CLASSIFICATION OF WATER QUALITY FOR AQUACULTURE USING CNN

Vaishnavi Kanchi[1], Komala Priya Gunturu [2], Priyanka Vattikonda[3], Sreenidhi Govada[4], [1,2,3,4]
Department of Computer Science & Engineering, VNITSW, Guntur.
A.Peda Gopi[5], [5]Associate Professor, Department of Computer Science & Engineering, VNITSW, Guntur
kvyshu2016@gmail.com[1], komalapriya118@gmail.com[2]priyankavattikonda@gmail.com[3],
sreenidhinirula@gmail.com[4], [5]gopiarepalli2@gmail.com

Abstract
Water is a vital for all aspects of human and ecosystem survival and health. Thus, its quality is also important. Water quality refers to the composition of a water sample. Evaluations of water quality parameters are necessary to enhance the performance of an assessment operation and develop better water resources management and plan. Water Quality plays an important role in attaining a sustainable aquaculture system, its cumulative effect can make the entire system. Early detection of fish diseases and identifying the underlying causes are crucial for farmers to take necessary steps to mitigate the potential outbreak. Typically, fish diseases are caused by virus and bacteria; and this may affect the level of pH, DO, BOD, COD, EC, PO43-, NO3-N, and NH3-N in water, resulting in the death of fishes. Being motivated by the recent successes of trending techniques ANN and CNN models has been adopted to detect and predict the degradation of water quality timely and accurately, thus it helps taking pre-emptive steps against potential fish diseases. Examining the results of the models showed that all of them had reached estimation properties. In additional, Conventional Neural Network (CNN) captures the embedded spatial and unsteady behaviour in the investigated problem using its architecture and nonlinearity nature compared with the other classical modelling techniques. The results show that the proposed CNN prediction model has a great potential to simulate and predict the total dissolved solids, electrical conductivity, and turbidity with absolute mean error 10% for different water bodies.

Keywords: Water quality prediction, Water quality parameters, artificial neural network, Conventional neural network

1. Introduction

Water quality directly affects virtually all water uses. Fish survival, diversity and growth; recreational activities such as swimming and boating, municipal, industrial, and private water supplies, agricultural uses such as irrigation and livestock watering, waste disposal, and general aesthetics-all are affected by the physical, chemical, biological, and microbiological conditions that exist in watercourses and in subsurface
Aquifers. Water quality impairment is often a trigger for conflict in a watershed, simply because degraded water quality means that desired uses are not possible or not safe (Heathcote, 1998). Malaysia is a developing country that moves towards the vision 2020. Unfortunately the development that had been carried throughout the country also contributes bad impact to the environment especially water quality. This issue has become sensitive, which not only affects human health, but also the entire environment .The development not only affects the water quality, but also the aquatic lives that live in it. Most acceptable ecological and social decisions are difficult to make without careful modeling, prediction, and analysis of river water quality for typical development scenarios. Water quality prediction enables a manager to choose an option that satisfies large number of identified conditions. For instance, water quality parameters, such as dissolved solids, electrical conductivity and turbidity in water describe a complex process governed by a considerable number of hydrologic, hydrodynamic, and ecological controls that operate at a wide range of spatiotemporal scales. Sources of the admixtures often cannot be clearly identified, and the locally influenced complex mass exchange between the variables may not be known.

Aquaculture is the controlled process of cultivating aquatic organisms, especially for human consumption. It's a similar concept to agriculture, but with fish instead of plants or livestock. Aquaculture is also referred to as fish farming. Aquaculture is breeding, raising, and harvesting fish, shellfish, and aquatic plants. Aquaculture is an environmentally responsible source of food and commercial products, helps to create healthier habitats, and is used to rebuild stocks of threatened or endangered species. Fishes account for approximately 15% of the animal protein intake of the human population globally.

2. Literature survey

Applications of ANNs in the areas of water engineering, ecological sciences, and environmental sciences have been reported since the beginning of the 1990s. In recent years, ANNs have been used intensively for prediction and forecasting in a number of water-related areas, including water resource study (Liong et al., 1999, 2001; Muttil and Chau, 2006; El-Shafie et al., 2008), oceanography (Makarynskyy, 2004), and environmental science (Grubert, 2003). The use of data-driven techniques for modeling the quality of both freshwater (Chen and Mynett, 2003) and seawater (Lee et al., 2000, 2003) has met with success in the past decade. Reckhow (1999) studied Bayesian probability network models for guiding decision making regarding water 424 Ali Najah, Ahmed Elshafie, Othman A. Karim and Othman Jaffar quality in the Neuse River in North Carolina. Chau (2006) has reviewed the development and current progress of the integration of artificial intelligence (AI) into water quality modeling. J. A. Bowers (2000) developed model to predict suspended solids conceder local precipitation, stream flow rates and turbidity as input. Hatim (2007) employed an ANN approach using six variables for the initial prediction of suspended solids in the stream at Mamasin dam. Most of them employed almost all possible environmental parameters as input variables without considering the optimal choice amongst them. The present study attempted to model Johor River Basin water quality parameters using ANN modeling for the first time. Limited water quality data and the high cost of water quality monitoring often pose serious problems for process-based modeling approaches. ANNs provide a particularly good option, because they are computationally very fast and require many fewer input parameters and input conditions than deterministic models. ANNs do, however, require a large pool of representative data for training. ANNs are, however, still not widely used tools in the fields of water quality prediction and forecasting. ANNs are able to approximate accurately complicated non-linear input– output relationships. Like their physics-based numerical model counterparts, ANNs require training or calibration. After training, each application of the trained ANN is an estimation of a simple algebraic expression with known coefficients and is executed practically instantaneously. The ANN technique is flexible enough to accommodate additional constraints that may arise in the application. Moreover, The ANN model can reveal hidden relationships in the historical data, thus facilitating the prediction and forecasting of water quality. This paper demonstrates the application of ANNs to model the values of selected river water quality parameters, having the dynamic and complex processes hidden in the monitored data itself. In addition, objective of this study is to investigate whether it is possible to predict the values of water quality parameters measured by a water quality monitoring program; this task is quite important for enabling selective monitoring of water quality parameters.

