}^{m} L_{\text{RMSE}} \quad (11)$$

where L is the joint loss and m is the number of tasks.

D. Results and Discussion

The optimal time lag in the multitask NARMAX – LSTM structure for each network were determined by using the lowest joint loss value obtained after performing empirical analysis. To find best neural networks architecture, the parameter settings for all the network models are as follows: the iteration number for each time horizon case is set to be 200; the number of hidden layers, hidden nodes, and time lags for input and output are set to be from 1 to 5, 32 to 256, and 1to 4, respectively.

The details of joint loss results for three different time horizon based on the best parameter setting are summarized in Table I, II and III.

It was observed from Table I that for one step ahead prediction, the optimal setting for the multi-task NARMAX-LSTM are as follows: time lag for input is 2 (n_u), time lag for output is $1(n_y)$, time lag for noise is 1 (n_e) and there are a total of 128 nodes in the hidden layer.

The one step ahead forecast model based on the optimal model in table I can be written as:

$$\hat{y}_i(t) = f_i\ (y_1(t-1), y_1(t-2), \dots, y_4(t-1), y_4(t-2), u_1(t-1), \dots, u_4(t-1), e_1(t-1), \dots, e_4(t-1)) \quad (12)$$

For two step ahead case, the best model setting as follow: time lag for input is 4 (n_u), time lag for output is $1(n_y)$, time lag for noise is 1 (n_e) and there are 128 nodes in the hidden layer as shown in Table I.

$$\hat{y}_i(t) = f_i\ \big(y_1(t-2), y_1(t-3), \dots, y_4(t-2), y_4(t-3), u_1(t-1), u_1(t-2), u_1(t-3), u_1(t-4), \dots, u_4(t-1), u_4(t-2), u_4(t-3), u_4(t-4), e_1(t-1), \dots, e_4(t-1)\big) \quad (13)$$

TABLE I. TIME LAG PERFORMANCE FOR ONE STEP AHEAD FORECASTING

No	Time Lag	LSTM Setting		Joint Loss (average RMSE)
		Hidden Layer	Hidden Node	
1.	$(n_y = 1, n_u = 1, n_e = 1)$	4	64	0.3088
2.	$\mathbf{(n_y = 1, n_u = 2, n_e = 1)}$	**1**	**128**	**0.1542**
3.	$(n_y = 1, n_u = 3, n_e = 1)$	3	32	0.3185
4.	$(n_y = 1, n_u = 4, n_e = 1)$	1	32	0.2578
5.	$(n_y = 2, n_u = 1, n_e = 1)$	2	64	0.2416
6.	$(n_y = 2, n_u = 2, n_e = 1)$	1	32	0.1931
7.	$(n_y = 2, n_u = 3, n_e = 1)$	1	64	0.1774
8.	$(n_y = 2, n_u = 4, n_e = 1)$	2	64	0.2126
9.	$(n_y = 3, n_u = 1, n_e = 1)$	2	64	0.2248
10.	$(n_y = 3, n_u = 2, n_e = 1)$	2	32	0.2276
11.	$(n_y = 3, n_u = 3, n_e = 1)$	1	64	0.1854
12.	$(n_y = 3, n_u = 4, n_e = 1)$	1	64	0.1953
13.	$(n_y = 4, n_u = 1, n_e = 1)$	3	32	0.2988
14.	$(n_y = 4, n_u = 2, n_e = 1)$	1	32	0.2101
15.	$(n_y = 4, n_u = 3, n_e = 1)$	2	32	0.2380
16.	$(n_y = 4, n_u = 4, n_e = 1)$	3	64	0.2683

As shown in Table III, the best model for the three steps ahead horizon case is as follow: two lagged input variables, two lagged output variables, one lagged noise variable and 64 nodes in hidden layer. For three steps ahead prediction, the model is:

TABLE II. TIME LAG PERFORMANCE FOR TWO STEPS AHEAD FORECASTING

No	Time Lag	LSTM Setting		Joint Loss (average RMSE)
		Hidden Layer	Hidden Node	
1.	$(n_y = 1, n_u = 1, n_e = 1)$	1	64	0.1287
2.	$(n_y = 1, n_u = 2, n_e = 1)$	1	32	0.1607
3.	$(n_y = 1, n_u = 3, n_e = 1)$	3	128	0.2273
4.	$\mathbf{(n_y = 1, n_u = 4, n_e = 1)}$	**1**	**128**	**0.1262**
5.	$(n_y = 2, n_u = 1, n_e = 1)$	1	64	0.1607
6.	$(n_y = 2, n_u = 2, n_e = 1)$	3	256	0.2340
7.	$(n_y = 2, n_u = 3, n_e = 1)$	1	32	0.1798
8.	$(n_y = 2, n_u = 4, n_e = 1)$	4	64	0.2326
9.	$(n_y = 3, n_u = 1, n_e = 1)$	3	32	0.2697
10.	$(n_y = 3, n_u = 2, n_e = 1)$	1	32	0.1633
11.	$(n_y = 3, n_u = 3, n_e = 1)$	1	64	0.1693
12.	$(n_y = 3, n_u = 4, n_e = 1)$	1	64	0.1621
13.	$(n_y = 4, n_u = 1, n_e = 1)$	4	64	0.5914
14.	$(n_y = 4, n_u = 2, n_e = 1)$	2	64	0.2340
15.	$(n_y = 4, n_u = 3, n_e = 1)$	2	128	0.2083
16.	$(n_y = 4, n_u = 4, n_e = 1)$	1	128	0.1731

$$\hat{y}_i(t) = f_i\ \big(y_1(t-3), y_1(t-4), y_1(t-5), \dots, y_4(t-3), y_4(t-4), y_4(t-5), u_1(t-1), u_1(t-2), \dots, u_4(t-1), u_4(t-2), e_1(t-1), \dots, e_4(t-1)\big) \quad (13)$$

TABLE III. TIME LAG PERFORMANCE FOR THREE STEPS AHEAD FORECASTING

No	Time Lag	LSTM Setting		Joint Loss (average RMSE)
		Hidden Layer	Hidden Node	
1.	$(n_y = 1, n_u = 1, n_e = 1)$	1	64	0.1739
2.	$(n_y = 1, n_u = 2, n_e = 1)$	2	32	0.2317
3.	$(n_y = 1, n_u = 3, n_e = 1)$	2	32	0.2252
4.	$(n_y = 1, n_u = 4, n_e = 1)$	2	128	0.1929
5.	$(n_y = 2, n_u = 1, n_e = 1)$	5	64	0.2767
6.	$\mathbf{(n_y = 2, n_u = 2, n_e = 1)}$	**1**	**64**	**0.1546**
7.	$(n_y = 2, n_u = 3, n_e = 1)$	3	128	0.2427
8.	$(n_y = 2, n_u = 4, n_e = 1)$	1	256	0.1566
9.	$(n_y = 3, n_u = 1, n_e = 1)$	3	64	0.2723
10.	$(n_y = 3, n_u = 2, n_e = 1)$	1	32	0.2037
11.	$(n_y = 3, n_u = 3, n_e = 1)$	2	256	0.1802
12.	$(n_y = 3, n_u = 4, n_e = 1)$	2	128	0.2324
13.	$(n_y = 4, n_u = 1, n_e = 1)$	1	32	0.2733
14.	$(n_y = 4, n_u = 2, n_e = 1)$	1	128	0.1723
15.	$(n_y = 4, n_u = 3, n_e = 1)$	3	64	0.2985
16.	$(n_y = 4, n_u = 4, n_e = 1)$	3	64	0.2596

By analyzing the time lag variation and the LSTM setting from three different time horizons in Table I, II and III, we can see that the numbers of input time lag, output

time lag, hidden layers and hidden nodes have no correlations at all. The increase of the network complexity, namely, the increase of hidden layers, number of nodes and time lag does not have significant effect on the accuracy or joint loss value of the models. In some cases, it needs a relatively larger number of hidden layers to generate the lowest joint loss value while in other cases it only needs a single hidden layer with a minimal number of hidden nodes. This means that manual intervention is still needed in the fine-tune process of the neural network models.

