Contents lists available at ScienceDirect

Journal of Hydrology

journal homepage: www.elsevier.com/locate/jhydrol

Research papers

Use long short-term memory to enhance Internet of Things for combined sewer overflow monitoring

Duo Zhang [a,*], Geir Lindholm [b], Harsha Ratnaweera [a]

[a] Faculty of Sciences and Technology, Norwegian University of Life Sciences, 1432 Ås, Norway
[b] Rosim AS, Brobekkveien 80, 0582 Oslo, Norway

ARTICLE INFO

Article history:
Received 29 July 2017
Received in revised form 8 November 2017
Accepted 10 November 2017
Available online 14 November 2017
This manuscript was handled by G. Syme, Editor-in-Chief.

Keywords:
Combined sewer overflow
Deep learning
Long short-term memory
Gated recurrent unit
Internet of Things

ABSTRACT

Combined sewer overflow causes severe water pollution, urban flooding and reduced treatment plant efficiency. Understanding the behavior of CSO structures is vital for urban flooding prevention and overflow control. Neural networks have been extensively applied in water resource related fields. In this study, we collect data from an Internet of Things monitoring CSO structure and build different neural network models for simulating and predicting the water level of the CSO structure. Through a comparison of four different neural networks, namely multilayer perceptron (MLP), wavelet neural network (WNN), long short-term memory (LSTM) and gated recurrent unit (GRU), the LSTM and GRU present superior capabilities for multi-step-ahead time series prediction. Furthermore, GRU achieves prediction performances similar to LSTM with a quicker learning curve.

1. Introduction

The sewer system is one of the most important infrastructures in modern cities. Although separate sewer systems have been introduced, decades-old combined sewer system still continues to service many developed cities. Under normal circumstances, these systems transport all wastewater to the wastewater treatment plant. During heavy rainfall events, when wastewater volume exceeds the capacity of the sewer system, the combined sewer system is designed to overflow and discharge wastewater directly to water bodies. This phenomenon is called Combined Sewer Overflow (CSO). Climate change and rapid urbanization in metropolitan areas have resulted in an increase in the CSO over the last decades. Decreasing the influence of overflows is an important part of reducing pollution in water bodies (Garofalo et al., 2017; Autixier et al., 2014; Lucas and Sample 2015). Several cities have employed Internet of Things (IoT) to monitor the performance of sewer systems and to provide useful data to managers and engineers (Montserrat et al., 2015). The basic idea of IoT is to let 'things'-such as sensors, actuators, mobile or desktop devices, etc. -be able to interact and cooperate with each other through wireless communication protocols (Giusto, 2010). IoT is a new catalyst for the data explosion. The world will have 50 billion IoT devices by 2020, these devices will generate massive data, to extract insight from data collected by IoT and convey the extracted insight to stakeholders, the analysis and application of IoT data are as important as data collection. For example, for the CSO structures, in addition to properly monitoring, it is also imperative to construct a model to predict the CSO events by utilizing data from the realm of IoT. The model is expected to provide sufficient response time for making decisions about CSO control, protect downstream hydraulic infrastructures during extreme rainfall events and mitigate the impact of CSO on the receiving waters (Garofalo et al., 2017; Chang et al., 2014; Mounce et al., 2014; Darsono and Labadie, 2007; Grum et al., 2011).

Most IoT data collected by sensors have a temporal dimension and can be modeled as a time series. Characterized by high complexity, dynamism, and non-stationarity, time series forecasting has always presented a challenge to hydrologists (Nourani et al., 2014). Recent years have seen a significant rise in the number of machine learning approaches applied to hydrologic time series forecasting. Among numerous machine learning algorithms such as support vector machine, k-nearest neighbors or linear discriminant analysis, Artificial Neural Network (ANN) had shown superior performance for IoT time series data (Alam et al., 2016). As the most traditional ANN, multilayer perceptron (MLP) is a very popular tool for handling hydrologic time series problems in previous researches. A few studies have explored using MLP and data

* Corresponding author.
E-mail addresses: Duo.Zhang@nmbu.no (D. Zhang), geir@rosim.no (G. Lindholm), Harsha.Ratnaweera@nmbu.no (H. Ratnaweera).

collected by IoT to predict CSO events (Kurth et al., 2008; Mounce et al., 2014). Another common practice in solving hydrologic time series problems is couple Wavelet Transforms (WT) with different ANN such as the MLP. WT is a kind of technique that can illustrate how the frequency content of a signal changes over time. WT relies on the mother wavelet, which can be changed according to the shape and compactness of the signal (Barzegar et al., 2017). ANN that use mother wavelet as activation functions instead of traditionally used sigmoid function are called Wavelet Neural Network (WNN) (Wang and Ding, 2003). WNN had shown have better fitting in estimating hydrologic time series data (Nourani et al., 2014). The advances of WNN had led to an increase in hydrological studies for different applications such as hydro-climatology time series prediction (Alexandridis and Zapranis, 2013), reservoir inflow modeling (Abghari et al., 2009; Chen et al., 2006) and evaporation prediction (Abghari et al., 2012).

A state-of-the-art branch of ANN is deep learning (Hinton et al., 2006). ANN have been less active during a period called Artificial Intelligence (AI) winter (Marçais and de Dreuzy, 2017). In recent years, renewed interest in ANN surged, in part due to exposure in popular media when the computer program (Google DeepMind's AlphaGo) defeated Go game world champion (Silver et al., 2016). The game changer behind the revival of ANN is deep learning. Deep learning is a topic that is making big waves now, alongside AlphaGo, a fascinating application of deep learning is the latest Google Neural Machine Translation (GNMT) system released in 2016. The GNMT system produces translation quality that is vastly improved compared to previous systems, even for a notoriously difficult language pair: Chinese to English (Google, 2016). The technology utilized by the GNMT system is a kind of Recurrent Neural Network (RNN), Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997).