3. Proposed Model:

Water Quality Index (WQI) Calculation:

The WQI, which is calculated using several parameters that affect WQ , was used to measure water quality. The performance of the proposed system was evaluated on the published dataset, with seven important water quality parameters. The WQI was calculated using the following formula:

$$WQI = \frac{\sum_{i=1}^{N} q_i \times w_i}{\sum_{i=1}^{N} w_i} \qquad (1)$$

where N denotes the total number of parameters included in the WQI formula, qi denotes the quality estimate scale for each parameter i calculated by Formula (2), and wi denotes the unit weight of each parameter in Formula (3).

$$q_i = 100 \times \left(\frac{V_i - V_{Ideal}}{S_i - V_{Ideal}}\right) \qquad (2)$$

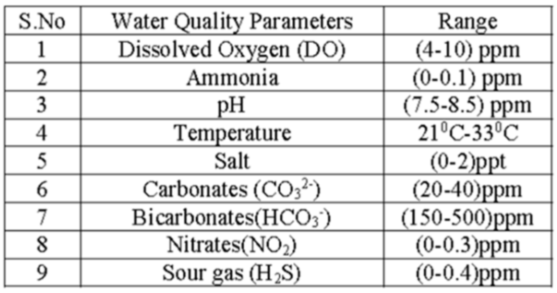
Where Vi is a measured value that refers to the water samples tested, VIdeal is an ideal value and indicates pure water (0 for all parameters except OD = 14.6 mg/L and pH = 7.0), and Si is a standard value recommended for parameter i.

$$w_i = \frac{K}{S_i} \qquad (3)$$

Where K denotes the constant of proportionality, which is calculated using the following formula:

$$K = \frac{1}{\sum_{i=1}^{N} S_i} \qquad (4)$$

WQI can be used to calculate more parameters, including our selecting parameters. The WQI depends on the variable data. The proposed system can test any parameters with any water quality data.



S.No	Water Quality Parameters	Range
1	Dissolved Oxygen (DO)	(4-10) ppm
2	Ammonia	(0-0.1) ppm
3	pH	(7.5-8.5) ppm
4	Temperature	21^0C-33^0C
5	Salt	(0-2)ppt
6	Carbonates (CO_3^{2-})	(20-40)ppm
7	Bicarbonates(HCO_3^-)	(150-500)ppm
8	Nitrates(NO_2)	(0-0.3)ppm
9	Sour gas (H_2S)	(0-0.4)ppm

Table 1: Water quality Parameters with threshold ranges

(1) Convolution layers:

Their roles are too abstract local features at different locations among the whole raw input or the intermediate feature maps with learnable filters (kernels). The size of the convolutional window can be determined by the anomalous object in the modeling of complex geochemical patterns. The advantage of convolution operation is reflected mainly in the implementation of weights sharing and spatial correlation among neighbors (Guo et al., 2016).

Pooling layers:

To speed up the training process and reduce the amount of memory consumed by the network, we try to reduce the redundancy present in the input feature. There are a couple of ways we can down sample an image, but for this post, we will look at the most common one: **max pooling**.

In max pooling, a window passes over the data (how many units to move on each pass). At each step, the maximum value within the window is pooled into an output matrix, hence the name max pooling

Fully-connected layers:

To speed up the training process and reduce the amount of memory consumed by the network, we try to reduce the redundancy present in the input feature. There are a couple of ways we can down sample an image, but for this post, we will look at the most common one: **max pooling**.

In max pooling, a window passes over the data (how many units to move on each pass). At each step, the maximum value within the window is pooled into an output matrix, hence the name max pooling

After multiple convolutional layers and down sampling operations, the output representation is converted into a feature vector that is passed into a Multi-Layer Perceptron, which merely is a neural network with at least three layers. This is referred to as a Fully-Connected Layer.

Dense layer:

The output layer of a CNN is in charge of producing the probability of each class (each digit) given the input data. To obtain these probabilities, we initialize our final dense layer to contain the same number of neurons as there are classes. The output of this dense layer then passes through the **Softmax activation function**, which maps all the final dense layer outputs to a vector whose elements sum up to mean.

