Australasian Transport Research Forum 2023 Proceedings
29 November – 1 December, Perth, Australia
Publication website: http://www.atrf.info

Evaluation of incremental deep learning approach on real-time traffic prediction

Mohammadreza Ghanbari[1], Saeed Asadi Bagloee[2], Zay Maung Maung Aye[3], Majid Sarvi[4]

[1, 2, 3, 4] Faculty of Engineering and Information Technology, University of Melbourne, Melbourne, Australia

Email for correspondence: mghanbarimal@student.unimelb.edu.au

Abstract

Traffic prediction is a crucial element in managing traffic. Its goal is to comprehend traffic patterns over time in order to anticipate future behavior. In recent times, many research papers have contributed to this area, with a focus on developing machine learning algorithms, particularly deep learning algorithms. Despite producing promising outcomes, real-time traffic prediction faces obstacles due to traffic data being a subset of big data streams that are continuously updated. Nonetheless, over the years, incremental learning has evolved to tackle real-time problems. To address this challenge, this paper evaluates an incremental deep learning approach using overlapping rolling window, tap delay line method and LSTM on a real-world traffic flow dataset collected in Melbourne.

1. Introduction

Intelligent transport systems (ITS) have undergone vast development and advancements, resulting in a large amount of data being generated continuously and in real-time at a low cost. One of the critical tasks of ITS is to leverage such data for various optimization tasks and applications. In this paper, we focus on one of the applications, namely, traffic prediction, which aims to predict the traffic status of a road network given recent past traffic observations.

Traffic data often comes in the form of time series, and an important task over traffic data, i.e., traffic prediction, is often modelled as a time series prediction problem. A time series refers to a set of data with a temporal correlation. The objective of time series prediction is to anticipate future values by analyzing past data. Time series prediction has various applications in diverse fields such as transportation (Bao et al., 2023), finance (Ghanbari and Arian, 2019), (Son et al., 2023), climate (Mudelsee, 2019), (Liu et al., 2022), and biology (Costello and Martin, 2018).

In the last decade, significant efforts have been made to develop techniques for time series prediction, ranging from statistical approaches to advanced deep learning algorithms. Statistical methods are mathematical techniques used to infer variable relationships. Exponential Smoothing (Gardner Jr, 1985), Vector Autoregressive (Chandra and Al-Deek, 2009), and Autoregressive Integrated Moving Average (Kumar and Hariharan, 2022) are the most commonly used statistical models. However, despite their popularity, they are often insufficient since they rely heavily on the assumption of stationary data.

With the emergence of machine learning and deep learning algorithms such as Support Vector Regression (Nidhi and Lobiyal, 2022), Artificial Neural Network (More et al., 2016), Convolution Neural Network (He et al., 2019), (Li et al., 2022), Recurrent Neural Network (Li and Ren, 2022), (Nourmohammadi et al., 2023), and Graph Convolutional Network (Kipf and

Welling, 2016), (Bao et al., 2023), there has been a growing interest in deep learning-based traffic prediction techniques. A substantial number of studies on traffic prediction have been conducted in recent years as these methods can potentially fit a wide range of functions and have a better ability to identify patterns (Lee et al., 2021).
However, the substantial and continuous flow of time series data presents a significant challenge for real-time traffic prediction, leading to a host of other difficulties. For example, the models used for prediction need to be updated frequently to incorporate new traffic patterns. Additionally, due to space constraints, it is impractical to store and analyze all data simultaneously, making computation infeasible (Gomes et al., 2019). To address these challenges, different paradigms have been introduced. One important paradigm of such is online learning, which refers to methods that learn dynamically from incoming data (Hoi et al., 2021).
In this paper, we present an attempt to evaluate the aforementioned challenges through the combination of LSTM, Tap Delay Line and Rolling Window techniques. Despite the perceived simplicity of the solution, it is particularly effective for learning from a continuous data stream.