The models for the above three different time horizons were tested using four new tasks, their performance were analyzed and summarized as follows. As can be seen in Table IV, individual loss values of the tasks (old and new) in a same time horizon are slightly different from each other. No task has an obvious larger loss value than other tasks. This means that the performance of some specific tasks does not significantly impact the performance of the models for other tasks. The joint loss value results show that both new and old tasks do not encounter overfitting.

TABLE IV. COMPARISON OF INDIVIDUAL LOSS VALUE OF OLD TASKS AND NEW TASKS

Multitask LSTM-NARMAX with Old Tasks					
Horizon	Task 1	Task 2	Task 3	Task 4	Joint Loss
1	0.0981	0.2167	0.1492	0.1527	0.1542
2	0.1228	0.1356	0.1136	0.1331	0.1262
3	0.0973	0.2026	0.1716	0.1472	0.1546
Multitask LSTM-NARMAX with New Tasks					
Horizon	Task 1	Task 2	Task 3	Task 4	Joint Loss
1	0.1201	0.1717	0.1959	0.1608	0.1621
2	0.1504	0.1058	0.1462	0.103	0.1263
3	0.0896	0.1415	0.2817	0.129	0.16045

To further assess the capability of the proposed multi-task NARMAX-LSTM model, its performance was compared with LSTM, ARIMA and Backpropagation models. The comparison results are shown in Figure 2, from which it is clear that in term of joint loss, multi-task NARMAX LSTM outperforms all the other models in all task in three different time horizon.

From Figure 2, it can be seen that ARIMA severely suffers negative transfer: its RMSE values for Tasks 1 and 4 are much higher than those of the other models. The ARIMA model also shows the worst performance for all tasks and for all time horizons. Backpropagation also suffers negative transfer in all time horizons. The negative transfer occurs in Task 2 and Task 4 for the Backpropagation model.

The LSTM model performs good in one step ahead especially in task 1, task 2 and task 3 but have poor results in other time horizons.

V. CONCLUSION AND FUTURE WORKS

The paper proposed a novel multi-task learning approach for time series forecasting by effectively integrating the NARMAX and LSTM models. Unlike traditional time series forecasting methods which follow 'a single model for a single task' practice, the proposed MLT-NARMAX-LSTM method deals with time series modelling and forecasting from a multi-task viewpoint: it does not treat a single time series modelling and forecasting as an isolated task, but one of a set of tasks; the individual tasks are not the same but closely associated to each other (e.g. common features, similar change patters, sharable knowledge and information). The proposed method has several advantages, for example, it generates robust models by making use of information shared by all the tasks; it avoids overfitting that can easily occur when using traditional time series modelling methods; the combination of the two different types of modelling methods, namely, NARMAX and LSTM, makes the proposed modelling framework immune to negative transfer. The predictive capability of MTL-NARMAX-LSTM network models was tested and evaluated on several real datasets of tide levels. Experimental results confirm the good performance of the proposed method.

In future, more benchmarks and case studies for applications in different areas will be investigated to further explore and improve the performance of the proposed method. To explore a new or improved MTL framework by employing evolutionary computation methods and transfer learning approach will be another interesting direction.

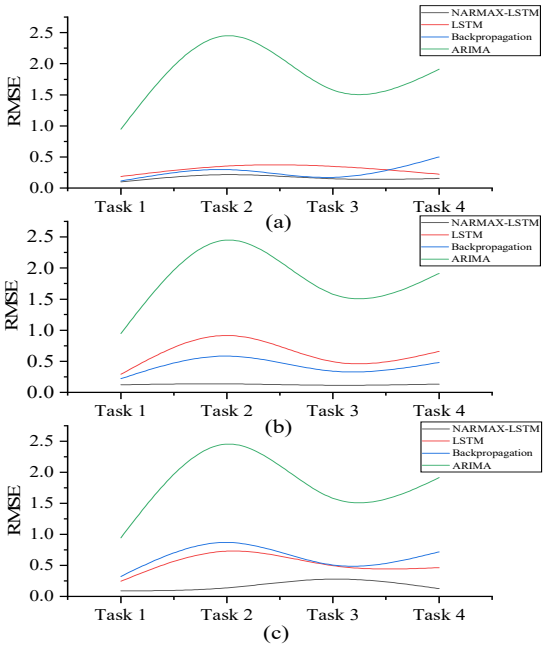


Figure 2. Comparison of the joint loss of the NARMAX-LSTM, LSTM, Backpropagation, and ARIMA. (a) One Step Ahead (b) Two Steps Ahead (3) Three Steps Ahead

ACKNOWLEDGMENT

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), EPSRC (Ref. EP/I011056/1), EU Horizon 2020 (Ref. 637302). Nerfita Nikentari gratefully acknowledges a scholarship from the Indoensia Endowment Fund for Education/Lembaga Pengelola Dana Pendidikan (LPDP).

REFERENCES

[1] P. Liu, X. Qiu, and X. Huang, "Recurrent Neural Network for Text Classification with Multi-Task Learning," Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI-16, pp. 2873–2879, 2016.

[2] M. Dorado-Moreno, N. Navarin, P.A. Gutiérrez, L. Prieto, A. Sperduti, S. Salcedo-Sanz, C. Hervás-Martínez, "Multi-task learning for the prediction of wind power ramp events with deep neural networks, "Neural Networks, Volume 123, 2020,pp. 401-411, 2020

[3] A. Chantal, "Learning What to Share in Multi-Task Learning, " Master Dissertation, School of Informatics University of Edinburgh, Edinburgh, 2019..

[4] Y. Zhang and Q. Yang, "An overview of multitask learning," National Science Review, vol. 5, no. 1, pp. 30–43, Sep. 2017.

[5] X, Jianpeng, "Multitask learning and its application to geospatio-temporal data, " Ph. D Dissertation, Computer Science and Engineering, Michigan State University, Michigan, 2017..

[6] R. Caruana, "Multitask learning," Machine Learning, vol. 28, no. 1, pp. 41–75, 1997, doi: 10.1023/a:1007379606734

[7] X. Sun, W. Xu, H. Jiang, and Q. Wang, "A deep multitask learning approach for air quality prediction," Annals of Operations Research, 2020.

[8] M. Nunes, E. Gerding, F. McGroarty, and M. Niranjan, "A comparison of multitask and single task learning with artificial neural networks for yield curve forecasting," Expert Systems with Applications, vol. 119, pp. 362–375, 2019.

[9] J. Stadermann, W. Koska, and G. Rigoll, "Multi-task Learning Strategies for a Recurrent Neural Net in a Hybrid Tied-Posteriors Acoustic Model," INTERSPEECH 2005 - Eurospeech, 9th European Conference on Speech Communication and Technology, Lisbon, 2005.

