Acta Scientific COMPUTER SCIENCES

Volume 7 Issue 5 August 2025

Research Article

Assessment of the Optimization of Hyperparameters in Deep LSTM for Time Series Sea Water Tidal Shift

Nosius Luaran[1], Rayner Alfred[1]*, Xu Fengchang[2] and Haviluddin[3]

[1]*Creative Advanced Machine Intelligence Research Centre, Faculty of Computing and Informatics, Universiti Malaysia Sabah, Jalan UMS, 88400 Kota Kinabalu, Sabah, Malaysia*

[2]*Department of Information Engineering, Shandong Light Industry Vocational College, Zibo City, Shandong Province, China*

[3]*Department of Informatics, Faculty of Engineering, Universitas Mulawarman, East Kalimantan, Indonesia*

***Corresponding Author:** Rayner Alfred, Creative Advanced Machine Intelligence Research Centre, Faculty of Computing and Informatics, Universiti Malaysia Sabah, Jalan UMS, 88400 Kota Kinabalu, Sabah, Malaysia.

Received: June 13, 2025

Published: July 10, 2025

Abstract

Generating accurate and reliable tidal forecasts is very crucial to support routine coastal decision-making. The forecasting of univariate time series behavior becomes more challenging and requires state-of-the-art techniques to achieve realistic forecasting outcome. However, the larger the initial hyperparameters space that must be searched, the more iterations the automated hyperparameters tuning technique should have. As a result of the complexity to tune the deep learning architecture in producing an optimal and consistent result, tuning several hyperparameters to improve deep learning results in learning time series data is highly needed. Therefore, it is necessary to investigate the best approach to fine tune machine learning algorithms that have multiple hyperparameters (e.g., deep Long Short-Term Memory (LSTM)) to learn and gather collective knowledge about time series sea water tidal shift data that could be used to make better forecasts for the individual time series. Thus, the aim of this paper is to investigate the effects of varying the values of the hyperparameters of the LSTM architecture on the forecasting accuracy of the sea water tidal height variation under nonextreme conditions. In this paper, we empirically evaluate the proposed LSTM forecasting framework to model the tidal dataset based on the RMSE measurements as the hyperparameters change. These hyperparameters are divided into static hyperparameters (e.g., mini batch size, number of epochs, number of iteration and percentage of neuron dropout) and dynamic hyperparameters (e.g., number of layers, number of hidden units, learning rate and L2 regularization). The initial static hyperparameters are used to estimate the optimal values of the dynamic parameters using the Bayesian Optimization method. Based on the results, we highlighted some findings that will aid in improving the performance of the LSTM model for tidal forecasting for hourly short term sea water level and these allow users to choose the best static and dynamic hyper-parameters combination that can produce the optimum outcomes. Finally, the paper is concluded by suggesting some of the extended works that can be performed to improve the results of forecasting the sea water tidal shift.

Keywords: Forecasting Sea Water Tidal Shift; Long Short-Term Memory; Time-Series Data; Hyperparameters Setting; Deep Learning; Bayesian Optimization

Introduction

In the recent years, there have been several efforts made in designing and investigating the performance of several models in learning and forecasting sequential or time series dataset. The evolution of Industry 4.0 and the Internet of Thing [1,2] have produced abundant large data that makes the application of multivariate time series more widespread in many fields [3]. Therefore, this have established a significant computational challenge in shallow machine learning method to identify time series behaviour predictions which become more complex [2]. For instance, a realistic forecasting method for tidal height variation under non-extreme

conditions (e.g., storm surges are not captured) is very crucial to support routine coastal decision-making and provide a benchmark for potential Mean Sea Level forecasts [4].

One of the deep Learning algorithms, Long short-term memory (LSTM), is popularly used in research areas with complex solutions of time-series data to resolve long-term dependency problems in its cell structure (logic cell or sigma) [5]. LSTM was designed to deal with the vanishing gradient problems that classic Recurrent Neural Networks (RNNs) have while learning long-term dependencies using sequential data [6]. Deep learning modules like LSTM, on the other hand, use a few hyper-parameters variables to solve problems related to learning time series data.

The LSTM architecture consists of many hyper-parameters that may affect the generalization efficiency, posing a difficult tuning problem for human users. The hyperparameters are critical for machine learning algorithms because they directly influence the behavior of the training algorithm and thus affect the accuracy performance of machine learning models [7]. The level of accuracy of the data prediction results is related to the parameters used during the model training session and uses several standard parameter variables to achieve certain level of accuracy for various types of problems. Therefore, applying different values to each parameter would achieve different degree of optimized accuracy performance.

Consequently, to achieve the highest degree of optimized accuracy performance of predictions in historical time-series performance, a tuning experiment is needed to be carried out to understand the effects of each parameter individually. It is necessary to understand the characteristics of each parameter to get an optimum model for learning time series data for a particular problem (e.g., sea water tidal shift). This can be considered as a fine-tuning experiment to find the best combination of parameter values that achieves the highest degree of accuracy. The tuning of these hyperparameter values is not straightforward task, particularly in time series applications. There are, on the other hand, several common computational methods and search techniques for optimizing hyper-parameters. For instance, Recurrent Neural Networks (RNNs) have become competitive forecasting methods [8].

This is due to the fact that LSTM neural network algorithms are well known to be dominated by several layers, rendering deep learning neural network computation more complex and difficult. Hyper-parameters are important for deep machine learning algorithms because they directly influence the training behaviours and have a significant effect on the performance of the deep learning model. The hyperparameters are those variables for which we provide the value "manually" to the model to aid in learning and estimating model parameters, and which are then used to predict unobserved data [7]. In other words, in machine learning, a hyperparameter is a parameter whose value is used to control the learning process. Tuning these hyper-parameters is difficult when it comes to optimizing the performance of the deep learning for time series applications [8].