The rows are concatenated to form a long feature vector. If multiple input layers are present, its rows are also concatenated to form an even longer feature vector. The feature vector is then passed through multiple dense layers. At each dense layer, the feature vector is multiplied by the layer's weights, summed with its biases, and passed through a non-linearity.

CNN recognizes the patterns to represent image features by utilizing the convolutional layers. CNNs can receive images or a multi-dimensional matrix, and the neurons in CNN are connected to a smaller feature from the previous layer. This algorithm can reduce computations and prevent over fitting problems. Therefore, CNN has been adopted in numerous studies focusing on the image objects from digital images. A convolutional layer consists of the filter size, padding, and stride as the layer parameters. The filter having a specific size (e.g., FH: filter height, FW: filter width) moves around the input image. The padding inserts the zero values around the input image, which prevents the loss for the feature extraction.

The stride can define the step size of the filter in convolutions. In each convolutional layer, the output size is calculated as:

$$\mathrm{OH} = \frac{\mathrm{IH} + 2\mathrm{PH} - \mathrm{FH}}{\mathrm{SH}} + 1 \qquad (1)$$

$$\mathrm{OW} = \frac{\mathrm{IW} + 2\mathrm{PW} - \mathrm{FW}}{\mathrm{SW}} + 1 \qquad (2)$$

where, OH is the height of output, IH is the height of input, FH is the height of the filter, SH is the height of the stride, OW is the width of output, IW is the width of input, PH is the height of the padding, PW is the width of the padding, FW is the width of the filter, and SW is the width direction of the stride. In general, the convolutional layer needs the activation function to transform the signal from linear to non-linear. The rectified linear unit (ReLU) is employed as the activation function in this study. This function improves the computational speed and accuracy compared with the other activation functions (e.g., tangent sigmoid function). Especially, the ReLU function prevents the vanishing gradient problem by an exponentially decreasing the training gradient.

The ReLU function is defined as:

$f(x) = \max(0, x)$ (3)

where, f(x) is the output of ReLU and x is the input signal.

The max-pooling layer was used to extract the invariant features with an efficient convergence rate. This layer can eliminate the non-maximal values by the non-linear downsampling that can reduce the computational sampling during the CNN process [26]. The fully connected vector connects a loss function to calculate errors between the observed and simulated values by the vectorizing the input signal. The MSE is used as a loss function in our study. This calculates errors between simulated and observed values. The mathematical equation of the MSE is as follows:

$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}(Y_i - O_i)^2 \qquad (4)$$

where, Yi is the simulated result, Oi is the observed data, and N is the number of the dataset.

The stochastic gradient descent (SGD) optimization was applied to train a CNN network. SGD optimizes the parameters of a CNN network by reducing the loss function, as:

$$\vartheta = \operatorname{argmin}\frac{1}{N}\sum_{i=1}^{N}\ell(x_i, \varnothing) \qquad (5)$$

where, ϑ is the network parameter, x is the training dataset, N is the number of the dataset, and ` is the loss function.

The deep learning models such as CNN and LSTM require e an epoch number, a batch size, and a learning rate as the hyper parameters for the model training. The epoch number is the number of the learning in the entire training dataset, while the batch size is the number of samples that worked in the training at a time. The learning rate is the step size at each iteration to minimize the loss function. In this study, the assigned epoch number and mini batch of CNN were 1000 and 16, respectively, and the applied learning rate was 0.001

Training the Model:

Training deep neural networks with tens of layers is challenging as they can be sensitive to the initial random weights and configuration of the learning algorithm.

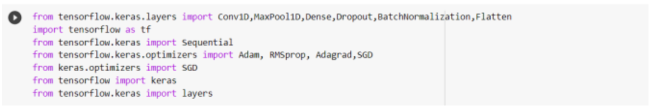
One possible reason for this difficulty is the distribution of the inputs to layers deep in the network may change after each mini-batch when the weights are updated. This can cause the learning algorithm to forever chase a moving target. This change in the distribution of inputs to layers in the network is referred to the technical name "*internal covariate shift*."

Batch normalization is a technique for training very deep neural networks that standardizes the inputs to a layer for each mini-batch. This has the effect of stabilizing the learning process and dramatically reducing the number of training epochs required to train deep networks.

Steps

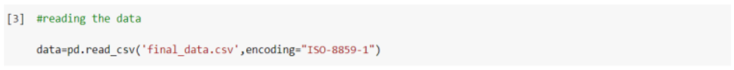
1. Importing required libraries
2. Importing the data
3. Data pre-processing
4. creating a model
5. Fitting the model

Step 1: Importing the required libraries.