[10] S. A. Billings, Nonlinear System Identification: NARMAX Methods in the Time, Frequency, and Spatio-Temporal Domains. Chichester, West Sussex: Wiley, 2013.

[11] B. R. Paredes, "Multitask and Transfer Learning for Multi-Aspect Data." PhD Dissertation, University College London, 2014.

[12] H. B. Lee, E. Yang, and S. J. Hwang, "Deep Asymmetric Multi-task Feature Learning," Proceedings of the 35 th International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018.

[13] S. Liu, Y. Liang, and A. Gitter, "Loss-Balanced Task Weighting to Reduce Negative Transfer in Multi-Task Learning," Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, no. 1, pp. 9977–9978, Jul. 2019.

[14] H. Luo, J. Cai, K. Zhang, R. Xie, and L. Zheng, "A multitask deep learning model for short-term taxi demand forecasting considering spatiotemporal dependences," Journal of Traffic and Transportation Engineering (English Edition), vol. 8, no. 1, pp. 83–94, 2021.

[15] S. Ruder, "An Overview of Multi-Task Learning in Deep Neural Networks," 2017, arXiv:1706.05098.

[16] M. Crawshaw, "Multi-Task Learning with Deep Neural Networks: A Survey," 2020, arXiv:2009.09796.

[17] Partoo Vafaeikia, Khashayar Namdar, and Farzad Khalvati, "A Brief Review of Deep Multi-task Learning and Auxiliary Task Learning," 2020, arXiv:2007.01126.

[18] S. Wang, Q. Wang, and M. Gong, "Multi-Task Learning Based Network Embedding," Frontiers in Neuroscience, vol. 13, 2020.

[19] J. Baxter, "A Bayesian/Information Theoretic Model of Learning to Learn via Multiple Task Sampling.," Machine Learning, vol. 28, pp. 7–39, 1997.

[20] Partoo Vafaeikia, Khashayar Namdar, and Farzad Khalvati, "A Brief Review of Deep Multi-task Learning and Auxiliary Task Learning," 2020, arXiv:2007.01126.

[21] J. R. Ayala 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 Kp index using NARX models," Space Weather, vol. 14, no. 10, pp. 899–916, 2016.

[22] H.-L. Wei and S. A. Billings, "An efficient nonlinear cardinal B-spline model for high tide forecasts at the Venice Lagoon," Nonlinear Processes in Geophysics, vol. 13, no. 5, pp. 577–584, 2006.

[23] H.-L. Wei and S. A. Billings, "Long term prediction of non-linear time series using multiresolution wavelet models," International Journal of Control, vol. 79, no. 6, pp. 569–580.

[24] Q. Zhang, S. Wu, X. Wang, B. Sun, and H. Liu, "A PM2.5 concentration prediction model based on multitask deep learning for intensive air quality monitoring stations," Journal of Cleaner Production, vol. 275, p. 122722, 2020.

B

Paper 2

Proceedings of the 27th International Conference on Automation & Computing, University of The West England, Bristol, UK, 1-3 September 2022

Tide Level Prediction Using NARX-based Recurrent Neural Networks

Nerfita Nikentari, Hua-Liang Wei
Department of Automatic Control and Systems Engineering
University of Sheffield, Sheffield
nnikentari1@sheffield.ac.uk,w.hualiang@sheffield.ac.uk (*corresponding author: H.-L. Wei)

2022 27th International Conference on Automation and Computing (ICAC) | 978-1-6654-9807-4/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICAC55051.2022.9911163

***Abstract*—Tide level forecasting is one of major components in oceanography and marine research. The future prediction of tides is an important issue for human and environment. This study proposes new tide level prediction methods with multiple input multiple output (MIMO) approach based on nonlinear autoregressive with exogenous inputs (NARX) model and three types of recurrent neural networks (RNNs), namely, long short-term memory (LSTM), bi-directional long short-term memory (biLSTM) and gated recurrent unit (GRU) networks. We develop three types of NARX models: NARX-LSTM, NARX-biLSTM and NARX-GRU. We evaluate the three models and compare their performances with LSTM, biLSTM and GRU network models. It turns out that the proposed MIMO NARX based RNN models, especially NARX-LSTM and NARX-GRU have better performance in comparison with their RNN counterparts (i.e., LSTM, biLSTM, and GRU). The NARX-GRU model shows the best performance for one step and three step ahead prediction in terms of RMSE in each location and average RMSE, while NARX-LSTM performs better for two steps ahead prediction in terms average RMSE. The results also show that the mixture of RNN and NARX improves the performance of RNN model for all the four tasks at four locations in the UK (Wick, Workington, Ullapol and New Heaven) and for the three time horizons (i.e., 1, 2 and 3 steps ahead).**

***Keywords*- tide level forecasting, MIMO, NARX, LSTM, biLSTM, GRU**

I. *INTRODUCTION*

Tides are an ocean natural phenomenon, which are mainly influenced by the gravitational forces of the rotation of the earth and the alignment of the sun and moon. The rising and the falling of the sea levels can also be induced by meteorological conditions and shallow water. Predicting future tide levels plays a vital part in keeping people and the environment safe because the tide level fluctuations play a substantial role in marine meteorology, the operational of oceanography and coastal zone management. In the fields of marine meteorology and oceanography the future tide level data will be helpful for the disposal and movements of sediments, tracers and pollutants. Besides that, tide level data is also crucial for coastal structure harbor project such as constructing cross sea bridges or navigating large engineering. For coastal zone management, the tide level prediction can be used as early warning systems of occurrences of Tsunamis and prevent flooding caused by sea rise [1] – [4].

The conventional method to predict tidal is to build a model based on astronomical factors. This method is called harmonic analyses, but this approach can be difficult to implement when the non-astronomical features play more significant role than astronomical ones. The tide level data for forecasting is usually presented as a time series [5,6]. An alternative technique to time series prediction, for instance Nonlinear Autoregressive Neural Network with Exogenous inputs (NARX) models and Recurrent Neural Networks (RNNs).

Many studies have been conducted to forecast tide levels using different linear or nonlinear models with single input single output (SISO) or multiple input single output (MISO). NARX model and RNN networks are among the commonly used techniques for tide level prediction, which usually involves using nonlinear MIMO models.

Wu et al. [7] carried out tide level forecasting using NARX neural network models and harmonic analysis with multiple inputs and single output. Nunno et al. [8] applied NARX network and LSTM models for forecasting high tide level of the city of Venice. In [9] and [10], LSTM was applied to tide level prediction. The models used in [8,9,10] involved multiple outputs, representing multiple tide levels of h steps ahead. These techniques can be easily implemented with LSTM models but these models still need to run n times if there are a number of n locations to forecast. An alternative MIMO method for tide level prediction is to combine and integrate neural networks and non-linear MIMO models, which can simultaneously produce predictions of tide levels at several locations (geological areas).

This study proposes a new class of nonlinear models and neural networks with multiple input and multiple output to predict tide level in several areas simultaneously. It aims to demonstrate the feasibility of using MIMO models based on two types of neural networks: a) solely using recurrent neural networks models, and b) recurrent neural networks combined with NARX models. Three RNN models, namely, LSTM, biLSTM and Gated Recurrent Unit (GRU), are built in this study. These three neural networks are then integrated with the NARX model, respectively. Therefore, a total of six models are developed: LSTM, biLSTM, GRU, NARX-LSTM, NARX-biLSTM and NARX-GRU. Comparative studies are performed to evaluate the two types of models based on the same real data and for tide level prediction with three different time horizons.

