Salinity Prediction Of Raw Water Using Deep Learning Based Time Series Model

Duc-Tam Huynh
University of Information Technology
Viet Nam National University Ho Chi Minh City
Ho Chi Minh City, Vietnam
huynhductam96@gmail.com

Trong-Hop Do
University of Information Technology
Viet Nam National University Ho Chi Minh City
Ho Chi Minh City, Vietnam
hopdt@uit.edu.vn

***Abstract*—Water is a precious and indispensable resource for human life. In recent years, the issue of water pollution and scarcity has attracted increasing attention from the international community. In this context, monitoring and forecasting water quality is an urgent requirement to ensure safe and clean water for the community. In the field of water production and supply, water salinity is an important factor affecting the water treatment process. According to QCVN 01-1:2018/BYT, the maximum allowable salinity of drinking water is 250 mg/L. Salinity varies seasonally and over time, and forecasting surface water salinity for water treatment is a challenge. To meet this need, the use of time series forecasting models has become increasingly important. This paper proposes appropriate machine learning models for forecasting water salinity. Our method learns the trends and patterns in river salinity data using statistical and machine learning techniques to build time series forecasting models. Also, we compared performance of using one-variable data with three-variable data, and explain how the size of window affects model performance, and impacts of forecast period as well. We evaluated our method on a 6-year dataset of river salinity data. The results illustrate that our selected models can forecast river salinity with a high performance. Our method can be employed to a wide range of water supply systems to improve the effectiveness of procedures for treating water and ensure the safety and sustainability of water resources.**

Keywords—water salinity, machine learning, deep learning, time series.

I. Introduction

Water is a vital resource for human existence and development. Ensuring safe and clean water is essential for protecting public health and improving the quality of life. Therefore, research and forecasting of raw water salinity before treatment is extremely important to ensure the efficient and sustainable use of water resources.

Challenges of pollution and climate change have significantly impacted water quality and salinity in water production areas. By building an accurate forecasting model, predicting the salinity of raw water before treatment could help to make appropriate decisions and treatment plans to minimize the impact of salinity on water supply systems and treatment processes.

Building time series models has challenges of determining the window size, forecast period, and the number of features used. In this research, transitional data were generated from raw data based on window size and forecast period to determine whether these factors affect the performance of models. In addition, the correlation between the quantity of characteristics and model performance is clarified. We also conduct comparison of state-of-the-art models (including ANNs, ARIMA, CNNs, GRU, LSTM, and TCN) for time series forecasting and selected appropriate models for certain contexts. These achievements could support to make science-based predictions, which can improve the efficiency and reliability of water resource management.

This study contributes to comparison of different models' performance on river salinity data, considering the number of variables. Additionally, we evaluate these models based on window size as well as forecast period.

II. Related Work

A. ARIMA

ARIMA is widely used for forecasting and modeling time series. The basic formula of the ARIMA model is represented as a set of three parameters (p, d, q). Value of d can be determined by the Augmented Dickey-Fuller (ADF). If the p-value is greater than the significance level (typically 0.05), then the time series needs to be integrated further until it becomes weakly stationary (has a lower p-value) [1]. The partial auto-correlation function (PACF) can be employed to determine the value of p. The significant lag on the PACF plot often determines the value of p. If it crosses the first lower level on the horizontal line, the value of p is typically equal to that position. To determine the value of q, the ACF plots can be used. The significant lag on the ACF plot often determines the value of q. If it crosses the first lower level on the horizontal line, the value of q is typically equal to that position [2].

B. Artificial Neural Networks

Artificial neural networks are composed of linked neurons, mirroring the structure of biological neural networks. ANNs comprise three layers, including one input layer, one output layer, and minimum one concealed layer.

Traditional neural networks are trained using a process called supervised learning. A collection of input-output pairs is fed into the network. The network then learns to associate the inputs with the outputs. To accomplish this, the connections' weights among neurons undergo adjustments. These adjustments rely on an optimization algorithm to modify the weights.

The weights are adjusted based on the error using a method called gradient descent, which calculates concerning its weights and then modifies these weights to reduce the errors. This procedure is iterated multiple times until the weights of the neural network reach convergence. A neural network's training is basically an optimization issue. After the training process, providing the network with a dataset that was not used in the training process as input, and the network will give you a predicted output [3].