Generally, the larger the initial hyper-parameter space that must be searched, the more iterations the automated hyper-parameter tuning technique should have. As a result of this difficulty of fine-tuning deep learning architecture in producing an optimal and consistent result, tuning several hyper-parameters to improve deep learning results in time series applications is needed [8]. Therefore, it will be useful to train an algorithm on multiple similar datasets and gather some collective knowledge that could be used to make better forecasts for the individual time series [9,10].

Thus, the aim of this paper is to investigate the effects of varying the values of the hyperparameters of the LSTM architecture on the forecasting accuracy of the sea water tidal height variation under non-extreme conditions (e.g., storm surges are not captured). These hyper-parameters include the learning algorithms, optimizer, number of training epochs, initial learning rate, number of hidden units, L2 Regularization, number of Layers, batch size and dropout. The effects of varying the combination of the hyper-parameter values will be evaluated based on the time series forecasting accuracy using the Root Mean Square Error (RMSE) measurements.

The rest of the paper is organized as follows. Section 4 describes related works on the application of machine learning methods in learning time-series datasets (e.g., sea water tidal height variation) and the experimental setup and configuration will be described in Section 5. Section 6 provides the detailed result analysis and discussion. Finally, Section 7 draws the conclusions.

Related work

LSTM models are found to be used effectively in learning time-series data in many applications [11]. These tasks include forecasting of flood [12], prediction of sea surface temperature (SST) [13], forecasting El Nino-Southern Oscillation (ENSO) climate phenomenon [14], forecasting model for wave height [15], prediction of electricity usages [16] and prices [17], forecasting tourist arriv-

als for efficient allocation of tourism resources [18], forecasting of future agriculture load [19], time series forecasting in Cryptocurrency (Bitcoin) daily price prediction [20], flow forecasting of high-speed railways [21], forecasting spot instance price [22], estimating and predicting COVID-19 outbreak in Asia Pacific [23], and finally in forecasting the numbers of reservations [24].

Individual statistical models such as ARIMA cannot compete with deep learning model (e.g., LSTM) in learning time series data. For instance, LSTM deep-learning model was found to be more suitable and outperformed the ARIMA model when predicting Solar Cycle 25 based on the value of SunSpot Number (SSN) with RMSE of 35.9 [25], forecasting network traffic and displaying updates in real-time to convey the information [26], forecasting covid19 cases in Turkey [27], and also predicting the number of patients visit to hospital [28].

Among the machine learning models, LSTM models have been found to outperform other six machine learning (ML) models which include Decision Tree, Multilayer Perceptron, Lasso, Linear Regression, Random Forest, and Ridge, in forecasting the numbers of reservations [24]. The empirical result shows that LSTM models improved 3.0% over ML models. In another work, LSTM models have also been found Support Vector Machine (SVM), ANNs and Hidden Markov Machine models in predicting the high-frequency stock trend of a short-term [29].

The hyper-parameters settings used for training the LSTM algorithms vary in a variety of problem domains and these hyper-parameters settings are outlined in Table 1. Table 2 shown below summarizes the values of hyper-parameters used in the deployment of LSTM in learning time-series data in real-world applications. These problem domains include forecasting wave height [15], forecasting future agriculture load [19], forecasting service blocking probability [21], forecasting cryptocurrency daily price [20], flood forecasting [12], stock price forecasting [16], prediction of electricity price [17], forecasting Covid-19 cases [27], predicting number of patients [28], and using a common ADAM (Adaptive Moment) to handle real-time, large datasets, and high dimensional parameters as momentum optimizer compared to ADAGRAD for forecasting sea surface temperature [13] and RMSPROP for short-term stock prediction [29].

Short	Notations Hyperparameters
LR	Learning Rate
HU	Hidden Units
NI	Number of Iterations
NL	Number of Layers
EP	Number of Epochs
MBS	Mini Batch Size
TTR	Training Testing Ratio
DO	Percentage of Dropouts

Table 1: The hyperparameters settings used for training the LSTM algorithms.

Ref	LR	HU	NI	NL	EP	MBS	TTR	DO
[11]	-	180	-	-	500	-	80:20	0.3
[12]	0.0001	20;30 ;50	-	-	100,000	-	-	-
[13]	0.1	-	-	-	-	100	-	-
[14]	0.0005	50	-	2	200	2;4;8	85:15	-
[15]	0.0001	-	125	-	250	-	-	0.2
[16]	0.001	50	-	2;4	15;25;50; 100	32	-	-
[17]	-	64;128	150	2	150	64	-	0.2
[18]	-	-	300	6	300	60	-	-
[19]	0.01	50	-	-	-	-	-	-
[20]	-	1000	-	-	300	72	-	-
[21]	-	8	5	-	200	-	-	0.6
[22]	-	4	-	1	1000	-	-	-
[23]	-	-	10	-	100	1	-	-
[24]	0.1	-	-	1	300	15	-	0.6
[25]	-	-	60;120	-	100	10	-	-
[27]	-	32	-	-	2000	1	-	-
[28]	0.001	-	-	-	100	-	-	-
[29]	0.05	200	-	1	-	64	-	-
[30]	0.01	200	-	-	800	-	-	-

Table 2: Several works related to the deployment of LSTM in learning time-series data.