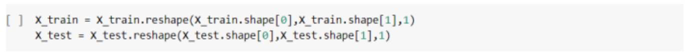
```
from tensorflow.keras.layers import Conv1D,MaxPool1D,Dense,Dropout,BatchNormalization,Flatten
import tensorflow as tf
from tensorflow.keras import Sequential
from tensorflow.keras.optimizers import Adam, RMSprop, Adagrad,SGD
from keras.optimizers import SGD
from tensorflow import keras
from tensorflow.keras import layers
```

Step 2: As mentioned below the input file present in "final_data.csv" We import it using pandas.



```
[3]  #reading the data

     data=pd.read_csv('final_data.csv',encoding="ISO-8859-1")
```

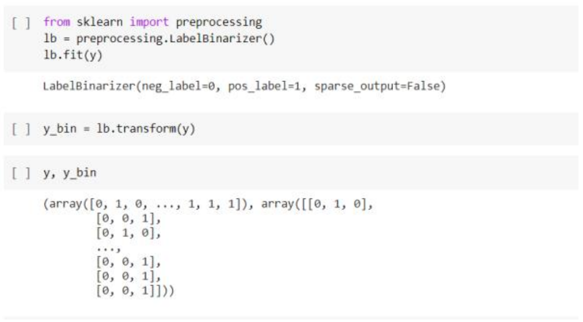
Step3: This is important step because the keras function accepts the inputs as images. But here we are using CSV file which is an array. So at first we have to convert these into image (All images are the 3-D matrix of pixels). So now we re-shape the matrix into a 3-D matrix. We separate the X (input) and Y (output) from Data We also re-scale the data in the range 0-1 because it will be faster to process. So we divide all the values by 255(The max value in the matrix (pixel) is 255)



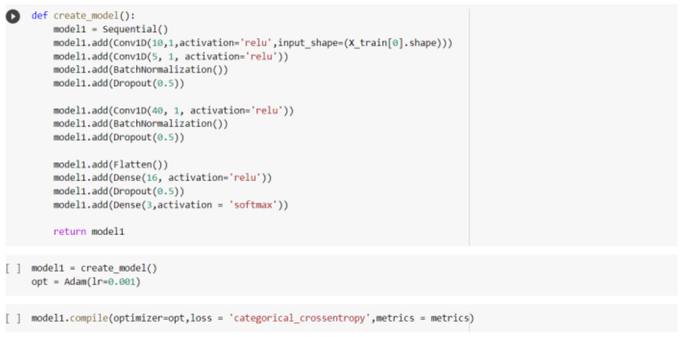
```
[ ]  X_train = X_train.reshape(X_train.shape[0],X_train.shape[1],1)
     X_test = X_test.reshape(X_test.shape[0],X_test.shape[1],1)
```

The model always gives a matrix of probability as output. The max value in that will be the output label. So the next process of data pre-processing is to encode the output. As the output will be in a matrix of 10 values. So we will define 10 neurons in the output layer. But our csv file contains single values. So we encode the values. Example: In the classification of cat(0), dog(1) and monkey(2).The output of the model will look like this [0.23 , 0.01 , 0.91]. But in csv file it will be a single digit i.e. 2. so we encode 2 as [0 , 0 , 1], 1 as [0 , 1 , 0] and 0 as [1 , 0 , 0].

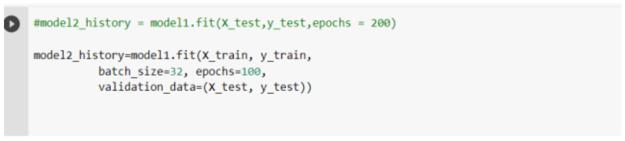
This is done by LabelBinarizer() function.


```
[ ] from sklearn import preprocessing
    lb = preprocessing.LabelBinarizer()
    lb.fit(y)

    LabelBinarizer(neg_label=0, pos_label=1, sparse_output=False)

[ ] y_bin = lb.transform(y)

[ ] y, y_bin

    (array([0, 1, 0, ..., 1, 1, 1]), array([[0, 1, 0],
            [0, 0, 1],
            [0, 1, 0],
            ...,
            [0, 0, 1],
            [0, 0, 1],
            [0, 0, 1]]))
```

Step 4: Now we build a model. The flatten mainly used to flatten the 3-D array of previous layer into a single layer (Because the Ann model only take 1-D array as input).So the flatten creates the input layer. There are 10 labels so there will be 10 neurons in the output layer. Dropout () is used to avoid over-fitting.


```
def create_model():
    model1 = Sequential()
    model1.add(Conv1D(10,1,activation='relu',input_shape=(X_train[0].shape)))
    model1.add(Conv1D(5, 1, activation='relu'))
    model1.add(BatchNormalization())
    model1.add(Dropout(0.5))

    model1.add(Conv1D(40, 1, activation='relu'))
    model1.add(BatchNormalization())
    model1.add(Dropout(0.5))

    model1.add(Flatten())
    model1.add(Dense(16, activation='relu'))
    model1.add(Dropout(0.5))
    model1.add(Dense(3,activation = 'softmax'))

    return model1

[ ] model1 = create_model()
    opt = Adam(lr=0.001)

[ ] model1.compile(optimizer=opt,loss = 'categorical_crossentropy',metrics = metrics)
```

Step 5: Now we fit the model with the data. You can increase the accuracy by increasing the number of epochs, Conv layer,andMaxpool layers. (In this case). But if you are dealing with images then best way is to use Image Data Generator. This uses real-time data augmentation to produce wide variety of images for a same label. The main advantage of this is the amount of Data is reduced.


```
#model2_history = model1.fit(X_test,y_test,epochs = 200)

model2_history=model1.fit(X_train, y_train,
          batch_size=32, epochs=100,
          validation_data=(X_test, y_test))
```