All the six models are used to forecast the tide levels of four different locations in the UK. This study is concerned with one step ahead, two steps ahead and three steps ahead predictions. The main contributions of the work are summarized as follows:

1) The proposal of using different recurrent neural network models, namely, LSTM, biLSTM, GRU for tide level prediction by exploring MIMO nonlinear autoregressive with exogenous inputs with GRU, biLSTM and LSTM and MIMO with GRU, biLSTM and LSTM.
2) The proposal of integrated NARX and RNN models, namely, NARX-LSTM, NARX-biLSTM and NARX-GRU, for tide level prediction.
3) Comprehensive comparative studies of the six models for tide leve prediction at three different horizons based on real data.

II. METHODOLOGY

A. MIMO Approach

The MIMO approach is where several or many outputs are simultaneously predicted using multiple inputs, in which the output of the prediction model is a vector of future values predicted by using only single model structure and single dataset [11,12]. The multiple output in this forecasting strategy is several h steps ahead data.

This study proposes different implementation ways of MIMO models. The designed MIMO architecture consists of multiple input and multiple output from several datasets (measured at different locations). The multiple outputs in our model is h steps ahead forecast results in multiple locations. The model only delivers single h steps ahead prediction but data are from multiple locations. The model is able to forecast the tide level at different locations one time by using the results from a few single models. The inputs are tide levels in four locations at the present time instant t and the output are the predicted data of tide level for h step/s ahead. This study implements direct forecasting models where there are h different models for h different time step/s.

B. MIMO RNN

As mentioned earlier, LSTM, biLSTM and GRU will be considered for implementing MIMO forecasting. The general architecture of the MIMO RNN is shown in Figure 1, where $u_k(t)(k = 1,2,\dots,r)$ and $y_j(t)(j = 1,2,\dots,i)$ are the system inputs and outputs, n, r, i are the numbers of lags, stations/locations and output variables, respectively. The outputs in this study assumed to be same with the number of the inputs. For example if the inputs are the location in four locations (r = 4), the output are the h steps ahead forecasting in four locations (i =4).

The inputs layer of the proposed network model are sequences of the tide data in r locations followed by an RNN layer (LSTM or biLSTM or GRU), or a dense layer or fully connected layer and the outputs layer are tide level data in i location at time t.

LSTM is a variant of RNN that was proposed by Hochreiter and Schmidhuber [13]. RNNs have disadvantages in training sequences with long term dependencies which cause gradient explosion or vanishing, and usually the longer the interval becomes, the more affected the gradient become [13]-[16]. LSTM, which works well on a wide variety of problems, is explicitly designed to address these issues by introducing memory cells. The functions of the cells are to determine the extent of the information to be erased, updated and exposed [13],[14], [16] – [18].

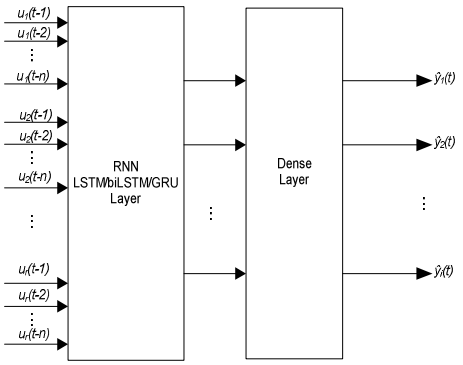


Figure 1. MIMO RNN Architecture

The basic structure of LSTM usually comprises one memory cell (c_t) and three gates: forget gate (f_t), input gate (i_t), and output gate (o_t). The forget gate decide what information will be thrown away, and how the information is taken out from the memory state of the previous step. The input gate decides what information will be added in the cell state and the output gate manage what information should be issued from the current state [14, 16, 19]. BiLSTM is combination of two LSTM layers, where one layer is to move forward and the other one move backward. The backward and forward LSTM was proposed to overcome the limitation of LSTM that was only able to absorb previous information but failed to gain future information [19]-[21].

GRU model is adapted from LSTM, aiming to simplify LSTM by using only two gates: update gates (z_t) and reset gate (r_t). The input gate and forget gate in LSTM are merged into update gate in GRU. This approach makes GRU require less training time and reduce computational cost [21]-[25].

C. MIMO NARX RNN

NARX is a nonlinear identification model that uses lagged inputs and outputs to represent the system behavior. NARX model is a special case of the nonlinear autoregressive moving average model with exogenous inputs (NARMAX) model that does not contain lagged noise variables. NARX model, implemented using a neural network structure, is usually called recurrent NARX network [26].

This study employs a direct approach using RNN to predict tide levels. The basic idea behind the direct approach is that forecasting for h time horizons (h steps ahead forecasting) are realized using different models [27-30]. The h steps prediction of tide level can be obtained using NARX LSTM or NARX biLSTM. The model for h steps ahead prediction can be expressed as [27-29]:

$$y(t) = f\big(y(t-h), y(t-h-1), \dots, y\big(t-h-n_y\big), u(t-1), u(t-2) \dots, u(t-n_u),\big) \quad (1)$$

where $y(t)$ *and* $u(t)$ and $e(t)$ are the system output and inputrespectively;$n_y$and n_u are the maximum lags of the output and input; $f(.)$ is nonlinear function or mapping.

MIMO NARX RNN is an MIMO forecasting model that employs NARX by incorporating LSTM, biLSTM or GRU. The MIMO model based on these three neural networks for direct forecasting approach can be defined as [26,31]:

$$y_i(t) = f_i\big(y_1(t-h), y_1(t-h-1), y_1(t-h-n_y), \dots, y_m(t-h), y_m(t-h-1), y_m(t-h-n_y), u_1(t-1), u_1(t-2), \dots, u_1(t-n_u), \dots, u_r(t-1), u_r(t-2), \dots, u_r(t-n_u)\big),\ i = 1, \dots, m \quad (2)$$

The general architecture of MIMO NARX RNN is shown in Figure 2. The architecture of MIMO NARX RNN is basically similar to that of MIMO RNN in that both have four layers: input layer, RNN layer, dense layer and output layer. The difference is in the inputs: the MIMO NARX RNN model needs previous output values. This study focuses on the direct approach, and the observational values of the output are assumed to be known [24].

In (2), f_i (i=1,2,…,m) are approximated by using LSTM or biLSTM and GRU. The NARX-LSTM and NARX-biLSTM models can be represented as [13]:

Forget Gate:

$$f(t) = \sigma\,(W_f[y(t-d), u(t-d)] + b_f) \quad (3)$$

Input Gate:

$$i(t) = \sigma\,(W_i[y(t-d), u(t-d)] + b_i) \quad (4)$$

$$\hat{c}(t) = tanh\,(W_c[y(t-d), u(t-d)]) + b_c \quad (5)$$

$$c(t) = f(t).c(t-1) + i(t).\hat{c}(t) \quad (6)$$

Output Gate:

$$o(t) = \sigma\,(W_o[y(t-d), u(t-d)] + b_o) \quad (7)$$

$$h(t) = o(t)\,tanh(c(t)) \quad (8)$$

GRU for NARX model can be represented as [32]:

Hidden Gate:

$$h(\mathrm{t}) = \big(1 - \mathrm{z(t)}\big).h(\mathrm{t}-1) + z(\mathrm{t}).h(\mathrm{t}) \quad (9)$$

Update Gate:

$$z(\mathrm{t}) = \sigma\,(\mathrm{W_z}[\mathrm{y(t-d), u(t-d)}]) \quad (10)$$

Reset Gate:

$$r(\mathrm{t}) = \sigma\,(\mathrm{W_r}[\mathrm{y(t-d), u(t-d)}]) \quad (11)$$

III. Results and Discussion

A. Data Preparation

The multiple inputs considered in this study are the tide levels in four stations/locations of the UK (Wick, Workington, Ullapol and New Heaven), and the multiple outputs are the predicted values (of *h* steps ahead) of tide levels in the four locations, each using its own *h* steps ahead prediction model.