C. Convolutional Neural Networks

The model (CNNs) is primarily applied for computer vision. The model's applications are image classification, object detection, and segmentation issue. They can also be applied in recommender systems, natural language processing, and time series. An example a CNN regression architecture as illustrated in Fig. 1.

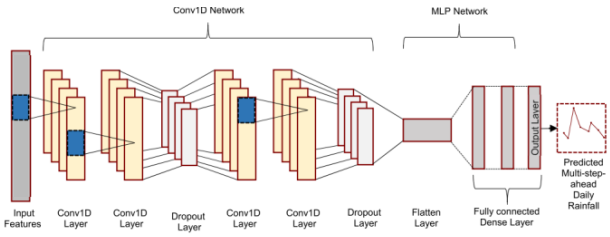

Fig. 1. A schematic representation of the Conv1D architecture *[4]* .

According to output layer for classification issue, the number of output units depends on the quantity of categories. For neural networks used for regression, there is one output [5].

D. Long Short-Term Memory

LSTM is able to learn long-term dependencies. The Fig. 2 depicts the alterations implemented on the concealed at time step t. These equations below represent an LSTM:

$$i = \sigma(W_i\, h_{t-1} + U_i x_t + V_i c_{t-1}) \quad (1)$$
$$f = \sigma(W_f h_{t-1} + U_f x_t + V_f c_{t-1}) \quad (2)$$
$$o = \sigma(W_o\, h_{t-1} + U_o x_t + V_o c_{t-1}) \quad (3)$$
$$g = tanh W_g\, h_{t-1} + U_g x_t \quad (4)$$
$$c_t = (f * c_{t-1}) + (g * i) \quad (5)$$
$$h_t = \tanh(c_t) * o \quad (6)$$

In which, i, f, and o interpret for the input gate, forget gate, and output gate, respectively. The calculations are performed employing identical equations, albeit utilizing distinct sets of matrices of parameters W_i, U_i, W_f, U_f, and W_0, U_0.

The forget gate is used to calculate the degree of the previous state h_{t-1} can pass through. The input gate is employed to determine the degree of new state calculated for the existing input x_t can pass through, and the output gate determines the degree, which revealed the internal hidden state to the next layer. Calculating of the state g relies on both the input x_t and the preceding state h_{t-1}.

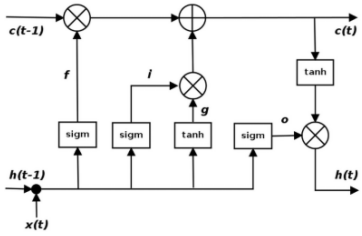

Fig. 2. A visual depiction of LSTM cell *[5]*.

Derived from i, f, o, and g, the calculation of state c_t at time t follows a specific procedure: it involves the multiplication of the cell state c_{t-1} at time (t-1) by the forget gate f, along with the product of the state g and the input gate i. Fundamentally, this process represents a mechanism for amalgamating prior memory and incoming information; specifically, adjusting the forget gate to 0 disengages the retention of former memory, while setting the input gate to 0 excludes the integration of the newly computed state. Subsequently, the computation of the hidden state h_t at time t is contingent upon the cell state c_t at time t, guided by the output gate o [5].

E. Gated Recurrent Units

The Gated Recurrent Units (GRU) represent a variation of the LSTM model, preserving the capacity to mitigate the vanishing gradient issue akin to LSTM. However, GRU exhibits a comparatively simpler internal architecture, leading to expedited training due to its reduced computational demands for updating the hidden state. A GRU cell includes the update gate as z and the reset gate as r. The proportion of the previous memory to preserve can be determined by update gate, while the reset gate is used to determine the method for integrating the new input with the existing memory.

The GRU cell determines how to compute the state h_t at time t from the hidden state h_{t-1} following set of equations:

$$z = \sigma(W_z\, h_{t-1} + U_z x_t) \quad (7)$$
$$r = \sigma(W_r\, h_{t-1} + U_r x_t) \quad (8)$$
$$c = tanh(W_c\, (h_{t-1} * r) + U_c x_t) \quad (9)$$
$$h_t = (z * c) + ((1 - z) * h_{t-1}) \quad (10)$$

The computations for the update gate z and reset gate r involve a fusion of the prior hidden state h_{t-1} and the present input x_t. The cell state c is determined through a functional relationship between the output of the reset gate r and the input x_t. Ultimately, the derivation of the hidden state h_t at time t hinges upon the relationship between the cell state c and the preceding hidden state h_{t-1}. The parameters W_z, U_z, W_r, U_r, and W_c, U_c are learned during training [5].