Algorithm

Back propagation is used solely for training all parameters (weights and biases) in CNN. Here is a brief description of the algorithm. The cost function with respect to individual training example (x, y) in hidden layers can be defined as:

$I\ (W, \theta; n, m) = \frac{1}{2}\|gw, \theta(n) - m\|^2$

The equation for error term δ for layer L is given by:

$\delta^{(L)} = ((W^{(L)})^T \delta^{(L+1)}) .h'(z^{(l)})$

Where δ (L+1) is the error for (L + 1) th layer of a network whose cost function is I (W, θ; n,m). h' $(z^{(l)})$ represents the derivate of the activation function.

$\nabla w^{(L)} I(W, \theta; n, m) = \delta^{(L+1)} (i^{(L+1)})^T$

$\nabla \theta^{(L)} I\ (W, \theta; n, m) = \delta^{(L+1)}$ where i is the input, such that $i^{(1)}$ is the input for 1st layer (i.e., the actual input) and $i^{(L)}$ is the input for L – th layer.

Error for sub-sampling layer is calculated as:

$\Delta_q^{(L)} = \text{upsample}((W_q^{(L)})^T \delta_k^{(L+1)}) \cdot h'(z_q^{(L)})$

Where q represent the filter number in the layer. In the subsampling layer, the error has to be cascaded in the opposite direction, e.g., where mean pooling is used, upsample evenly distributes the error to the previous input unit. And finally, here is the gradient w.r.t. feature maps:

$\nabla w_k^{(L)} I(W, \theta; n, m) = \Sigma_{t-1}(i_t^{(L)}) * \text{rot90}(\delta_k^{(L+1)}, 2)$

$\nabla \theta_k^{(L)} I\ (W, \theta; n, m) = \Sigma_{i,j} (\delta_q^{(L+1)})_{i,j}.$

Backpropagation Algorithm in CNN

```
1: Initialization weights to randomly generated value(small)
2: Set learning rate to a small value (positive)
3: Iteration x = 1; Begin
4: for x< max iteration OR Cost function criteria met, do
5: for input $n_1$ to $n_i$ , do
6:      a. Forward propagate through convolution, pooling and then fully conflected layers
7:      b. Derive Cost Function value for the input
8:      c. Calculate error term $\delta^{(L)}$ with respect to weights for each type of layers.
9:      Note that the error gets propagated from layer to layer in the following sequence
10:     i.  Fully connected layer
11:     ii.  Pooling layer
12:     iii. Convolution layer
        13:     d. Calculate gradient $\nabla w_q^{(L)}$ and $\nabla\theta_q^{(L)}$ for weights $\nabla w_q^{(L)}$ and bias respectively for each
        layer
14:             Gradient calculated in the following sequence
15:     i.  Convolution layer
16:     ii.  Pooling layer
17:     iii. Fully connected layer
18:     e.Update weights
19:             $w_{ji}^{(L)} \leftarrow w_{ji}^{(L)} + \nabla w_{ji}^{(L)}$
20:     f.Update bias
21:             $\theta_j^{(L)} \leftarrow \theta_j^{(L)} + \nabla\theta_j^{(L)}$
```

4. Experimental results

Performance measurement approaches, such as MSE, were applied to evaluate the ability of the proposed model to predict the WQI. Furthermore, the accuracy, specificity, sensitivity, precision, recall, and F-score performance measurements were determined to evaluate the FFNN and KNN classification algorithms to classify the WQC. The statistical methods used are defined as follows:

- Mean square error (MSE)

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

- Root mean square error (RMSE)

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

$$R = \frac{n\sum(x \times y) - (\sum x)(\sum y)}{[n\sum(x^2) - \sum(x^2)] \times [n\sum(y^2) - \sum(y^2)]} \times 100\%$$

Where R is Pearson’s correlation coefficient, x is the observation input data in the first set of the training data, y is the observation input data of the second set of the training data, and n is the total number of input variables.

- Accuracy

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \times 100\%$$

- Specificity

$$Specificity = \frac{TN}{TN + FP} \times 100\%$$

- Sensitivity

$$Sensitivity = \frac{TP}{TP + FN} \times 100\%$$

- Precision

$$Precision = \frac{TP}{TP + FP} \times 100\%$$

- F-score

$$\text{F} - \text{score} = \frac{2 * preision * \text{Sensitivity}}{preision + \text{Sensitivity}} \times 100\%$$

Where TP, TN, FP, and FN are the true positive, true negative, false positive, and false negative, respectively.