The training set, the validation set, and the test set have the sample sizes around 13,000, 3,000,and 3,000, respectively.

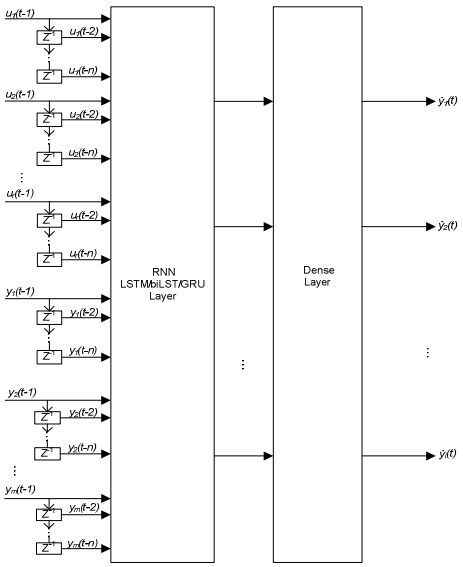


Figure 2. MIMO NARX RNN Architecture

B. Parameter Setting

In order to fairly compare the six models, all RNN models are trained, validated and tested using the same parameter setting. The setting can be seen in table I.

TABLE I. Parameter Setting for MIMO RNN/MIMO NARX RNN

Parameter	Values
Number of hidden layer	1,2,3,4,5
Number of nodes	25,32,64,128,256
Activation function	SGDM
Number of iterations	200
Time Lag for NARX model	n_y = 1,2,3,4 n_u = 1,2,3,4

C. MIMO RNN

For the MIMO RNN models, a number of different neural network settings were implemented. The optimal settings obtained for the three RNN network models are shown in Table II. These results were achieved after fine tuning of the RNN parameters. The best results for LSTM model is by applying and testing a large number of hidden layers and using large time horizons. GRU delivers the best accuracy for forecasting in all time horizons while biLSTM failed to gain good overall performance.

TABLE II. MIMO RNN Performance

RNN Model	Forecast Horizon	Number of Hidden Layer	Number of Hidden Node	Average RMSE
LSTM	1	1	256	0.2781
	2	4	128	0.4894
	3	5	256	0.4841
biLSTM	1	1	32	0.5653
	2	1	128	0.7131
	3	1	32	0.6285
GRU	1	2	256	**0.1383**
	2	1	64	**0.1082**
	3	5	25	**0.4447**

D. MIMO NARX RNN

The optimal structure of MIMO NARX RNN model are determined using the lowest average RMSE values obtain after performing an empirical analysis as shown in Table III. As seen from Table I, for one step ahead and two steps ahead forecasting, the GRU models, with time lags n_y=4 and n_u=2, perform best among the group of MIMO NARX RNN. The hidden layer of the optimal NARX-GRU model for one step and two steps ahead includes 128 and 64 nodes, respectively. For the three steps ahead case, the best model is still NARX-GRU, with the following setting: time lags n_y=1 and n_u=4; there are 128 nodes in the hidden layer. The details of network parameter setting for the best NARX RNN model are shown in Table IV.

The models identified by the NARX RNN based on the optimal results in Table III for each horizon are listed below:

1) One Step Ahead

$$\hat{y}_i(t) = f_i\ \big(y_1(t-1), y_1(t-2), y_1(t-3), y_1(t-4), y_1(t-5), \dots, y_4(t-1), y_4(t-2), y_4(t-3), y_4(t-4), y_4(t-5), u_1(t-1), u_1(t-2), \dots, u_4(t-1), u_4(t-2)\big) \quad (12)$$

2) Two Steps Ahead

$$\hat{y}_i(t) = f_i\ \big(y_1(t-2), y_1(t-3), y_1(t-4), y_1(t-5), y_1(t-6), \dots, y_4(t-2), y_4(t-3), y_4(t-4), y_4(t-5), y_4(t-6), u_1(t-1), u_1(t-2), \dots, u_4(t-1), u_4(t-2)\big) \quad (13)$$

3) Three Steps Ahead

$$\hat{y}_i(t) = f_i\big(y_1(t-3), y_1(t-4), \dots, y_4(t-3), y_4(t-4), u_1(t-1), u_1(t-2), u_1(t-3), u_1(t-4) \dots, u_4(t-1), u_4(t-2), u_4(t-3), u_4(t-4)\big) \quad (14)$$

TABLE III. MIMO NARX RNN PERFORMANCE

Network lags	Average RMSE								
	One Step Ahead			Two Steps Ahead			Three Steps Ahead		
	LSTM	biLSTM	GRU	LSTM	biLSTM	GRU	LSTM	biLSTM	GRU
$(n_y = 1, n_u = 1)$	0.1576	0.5364	0.1089	0.1990	0.5070	0.134	0.2840	0.7233	0.1412
$(n_y = 1, n_u = 2)$	0.1717	0.4244	0.1358	0.6959	0.6899	0.1708	0.3054	0.6735	0.1638
$(n_y = 1, n_u = 3)$	0.2483	0.5483	0.1238	0.2372	0.5179	0.1209	0.1787	0.6728	0.1493
$(n_y = 1, n_u = 4)$	0.2041	0.4591	0.1427	0.1999	0.5390	0.1514	0.1755	0.6401	**0.1379**
$(n_y = 2, n_u = 1)$	0.1875	0.4650	0.1037	0.2742	0.4633	0.1462	0.4760	0.7084	0.1545
$(n_y = 2, n_u = 2)$	0.1120	0.5493	0.1053	0.1913	0.7097	0.2333	0.2881	0.7025	0.2111
$(n_y = 2, n_u = 3)$	0.2504	0.5148	0.1531	0.2443	0.6501	0.1309	0.2185	0.6991	0.1532
$(n_y = 2, n_u = 4)$	0.154	0.3841	0.1263	0.2847	0.5355	0.1537	0.1464	0.4114	0.1732
$(n_y = 3, n_u = 1)$	0.3640	0.5163	0.1541	0.2491	0.5312	0.2721	0.4883	0.5957	0.1703
$(n_y = 3, n_u = 2)$	0.1735	0.5553	0.1428	0.2635	0.5132	0.1751	0.2006	0.6928	0.1964
$(n_y = 3, n_u = 3)$	0.1948	0.4520	0.1214	0.2361	0.6943	0.2000	0.2788	0.7154	0.1533
$(n_y = 3, n_u = 4)$	0.1909	0.4619	0.1094	0.2199	0.2969	0.1619	0.2317	0.6299	0.1553
$(n_y = 4, n_u = 1)$	0.1822	0.3736	**0.0955**	0.2167	0.4606	**0.1265**	0.1591	0.5741	0.1455
$(n_y = 4, n_u = 2)$	0.1098	0.4163	0.1704	0.1385	0.5287	0.1284	0.2341	0.7481	0.1699
$(n_y = 4, n_u = 3)$	0.1660	0.4733	0.1216	0.2099	0.432	0.1454	0.1962	0.6849	0.1698
$(n_y = 4, n_u = 4)$	0.1341	0.5130	0.1216	0.1689	0.5524	0.2707	0.2346	0.7377	0.1796