F. Temporal Convolutional Networks

Temporal convolutional networks can be described by a set of simple architectures, which is shown in Fig. 3. The distinct features of TCNs is that no information "leaks" from the future into the past. In addition, the structure is capable of receiving input sequences of varying lengths and mapping them into output sequences of equivalent lengths, just like an RNN.

TCNs include dilated convolutions and residual connections. A residual block is represented as (11).

$$o = \text{Activation}(x + F(x)) \quad (11)$$

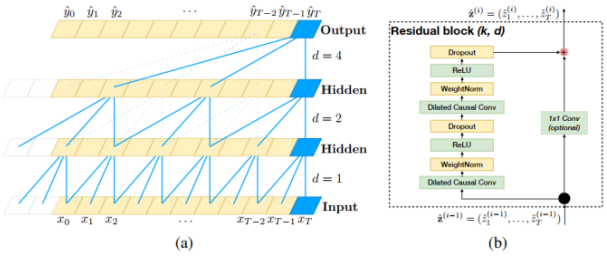

Fig. 3. Structure of the Temporal Convolutional Networks. (a) A dilated causal convolution with dilation factors d = 1; 2; 4 and filter size k = 3. (b) TCN residual block

In TCNs, the widths of input may differ output. To address the mismatch between input and output widths, the authors use 1x1 convolution to guarantee that the element-wise addition ⊕ receives tensors of the same size [6].

III. DATA

Experimental data**:** water quality data of the Saigon River, which are monitored at the Hoa Phu pumping station, belongs

to Tan Hiep Water Plant, where position shown in Fig. 4. Data are mean values of each day, collected over a period of 6 years (from January 1, 2017 to December 31, 2022). It contains 11 features.

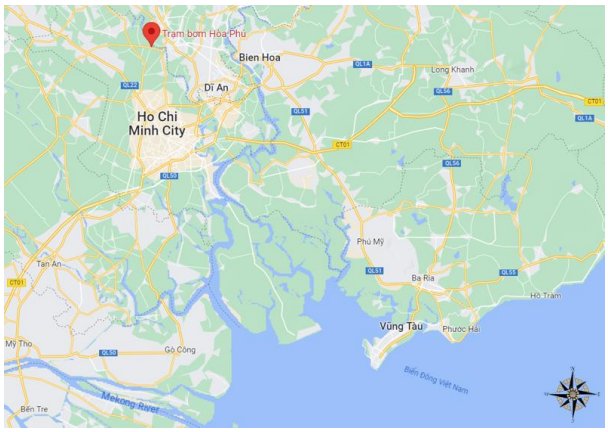

Fig. 4. Location of Hoa Phu pumping station.

IV. Experiment and Result

A. Experimental Procedure

The data are experienced cleaning process, which is to remove outliers, which values greater than and less than 3SD (standard deviation) [7]. Missing data will be replaced with linear interpolation values as (12). Let's consider that the x-data points are arranged in increasing order [8].

$$\hat{y}(x) = y_i + \frac{(y_{i+1}-y_i)(x-x_i)}{(x_{i+1}-x_i)} \qquad (12)$$

Where: $\hat{y}(x)$ is the interpolated value at x, y_i and y_{i+1} are the known values at x_i and x_{i+1}, x is the point at which we want to interpolate

The next process is feature selection. The dataframe has 11 features, after calculating Pearson correlation coefficient of these features to salinity as (13), the three features are selected, including pH, electrical conductivity, and salinity, because the Pearson correlation coefficient of those are higher than 0.50 and those parameters can be monitored by sensors instead of manually tested in chemical laboratory of the water plant.

$$r_{XY} = \frac{\sum(X_i-\bar{X})(Y_i-\bar{Y})}{\sqrt{\sum(X_i-\bar{X})^2\sum(Y_i-\bar{Y})^2}} \qquad (13)$$

Where: X_i and Y_i are the values of feature X and Y for the i-th observation. $\bar{X}$ and $\bar{Y}$ are the means of features X and Y, respectively [9].