Results

Experimental Results:

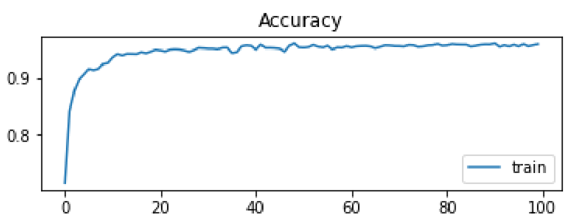


Fig1: Plot accuracy during training

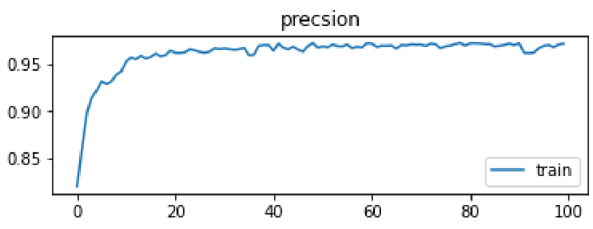


Fig 2: plot Precision during training

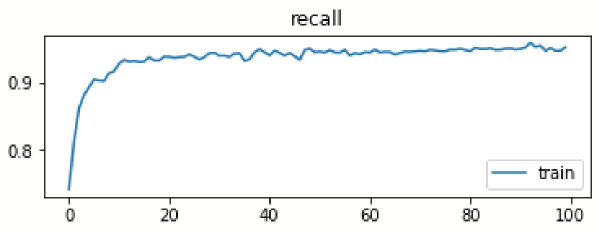


Fig 3: plot Recall during training

Comparing the accuracies of two models:

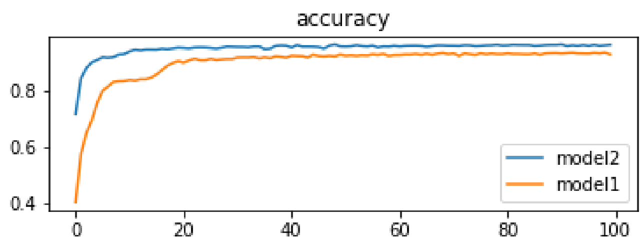


Fig4: Plot precision of the both model1 and model2

Comparing the precisions of two models:

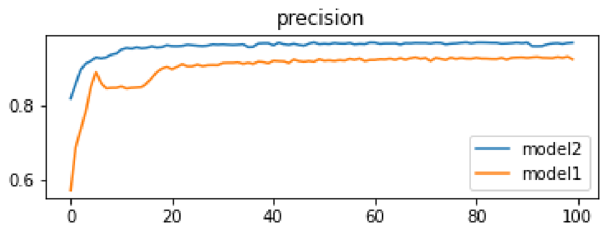


Fig5: Plot precision of the both model1 and model2

Comparing the recalls of Two Models:

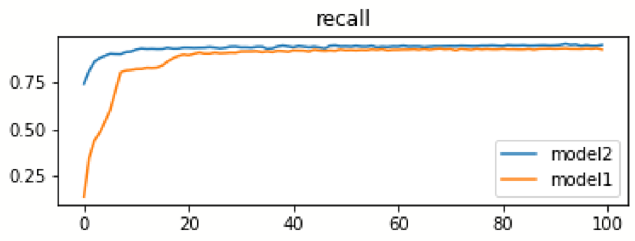


Fig6: Plot recall of the both model1 and model2

5. Conclusion

In this study, deep learning models (CNN) to simulate the water level and the water quality parameters, DO, BOD, COD, EC, PO43-, NO3-N, and NH3-N. Among the deep learning models, the CNN model was adopted to simulate the water level. We found the following in this study: The water level from the CNN model produced the NSE value of 0.933 that can be regarded as acceptable model performance. The water levels increased in the rainy season, while those were low in the dry season. For all of the pollutants, the NSE values of the CNN model for the training and validation periods were above 0.75 which is within the “very good” performance range. The CNN model in this study well represented the different temporal variations of each pollutant type.

This study suggests that the approach of the two deep learning techniques proposed in this study has promise as a tool in accurately simulating the water level and water quality and that this approach can contribute to developing effective strategies for better water sustainability and management. Although our model showed the acceptable model performance, only the three different pollutants were investigated in this study. However, most process-based models can simulate a lot more water quality including the three pollutants (e.g., chlorophyll, algae, dissolved oxygen, and fecal bacteria). A further study is recommended to develop deep learning models so that more pollutants including chlorophyll, algae, dissolved oxygen, and fecal bacteria can be simulated. In addition, further study on the deep learning model with “visual explanations” is required, such as Gradient-weighted-Class Activation Mapping (Grad-CAM) and CAM, because the deep learning model is a black-box model that has general difficulty in identifying physical features. In addition, the approach outlined in this study should be replicated with other datasets

References

[1] Han, H.G.; Qiao, J.F.; Chen, Q.L. Model predictive control of dissolved oxygen concentration based on a self-organizing RBF neural network. *Control Eng. Pract.* **2012**, *20*, 465–476. [Google Scholar] [CrossRef]

[2] Liu, S.; Yan, M.; Tai, H.; Xu, L.; Li, D. Prediction of dissolved oxygen content in quaculture of hyriopsiscumingii using elman neural network. *IFIP Adv. Inf. Commun. Technol.* **2012**, *370 AICT*, 508–518. [Google Scholar] [CrossRef]

[3] Chen, S.; Fang, G.; Huang, X.; Zhang, Y. Water quality prediction model of a water diversion project based on the improved artificial bee colony-backpropagation neural network. *Water* **2018**, *10*, 806. [Google Scholar] [CrossRef]