TABLE IV. NEURAL NETWORKS SETTING FOR OPTIMAL RNN NARX MODELS

Model	Forecast Horizon	Number of Hidden Layer	Number of Hidden Node
NARX- LSTM	1	1	256
	2	1	256
	3	1	128
NARX- biLSTM	1	1	25
	2	1	25
	3	1	25
NARX- GRU	1	1	128
	2	1	64
	3	1	128

E. Performance Comparison Between MIMO RNN and MIMO NARX RNN

The comparison of RNN and NARX RNN models were performed based on two metrics: average RMSE and individual RMSE for tide level prediction as each location.

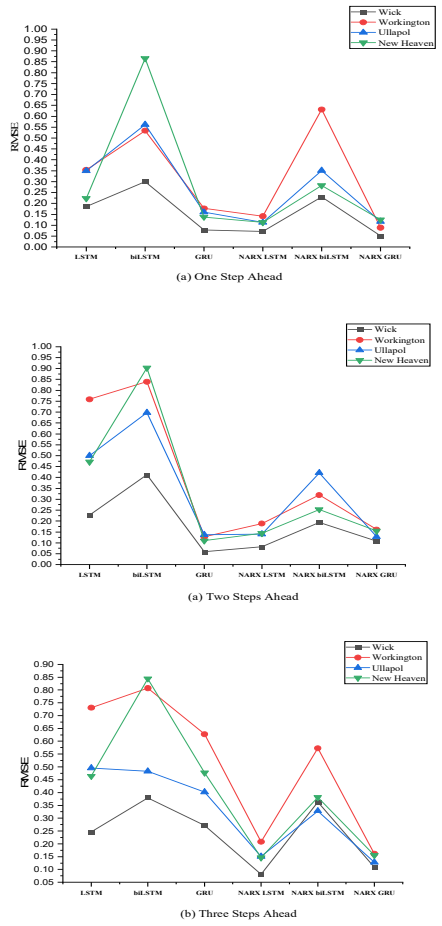
TABLE V. COMPARISON PERFORMANCE OF MIMO-RNN AND MIMO-NARX IN TERM OF AVERAGE RMSE

Model	Time Horizon		
	1	2	3
LSTM	0.2781	0.4894	0.4841
biLSTM	0.5653	0.7131	0.6285
GRU	0.1383	**0.1082**	0.4447
NARX-LSTM	0.1098	0.1385	0.1464
NARX-biLSTM	0.3736	0.2969	0.4114
NARX-GRU	**0.0955**	0.1209	**0.1379**

Table V shows the comparison results based on average RMSE for one step ahead, two steps ahead and three steps

ahead forecasting. It can be seen that in both GRU and NARX GRU groups, GRU gives the best overall performance in comparison with LSTM and biLSTM for one step and three steps ahead. The performance of the biLSTM is the worst among three models: the average RMSE values are larger than 0.5. The biLSTM forecast results are slightly improved when the NARX model deployed in the RNN network where the average RMSE values in the three different time horizons are below 0.45.

Now we compare the RMSE values of the individual models for tide level prediction for different locations. The RMSE values for three prediction time horizons for the four locations are shown in Figure 4. We have the following findings:



(a) One Step Ahead

(a) Two Steps Ahead

(b) Three Steps Ahead

Figure 4. Comparison performance of MIMO RNN and MIMO NARX for every location

1) For one step ahead prediction, the best model for Wick and Workington is NARX GRU with RMSE values of 0.0508 and 0.0888, respectively, whilst the optimal model for Ullapol and New Heaven is NARX LSTM.

2) The optimal model for two steps ahead prediction of the tide level in Wick, Workington and New Heaven is GRU, whilst NARX-GRU gives the best results for Ullapol.

According to the results shown in Figure 4, it can be noticed that the NARX LSTM model performs best for three steps ahead prediction in Wick and New Heaven, whereas NARX GRU performs best for the data at Workington and Ullapol. Similar to the case of using average RMSE, the biLSTM model demonstrates poor results in single models for each of the locations for almost all locations and time horizons. The model performs good only in Wick for one step ahead and two steps ahead forecasting.

Based on the RMSE values for all the three different horizons and all the four stations, it also can be seen that the RMSE values are not at the same level for the four locations. The location that has a relatively low and stable RMSE is Wick, while others locals have slightly higher RMSE values. This may suggest that one of the inputs or features can be detrimental to the performance of other features while using an MIMO model.

The proposed models show the positive benefits of using several similar data to forecast the tide levels at different locations or stations simultaneously. However, the combination of NARX networks and the MIMO models requires high computational costs. The MIMO models need to be well designed; in doing so, it does not only need to find the best number of delays (for MIMO NARX) but also the optimal number of hidden layers and hidden nodes.

IV. Conclusion

A total of six MIMO RNN models, namely, LSTM, bilSTM, GRU, NARX-LSTM, NARX-biLSTM, and NARX-GRU have been investigated for tide level prediction from one to three steps ahead. All models were initialized with the same neural network setting, e.g. the number of hidden layers and the number of nodes in each hidden layer. It turned out that GRU in both individual RNN models and combined NARX-RNN models outperform other models in terms of average RMSE. It was also observed that the prediction performances of the three individual RNN models, namely, LSTM, biLSTM and GRU, were improved when they were combined with the NARX model. In other words, the three hybrid models NARX-LATM, NARX-biLATM and NARX-GRU perform better than LSTM, biLSTM and GRU, respectively, for all the cases of the three time horizons (i.e., 1-, 2-, and 3-steps ahead prediction). The study also found that the biLSTM and NARX biLSTM may not be suitable for predicting tide level with MIMO models due to their poor performance for all the three different prediction cases (three time horizons).

Future work will broaden the scope to include a feasibility investigation of the proposed method for other non-linear types of models.

Acknowledgment

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), EPSRC (Ref. EP/I011056/1), EU Horizon 2020 (Ref. 637302). Nerfita Nikentari gratefully acknowledges a scholarship from the Indoensia Endowment Fund for Education/Lembaga Pengelola Dana Pendidikan (LPDP).

REFERENCES

[1] M. A. Rizkina, D. Advtia and N. Subasita, "Nonlinear Autoregressive Neural Network Models for Sea Level Prediction, Study Case: in Semarang, Indonesia," *2019 7th International Conference on Information and Communication Technology (ICoICT)*, 2019, pp. 1-5.

[2] A. Y. Winona and D. Advtia, "Short Term Forecasting of Sea Level by Using LSTM with Limited Historical Data," *2020 International Conference on Data Science and Its Applications (ICoDSA)*, 2020, pp. 1-5.

[3] 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," in *IEEE Journal of Oceanic Engineering*, vol. 46, no. 4, pp. 1261-1271, Oct. 2021.