After selecting features, new datasets were generated from clean dataset to clarify whether or not using three variables has better performance than one variable, increasing window size could improve model performance, and the longer the forecast period, the higher level of error. To explore how window size, the number of variables, and forecast period affect performance of models, the period of time applied for window size will be altered from 7 days to 15 days and 30 days, and forecast period will be changed from 1 day to 3 days and 7 days, the number of variables varied as 1 variable and 3 variables. These generated 18 datasets with 9 three-variable datasets and 9 one-variable datasets. For example, "Dataset 3var, 30-1" in Fig. 5 illustrates for data using three variables, window size is 30 days, and forecast period is 1 day. The research employed ANN, CNN, GRU, LSTM, TCN, TCN, and ARIMA for training data. These models are trained on one-variable and three-variable, excepting to the ARIMA model is only trained on one-variable dataset. The matrix of datasets and models need to be trained and tested in 99 running times.

The datasets were divided into three subsets: training set - 60%, validating set - 20%, and test set – 20% of data [10]. The workflow of the research is summarized in Fig. 5. To split data, the split points in time were applied as the first 60% for training, the next 20% for validating, and the rest 20% for testing.

After splitting process, the data will be standardized as follows (14) before training and testing.

$$z = \frac{x-\mu}{\sigma} \qquad (14)$$

Where: z is the standardized value, x represents the original data, μ represents the average of the dataset, σ represents the standard deviation of the dataset [11].

B. Evaluation Metrics

We used two performance metrics: Mean Absolute Error (MAE) and Mean Absolute Percentage Error (MAPE), defined as follows:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|x_i - \hat{x}_i| \qquad (15)$$

$$MAPE = \frac{100}{n}\sum_{i=1}^{n}\frac{|x_i - \hat{x}_i|}{x_i} \qquad (16)$$

In which, x_i is the actual salinity data, $\hat{x}_i$ is the predicted salinity data, and n represents the quantity of data points [12].

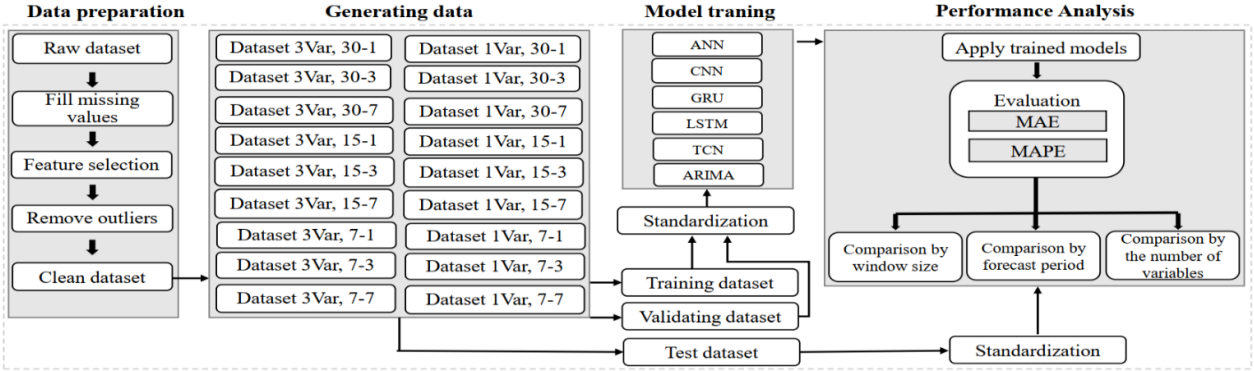

Fig. 5. The workflow of data processing.

C. Experimental Environment

The research had implemented on Window 10, used Python programming language. Scripts were compiled on Visual Studio Code. The computer configurations are 11th Generation Intel® Core™ i5 Processor and Ram DDR4 8GB 2666Mhz.