RNN is a kind of advanced ANN that involves feedback connections in the architecture. Unlike traditional ANN that all the inputs and outputs are independent of each other, the output of RNN depends on previous computations as well as calculations of the current time step. One way to think about RNN is that they have a "memory" that captures information from previous computations. RNN are considered very effective in modeling complex hydrologic time series. However, due to the difficulty of learning long-range dependencies, the training of RNN can be extremely challenging. This problem is commonly known as the vanishing/-exploding gradient problem (Hochreiter and Schmidhuber, 1997). Therefore, certain types of RNN, such as LSTM, were designed to solve these problems. LSTM is today's dominant technology in the field of natural language processing. When processing natural language, the model has to consider not only the current word but also other adjoining words in the sentence or even paragraph. Data with this kind of context information called sequential data. Time series data are the most popular form of sequential data. Stimulated by the success of LSTM on machine translation, a few studies have explored the power of LSTM on time series prediction and obtained promising results (Zaytar and Amrani, 2016; Ma et al., 2015; Xingjian et al., 2015).

One drawback of LSTM is its complexity. Therefore, simplifying LSTM, has become a highly researched topic in the field of computer science. The Gated Recurrent Unit (GRU) (Cho et al., 2014) first proposed in 2014 is one of the most successful LSTM variants. Chung et al., (2014) evaluated the performance of LSTM and GRU on the tasks of polyphonic music modeling and speech signal modeling. The evaluation revealed the GRU to be comparable to LSTM, and better performing than more traditional RNN. To the best of the author's knowledge, no prior studies have employed GRU to analyze time series data. Given the success of the GRU in modeling sequential data, such as speech synthesis (Wu & King, 2016), sentiment classification (Tang et al., 2015), targeted sentiment analysis (Zhang et al., 2016), etc., the effectiveness of GRU on the prediction of hydrologic time series has to be investigated.

The objectives of this research are to investigate the performance of LSTM and GRU on predicting the multi-step ahead hydrologic time series data collected by the IoT, and to compare with the performance of traditional methods such as MLP and WNN. The remainder of this paper is organized as follows: a general description about the study area and different algorithms are provided in the first section. The simulation results are presented in section two, where the prediction efficiencies of the studied algorithms are compared. Conclusions and envisioned future developments are discussed at the end of this paper.

2. Method and data

2.1. Case study area

Drammen is a coastal city with approximately 150, 000 inhabitants located in the southeast of Norway. It is the fourth largest city in Norway, and the capital of Buskerud County. The catchment of Drammen's sewer system is around 15 km^2 and the total length of the sewer system is approximately 500 km. Fig. 1 is an overview of Drammen. In Fig. 1, the rain gauges are denoted by squares and CSO structures are denoted by triangles. The traditional city center distributes along the Drammen Fjord, which flows through Drammen. The sewer system in the city center mainly consists of the combined sewer system. During heavy rainfall events, the combined sewer system in the city center discharges overflows directly into the Drammen Fjord though CSO structures, cause heavy pollution. In order to mitigate pollutions from the combined sewer system, the government of Drammen initialized the Regnbyge 3 M project. The ultimate goal of the Regnbyge 3 M project is integrates intelligent monitoring, modeling and control solutions, manage the sewer system and water bodies in a holistic way.

This study focuses on the CSO structure of Dr_Hansteensgate, which located adjacent to important infrastructures such as the train station, shopping center, and the stadium. The risks of overflow and consequent pollution risks may be decreased if accurate hydrological time series prediction can be provided. Therefore, to test the performance of different ANN models against this objective, Dr_Hansteensgate CSO (highlighted by a circle in Fig. 1) was selected for consideration.

2.2. The Regnbyge.no IoT

In order to monitor the hydraulic behavior of CSO structures, data must be provided for further analysis and model construction. In the first phase of the Regnbyge 3 M project, Rosim AS, Norway developed an IoT, Regnbyge.no, to monitor the sewer system. Typical IoT architecture consists of three layers: sensing layer, network layer, and application layer (Atzori et al., 2010). The sensing layer includes devices that are deployed in the field, such as sensors, actuators, and wireless transmitters. The major function of the sensing layer is to collect data. The network layer provides a bridge between the sensing layer and the application layer. In the network layer, data sensed by sensors transmit through wireless communication protocols to the application layer. On the top of the IoT architecture is the application layer, it enabling users, algorithms, or models to interact with devices in the sensing layer.

The sensing layer of the Regnbyge.no IoT consists of ultrasonic water level sensors produced by NIVUS GmbH, Germany and rain gauges. The measurements collected by the water level sensors and rain gauges transmit using wireless telemetry to a remote server located inside Rosim AS. A spatial database is designed to manage the transmitted data, simplifying the process for searching,