The remainder of this paper is organized as follows: Section 2 describes the technical background. Section 3 presents the proposed method. Experimental results are discussed in Section 4. Section 5 contains the discussion and Section 6 concludes the paper.

2. Technical background

In this section we briefly review the techniques used to form the approach.

2.1. Long short-term memory

In a traditional feed-forward neural network, the error term for each layer is determined by multiplying the errors of all previous layers. This can lead to an issue of vanishing gradients when using activation functions such as the sigmoid function, which have small derivatives that get multiplied multiple times as the number of layers increases. This problem was addressed with the introduction of Long Short-Term Memory (LSTM), an artificial recurrent neural network (RNN) architecture that was developed by Hochreiter in 1997 (Hochreiter and Schmidhuber, 1997).

The purpose of an LSTM recurrent unit is to retain all previous knowledge that the network has encountered while disregarding irrelevant data. This is achieved through the use of various activation function layers, known as "gates". Additionally, each unit maintains an Internal Cell State vector, which represents the information that was deemed important and retained by the previous LSTM recurrent unit. An LSTM recurrent network consists of three different gates as follows:

a. Forget Gate: It determines to what extent to forget the data from the previous unit.

$$f_t = \sigma\big(w_f \,.\, [h_{t-1}, x_t] + b_f\big) \tag{1}$$

b. Input Gate: It determines the extent of information to be written onto the Internal Cell State.

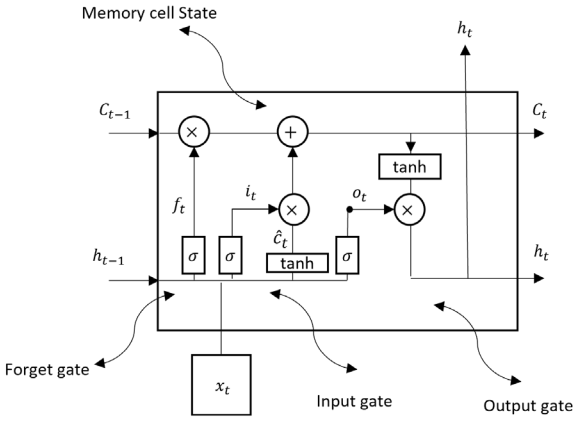
$$\begin{aligned} i_t &= \sigma(w_i \,.\, [h_{t-1}, x_t] + b_i) \\ \hat{C}_t &= tanh(w_{\hat{c}} \,.\, [h_{t-1}, x_t] + b_{\hat{c}}) \end{aligned} \tag{2}$$

c. Output Gate: It determines what output (next hidden state) to generate from the current Internal Cell State.

$$C_t = f_t * C_{t-1} * i_t * \hat{C}_t \quad (3)$$
$$o_t = \sigma(w_o \,.\, [h_{t-1}, x_t] + b_o)$$
$$h_t = o_t * \tanh\,(C_t)$$

Here, $x_t$is the data sample at time t, $w_f, w_i, w_{\hat{c}}, w_o$ are the weights and $b_f, b_i, b_{\hat{c}}, b_o$ are bias, h_{t-1} is the previous state, σ and $tanh$ are the activation functions and $*$ is element-wise multiplication. Figure 1 illustrates the LSTM structure.

Figure 1: The structure of LSTM model.



2.2. Tapped delay line

Tap Delay Line is a technique commonly used in time series data analysis to improve the accuracy of predictions. It involves creating a sequence of delayed versions of the original time series data, each with a specified time delay. These delayed versions are then used as additional inputs to a time series data analysis model (e.g., a prediction model), along with the original time series data. The technique is particularly useful for time series data that exhibits periodicity or seasonality, as it allows the time series data analysis model to capture the patterns in the data more effectively. Simply, the Tap Delay Line (TDL) is an input vector that consists of the current time-step and past time-step data instances. Let $[x_1, \dots, x_N]$ be a univariate vector of a time series. Then every single input value s_t to a model is converted to a vector as follows:

$$\text{TDL} = [x_t, x_{t-1} \dots, x_{t-d}] \quad (4)$$

where d *is the delay* number. Likewise, if $A = [X_1, \dots, X_k]$ is a multivariate dataset with $X_i = [x_i^1, \dots, x_i^N]$, Then the input is enriched to be:

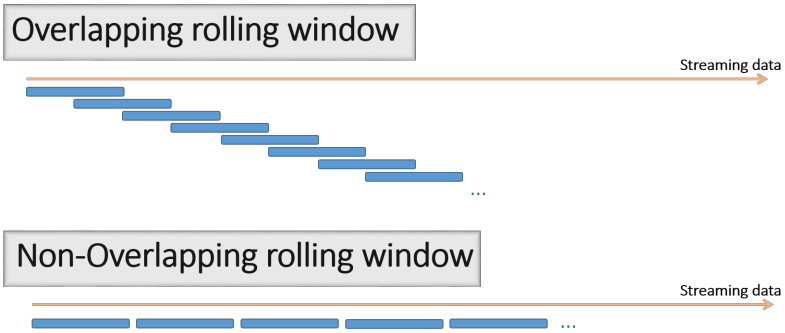
$$\text{TDL} = [x_1^t, \dots, x_1^{t-d}, \dots, x_i^t, \dots, x_i^{t-d}, \dots, x_k^t, \dots, x_k^{t-d}] \quad (5)$$

This equation means that the input vector consists of the current and past d values of all columns of the input dataset. From a mathematical standpoint, tapped delay line relies on the Takens Theorem (Takens, 2006). Despite its benefits, it adds an additional hyperparameter (i.e., the delay number) to the model and requires the delay number to be predetermined.

2.3. Rolling window

The rolling window technique is a powerful technique to optimally feed data into a model (Zivot and Wang, 2006). By dividing the data into fixed-sized batches and sequentially feeding them into a model, this approach has shown to be effective for various time series prediction tasks. There are two different approaches within the rolling window technique, each with its own benefits. The first is the overlapping rolling window approach, where data batches are acquired by sequentially moving a pre-specified window with a fixed overlap size. The second approach is the non-overlapping rolling window, where the data is separated into independent batches. The rolling window technique introduces another hyper-parameter, the batch size, and needs to be set in advance. Figure 2 depicts the overlapping rolling window technique.

Figure 2: Overlapping and non-overlapping rolling windows approaches.

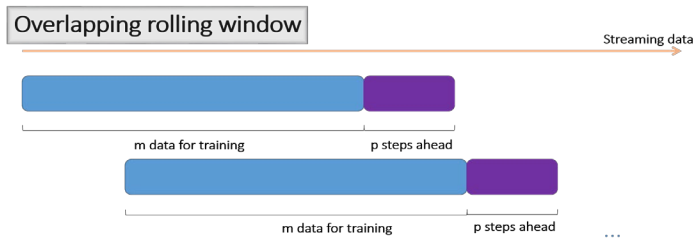


3. Overall approach

Suppose there is a time series of incoming data, $\{X_i\}_{i=1}^{\infty}$, and B_1 denotes the first batch of data where $B_1 = \{X_1, X_2, \dots, X_m\}$ with utlizing the tap delay line. At the first stage, an LSTM model is initialized and then B_1 is fed to the model as training data. Here, the model initializes the parameters including weghts and biases randomly, and after training, these parameters are optimized based on B_1. Next, the model is used to predict for p steps ahead. After making prediction, the next batch of data B_2 is created and then fed to the model. Here B_2 is created based on p new incoming real data points and the last $m - p$ data points of batch B_1, i.e., $B_2 = \{X_{m-p}, X_{m-p+1}, \dots, X_m, X_{m+1}, \dots, X_{m+p}\}$ where $\{X_{m+1}, \dots, X_{m+p}\}$ is the receiving real dataset. At this step, the saved model (i.e., the one trained from B_1, is loaded and is further trained on B_2. In the next step, the trained model on batch B_2 is again used to make prediction. and the above precedure is repeated.