D. Results and Discussion

D.1. Comparison by Forecast Priod

From the results of Fig.6, Fig. 7, Fig. 8, Fig. 12, Fig. 13, and Fig. 14, the experiment intentionally fixed the window size as 7 days, 15 days, and 30 days to compare performance of models when increased forecast period. They indicate that the MAE and MAPE error tend to increase as the forecast period increases from 1 day to 3 days and 7 days, while the window sizes were fixed as 7 days, 15 days, and 30 days. The ARIMA model has a higher level of error than the other models because it can only forecast well for stationary datasets [13]. The best 1-day forecast model is CNN_1Var with an MAE error of 1.40mg/L (MAPE error of 4.31%) with a window size of 7 days. The GRU_1Var model has the lowest MAE error (3.20 mg/L) of all the 3-day forecast models, but GRU_1Var and CNN_1Var have the same level of error by MAPE as 10.0% with a window size of 7 days. The ANN_1Var model with a 7-day window size has the lowest MAE error (5.38 mg/L) of all the 7-day forecast models, while comparing by MAPE error, the LSTM_3Var has better performance with MAPE error as 15.9% comparing to ANN_1Var as 16.7% in the same window size of 7 days. When comparing in different metrics, the best order performance model may change, because the metrics measuring certain aspects of the data or the random sampling used [14]. The CNN_1Var model has Conv1D layers that are able of extracting spatial features such as trends, peaks, or variations in time series. This allows the model to learn important short-term patterns in data, leading to better forecasting performance [15]. The GRU_1Var outperform other models in case short-term forecasting task because its few parameters with simple structure [16, 17]. The ANN_1Var models could provide comparable forecasting results because data are relatively short and dependencies between the data points are not complex [18].

D.2. Comparison by Window Size

As the results of Fig. 9, Fig. 10, Fig. 11, Fig. 15, Fig. 16, and Fig. 17, the MAE error also tends to rise as the window size increases from 7 days to 15 days and 30 days, while forecast period was fixed as 1 day, 3 days, and 7 days. That also happens when comparing by MAPE metric. It is notable that increasing window size could improve performance of models for longer forecast period according to models which have buffer memory like GRU and LSTM. As Fig. 17 show that using appropriate window size and multivariate facilitate the LSTM_3Var to achieve the best performance with forecast period as 7 days with MAPE 15.9%. For TCN model, dilated convolution and residual block may enhance model performance [6].

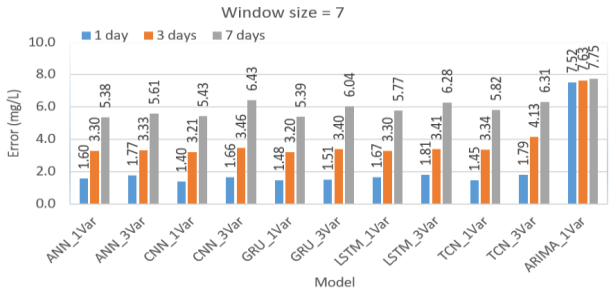

Fig. 6. MAE of models using window size as seven days, and forecast period as one day, three days, and seven days.

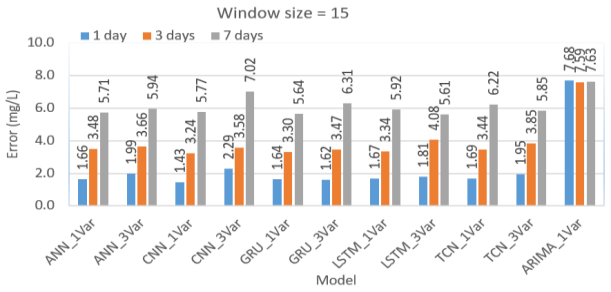

Fig. 7. MAE of models using window size as 15 days, and forecast period as one day, three days, and seven days.

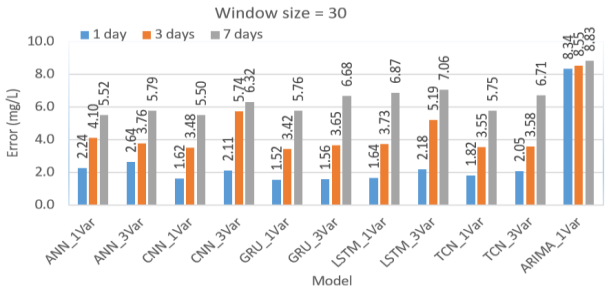

Fig. 8. MAE of models using window size as 30 days, and forecast period as one day, three days, and seven days.

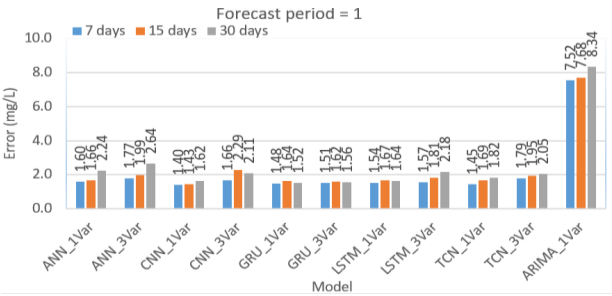

Fig. 9. MAE of models using forecast period as 1 day, and window size as one day, three days, and seven days.