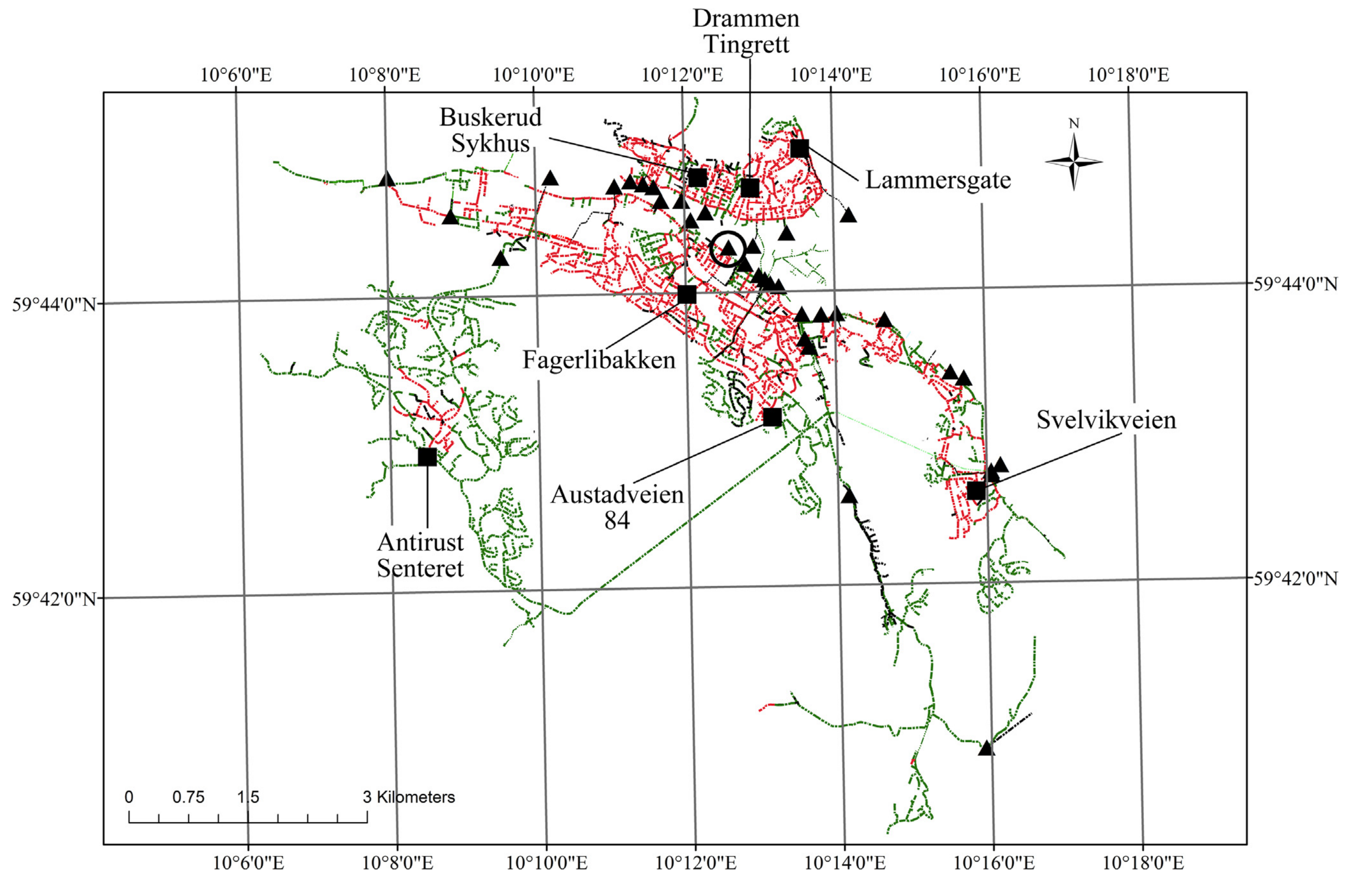

Fig. 1. Overview of Drammen sewer system, Norway, squares denote rain gauges and triangles denote CSO structures.

editing, and giving real-time information in a user-friendly way. Once data is stored in the database, information can be shared via the web and accessed through a desktop or mobile devices. Furthermore, the data in the database is visualized using web-based geographic information system (Web-GIS). Currently, the Web-GIS only provides an interface for displaying real-time and historical data of the water level and rainfall. In the next phase of the Regnbyge 3 M project, forecasting models and actuators will be implemented in the Regnbyge.no IoT. The forecasting model will predict future water levels, according to the predicted water level, managers of the sewer system can operate actuators timely using control commands sent from the server, thus guide the behavior of CSO structures.

Fig. 2 displays major components of the Regnbyge.no IoT platform, Fig. 2 (a) shows an ultrasonic water level sensor installed on the top of a CSO structure, Fig. 2 (b) illustrates how engineers embed a sensor's antenna in the road surface, Fig. 2 (c) is the installation of a transmitter on a lamp post. Fig. 2 (d) demonstrates the user interface of the Regnbyge.no IoT.

2.3. ANN

2.3.1. MLP

MLP is one of the most popular ANN, which is usually comprised of input layer, hidden layer, and output layer. There are some neurons in each layer and different layers are connected by weights and bias. Fig. 3 shows the architecture of a three-layered MLP. In Fig. 3, the circles denote neurons and lines between circles denote weights. The MLP first computes the weighted sum of the inputs, which can be mathematically represented as:

$$s = \sum_{i=1}^{n} w_i x_i + b \tag{1}$$

where w_i represents the weights, x_i is the inputs, b is the bias. Afterwards, the computed weighted sum s is fed into the neuron. The neuron consists of an activation function. There are various functions that can be used and the most classical one is the sigmoid function. The sigmoid function is defined as:

$$f(s) = \frac{1}{1+e^{-s}} \tag{2}$$

When training the MLP, the ultimate goal is to minimize the cost function. The cost function can be defined as:

$$C = \frac{1}{2}(f(s) - f(s)_{observed})^2 \tag{3}$$

where C is the cost of the cost function, $f(s)$ is the predicted output from neuron and $f(s)_{observed}$ is the observed true value.

Back propagation (BP) is the most commonly used training algorithms. BP uses the chain rule of differentiation to calculate the partial derivative or gradient of the cost corresponding to the weights. For a single training example of a neuron, the gradient of cost C corresponding to a weight w_i can be represented as:

$$\frac{\partial C}{\partial w_i} = \frac{\partial C}{\partial f(s)} \frac{\partial f(s)}{\partial s} \frac{\partial s}{\partial w_i} \tag{4}$$

The partial derivative of the cost function corresponding to activation function is:

$$\frac{\partial C}{\partial f(s)} = f(s) - f(s)_{observed} \tag{5}$$

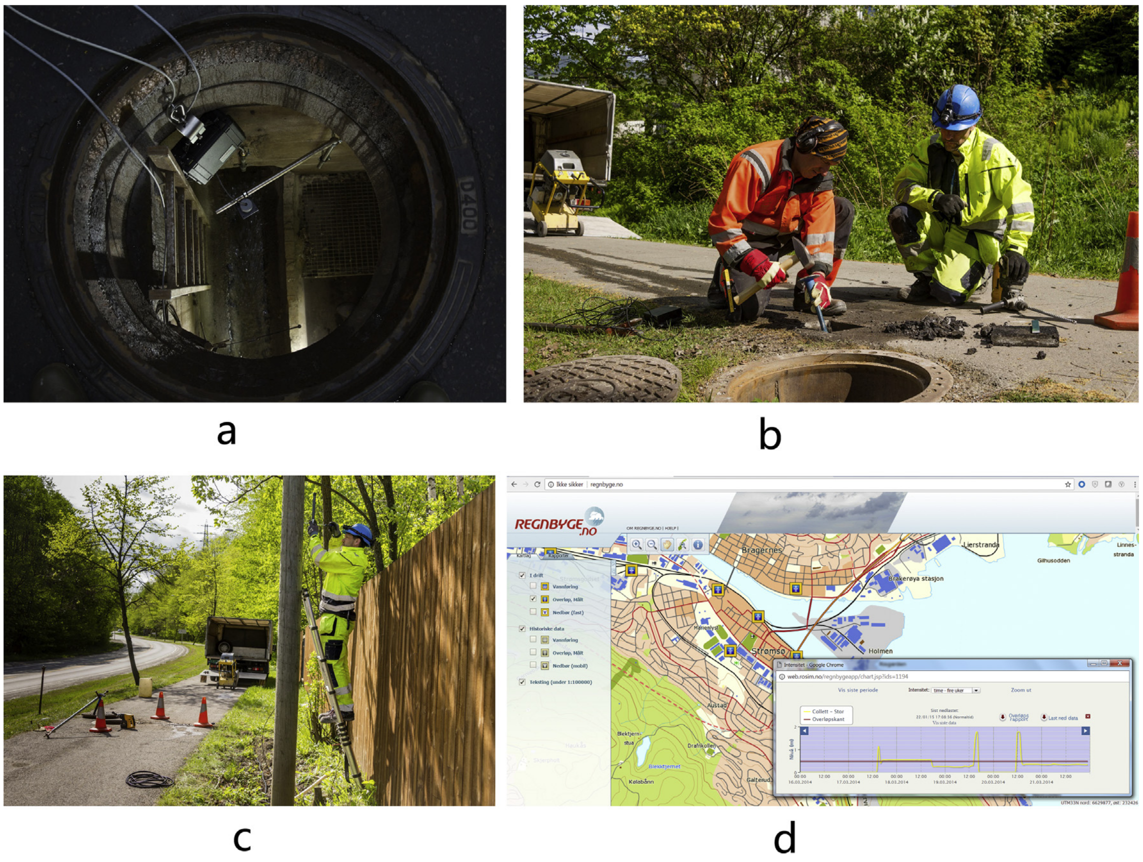

Fig. 2. Major components of the Regnbyge.no IoT; (a) an example of field-deployed water level sensor; (b) engineers are embedding antenna in the road surface; (c) engineer are installing wireless transmitter on a lamp post; (d) the user interface of the Regnbyge.no IoT.

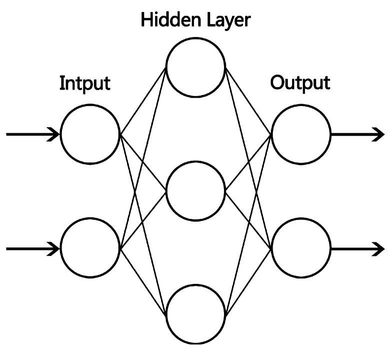

Fig. 3. Architecture of a three-layered MLP.

The partial derivative of the sigmoid activation function corresponding to the weighted sum input of the neuron s is:

$$\frac{\partial f(s)}{\partial s} = f(s)(1 - f(s)) \tag{6}$$

BP calculates the error contribution of each neuron, the cost function, i.e. error between predicted and observed value can be minimized through adjusting the weights of each neuron.

2.3.2. WNN

WT is a multi-resolution analysis in time and frequency domain. For example, for time series $f(t)$, WT is the sum over all time of $f(t)$ multiplied by the scale:

$$W_f(a,b) = |a|^{-1/2}\int_{-\infty}^{+\infty} f(t)\bar{\psi}\left(\frac{t-b}{a}\right)dt \tag{7}$$

where $\psi(\mathrm{t})$ is the mother wavelet, a is the parameter defining the window of analysis, b is the parameter localizing the wavelet function in the time domain, and $f(t)$ is the complex conjugate of the basic wavelet function. $W_f(a,b)$ represents the correlation between the signal $f(t)$ and a scaled version of the function $\psi(\mathrm{t})$.

Similar to MLP, the WNN also consists of an input layer, hidden layer, and output layer. The difference of WNN is its use mother wavelet as an activation function. The neuron of WNN often referred as wavelons, which transfer the input variables to the dilated and translated version of the mother wavelet. The mother wavelet used in this study is the Morlet mother wavelet. Morlet mother wavelet is a complex exponential with a Gaussian envelope that ensures localization. Fig. 4 shows the Morlet mother wavelet. The mathematical representation of the Morlet wavelet activation function is given as:

$$f(x) = \cos(1.75x) * e^{-x^2/2} \tag{8}$$

When training the WNN through BP, the partial derivation of the Morlet wavelet activation function can be written as:

$$\frac{\partial f(x)}{\partial x} = -1.75 * \sin(1.75x) * e^{-x^2/2} - x * \cos(1.75x) * e^{-x^2/2} \tag{9}$$

According to several recent studies (Majeed et al., 2017; Chitsaz et al., 2015; Xu & Liu, 2013; Alexandridis and Zapranis, 2013), Morlet wavelet ANN is a good alternative in cases with difficult nonlinear systems. The Morlet Wavelet ANN is reliable and robust, it can solve the problem with almost the same accuracy as other WNNs with the Mexican hat wavelet or Haar wavelet based activation function.