Figure 3 illustrates the proposed method (with a fixed data size on every batch). The first m data points are used for training (showed in blue color) and then after that the trained model predicts for the next p steps ahead (showed in purple color). Then in the next step the actual values of the predicted data are attached to the last training data and just as much data is dropped from the beginning of training data.

Figure 3: The overlapping rolling window combined with tap delay line method for making prediction in real time.



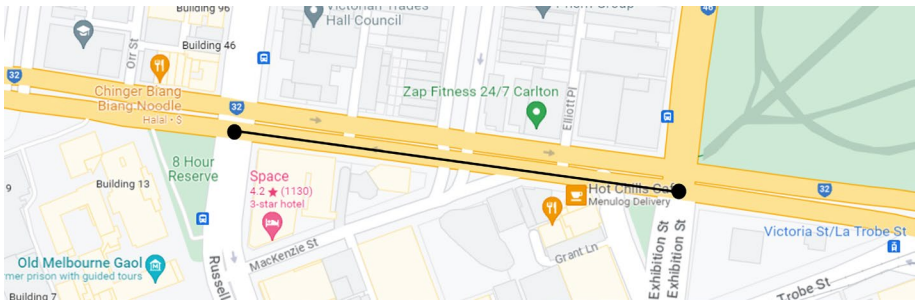
4. Experimental results

In this section, an experiment on a real-world dataset is conducted to answer whether the proposed method is capable of handling streaming data. All the experiments are carried out in Python 3.9.7 environment using TensorFlow (Abadi et al., 2016) on a laptop with 11th Gen Intel(R) Core i7 and 16 GB memory.

4.1. Data description

The model is evaluated on a real-world traffic dataset. Data was collected from the Victoria Street in Melbourne, Australia (Figure 4). The data contains the average speed and flow for 40 days and the values are aggregated every 15 minutes. The task is traffic flow prediction.

Figure 4: The location of data collection



4.2. Measures of effectiveness

In this study, to evaluate the accuracy of prediction, the root mean square error (RMSE) is used.

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

where y_i and $\hat{y}_i$ are the ground-truth and predicted values of the ith data point, respectively and N is the number of samples.

4.3. Numerical results

In the experiment we test both well-known methods LSTM and CNN. The number of hidden inputs are set to 20 for both and one dense layer is used at the end. The mean square error is used as the loss function and ADAM as the optimization algorithm. We used the sigmoid activation function and 10 training epochs. The size of tap delay line is 8 which means it uses

data from the past two hours for predicting the next value(s). Also, we select 30 days (2,880 samples) as the batch size of the window (m) to train the model and the rest for testing and updating. For data normalization, the min-max method was used to bring the data values into the range of [0, 1].

Figure 5 illustrates the comparison of forecasted and actual values of traffic flow for single step ahead (next 15 minutes) and 4 step ahead for a period of 10 days after denormalization and corresponding loss history. Note that each batch of data is trained with 10 epochs.

Figure 5: The actual and predicted values of traffic flow for 1 step (top) and 4 steps (bottom) ahead for 10 days, and their corresponding training loss history using LSTM.