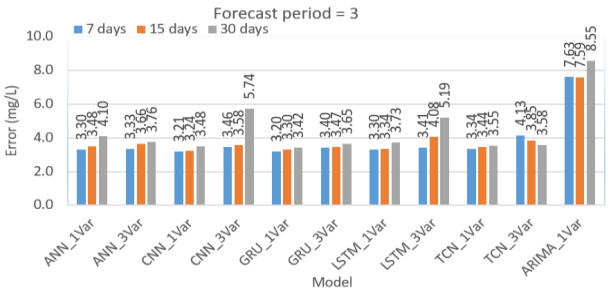

Fig. 10. MAE of models using forecast period as 3 days, and window size as one day, three days, and seven days.

D.3. Comparison by The Number of Variables

As the same window size and forecast period, increasing the number of variable results in rise of error in both metrics. This phenomenon is known as the "curse of dimensionality". As the number of variables increases, the available data becomes sparse in the high-dimensional space. This sparsity can lead to difficulties in accurately estimating model parameters, identifying meaningful patterns, and generalizing well to unseen data [19].

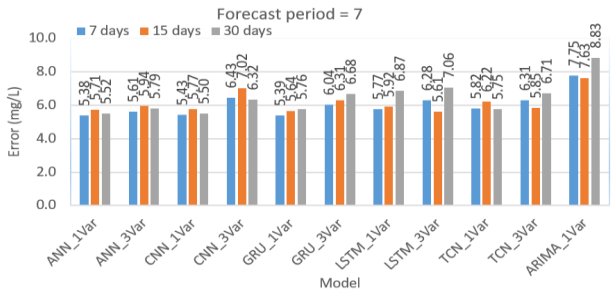

Fig. 11. MAE of models using forecast period as 7 days, and window size as one day, three days, and seven days.

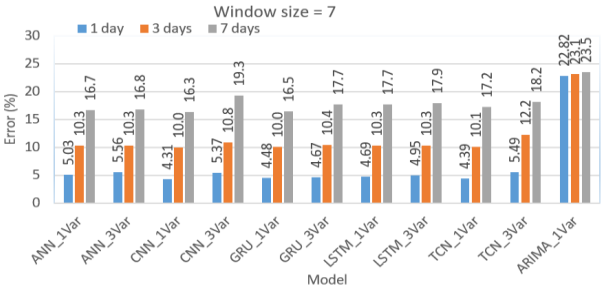

Fig. 12. MAPE of models using window size as 7 days, and forecast period as one day, three days, and seven days.

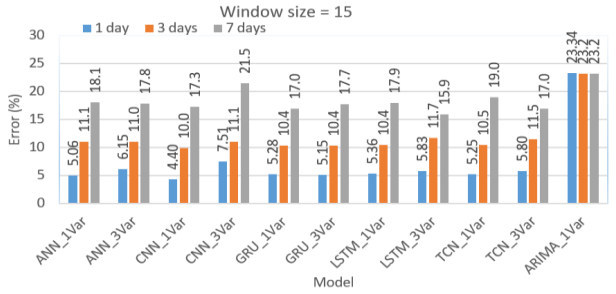

Fig. 13. MAPE of models using window size as 15 days, and forecast period as one day, three days, and seven days.

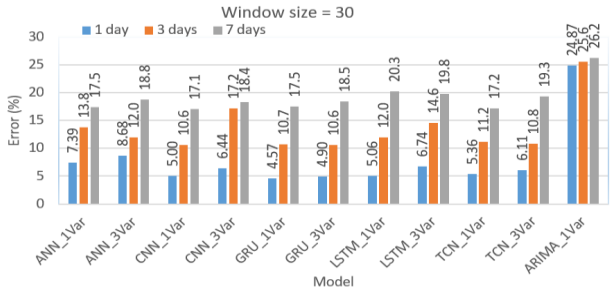

Fig. 14. MAPE of models using window size as 30 days, and forecast period as one day, three days, and seven days.