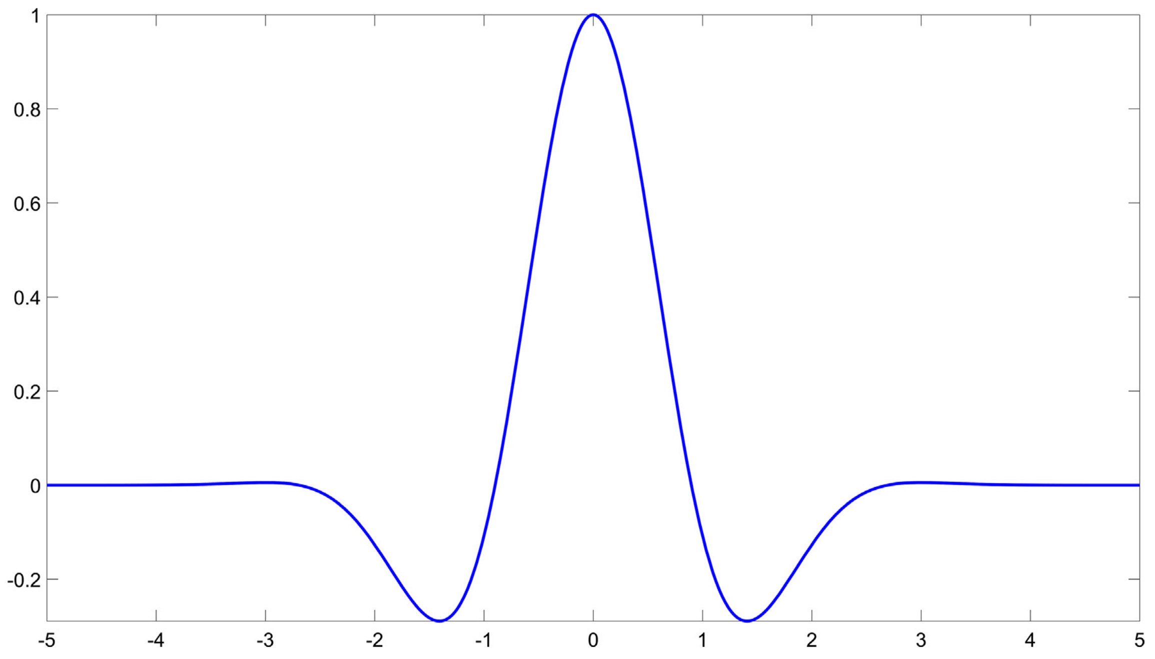

Fig. 4. The Morlet mother wavelet.

2.3.3. LSTM

Different from traditional ANN, RNN (Elman, 1990) takes the state of the hidden neuron at the previous time steps as an additional input for the next time step (Ishak et al., 2003).

As shown in Fig. 5, the neuron state of the present time step h_{t+1} is calculated by the equation:

$$h_{t+1} = w_{h+1}h_t + w_{i+1}x_{t+1} + b \tag{10}$$

Similar to h_{t+1}, the h_t is calculated by:

$$h_t = w_h h_{t-1} + w_i x_t + b \tag{11}$$

where h_{t+1}, h_t, and h_{t-1} are states of the hidden neuron at the time step t + 1, t, and t-1 respectively, w_{i+1}, w_i and w_{h+1}, w_h are weights between input values and hidden neurons (input weights), and between hidden neurons (hidden weights) respectively.

The training of RNN relies on an extended version of BP called backpropagation through time (BPTT). Unlike BP, BPTT not only calculates the gradient of the cost corresponding to the input weights but also the gradient of the cost corresponding to the hidden weights of the previous time steps. As shown in Fig. 5, the dashed line and arrow is the direction of gradient calculation of BPTT, BPTT first calculates the gradient of output at time step t + 1 (O_{t+1} in Fig. 5) corresponding to the state of hidden neuron at time step t + 1 (h_{t+1} in Fig. 5), i.e. $\frac{\partial O_{t+1}}{\partial h_{t+1}}$ in Fig. 5. It then calculates the gradient of the state of hidden neuron at time step t + 1 corresponding to the state of hidden neuron at previous time step ($\frac{\partial h_{t+1}}{\partial h_t}$ in Fig. 5), and backpropagation to earlier neurons step by step in this way. When using the BPTT method, with gradient calculation, the error of partial derivative accumulates through time steps. Hence, it will be extremely hard to learn and tune the parameters of the earlier neurons and learn long-term dependencies. Because the gradient going through the network either gets very small and vanish, or gets very large and explode, this problem is commonly known as the vanishing/exploding gradient problem. In recent years, modern RNN combat vanishing/exploding gradient, such as LSTM, had been proposed (Lipton et al., 2015; Gers 2001; Gers et al., 2000).

The LSTM replaced the ordinary neuron in the hidden layer with a memory cell and three gates: the input gate, forget gate and output gate. LSTM can selectively update the memory cell state based on the new input, forget irrelevant content, or selectively output part of the memory cell state as the new hidden neuron state according to the state of the input, forget and output gate respectively, similar to data in a computer's memory. In this way, the LSTM is able to learn long time span time series.

Fig. 6 shows the neuron in the hidden layer of LSTM, i, f and o represent the input, forget and output gate respectively. c and $\bar{c}$ denote the memory cell and the new memory cell. The principal

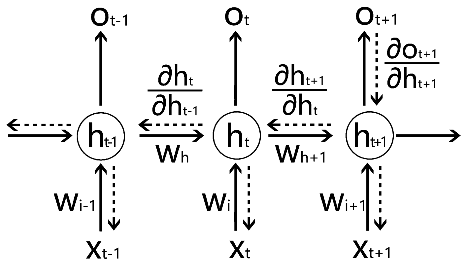

Fig. 5. The architecture of the RNN and schematic diagram of BPTT. The solid lines indicate how RNN inherits previous hidden neuron states. The dashed lines indicate the direction of BPTT.