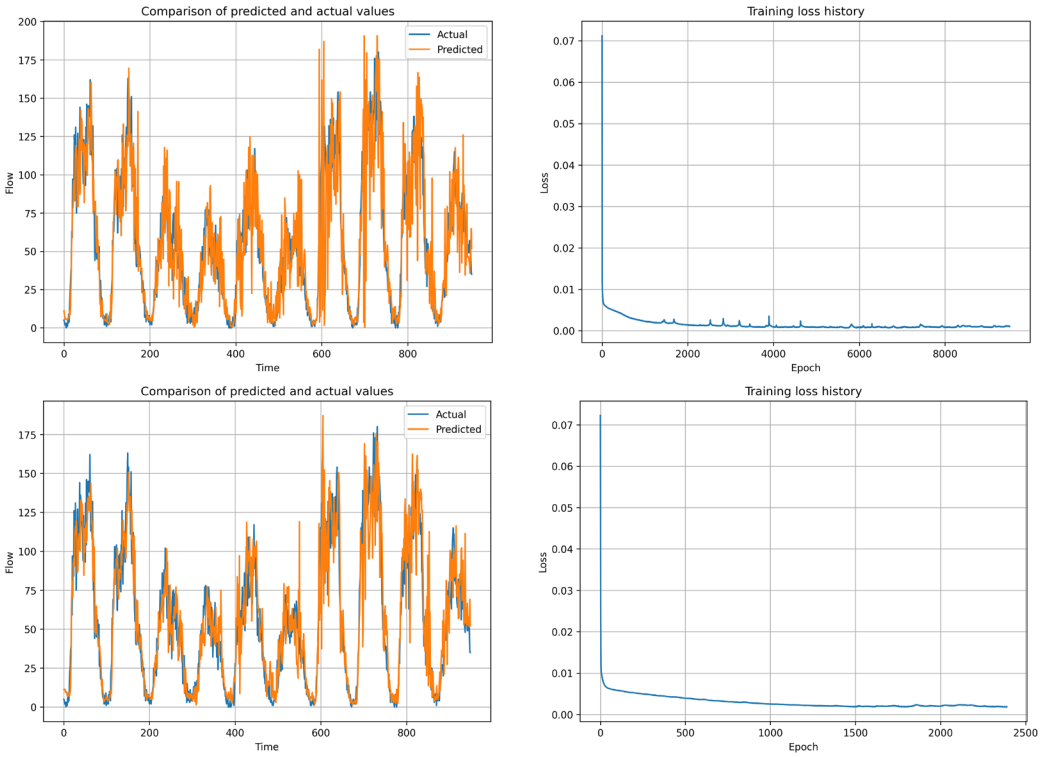


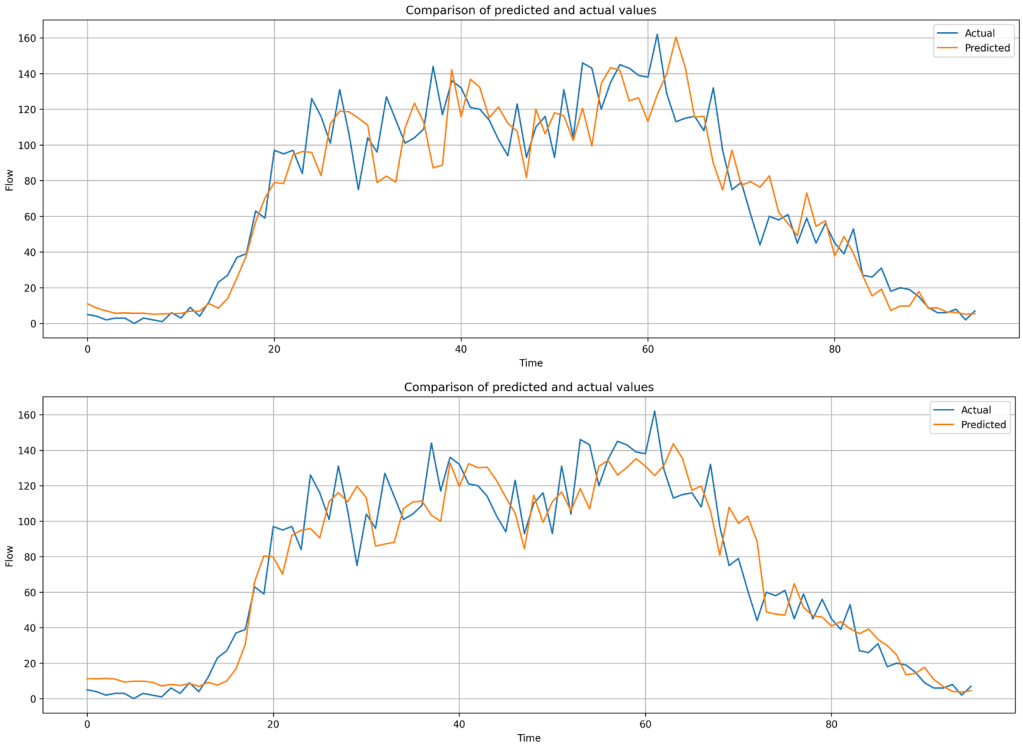
Table 1 shows the cumulative RMSE values of the results for time horizon 1 and 4 for both LSTM and CNN model utilized in the framework. And as can be seen, the smaller the number of horizon steps results in better performance. Also, the LSTM outperforms the other one since it is capable of capturing the inherently sequential dependencies of traffic data time series.