Based on Table 2, the mini batch size ranges from 1 to 100, and the epoch value ranges from 15 to 100,000. An epoch's value of 300, on the other hand, is frequently used in forecasting tourist arrivals for efficiency [18], cryptocurrency daily price [20], and hotel reservation numbers [24]. The number of iterations for the basic and hybrid LSTM models ranges from 4 to 300. The number of hidden units used in most forecasting tasks ranged from more than 50 to 1000 neurons to extract data features [20]. Most of the works conducted previously used 200 hidden units or below [11,16,17,19,30].

Using single layer [22,29] or 2 layers [17,16] of hidden units is adequate to get good results using LSTM algorithms. It can be increased to 50 to maintain data consistency. Dropout and L2 Regularization parameters ranging from 0.2 to 0.6 and 0.0001.

respectively are used to avoid overfitting. The model's convergence speed is measured by the Learning Rate factor, which ranges from 0.1 to 0.0001. Another factor to consider is the ratio of training to test, which is not fixed in any problem domains [6] and thus affects the RMSE assessment [8,9]. Forecasting the El Nio-Southern Oscillation (ENSO) [14] and prediction based on the arbitrary nature of human behaviour [11] were accomplished using a split training and test ratio of 85:15 and 80:20, respectively.

The findings of this study suggest that the LSTM model produce results remarkably [15,27], highly accurate and stable [12,17], achieved best prediction [11,13,16,30], improved prediction outcome compared to other methods such as Neural Network Autoregression (NNA), ARIMA, Naïv e, Seasonal Naïv e, Mean and Drift, and etc [20,24,28,29]. Apart from that, LSTM model capable of long-term temporal dependencies, model non-linear characteristics [14], show suitability for time-series dataset [25], perform good generalization ability [21]. Furthermore, LSTM improves the problem of input variables selection [20], LSTM RNN efficiently and effectively captures seasonal patterns and long-term trends, and the validation works efficiently, and the results obtained much faster when applying k-Fold Cross [22].

Hybrid models also have been used effectively as forecasting techniques in interpreting real problems which are often complex in nature and outperforming a single machine learning that is unable to capture a complex problem and improve the prediction accuracy. For instance, a hybrid deep learning using Random Forest (RF) and LSTM also was found to produce better results when forecasting tourist arrivals for efficient allocation of tourism resources [18]. In this work, RF was used reduce dimensionality of search query and LSTM was used to model the non-linear relationship. Next, the proposed hybrid ARIMA-LSTM model was also found to be very effective in achieving predictive precision and have wider application range compared to single ARIMA and LSTM [30].

Furthermore, Deep learning LSTM model also outperformed the GRU model, but GRU model converges faster than LSTM [19]. Recurrent Neural Network (RNN) helps to transform and updates both important and non-important dataset [32]. Nevertheless, above all, analysis reveal that the input data type has more influence than the quantity of the input data [12], hidden neurons and length of data impacted on prediction accuracy [15], increase number of epochs decrease error prediction and improve prediction accuracy [17].

Conversely, LSTM experienced several limitations that directly affect the performance of the LSTM model. The data resolution and characteristics [12,13], the independent and dependent variables selection [15,17], rapid changes in high-risk stock dataset [16] greatly influence the LSTM model performance. Deep learning performed poorly in fitting data with trend for non-stationary dataset [19]. The accuracy also compromises with the length sequence of the period observation to catch value information [28], external factors variables [30] and insufficient data for experiment [11].

Technically, Deep learning neural network algorithms are dominated by several layers, rendering deep learning neural network computation to enhance better forecasting accuracy for individual time series [8]. According to the review's findings, the ranges of values used for each training parameters set in any deep machine learning approaches vary depending on the field of the problems. It is worth to highlight that previous research analysis had revealed that the accuracy of the prediction is highly dependent on the training parameter settings. These hyper-parameters are critical for machine learning algorithms because they directly influence the behaviour of the training algorithms and thus affect the accuracy performances of the machine learning models [7]. It was suggested that further experiments with sufficient data and a wider scale are required to provide more clarity on hyper-parameter characteristic [31].

Materials and Methods

The goal of the experiment that will be conducted is to investigate the LSTM model performance using different hyper-parameter settings for the training phase. This section describes the

datasets and outlines the setup of the experiments and the hyper-parameter settings used in this work. At the end of this section, the metric benchmarking used to assess the performance of the LSTM will be discussed. Figure 1 describes the proposed research methodology framework in investigating the performance of the LSTM performance. The processes involved include data preparation or preprocessing, hyperparameters setting, training the LSTM model using different hyper-parameter settings and finally assessing the LSTM model according to different specifications of the hyperparameter values.

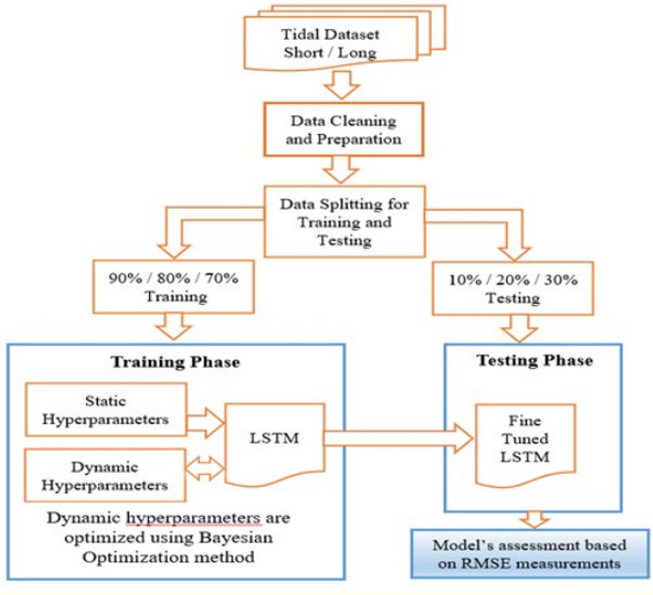

Figure 1: Proposed research methodology to investigate the performance of the LSTM model using different hyperparameter settings.