[4] W. Bao and W. Bin, "Real-time Tide Prediction Based on An Hybrid HA-WANN Model Using Wind Information," *2018 14th IEEE International Conference on Signal Processing (ICSP)*, 2018, pp. 604-608.

[5] S. Consoli, D. R. Recupero, and V. Zavarella, "A Survey on Tidal Analysis and Forecasting Methods for Tsunami Detection," arXiv:1403.0135 [cs.CE], Mar. 2014.

[6] B. L. Meena and J. D. Agrawal, "Tidal Level Forecasting Using ANN," *Procedia Engineering*, vol. 116, pp. 607–614, 2015.

[7] W. Wu, L. Li, J. Yin, W. Lyu and W. Zhang, "A Modular Tide Level Prediction Method Based on a NARX Neural Network," in *IEEE Access*, vol. 9, pp. 147416-147429, 2021.

[8] F. Di Nunno, F. Granata, R. Gargano, and G. de Marinis, "Forecasting of Extreme Storm Tide Events Using NARX Neural Network-Based Models," *Atmosphere*, vol. 12, no. 4, p. 512, Apr. 2021

[9] C. -H. Yang, C. -H. Wu and C. -M. Hsieh, "Long Short-Term Memory Recurrent Neural Network for Tidal Level Forecasting," in *IEEE Access*, vol. 8, pp. 159389-159401, 2020.

[10] A. Y. Winona and D. Advtia, "Short Term Forecasting of Sea Level by Using LSTM with Limited Historical Data," *2020 International Conference on Data Science and Its Applications (ICoDSA)*, 2020, pp. 1–5.

[11] S. B. Taieb, G. Bontempi, A. Sorjamaa and A. Lendasse, "Long-term prediction of time series by combining direct and MIMO strategies," *2009 International Joint Conference on Neural Networks*, 2009, pp. 3054-3061.

[12] S. Ben Taieb, A. Sorjamaa, and G. Bontempi, "Multiple-output modeling for multi-step-ahead time series forecasting," *Neurocomputing*, vol. 73, no. 10–12, pp. 1950–1957, Jun. 2010.

[13] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[14] P. Liu, X. Qiu, and X. Huang, "Recurrent Neural Network for Text Classification with Multi-Task Learning," *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI-16),* pp. 2873–2879, 2016.

[15] X. Sun, W. Xu, H. Jiang, and Q. Wang, "A deep multitask learning approach for air quality prediction," *Annals of Operations Research*, vol 303, no. 1-2, pp. 51-79. Jul. 2020.

[16] H. Luo, J. Cai, K. Zhang, R. Xie, and L. Zheng, "A multi-task deep learning model for short-term taxi demand forecasting considering spatiotemporal dependences," *Journal of Traffic and Transportation Engineering (English Edition)*, vol. 8, no. 1, pp. 83–94, Feb. 2021.

[17] D. Spathis, S. Servia-Rodriguez, K. Farrahi, C. Mascolo, and J. Rentfrow, "Sequence Multi-task Learning to Forecast Mental Wellbeing from Sparse Self-reported Data," in KDD '19: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery*, pp. 2886–2894,2019.

[18] Z. Shen, Y. Zhang, J. Lu, J. Xu, and G. Xiao, "A novel time series forecasting model with deep learning," *Neurocomputing*, vol. 396, pp. 302–313, Jul. 2020.

[19] P. L. Lu, T. B. Bai, and P. Langlais, "SC-LSTM: Learning Task-Specific Representations in Multi-Task Learning for Sequence Labeling," *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, vol. 1, pp. 2396–2406, Jun. 2019.

[20] Md. M. Rahman, Y. Watanobe, and K. Nakamura, "A Bidirectional LSTM Language Model for Code Evaluation and Repair," *Symmetry*, vol. 13, no. 2, p. 247, Feb. 2021.

[21] P. Singla, M. Duhan, and S. Saroha, "An ensemble method to forecast 24-h ahead solar irradiance using wavelet decomposition and BiLSTM deep learning network," *Earth Science Informatics*, vol. 15, no. 1, pp. 291–306, Nov. 2021.

[22] F. Shahid, A. Zameer, and M. Muneeb, "Predictions for COVID-19 with deep learning models of LSTM, GRU and Bi-LSTM," *Chaos, Solitons & Fractals*, vol. 140, p. 110212, Nov. 2020.

[23] K. E. ArunKumar, D. V. Kalaga, Ch. Mohan Sai Kumar, M. Kawaji, and T. M. Brenza, "Comparative analysis of Gated Recurrent Units (GRU), Long Short-Term Memory (LSTM) cells, Autoregressive Integrated Moving Average (ARIMA), Seasonal Autoregressive Integrated Moving Average (SARIMA) for forecasting COVID-19 trends," *Alexandria Engineering Journal*, vol. 61, no. 10, Jan. 2022.

[24] B. C. Mateus, M. Mendes, J. T. Farinha, R. Assis, and A. M. Cardoso, "Comparing LSTM and GRU Models to Predict the Condition of a Pulp Paper Press," *Energies*, vol. 14, no. 21, p. 6958, Jan. 2021.

[25] K. Zarzycki and M. Ławryńczuk, "LSTM and GRU Neural Networks as Models of Dynamical Processes Used in Predictive Control: A Comparison of Models Developed for Two Chemical Reactors," *Sensors*, vol. 21, no. 16, p. 5625, Aug. 2021.

[26] S. A. Billings, Nonlinear System Identification: NARMAX Methods in the Time, Frequency, and Spatio-Temporal Domains. Chichester, West Sussex: Wiley, 2013.

[27] J. R. Ayala 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.

[28] H.-L. Wei and S. A. Billings, "An efficient nonlinear cardinal B-spline model for high tide forecasts at the Venice Lagoon," *Nonlinear Processes in Geophysics*, vol. 13, no. 5, pp. 577–584, Oct. 2006.

[29] H.-L. Wei and S. A. Billings, "Long term prediction of non-linear time series using multiresolution wavelet models," *International Journal of Control*, vol. 79, no. 6, pp. 569–580, Jun. 2006.

[30] H. -L. Wei, D. Q. Zhu, S. A. Billings, and M. A. Balikhin, "Forecasting the geomagnetic activity of the Dst index using multiscale radial basis function networks," *Advances in Space Research*, vol. 40, no. 12, pp. 1863–1870, Jan. 2007.

[31] H.-L. Wei, *"Sparse, interpretable and transparent predictive model identification for healthcare data analysis*," in Advances in Computational Intelligence. IWANN 2019, pp. 103–114.

[32] K. Cho, B. van Merrienboer, D. Bahdanau and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches", *Proc. 8th Workshop Syntax Semantics Struct. Statist. Transl.*, pp. 103-111, 2014.