Table 1: The cumulative RMSE for different time horizon over the predictions.

Model\ Horizon	15 mins	60 mins
CNN	26.65	99.38
LSTM	24.32	73.82

For a better comparison, Figure 6 provides the actual and predicted values of the tested data for different time horizons for one day. The results and visualizations confirm that this approach is capable of handling huge and continuous dataset for real-time prediction.

Figure 6: The actual and predicted values of traffic flow for 15 mins (top) and 60 mins (bottom) for 1 day (96 steps) using LSTM.



5. Discussion

A standout advantage of this approach, compared to conventional implementation of LSTM models for time series prediction, is the noteworthy reduction in computational time. By employing incremental learning, we sidestep the need for complete model retraining when new data arrives. Instead, we incrementally update the model using the latest data while retaining past knowledge. This streamlined process not only enhances real-time adaptability to evolving patterns but also significantly cuts down on the computational resources required. The computational efficiency positions this approach as a practical solution for Real-time Traffic prediction, demonstrating its clear advantage over traditional implementation.

It is noteworthy that an essential challenge in deep learning and data-driven traffic prediction models is their inclination to focus on typical traffic patterns, neglecting the more crucial "tail end" of traffic data that holds significant value for real-time predictions. Additionally, these models often lack the capability to anticipate unfamiliar traffic patterns not encountered during training. Solutions include a real-time update by data assimilation (e.g., Kalman Filter), or fuse the machine learning model with a conventional traffic model that can predict irregular traffic patterns. Acknowledging these valid concerns, our study is centered around the evaluation of our existing approach within its current framework. While the integration of real-time updates or the fusing with conventional traffic models presents an intriguing direction for future research, our immediate focus lies in refining the efficacy of our model within its existing framework.

6. Conclusion

Real-time traffic prediction is crucial for effective traffic management. With advancements in communication, sensors, and data availability, accurate prediction is necessary. However, real-time prediction presents various challenges. Recently, deep learning techniques, particularly the RNN, have been shown to outperform other methods due to their recursive nature. In this study, the LSTM algorithm is employed for traffic prediction. In this regard, the combination of a rolling window technique and tap delay line is used to make a tricky approach to update the algorithm dynamically and using it for continuous streaming time series datasets. The model is tested on real world traffic flow dataset collected in Melbourne, Victoria.

7. References

ABADI, M., BARHAM, P., CHEN, J., CHEN, Z., DAVIS, A., DEAN, J., DEVIN, M., GHEMAWAT, S., IRVING, G. & ISARD, M. Tensorflow: a system for large-scale machine learning. Osdi, 2016. Savannah, GA, USA, 265-283.

BAO, Y., HUANG, J., SHEN, Q., CAO, Y., DING, W., SHI, Z. & SHI, Q. 2023. Spatial–Temporal Complex Graph Convolution Network for Traffic Flow Prediction. *Engineering Applications of Artificial Intelligence,* 121**,** 106044.

CHANDRA, S. R. & AL-DEEK, H. 2009. Predictions of freeway traffic speeds and volumes using vector autoregressive models. *Journal of Intelligent Transportation Systems,* 13**,** 53-72.