Dataset preparations

The dataset contains the sea level measurements taken hourly for 4 weeks (Jan 1997) of Kota Kinabalu's Tide gauge station (5.98300 Lat 116.06700 Long), which located in the west of the South China Sea and experiencing Semi Diurnal Tidal characteristic. The dataset is obtained from the University of Hawaii - Sea Level Center (UHSLC) (https://uhslc.soest.hawaii.edu/datainfo/) and accessed in October 2020. The LSTM model will be used in this experiment as it has outperformed other algorithms as reviewed in the previous section.

Table 3: Static and dynamic hyper-parameters considered for the LSTM model.

Static	Dynamic
Number of Epochs	No of Layers
Mini Batch Size	No of Hidden Units
No of Iterations	Initial Learning Rate
Dropout	L2 Regularization

Table 4: Ranges of values for the static hyper-parameters used based on literature.

Epoch	Mini Batch Size	Number of Iteration	Percentage of Dropout
1000 - 10000	10 - 200	10 - 400	0.1 - 0.9

Hyperparameter settings

In this experiment, the hyper-parameters are divided into static and dynamic hyperparameters, as shown in Table 3. The static hyperparameters include the number of epochs, the size of mini batch, the number of iteration and the percentages of dropping. On the other hand, the dynamic hyper-parameters include the number of layers, number of hidden units, initial learning rate and the value of L2 regularization. The ranges of values for the static hyper-parameters are determined and configured based on the findings found in literature review of related works, as it is expensive to compute or calculate for all ranges of hyper-parameter values. This is done in this manner to expediate the process of getting the best hyper-parameter settings for the LSTM model by using the known values of these static hyper-parameters obtained from literature. The Bayesian Optimization method is then used to find the optimal values of dynamic hyperparameters based on the pre-configured values of static hyper-parameters. Bayesian Optimization is often used in applied machine learning to tune the hyper-parameters of a given well-performing model on a validation dataset.

Thus, given series of values of the static hyper-parameters of the LSTM, the optimized values for the dynamic hyper-parameters will be generated and that will produce an optimized LSTM model with optimized configuration settings of the dynamic hyperparameters based on the static hyper-parameters. The characteristics of

the static hyperparameters can also be investigated with respect to the performance of the LSTM model in forecasting the sea water level. The ranges of static hyper-parameters used to train the LSTM model are shown in Table 4. Table 5, tabulates the ranges of dynamic hyperparameters' values that will be used and optimized based on the facts found in the literature to produce the optimized LSTM model.

Table 5: Ranges of values for the dynamic hyperparameters used based on literature.

No of Layers	No of Hidden Units	Initial Learning Rate	L2 Regularization
1 - 4	1 - 200	0.0001 - 1	0.000001 - 0.01

Table 6: Discretized initial values for all the static hyper-parameters.

Static Hyper-parameters	5 Discretized Bins
Mini Batch Size	10, 50, 100, 150, 200
Dropout	0.1, 0.3, 0.5, 0.7, 0.9
Number of Epochs	1,000, 3,000, 5,000, 7,000, 10,000
No of Iterations	10, 100, 200, 300, 400

Table 7: Discretized initial values for all the dynamic hyper-parameters.

Dynamic Hyperparameters	4 to 5 Discretized Bins
No of Layers	1, 2, 3, 4
No of Hidden Units	1 - 40, 41 - 80, 81 - 120, 121 - 160, 161 - 200
Learning Rate	0.0001 - 0.001, 0.001 - 0.01, 0.01 - 0.1, 0.1 - 1
L2 Legularization	0.000001 - 0.00001, 0.00001 - 0.0001, 0.0001 - 0.001, 0.001 - 0.01

The values of the static hyperparameters settings will be divided or discretized into several bins (e.g., 4 or 5) to reduce the search space and at the same time to investigate the effects of varying the values of the static hyperparameters, during the training phase, on the performance of the LSTM model in learning sea water level time series. The values of the dynamic hyperparameters settings will also be divided or discretized into several bins to investigate the characteristics of the static and dynamic hyper-parameters in producing an optimized LSTM model. Tables 6 and 7 tabulate the discretized initial values for all the static and dynamic hyperparameters.

Experimental setup and training of LSTM Model

The dataset is divided into 80% and 20% for the training and testing dataset. The experiment will be conducted for 20 times, for each setting of static and dynamic hyperparameters' values used to train and test the LSTM model. The performance of the LSTM model is then assessed using the averaged Root Mean Square Error (RMSE) and Coefficient of Determination (R^2) between the actual and predicted sea water level obtained from the 20 runs.

There are four experimental setups that will be used in this work. The setups are explained as follows.

- In the first part of the experiments, we varied the size of the mini batch (e.g., 10, 50, 100, 150, 200) and we fixed the other static hyper-parameters settings that include the percentage of Dropout (0.1%), the number of Iteration (10) and the number of epochs (1000). For instance, the first setting of hyperparameters consists of 10 size of batch, 0.1 as the value of dropout, 1000 epochs and 10 iterations as shown in Table 4. Then, the value of the size of the batch will be updated to 20 and there are no changes to the values of the remaining static hyper-parameters.
- In the second part of the experiments, we varied the percentages of dropout values (e.g., 0.1, 0.3, 0.5, 0.7, 0.9) and we fixed the other static hyperparameters settings that include the size of mini batch (10), the number of iterations (10), the number of epochs (1000).
- In the third part of the experiments, we varied the number of epochs (e.g., 1,000, 3,000, 5,000, 7,000, 10,000) and we fixed the other static hyperparameters settings that include the size of mini batch (10), the number of iterations (10), the percentage of dropout (0.3).
- In the final part of the experiments, we varied the number of iterations (e.g., 10, 100, 200, 300, 400) and we fixed the other static hyper-parameters settings that include the size of mini batch (10), the number of epochs (1000), the percentage of dropout (0.3).