[4] Fijani, E.; Barzegar, R.; Deo, R.; Tziritis, E.; Konstantinos, S. Design and implementation of a hybrid model based on two-layer decomposition method coupled with extreme learning machines to support real-time environmental monitoring of water quality parameters. *Sci. Total Environ.* **2019**, *648*, 839–853. [Google Scholar] [CrossRef] [PubMed]

[5] Gopi, A.P., Jyothi, R.N.S., Narayana, V.L. et al. Classification of tweets data based on polarity using improved RBF kernel of SVM. Int. j. inf. tecnol. (2020). https://doi.org/10.1007/s41870-019-00409-4

[6] Nourani, V.; Alami, M.T.; Vousoughi, F.D. Self-organizing map clustering technique for ANN-based spatiotemporal modeling of groundwater quality parameters. *J. Hydroinformatics* **2016**, *18*, 288–309. [Google Scholar] [CrossRef]

[7] Chang, F.J.; Chen, P.A.; Chang, L.C.; Tsai, Y.H. Estimating spatio-temporal dynamics of stream total phosphate concentration by soft computing techniques. *Sci. Total Environ.* **2016**, *562*, 228–236. [Google Scholar] [CrossRef]

[8] Li, L.; Jiang, P.; Xu, H.; Lin, G.; Guo, D.; Wu, H. Water quality prediction based on recurrent neural network and improved evidence theory: A case study of Qiantang River, China. *Environ. Sci. Pollut. Res.* **2019**, *26*, 19879–19896. [Google Scholar] [CrossRef] [PubMed]

[9] Zhang, L.; Zou, Z.; Shan, W. Development of a method for comprehensive water quality forecasting and its application in Miyun reservoir of Beijing, China. *J. Environ. Sci.* **2017**, *56*, 240–246. [Google Scholar] [CrossRef]

[10] Gholamreza, A.; Afshin, M.-D.; Shiva, H.A.; Nasrin, R. Application of artificial neural networks to predict total dissolved solids in the river Zayanderud, Iran. *Environ. Eng. Res.* **2016**, *21*, 333–340. [Google Scholar] [CrossRef]

[11] Zhao, J.; Zhao, C.; Zhang, F.; Wu, G.; Wang, H. Water Quality Prediction in the Waste Water Treatment Process Based on Ridge Regression Echo State Network. *IOP Conf. Ser. Mater. Sci. Eng.* **2018**, *435*. [Google Scholar] [CrossRef]

[12] Antanasijević, D.; Pocajt, V.; Povrenović, D.; Perić-Grujić, A.; Ristić, M. Modelling of dissolved oxygen content using artificial neural networks: Danube River, North Serbia, case study. *Environ. Sci. Pollut. Res.* **2013**, *20*, 9006–9013. [Google Scholar] [CrossRef]

[13] Qiao, J.; Hu, Z.; Li, W. Soft measurement modeling based on chaos theory for biochemical oxygen demand (BOD). *Water* **2016**, *8*, 581. [Google Scholar] [CrossRef]

[14] Baek, G.; Cheon, S.P.; Kim, S.; Kim, Y.; Kim, H.; Kim, C.; Kim, S. Modular neural networks prediction model based A 2/O process control system. *Int. J. Precis. Eng. Manuf.* **2012**, *13*, 905–913. [Google Scholar] [CrossRef]

[15] Hameed, M.; Sharqi, S.S.; Yaseen, Z.M.; Afan, H.A.; Hussain, A.; Elshafie, A. Application of artificial intelligence (AI) techniques in water quality index prediction: A case study in tropical region, Malaysia. *Neural Comput. Appl.* **2017**, *28*, 893–905. [Google Scholar] [CrossRef]

[16] Hu, Z.; Zhang, Y.; Zhao, Y.; Xie, M.; Zhong, J.; Tu, Z.; Liu, J. A water quality prediction method based on the deep LSTM network considering correlation in smart mariculture. *Sensors* **2019**, *19*, 1420. [Google Scholar] [CrossRef] [PubMed]

[17] Huang, M.; Zhang, T.; Ruan, J.; Chen, X. A New Efficient Hybrid Intelligent Model for Biodegradation Process of DMP with Fuzzy Wavelet Neural Networks. *Sci. Rep.* **2017**, *7*, 1–9. [**Google Scholar**] [**CrossRef**] [**PubMed**]

[18] Sakizadeh, M.; Malian, A.; Ahmadpour, E. Groundwater Quality Modeling with a Small Data Set. *Groundwater* **2016**, *54*, 115–120. [**Google Scholar**] [**CrossRef**]

[19] Gopi, A., et al. "Designing an Adversarial Model Against Reactive and Proactive Routing Protocols in MANETS: A Comparative Performance Study." International Journal of Electrical & Computer Engineering (2088-8708) 5.5 (2015).

[20] Kumar, S. Ashok, et al. "An Empirical Critique of On-Demand Routing Protocols against Rushing Attack in MANET." International Journal of Electrical and Computer Engineering5.5 (2015).