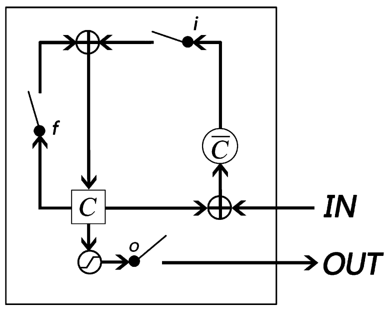

Fig. 6. Neuron in the hidden layer of LSTM.

of the memory cell in LSTM can be mathematically represented by the following equations:

Input gate:

$$i_t = \sigma_g(W_i * x_t + U_i * h_{t-1} + V_i{}^\circ c_{t-1} + b_i) \quad (12)$$

Forget gate

$$f_t = \sigma_g(W_f * x_t + U_f * h_{t-1} + V_f{}^\circ c_{t-1} + b_f) \quad (13)$$

Output gate:

$$o_t = \sigma_g(W_o * x_t + U_o * h_{t-1} + V_o{}^\circ c_{t-1} + b_o) \quad (14)$$

Cell state:

$$c_t = f_t{}^\circ c_{t-1} + i_t{}^\circ c_t \quad (15)$$

$$c_t = \sigma_c(W_c * x_t + U_c * h_{t-1} + b_c) \quad (16)$$

Output vector:

$$h_t = o_t{}^\circ \sigma_h(c_t) \quad (17)$$

where x_t is the input vector. W, U, V, and b are parameters for weights and bias. $\circ$ represents the scalar product of two vectors, σ_g is the sigmoid function, σ_h and σ_c are hyperbolic tangent function, for a given input z, the output of hyperbolic tangent function is:

$$f(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \quad (18)$$

2.3.4. GRU

The concept of a GRU is quite similar to LSTM. The GRU also use gates to modulate the flow of information inside the neuron in the hidden layer. The difference is the GRU has two gates (reset gate and an update gate), while LSTM has three gates. GRU combines the input and forget gates into an update gate to balance between previous activation and the candidate activation. The activation of h at time t depends on h at the previous time and the candidate h (the $\bar{h}$ in Fig. 7). The update gate z decides how much of the previous memory to keep around. The GRU unit forgets the previously computed state when the reset gate r is off. (Cho et al., 2014; Chung et al., 2014).

The GRU is formulated as:

$$z_t = \sigma_g(W_z * x_t + U_z * h_{t-1} + b_z) \quad (19)$$

$$r_t = \sigma_g(W_r * x_t + U_r * h_{t-1} + b_r) \quad (20)$$

$$h_t = z_t{}^\circ h_{t-1} + (1 - z_t)^\circ \bar{h}_t \quad (21)$$

$$\bar{h}_t = \sigma_h(W_h * x_t + U_t * (r_t{}^\circ h_{t-1}) + b_h) \quad (22)$$

where x_t is the input vector, h_t is the output vector, z_t is the update gate vector, h_t is the reset gate vector. W, U and b are parameters for weights and bias. $\circ$ represents the scalar product of two vectors, $\sigma(.)$ is the sigmoid function. σ_g represent the sigmoid activation function, σ_h represent the hyperbolic tangent activation function.

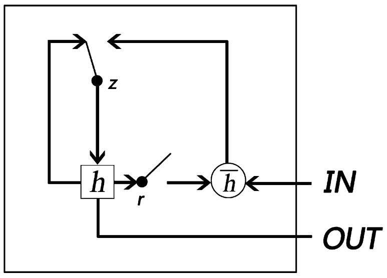

Fig. 7. Neuron in the hidden layer of GRU.

2.3.5. Model implementation

In this paper, the MLP and WNN were implemented using Matlab, R2016a. The LSTM and GRU were programmed using Keras (Chollet, 2015). Keras is a high-level deep learning library. It is written in Python and runs on top of either TensorFlow (Abadi et al., 2016) or Theano backend. TensorFlow backend was employed in this study. TensorFlow is an open-source software for deep learning, released by Google in 2015. Besides Keras and TensorFlow, other Python libraries such as matplotlib, numpy, pandas, sklearn were also used.

2.4. Model performance criteria

The performance of the developed models was evaluated by three criteria, the root mean square error (RMSE), Nash-Sutcliffe Efficiency (NSE) and the coefficient of determination (R^2).

The calculation of RMSE as shown below:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(Y_i^{obs} - Y_i^{sim})^2}{n}} \quad (23)$$

RMSE value of 0 means a perfect fit between observed and predicted values.

NSE is a parameter that determines the relative importance of residual variance (noise) compare to the variance in the measured data (information). The NSE is calculated by the following equation:

$$NSE = 1 - \left[\frac{\sum_{i=1}^{n}(Y_i^{obs} - Y_i^{sim})^2}{\sum_{i=1}^{n}(Y_i^{obs} - Y^{mean})^2}\right] \quad (24)$$

NSE varies from $-\infty$ to 1, NSE = 1 indicates a perfect correlation between simulated and observed data, values between 0.0 and 1.0 is generally acceptable.

The equation for the coefficient of determination (R^2) is:

$$R^2 = \left[\frac{\left(\sum_{i=1}^{n}(Y_i^{sim} - Y_{sim}^{mean})(Y_i^{obs} - Y^{mean})\right)^2}{\sum_{i=1}^{n}(Y_i^{sim} - Y_{sim}^{mean})^2 \sum_{i=1}^{n}(Y_i^{obs} - Y^{mean})^2}\right] \quad (25)$$

The R^2 values range between 0 and 1, R^2 value of 1 indicates a perfect correlation.

In above-listed equations: nY_i^{obs} = the i-th observed data.Y_i^{sim} = the i-th simulated data.Y^{mean} = mean value of observed data.Y_{sim}^{mean} = mean value of simulated data.

n = number of data.

3. Results and discussion

3.1. Datasets for ANN training

According to CSO water level data of Dr_Hansteensgate and rainfall intensity data from the Fagerlibakken rain gauge recorded by the Regnbyge.no IoT during 2014, 26 CSO events were selected for constructing ANN models in this study. During a CSO event, the water level in the CSO structures rises due to rainfall in the associated catchment. The Fagerlibakken rain gauge was selected using the cross correlation function XCORR function in Matlab, R2016a. The collected time series data have 2352 records with a temporal resolution of 10 min. In consideration of maintaining similar statistical characteristics, 20 events were selected as training sets and the remaining 6 events were allocated as testing sets. The difference between training and testing is regularization mechanisms, which is used to prevent overfitting, are turned off during the testing mode. Dropout and L2 weight penalty method were used as regularization methods in this study. Dropout is a technique that randomly drops selected neurons and set their associated weights to zero during training. L2 weight penalty method adds an extra squared term to the cost function to constrain the weights. It keeps the weights small unless they have big error derivatives. The summary statistics for water levels and rainfall are presented in Table 1.