Evaluation

In this study, the performance efficiency of the forecasting LSTM model in the field related to tidal variation is evaluated using the most common metrics to measure the accuracy of the model using the root-mean-square deviation (RMSE) statistical method and Coefficient of Determination (R^2) [32] to compare the predicted

values with observed values using the following formulas:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y^i - x^i)^2}$$

$$R^2 = 1 - \frac{\sum(y^i - x^i)^2}{\sum(y^i - z^i)^2}$$

Where i is a variable, n is the number of non-missing data points, y^i is the actual observations time series and x^i is the estimated time series, $\sum(y^i - x^i)^2$, is the sum of squares of residuals and $\sum(y^i - z^i)^2$ is the total sum of squares of errors from the average model.

All the results of the assessment can be referred from Table 8 to Table 11. Some of the mean RMSE measurements are not available, indicated by symbol '–', due to the fact that the Bayesian Optimization method cannot find the optimal values of RMSE of those ranges of values for static and dynamic hyper-parameters based on the pre-configured values, as shown in Tables 6 and 7. For instance, in Table 8, there is no RMSE value that can be obtained using range of 41 - 80 number of hidden units when the mini batch size is set to 10.

Results and Discussion

In this study, a total of 4 weeks (hourly = 672) of sea level water datasets are used to investigate the behaviour of the training parameter using LSTM Model. The results of the LSTM model training are discussed in this section and presented in Table 8 to Table 11.

Results of first experimental setup

As mentioned earlier, in the first part of the experiments, we varied the size of the mini batch (e.g., 10, 50, 100, 150, 200) and we fixed the other static hyperparameters settings that include the percentage of Dropout (0.1%), the number of Iteration (10) and the number of epochs (1000).

The results shown in Table 8 show that increasing the value of MBS parameter would result higher RMSE (loss in accuracy) and lower correlation between the actual against the predicted value. The highest performance of LSTM models was obtained when the Mini Batch Size's value is 10 having test RMSE value of 0.073 (R^2: 0.9527). The test RMSE is the average overall RMSE measurement that is obtained from all the 20 testing phases conducted for the LSTM model. The results of No of Layers indicated that multiple LSTM layers will give higher RMSE compared to single layer. This means that a single layer can produce better result compared to multiple layers. A higher number of Hidden units also provides lower RMSE values. For instance, the number of hidden units having the range of 121 - 200 provides lower RMSE. The initial learning rate ranging from 0.0001 -0.001 produced lower RMSE (0.071) for mini batch sizes of 10 and 150. In addition to that, the range of L2 regularization values between 0.00001 - 0.0001 produced better RMSE outcome (e.g., 0.070).

Table 8: LSTM performance with different sizes of mini batch (e.g., 10, 50, 100, 150, 200).

Mini Batch Size	10	50	100	150	200
Test RMSE	0.073	0.077	0.083	0.092	0.106
R^2	0.9527	0.9442	0.929	0.8912	0.8655
No Of Layers		Mean RMSE			
1	0.070	0.071	0.08	0.089	0.104
2	0.078	0.077	0.089	0.099	0.106
3	0.075	0.082	0.084	0.09	0.11
4	0.077	0.087	0.08	0.089	0.097
No of Hidden Units		Mean RMSE			
1 - 40	0.086	0.088	-	0.107	0.116
41 - 80	-	-	-	-	-
81 - 120	0.073	0.071	0.088	0.086	0.103
121 - 160	0.074	0.082	0.078	0.082	0.126
161 - 200	0.069	0.076	0.083	0.096	0.098
Learning Rate		Mean RMSE			
0.0001 - 0.001	0.071	0.077	0.082	0.085	0.111
0.001 - 0.01	0.074	0.074	0.078	0.096	0.104

0.01 - 0.1	-	0.08	0.086	0.099	0.105
0.1 - 1	-	0.08	0.079	0.103	0.107
L2 Regularization		Mean RMSE			
0.000001 - 0.00001	0.076	0.082	0.085	0.084	0.106
0.00001 - 0.0001	0.070	0.077	0.083	0.096	0.108
0.0001 - 0.001	0.073	0.075	0.082	0.095	0.098
0.001 - 0.01	-	-	0.080	0.111	0.130

Results of second experimental setup

In the second part of the experiments, we varied the percentages of dropout values, and we fixed the other static hyper-parameters settings that include the size of mini batch, the number of iterations and the number of epochs as shown in Table 9.

The performance of the LSTM model remains constant when the percentages of dropout values are varied from 0.1 to 0.9, with test RMSE values ranging from 0.072 to 0.076, as shown in Table 9. The best value of the test RMSE with 0.072 was obtained when the percentage of dropout is fixed to 0.3. The best result can be obtained when a single layer of LSTM model is used. At the same time, using hidden units ranging from 161 to 200 for all dropout values also can produce lower RMSE values for the LSTM model investigated in this work.

For the percentages of dropout values of 0.1, 0.3, and 0.5, the lowest RMSE can be obtained by using initial learning rate ranging from 0.001 to 0.0001. For the initial learning rate ranges of 0.001 - 0.01 and 0.01 - 0.1, the RMSE obtained are 0.075 and 0.065, respectively. When using the range of values between 0.0001 - 0.001 for the L2.