[21] A PEDA GOPI, V LAKSHMAN NARAYANA "Protected strength approach for image steganography" Signal, Image, Parole ARTICLE VOL 34/3-4-2017- pp.175-181 - doi:10.3166/ts.34.175-181.

[22] Kanumalli, S.S., Chinta, A.,ChandraMurty, P.S.R. (2019). Isolation of wormhole attackers in IOV using WPWP packet. Revue d'IntelligenceArtificielle, Vol. 33, No. 1, pp. 9-13. https://doi.org/10.18280/ria.330102

[23] Narayana, Vejendla Lakshman, et al. "Secure Data Uploading and Accessing Sensitive Data Using Time Level Locked Encryption to Provide an Efficient Cloud Framework." Ingénierie des Systèmesd'Information 25.4 (2020).

[24] Kotamraju, Siva Kumar, et al. "Implementation patterns of secured internet of things environment using advanced blockchain technologies." Materials Today: Proceedings (2021).

[25] Krishna, Komanduri Venkata Sesha Sai Rama, et al. "Classification of Glaucoma Optical Coherence Tomography (OCT) Images Based on Blood Vessel Identification Using CNN and Firefly Optimization." Traitement du Signal 38.1 (2021).

[26] Satya Sandeep Kanumalli, Anuradha Ch and Patanala Sri Rama Chandra Murty, "Secure V2V Communication in IOV using IBE and PKI based Hybrid Approach" International Journal of Advanced Computer Science and Applications(IJACSA), 11(1), 2020. http://dx.doi.org/10.14569/IJACSA.2020.0110157

[27] CHALLA, RAMAIAH, et al. "Advanced Patient's Medication Monitoring System with Ardunio UNO and NODEMCU." 2020 4th International Conference on Electronics, Communication and Aerospace Technology (ICECA). IEEE, 2020.

[28] Kanumalli, Satya Sandeep, Anuradha Ch, and Patanala Sri Rama Chandra Murty. "Advances in Modelling and Analysis B." Journal homepage: http://iieta. org/Journals/AMA/AMA_B 61.1 (2018): 5-8.

[29] Venkatramulu, S., et al. "Implementation of Grafana as open source visualization and query processing platform for data scientists and researchers." Materials Today: Proceedings (2021).

[30] Sandeep, Kanumalli Satya, Anuradha Chinta, and PatanalaMurty. "Isolation of Wormhole Attackers in IOV Using WPWP Packet." Rev. d'IntelligenceArtif. 33.1 (2019): 9-13.

[31] Gopi, ArepalliPeda, et al. "Classification of tweets data based on polarity using improved RBF kernel of SVM." International Journal of Information Technology (2020): 1-16.

[32] Narayana, Vejendla Lakshman, ArepalliPeda Gopi, and Kosaraju Chaitanya. "Avoiding Interoperability and Delay in Healthcare Monitoring System Using Block Chain Technology." Rev. d'IntelligenceArtif. 33.1 (2019): 45-48.

[33] Arepalli, Peda Gopi, et al. "Certified Node Frequency in Social Network Using Parallel Diffusion Methods." Ingénierie des Systèmesd'Information 24.1 (2019).

[34] Narayana, Vejendla Lakshman, ArepalliPeda Gopi, and R. S. M. Patibandla. "An Efficient Methodology for Avoiding Threats in Smart Homes with Low Power Consumption in IoT Environment

Using Blockchain Technology." Blockchain Applications in IoT Ecosystem. Springer, Cham, 2021. 239-256.

[35] Kotamraju, Siva Kumar, et al. "Implementation patterns of secured internet of things environment using advanced blockchain technologies." Materials Today: Proceedings (2021).

[36] Bharathi, C. R., et al. "A Node Authentication Model in Wireless Sensor Networks With Locked Cluster Generation." Design Methodologies and Tools for 5G Network Development and Application. IGI Global, 2021. 236-250.

[37] Vejendla, Lakshman Narayana, Alapati Naresh, and Peda Gopi Arepalli. "Traffic Analysis Using IoT for Improving Secured Communication." Innovations in the Industrial Internet of Things (IIoT) and Smart Factory. IGI Global, 2021. 106-116.

[38] Narayana, Vejendla Lakshman, ArepalliPeda Gopi, and Kosaraju Chaitanya. "Avoiding Interoperability and Delay in Healthcare Monitoring System Using Block Chain Technology Avoiding Interoperability and Delay in Healthcare Monitoring System Using Block Chain Technology."

[39] Yamparala, Rajesh, and Balamurugan Perumal. "EFFICIENT MALICIOUS NODE IDENTIFICATION METHOD FOR IMPROVING PACKET DELIVERY RATE IN MOBILE AD HOC NETWORKS WITH SECURED ROUTE." Journal of Critical Reviews 7.7 (2020): 1011-1017.

[40] Devi, S. Pramela, et al. "Likelihood based Node Fitness Evaluation Method for Data Authentication in MANET." International Journal of Advanced Science and Technology 29.3 (2020): 5835-5842.

[41] Yamparala, Rajesh, and Balamurugan Perumal. "Secure Data Transmission with Effective Routing Method Using Group Key Management Techniques-A Survey Secure Data Transmission with Effective Routing Method Using Group Key Management Techniques-A