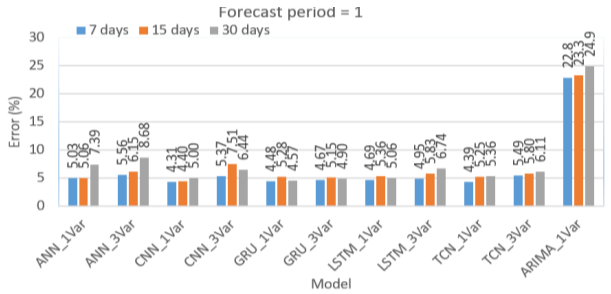

Fig. 15. MAPE of models using forecast period as 1 day, and window size as one day, three days, and seven days.

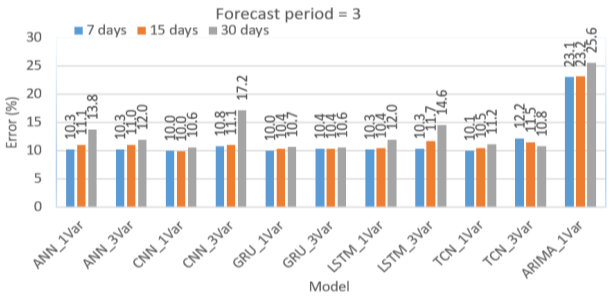

Fig. 16. MAPE of models using forecast period as 3 days, and window size as one day, three days, and seven days.

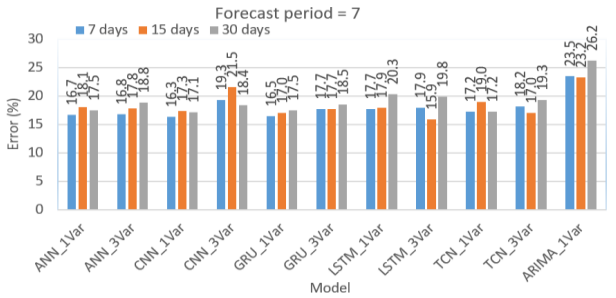

Fig. 17. MAPE of models using forecast period as 7 days, and window size as one day, three days, and seven days.

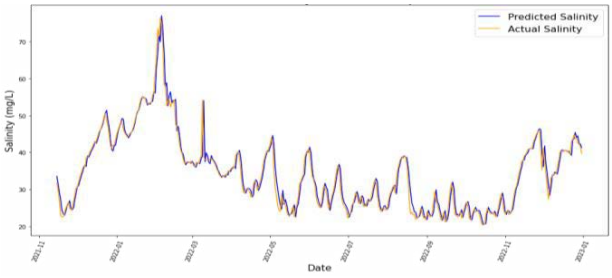

Fig. 18. Actual values and predicted values of CNN_1Var model with window size of as 7 days and forecast period as 1 day.

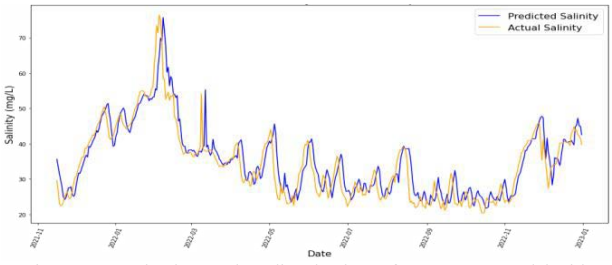

Fig. 19. Actual values and predicted values of CNN_1Var model with window size as 7 days and forecast period as 3 days.

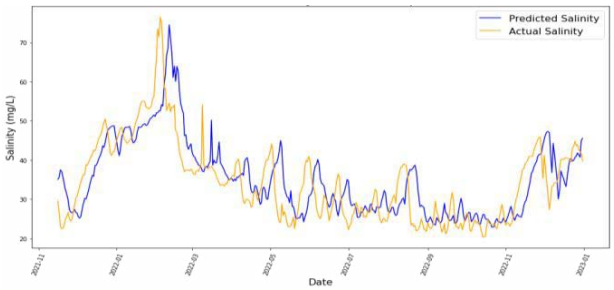

Fig. 20. Actual values and predicted values of CNN_1Var model with window size as 7 days and forecast period as 7 days.

V. Conclusion and Future Work

The study focuses on forecasting the salinity of untreated water, a topic not previously explored in Tan Hiep Water Plant. Extensive real-time data was employed for this purpose. Sophisticated techniques in deep learning and machine learning tailored for predicting time series were effectively put into practice. The research involved testing and comparing the models' effectiveness across diverse conditions.