Regularization, the LSTM model is found to produce better RMSE values when the percentages of dropout are 0.3, 0.5 and 0.9 having RMSE values of 0.068, 0.070 and 0.066 respectively.

Table 9: LSTM performance with different percentages of dropout values (e.g., 0.1, 0.3, 0.5, 0.7, 0.9).

% of Dropout	**0.1**	**0.3**	**0.5**	**0.7**	**0.9**
Test RMSE	0.073	0.072	0.074	0.076	0.073
R^2	0.953	0.954	0.950	0.987	0.950
No Of Layers			Mean RMSE		
1	0.070	0.069	0.072	0.071	0.069
2	0.078	0.082	0.082	0.079	0.081
3	0.075	0.074	0.078	0.091	0.082
4	0.077	-	0.081	0.082	-
No of Hidden Units			Mean RMSE		
1 - 40	0.086	0.088	0.088	0.082	0.082
41 - 80	-	0.075	0.074	0.091	0.057
81 - 120	0.073	-	0.076	0.079	0.071
121 - 160	0.074	0.076	0.082	0.079	0.075
161 - 200	0.069	0.068	0.073	0.07	0.071
Learning Rate			Mean RMSE		
0.0001 - 0.001	0.071	0.071	0.073	0.077	0.073
0.001 - 0.01	0.074	0.076	0.080	0.075	0.076
0.01 - 0.1	-	0.076	-	0.066	0.065
0.1 - 1	-	-	-	-	-
L2 Regularization			Mean RMSE		
0.000001 - 0.00001	0.076	0.075	0.078	0.080	0.077
0.00001 - 0.0001	0.070	0.076	0.079	0.072	0.077
0.0001 - 0.001	0.073	0.068	0.07	0.074	0.066
0.001 - 0.01	-	-	-	-	-

Results of third experimental setup

In the third outlined of the experimental setup, we varied the number of epochs and we fixed the other static hyper-parameters settings that include the size of mini batch, the number of iterations and the percentage of dropout.

The changes in the number of epochs in Table 10 showed that increasing the number of epochs degraded the LSTM model's performance. An Epoch's value of 1000 produced the lowest test RMSE of 0.072 and the highest R^2 of 0.950. It was found that the best results for the LSTM model can be achieved by using only one layer. With a mean RMSE of 0.068, the range of concealed units ranging from 161 to 200 can be used to produce the best RMSE result. The value of initial learning rate ranging from 0.0001 to 0.001 produced better RMSE values. In addition to that, the ranging values of L2 Regularization between 0.00001 and 0.0001 were found to produce good RMSE values, particularly when the epoch's value is greater than 3000.

Table 10: LSTM performance with different number of epochs (e.g., 1k, 3k, 5k, 7k, 10k).

Number of Epochs	1,000	3,000	5,000	7,000	10,000
Test RMSE	0.072	0.077	0.081	0.081	0.089
R^2	0.95	0.945	0.937	0.937	0.915
No Of Layers			Mean RMSE		
1	0.069	0.07	0.068	0.069	0.079
2	0.082	0.078	0.089	0.075	0.097
3	0.074	0.088	0.079	0.093	0.089
4	-	0.083	0.077	0.089	0.085
No of Hidden Units			Mean RMSE		
Jan-40	0.088	0.091	0.087	0.088	0.106
41 - 80	0.075	0.084	0.087	0.088	0.093
81 - 120	-	0.081	0.079	0.086	0.08
121 - 160	0.076	0.072	0.079	0.072	0.085
161 - 200	0.068	0.07	0.077	0.08	0.081
Learning Rate			Mean RMSE		
0.0001 - 0.001	0.071	0.076	0.079	0.073	0.088
0.001 - 0.01	0.076	0.08	0.074	0.086	0.087
0.01 - 0.1	0.076	0.076	0.09	0.107	0.096
0.1 - 1	-	-	-	-	-
L2 Regularization			Mean RMSE		
0.000001 - 0.00001	0.075	0.085	0.082	0.083	0.085
0.00001 - 0.0001	0.076	0.07	0.078	0.067	0.083
0.0001 - 0.001	0.068	0.076	0.085	0.081	0.102
0.001 - 0.01	-	0.079	-	0.075	-

Results of fourth experimental setup

In the final part of the experiments, we varied the number of iterations (e.g., 10, 100, 200, 300, 400) and we fixed the other static hyper-parameters settings that include the size of mini batch (10), the number of epochs (1000), the percentage of dropout (0.3).

Table 11 indicates that as the number of iterations grows, the best result of test RMSE remains constant at 0.047 with an R^2 of 0.996. The model's performance is optimal when only one layer is used having lower RMSE values for all predefined values of the number of iterations. In addition to that lower RMSE values can be obtained when using between 161 and 200 number of hidden units, between 0.0001 - 0.001 of initial learning rate, and between 0.00001 - 0.0001 L2 regularization, respectively.

Table 11: LSTM performance with different number of iterations (e.g., 10, 100, 200, 300, 400).