3.2. Training of ANN

As the primary objective of this paper, to anticipate the occurrence of CSO events and develop early warning and supply references for necessary CSO reduction measures, different ANN models were developed. The purposes of the developed models are to predict the one- to six-step ahead water level in the CSO structure based on data collected by IoT. To make a fair comparison, the input of the developed ANN models remain the same i.e. data from the previous six steps collected by IoT. After confirming input data and prediction goals, the training of ANNs was implemented through trial-and-error procedures. While the best-performed MLP has one hidden layer with fifteen hidden neurons, the WNN used in this study has one hidden layer with eighteen wavelons. The optimal structure of LSTM and GRU have two hidden layers with ten hidden neurons in each layer. In this study, the Levenberg–Marquardt algorithm was used for training of MLP. The WNN was trained using the standard BP method. For LSTM and GRU, the ADAM (Kingma and Ba, 2014) method was selected as the optimization algorithm.

3.3. Performance of ANN

All the four ANN models performed rather consistently in the training stages while showing different performance during the

Table 1
Summary statistics for water levels (m) and the rainfalls (mm/s).

Event	Model stage	Max level (m)	Average level (m)	Standard deviation level	Max rainfall (mm/s)	Average rainfall (mm/s)	Standard deviation rainfall
1	Training	1.19	0.73	0.31	4.76	1.92	1.22
2		1.18	0.81	0.32	0.44	0.17	0.11
3		0.54	0.30	0.17	0.14	0.07	0.03
4		1.24	0.76	0.30	25.00	3.66	4.45
5		0.50	0.38	0.12	4.55	1.16	0.93
6		0.83	0.61	0.20	5.56	2.00	1.44
7		1.27	0.72	0.34	33.33	11.00	7.32
8		1.29	0.80	0.33	25.00	8.48	6.36
9		1.25	0.87	0.28	50.00	10.38	10.65
10		0.67	0.50	0.17	2.27	0.94	0.55
11		1.11	0.68	0.28	2.22	1.07	0.50
12		1.20	0.76	0.32	9.09	2.13	1.81
13		1.20	0.90	0.34	10.00	1.90	1.79
14		1.20	0.71	0.33	11.11	1.93	1.55
15		1.17	0.93	0.27	2.22	0.88	0.43
16		1.07	0.59	0.29	2.04	0.73	0.48
17		1.17	0.70	0.30	2.86	0.60	0.37
18		0.62	0.52	0.14	0.97	0.52	0.25
19		0.81	0.66	0.18	2.04	0.69	0.35
20		1.20	0.86	0.35	4.00	1.48	0.74
21	Testing	1.18	0.57	0.11	1.31	0.58	0.37
22		1.22	0.83	0.31	25.00	6.73	6.55
23		1.21	0.89	0.31	9.09	2.96	1.60
24		1.18	0.71	0.30	3.33	1.37	0.72
25		0.81	0.48	0.22	1.41	0.39	0.26
26		1.25	0.81	0.31	14.29	4.04	2.98

Table 2
Performance of different models for one- to six-step ahead CSO water level prediction within the testing stage.

Time steps		GRU	LSTM	WNN	MLP
6	RMSE	0.1536	0.1529	0.1694	0.2233
	R^2	0.7960	0.7987	0.7525	0.7660
	NSE	0.7807	0.7828	0.7332	0.5133
5	RMSE	0.1353	0.1398	0.1402	0.2115
	R^2	0.8313	0.8336	0.8041	0.8120
	NSE	0.8138	0.8186	0.8075	0.5616
4	RMSE	0.1188	0.1263	0.1270	0.2011
	R^2	0.8786	0.8653	0.8768	0.8694
	NSE	0.8693	0.8522	0.8506	0.6138
3	RMSE	0.1004	0.1130	0.1030	0.1780
	R^2	0.9133	0.8991	0.9334	0.8861
	NSE	0.8717	0.8627	0.8445	0.7061
2	RMSE	0.0820	0.1021	0.0887	0.1656
	R^2	0.9542	0.9357	0.9397	0.8967
	NSE	0.8967	0.8816	0.8432	0.7648
1	RMSE	0.0641	0.0720	0.0654	0.0918
	R^2	0.9892	0.9765	0.9711	0.9531
	NSE	0.9267	0.9266	0.9319	0.9197