COSTELLO, Z. & MARTIN, H. G. 2018. A machine learning approach to predict metabolic pathway dynamics from time-series multiomics data. *NPJ systems biology and applications,* 4**,** 1-14.

GARDNER JR, E. S. 1985. Exponential smoothing: The state of the art. *Journal of forecasting,* 4**,** 1-28.

GHANBARI, M. & ARIAN, H. 2019. Forecasting stock market with support vector regression and butterfly optimization algorithm. *arXiv preprint arXiv:1905.11462*.

GOMES, H. M., READ, J., BIFET, A., BARDDAL, J. P. & GAMA, J. 2019. Machine learning for streaming data: state of the art, challenges, and opportunities. *ACM SIGKDD Explorations Newsletter,* 21**,** 6-22.

HE, Z., CHOW, C.-Y. & ZHANG, J.-D. STCNN: A spatio-temporal convolutional neural network for long-term traffic prediction. 2019 20th IEEE International Conference on Mobile Data Management (MDM), 2019. IEEE, 226-233.

HOCHREITER, S. & SCHMIDHUBER, J. 1997. Long short-term memory. *Neural computation,* 9**,** 1735-1780.

HOI, S. C., SAHOO, D., LU, J. & ZHAO, P. 2021. Online learning: A comprehensive survey. *Neurocomputing,* 459**,** 249-289.

KIPF, T. N. & WELLING, M. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

KUMAR, P. B. & HARIHARAN, K. 2022. Time Series Traffic Flow Prediction with Hyper-Parameter Optimized ARIMA Models for Intelligent Transportation System. *Journal of Scientific & Industrial Research,* 81**,** 408-415.

LEE, K., EO, M., JUNG, E., YOON, Y. & RHEE, W. 2021. Short-term traffic prediction with deep neural networks: A survey. *IEEE Access,* 9**,** 54739-54756.

LI, H., LI, X., SU, L., JIN, D., HUANG, J. & HUANG, D. 2022. Deep spatio-temporal adaptive 3d convolutional neural networks for traffic flow prediction. *ACM Transactions on Intelligent Systems and Technology (TIST),* 13**,** 1-21.

LI, Y. & REN, H. Vessel Traffic Flow Prediction Using LSTM Encoder-Decoder. Proceedings of the 2022 5th International Conference on Signal Processing and Machine Learning, 2022. 1-7.

LIU, Y., LI, D., WAN, S., WANG, F., DOU, W., XU, X., LI, S., MA, R. & QI, L. 2022. A long short-term memory-based model for greenhouse climate prediction. *International Journal of Intelligent Systems,* 37**,** 135-151.

MORE, R., MUGAL, A., RAJGURE, S., ADHAO, R. B. & PACHGHARE, V. K. Road traffic prediction and congestion control using Artificial Neural Networks. 2016 International Conference on Computing, Analytics and Security Trends (CAST), 2016. IEEE, 52-57.

MUDELSEE, M. 2019. Trend analysis of climate time series: A review of methods. *Earth-science reviews,* 190**,** 310-322.

NIDHI, N. & LOBIYAL, D. 2022. Traffic flow prediction using support vector regression. *International Journal of Information Technology,* 14**,** 619-626.

NOURMOHAMMADI, Z., NOURMOHAMMADI, F., KIM, I. & PARK, S. H. 2023. A deep spatiotemporal approach in maritime accident prediction: A case study of the territorial sea of South Korea. *Ocean Engineering,* 270**,** 113565.

SON, B., LEE, Y. Y., PARK, S. & LEE, J. 2023. Forecasting global stock market volatility: The impact of volatility spillover index in spatial-temporal graph-based model. *Journal of Forecasting*.

TAKENS, F. Detecting strange attractors in turbulence. Dynamical Systems and Turbulence, Warwick 1980: proceedings of a symposium held at the University of Warwick 1979/80, 2006. Springer, 366-381.

ZIVOT, E. & WANG, J. 2006. *Modeling financial time series with S-PLUS*, Springer.