Number of Iterations	10	100	200	300	400
Test RMSE	0.072	0.050	0.047	0.047	0.047
R2	0.95	0.975	0.996	0.996	0.996
No Of Layers		Mean RMSE			
1	0.069	0.050	0.047	0.047	0.047
2	0.082	-	-	-	-
3	0.074	-	-	-	-
4	-	-	-	-	-
No of Hidden Units		Mean RMSE			
Jan-40	0.088	0.048	-	-	-
41 - 80	0.075	-	-	-	-
81 - 120	-	-	-	0.048	-
121 - 160	0.076	0.05	0.051	0.047	0.046
161 - 200	0.068	0.05	0.047	0.046	0.047
Learning Rate		Mean RMSE			
0.0001 - 0.001	0.071	0.049	0.047	0.047	0.047
0.001 - 0.01	0.076	0.056	0.051	-	-
0.01 - 0.1	0.076	-	-	-	-
0.1 - 1	-	-	-	-	-
L2 Regularization		Mean RMSE			
0.000001 - 0.00001	0.075	0.051	-	-	-
0.00001 - 0.0001	0.076	0.049	0.047	0.046	0.046
0.0001 - 0.001	0.068	0.051	0.048	0.048	0.050
0.001 - 0.01	-	-	-	-	-

Summarized findings

The experimental results demonstrate that tuning static hyper-parameters significantly affects the forecasting accuracy of the LSTM model in predicting sea water tidal shifts.

Across all four experimental setups, the following key observations were made:

- **Mini-Batch Size (Experiment 1):** Smaller mini-batch sizes yield better forecasting accuracy. A mini-batch size of 10 achieved the lowest RMSE of 0.073 and the highest coefficient of determination ($R^2 = 0.9527$). Increasing the mini-batch size consistently degraded model performance, indicating that smaller batches allow for more precise gradient updates.
- **Dropout Rate (Experiment 2):** The model demonstrated robust performance across various dropout rates, with only slight variations in RMSE. The optimal dropout rate was 0.3, achieving an RMSE of 0.072. Single-layer LSTM architectures and high hidden unit counts (161-200) consistently led to improved generalization.
- **Number of Epochs (Experiment 3):** Contrary to common expectations, increasing the number of training epochs beyond 1000 did not improve performance. Instead, an epoch value of 1000 delivered the best result (RMSE = 0.072), while higher values introduced over-fitting and increased error.
- **Number of Iterations (Experiment 4):** Increasing the number of training iterations substantially improved performance up to a saturation point. The best RMSE of 0.047 and R^2 of 0.996 were achieved and maintained consistently across iterations 200 to 400, highlighting iteration count as a critical driver for convergence and stability.

Across all experiments, single-layer LSTM networks consistently outperformed multilayer architectures in this domain, and hidden units in the range of 161-200 yielded the best feature representations. Moreover, initial learning rates between 0.0001-0.001 and L2 regularization values between 0.00001-0.0001 emerged as optimal ranges that supported low error rates and strong generalization.

These results emphasize the importance of targeted hyperparameter tuning for LSTM based time series models and provide empirical guidelines for developing high-accuracy tidal forecasting systems.

Conclusion

This paper presents a novel framework for optimizing Long Short-Term Memory (LSTM) deep learning models in sea level forecasting by integrating static hyperparameter tuning with dynamic hyper-parameter optimization using Bayesian algorithms. Unlike prior studies that either focus on empirical tuning or rely on default parameters, our work systematically distinguishes between static and dynamic hyper-parameters, offering a structured approach that enhances model performance, interpretability, and reproducibility. The findings demonstrate that optimal configurations, specifically a single LSTM layer, 161 - 200 hidden units, a learning rate between 0.0001-0.001, and L2 regularization in the range of 0.00001 - 0.001, consistently yield the lowest RMSE and highest R^2 scores. Moreover, among static hyperparameters, smaller mini-batch sizes (e.g., 10), moderate epochs (1000), and higher iteration counts (≥400) significantly improve convergence and accuracy. Dropout rates of 0.3 or lower were also found to enhance generalization while preventing over-fitting.

What sets this study apart is its comprehensive and methodical assessment of hyperparameter influence on tidal forecasting, a level of granularity not extensively reported in previous works. Additionally, by evaluating performance across multiple parameter ranges using a controlled setup, this paper provides a reusable, domain-adaptable optimization strategy applicable to other time series forecasting problems beyond coastal applications. The impact of this work lies not only in the improved accuracy achieved, but also in its contribution to building more robust, interpretable, and generalizable LSTM based models.

This work can be further extended by increasing the observational period (e.g., from 4 to 52 weeks) and incorporating multivariate inputs such as meteorological variables, remote sensing data, and spatial characteristics, elements that were excluded in this univariate analysis. Such extensions will enable even more precise and context-aware sea level predictions. Overall, this paper makes a significant and original contribution to the field of deep learning for environmental forecasting and offers practical guidelines for LSTM deployment in critical coastal decision-making scenarios. These contributions, supported by rigorous experimentation and systematic methodology, justify the paper's acceptance in a high-impact journal.

Acknowledgments

The authors would like to express their sincere gratitude to Universiti Malaysia Sabah for their financial support, which made this research possible.

Author Contribution:

- **Nosius Luaran:** Conceived and designed the experiments; Performed the experiments; Analyzed and interpreted the data; Contributed reagents, materials, analysis tools or data; Wrote the paper.
- **Rayner Alfred:** Conceived and designed the experiments; Performed the experiments; Analyzed and interpreted the data; Contributed reagents, materials, analysis tools or data; Wrote the paper.
- **Fengchang Xu, Haviluddin:** Contributed reagents, materials, analysis tools or data.

Funding Statement

This work was supported in part by Universiti Malaysia Sabah.

Conflict of Interest

The authors declare no conflict of interest.

Additional Information

No additional information is available for this paper.

Bibliography

1. Yaqoob I., *et al*. "Internet of things architecture: Recent advances, taxonomy, requirements, and open challenges". *IEEE Wireless Communications* 24.3 (2017): 10-16.