C

Paper 3

Article:
Nikentari, N. orcid.org/0009-0008-6635-2945 and Wei, H.-L. orcid.org/0000-0002-4704-7346 (2024) Multi-task learning using non-linear autoregressive models and recurrent neural networks for tide level forecasting. International Journal of Electrical and Computer Engineering (IJECE), 14 (1). pp. 960-970. ISSN 2088-8708

https://doi.org/10.11591/ijece.v14i1.pp960-970

International Journal of Electrical and Computer Engineering (IJECE)
Vol. 14, No. 1, February 2024, pp. 960~970
ISSN: 2088-8708, DOI: 10.11591/ijece.v14i1.pp960-970

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):

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

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):

$$y_i(t) = f_i\left(y_1(t-1), \dots, y_1\left(t-n_y\right), \dots, y_m(t-1), y_m\left(t-n_y\right), u_1(t-1), \dots, u_1(t-n_u), \dots, u_r(t-1), \dots, u_r(t-n_u)\right) \quad (2)$$

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):

$$y(t) = f\left(y(t-1), \dots, y\left(t-n_y\right), u(t-d), \dots, u(t-n_u), e(t-1), e(t-2) \dots, e(t-n_e)\right) + e(t) \quad (3)$$

and

$$y_i(t) = f_i\left(y_1(t-1), \dots, y_1\left(t-n_y\right), \dots, y_m(t-1), \dots, y_m\left(t-n_y\right), u_1(t-1), \dots, u_1(t-n_u), \dots, u_r(t-1), \dots, u_r(t-n_u), e_1(t-1), \dots, e_1(t-n_e), \dots, e_m(t-1), \dots, e_m(t-n_e)\right) \quad (4)$$

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.

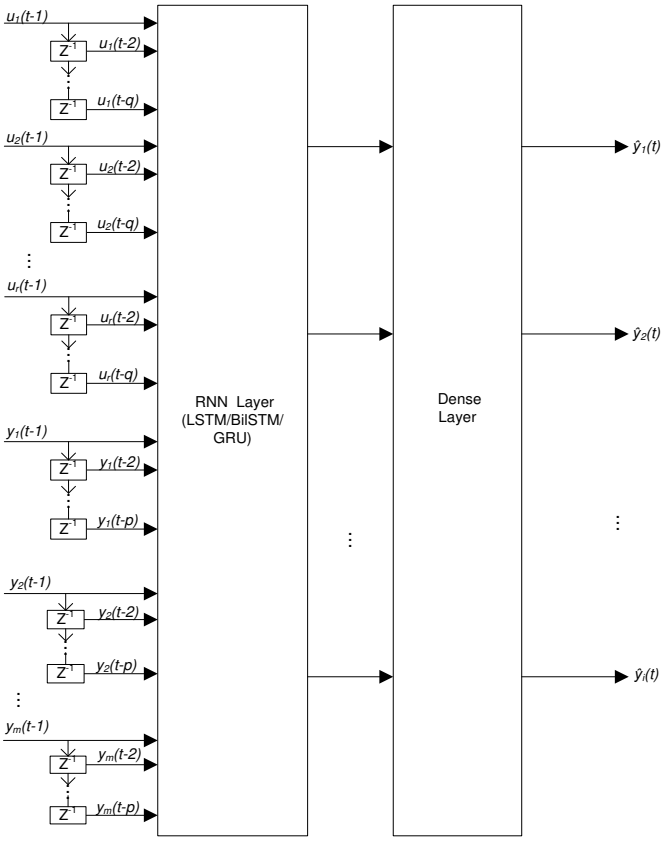


Figure 1. NARX RNN architecture

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:

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

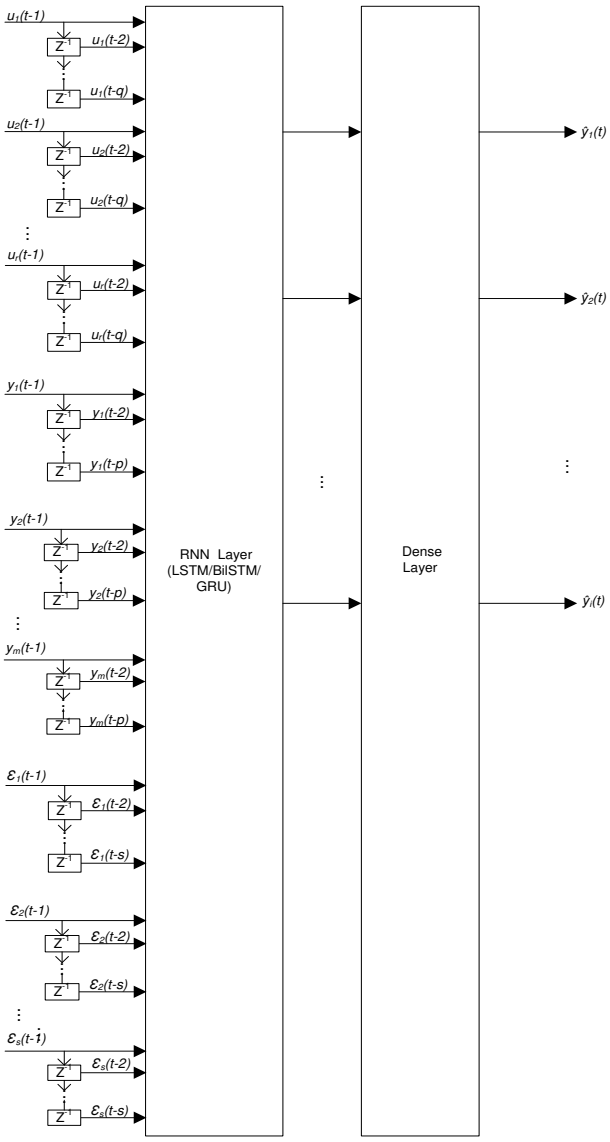


Figure 2. NARMAX RNN architecture

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):

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

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):

$$L = \frac{1}{m}\sum_{i=1}^{m} L_i \quad (8)$$

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):

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

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

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

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

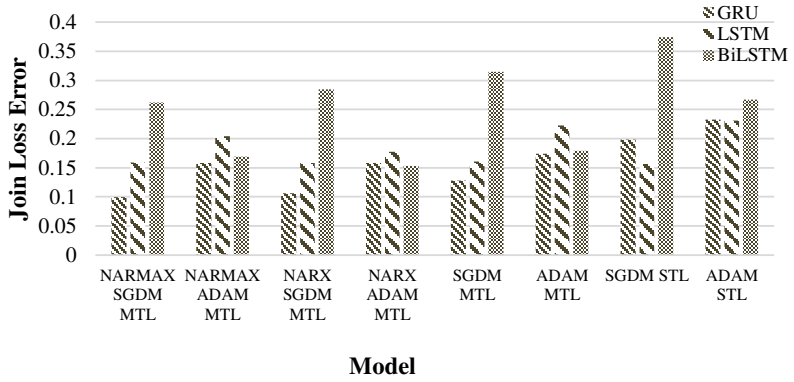


Figure 3. Comparison of joint loss error of different model structures builts based on three RNNs, namely, GRU, LSTM, and BiLSTM

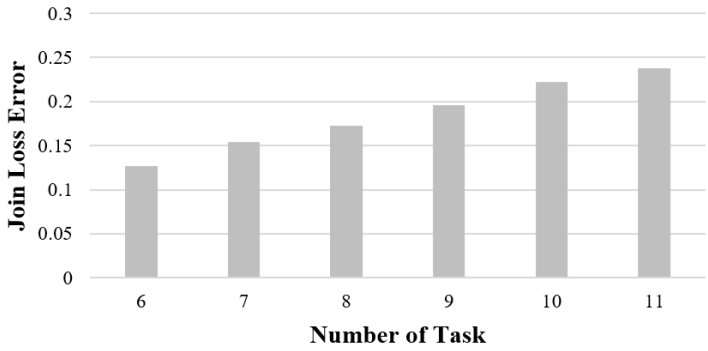


Figure 4. Joint loss error of MTL NARMAX with many tasks

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.