In the near future, we integrate additional variables into the model to enhance accuracy. Experimenting with advanced deep learning models based on transformer architecture for this forecasting problem. We also continue to build a real-time forecasting system, research and improve a more general data preprocessing algorithm, as well as inherit and develop machine learning models to achieve better performance, in order to lay the foundation for the development of higher-level problems, with the ultimate goal of this development direction being to build a real-time application that can help operators quickly grasp the trend of salinity changes as quickly as possible.

References

[1] M. A. N. M. Asri, N. Zaini and M. F. A. Latip, "Development of an LSTM-based Model for Energy Consumption Prediction with Data Pre-analysis," 2021 11th IEEE International Conference on Control System, Computing and Engineering (ICCSCE), Penang, Malaysia, 2021.

[2] George E. P. Box, Gwilym M. Jenkins, Gregory C. Reinsel, Greta M. Ljung, Time Series Analysis: Forecasting and Control, Wiley, 2015.

[3] Perry Xiao, Artificial Intelligence Programming with Python from Zero to Hero, John Wiley & Sons, Inc., 2022.

[4] Mohd Imran Khan and Rajib Maity, "Hydrid Deep Learning Approach for Multi-Step-Ahead Daily Rainfall Prediction Using GCM Simulations," *IEEE Access,* vol. 8, pp. 52774-52784, 2020.

[5] Amita Kapoor, Antonio Gulli, Sujit Pal, Deep Learning with TensorFlow and Keras: Build and deploy supervised, unsupervised, deep, and reinforcement learning models, Packt Publishing, 2022.

[6] Shaojie Bai ,J. Zico Kolter, Vladlen Koltun, "An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling," arXiv:1803.01271v2, 2018.

[7] N. N. R. Ranga Suri , Narasimha Murty M and G. Athithan, Outlier Detection: Techniques and Applications, Springer: https://doi.org/10.1007/978-3-030-05127-3, 2019.

[8] Qingkai Kong, Timmy Siauw, Alexandre Bayen, Python Programming and Numerical Methods: A Guide for Engineers and Scientist, https://doi.org/10.1016/C2018-0-04165-1: Academic Press, 2020.

[9] Robert S. Witte and John S. Witte, Statistics, Wiley, 2017.

[10] V. Roshan Joseph, "Optimal ratio for data splitting," *Wiley,* vol. 15, no. 4, pp. 531-538, 2022.

[11] Alice Zheng and Amanda Casari, Feature Engineering for Machine Learning, O'Reilly , 2018.

[12] Zhiyong Cui, Ruimin Ke, Ziyuan Pu, Yinhai Wang, "Stacked Bidirectional and Unidirectional LSTM Recurrent Neural Network for Forecasting Network-wide Traffic State with Missing Values," arXiv:2005.11627v1, 2020.

[13] Peter T Yamak, Li Yujian and Pius K Gadosey, "A Comparison between ARIMA, LSTM, and GRU for Time Series Forecasting," Computing and Artificial Intelligence, https://doi.org/10.1145/3377713.3377722, 2019.

[14] Max Kuhn and Kjell Johnson, Applied Predictive Modeling, https://doi.org/10.1007/978-1-4614-6849-3: Springer, 2013.

[15] Jun Zhang, Yongchuan Yu , Jianzhuo Yan and Jianhui Chen, "Data-Driven Parameter Prediction of Water Pumping Station," MDPI, https://doi.org/10.3390/w15061128, 2023.

[16] Shuai Gao, Yuefei Huang, Shuo Zhang, Jingcheng Han, Guangqian Wang, Meixin Zhang, Qingsheng Lin, "Short-term runoff prediction with GRU and LSTM networks without requiring time step optimization during sample generation," Journal of Hydrology, https://doi.org/10.1016/j.jhydrol.2020.125188, 2020.

[17] K.E. ArunKumar, Dinesh V. Kalaga, Ch. Mohan Sai Kumar, Masahiro Kawaji, 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," *Alexandria Engineering Journal,* vol. 61, no. 10, pp. 7785-7603, 2022.

[18] He-Ren Lou, Xin Wang, Ya Gao and Qiang Zeng, "Comparison of ARIMA model, DNN model and LSTM model in predicting disease burden of occupational pneumoconiosis in Tianjin, China," BMC Public Health, https://doi.org/10.1186/s12889-022-14642-3, 2022.

[19] Christopher M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.