2. Canizo M., *et al*. "Multi-head cnn-rnn for multi-time series anomaly detection: An industrial case study". *Neurocomputing* 363 (2019): 246-260.

3. Wan R., *et al*. "Multivariate temporal convolutional network: A deep neural networks approach for multivariate time series forecasting". *Electronics* 8.8 (2019).

4. Abubakar AG., *et al*. "A review of modelling approaches on tidal analysis and prediction". The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences XLII-4/W16 (2019): 23-34.

5. Van Houdt G., *et al*. "N´apoles: A review on the long short-term memory model". *Artificial Intelligence Review* 53 (2020): 5929-5955.

6. Yamak PT., *et al*. "A comparison between arima, lstm, and gru for time series forecasting". In: Proceedings of the 2019 2nd International Conference on Algorithms, Computing and Artificial Intelligence. ACAI (2019): 49-55. Association for Computing Machinery, New York, NY, USA (2019).

7. Peter G and Matskevichus M. "Hyperparameters tuning for machine learning models for time series forecasting". In: 2019 Sixth International Conference on Social Networks Analysis, Management and Security (SNAMS) (2019): 328-332.

8. Hewamalage H., *et al*. "Recurrent neural networks for time series forecasting: Current status and future directions". *International Journal of Forecasting* 37.1 (2021): 388-427.

9. Hewamalage H., *et al*. "Recurrent neural networks for time series forecasting: Current status and future directions". *International Journal of Forecasting* 37.1 (2021): 388-427.

10. Dixon M and London J. "Financial forecasting with a-rnns: A time series modeling approach". *Frontiers in Applied Mathematics and Statistics* 6 (2021): 59.

11. Casado-Vara R., *et al*. "Web traffic time series forecasting using lstm neural networks with distributed asynchronous training". *Mathematics* 9.4 (2021).

12. Le XH., *et al*. "Application of long short-term memory (lstm) neural network for flood forecasting". *Water* 11.7 (2019).

13. Zhang Q., *et al*. "Prediction of sea surface temperature using long short-term memory". *IEEE Geoscience and Remote Sensing Letters* 14.10 (2017): 1745-1749.

14. Van Houdt G., *et al*. "El ni˜nosouthern oscillation forecasting using complex networks analysis of lstm neural networks". *Artificial Life and Robotics* 24 (2019): 445-451.

15. G., *et al*. "A forecasting model for wave heights based on a long short-term memory neural network". *Acta Oceanologica Sinica* 40 (2021): 62-69.

16. Kelany O., *et al*. "Deep learning model for financial time series prediction". In: 2020 14th International Conference on Innovations in Information Technology (IIT) (2020): 120-125.

17. Zhou S., *et al*. "An optimized heterogeneous structure lstm network for electricity price forecasting". *IEEE Access* 7 (2019): 108161-108173.

18. P., *et al*. "Forecasting tourist arrivals via random forest and long short-term memory". *Cognitive Computation* 13 (2021): 125-138.

19. Saini U., *et al*. "Univariant time series forecasting of agriculture load by using lstm and gru rnns". In: 2020 IEEE Students Conference on Engineering Systems (SCES) (2020): 1-6.

20. Wu CH., *et al*. "A new forecasting framework for bitcoin price with lstm". In: 2018 IEEE International Conference on Data Mining Workshops (ICDMW) (2018): 168-175.

21. Zhou Y., *et al*. "Lstm based network flow forecasting in high-speed railway data network". In: 2019 IEEE International Conferences on Ubiquitous Computing Communications (IUCC) and Data Science and Computational Intelligence (DSCI) and Smart Computing, Networking and Services (SmartCNS) (2019): 71-75.

22. Sarah A., *et al*. "Lstm model to forecast time series for ec2 cloud price". In: 2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (2018): 1085-1088.

23. Rauf HT., *et al*. "Time series forecasting of covid-19 transmission in asia pacific countries using deep neural networks". *Personal and Ubiquitous Computing* (2021): 1-18.

24. W., *et al*. "Forecasting hotel reservations with long short-term memory-based recurrent neural networks". *International Journal of Data Science and Analytics* 9 (2020): 77-94.

25. Pala Z and Atici R. "Forecasting sunspot time series using deep learning methods". *Solar Physics* 294 (2019): 50.

26. Yang H., *et al*. "A network traffic forecasting method based on sa optimized arima-bp neural network". *Computer Networks* 193 (2021): 108102.

27. Helli SS., *et al*. "Short-term forecasting covid-19 cases in turkey using long short-term memory network". 2020 Medical Technologies Congress (TIPTEKNO) 1-4 (2020).

28. Karsanti HT., *et al*. "Deep learning-based patient visits forecasting using long short term memory. In: 2019 International Conference of Artificial Intelligence and Information Technology (ICAIIT) (2019): 344-349.

29. Yao S., *et al*. "High-frequency stock trend forecast using lstm model". In: 2018 13th International Conference on Computer Science Education (ICCSE) (2018): 1-4.

30. Han Y. "A forecasting method of pharmaceutical sales based on arima-lstm model". In: 2020 5th International Conference on Information Science, Computer Technology and Transportation (ISCTT) (2020): 336-339.

31. Yadav A., *et al*. "Optimizing lstm for time series prediction in Indian stock market". *Procedia Computer Science* 167 (2020): 2091-2100.

32. Chicco D., *et al*. "The coefficient of determination r-squared is more informative than smape, mae, mape, mse and rmse in regression analysis evaluation". *PeerJ Computer Science* 7 (2021).

33. Alfred R and Obit JH. "The roles of machine learning methods in limiting the spread of deadly diseases: A systematic review". *Heliyon* 7 (2021).