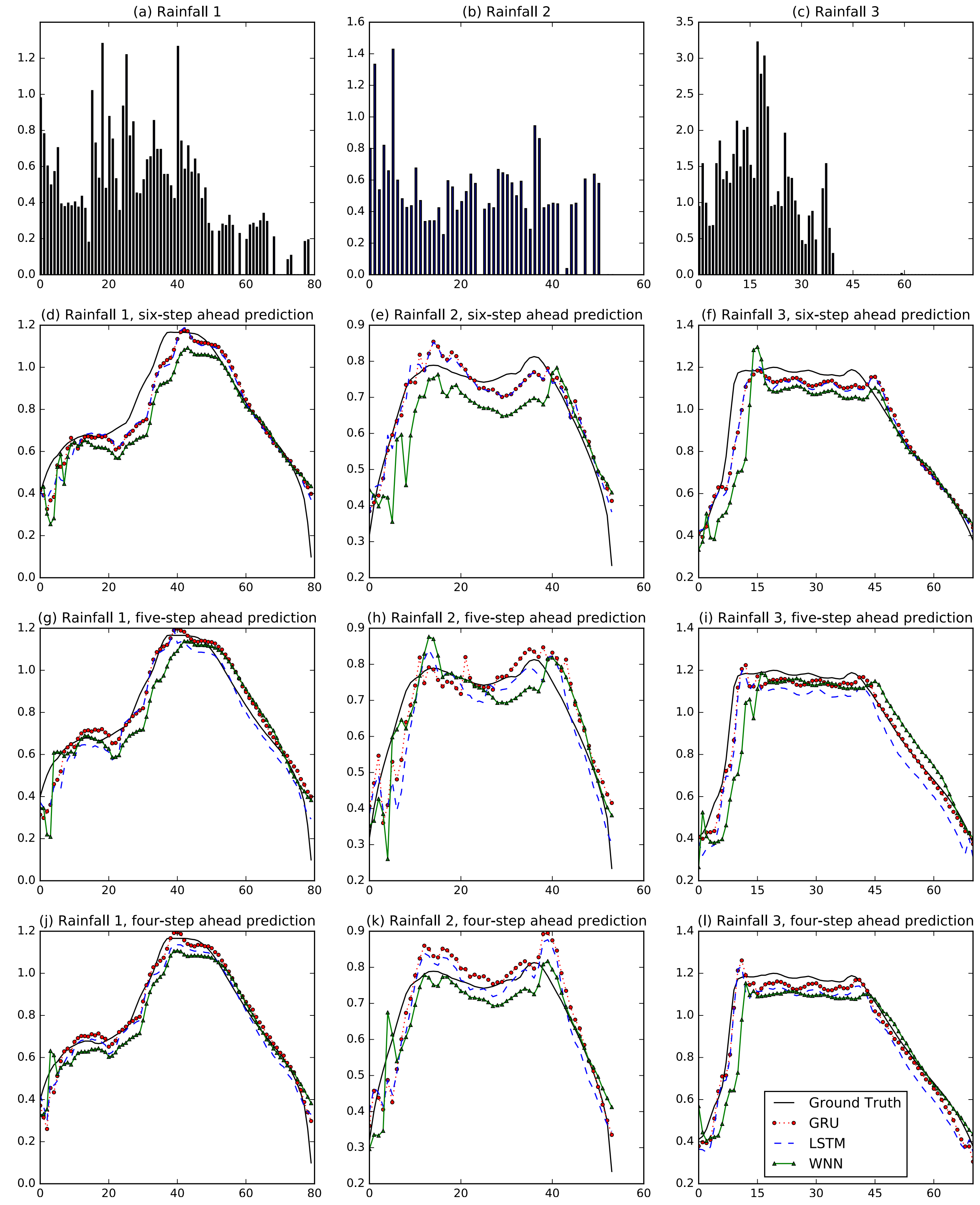

Fig. 8. Four- to six-step ahead forecasts of selected rainfall events with respect to the LSTM, GRU, and WNN.

testing stages. The results for the one- to six-step ahead CSO water level forecasting within the testing stages are provided in Table 2.

As shown in Table 2, the model prediction accuracy reduced as the prediction steps increased. For the one-step-ahead prediction, all the ANN models showed satisfying results, with low RMSE value, high NSE value, and high R^2 value. When turning the prediction time step to two, the performance of MLP deteriorated immediately. Compared to the one-step-ahead prediction, the RMSE

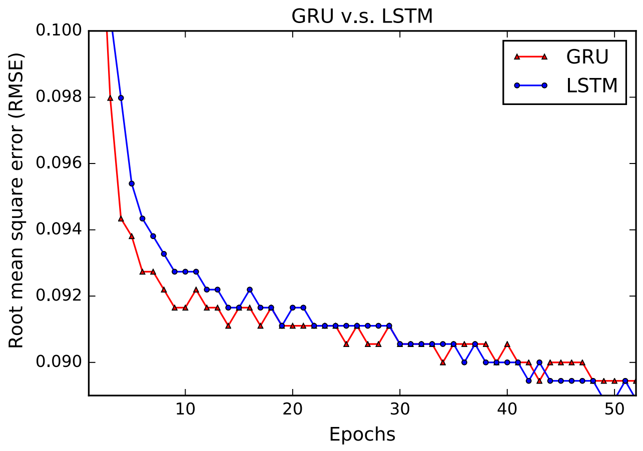

Fig. 9. Learning curves of LSTM and GRU.

value of MLP for the two-step ahead prediction almost doubled, increasing from 0.0918 to 0.1656, the R^2 value and NSE values reduced from 0.9531 and 0.9197 to 0.8967 and 0.7648 respectively. With longer time steps, the MLP performance decreased continually. It reveals that as a static feedforward ANN, the MLP is not suitable for multi-step-ahead forecasting. The WNN presented better performance than MLP. For three-step ahead prediction, the WNN even got the highest R^2 value, 0.9334. However, this fact alone doesn't prove that the performance of WNN is better than LSTM and GRU when comprehensively considering all the performance criteria. Because the remaining values of RMSE, NSE, and R^2 of WNN are very unstable across the different time steps compared to LSTM and GRU.

When the prediction time step exceeds four, LSTM and GRU outperform MLP and WNN in terms of all the performance criteria. LSTM and GRU even produce low RMSE value (around 0.15), high NSE value (around 0.8) and high R^2 value (around 0.8) as the prediction time step reaches six. Besides, LSTM shows a slightly better performance than GRU, but the difference is marginal.

To further illustrate the performance of LSTM, GRU, and WNN in a more intuitive way, hydrographs of the observed versus four-, five- and six-step ahead predicted water level for three rainfall events in the testing sets were drawn in Fig. 8.

For six-step ahead prediction (Fig. 8(d)–(f)), when the CSO water level starts to increase, the LSTM and GRU react quicker than WNN, whereas the WNN presents the time-lag problem. Moreover, it obviously shows that LSTM and GRU better predict the peak water level, while the WNN either underestimates or overestimates the peak value. The inconsistency of WNN indicates its poor performance compared to LSTM and GRU. Similar observations can also be found in the five- and four-step ahead prediction (Fig. 8(g)–(i)). Only in Fig. 8(k) does the WNN look more stable than LSTM and GRU.

As we can see from Fig. 8, in general, the LSTM and GRU can better capture the sudden change of water levels compared to WNN, and significantly alleviate the time-lag problem at the peak value. In terms of peak water level, the WNN either underestimates or undergoes strong fluctuations at the peak value. This is probably due to the insufficient learning capability of WNN. Predictions by both LSTM and GRU showed very similar behavior, especially for the six-step ahead prediction. However, the GRU has several advantages compared to LSTM.

Fig. 9 displays the learning curves of LSTM and GRU. In Fig. 9, the x-axis is epochs (one full training cycle on the training samples) and the y-axis is the RMSE value. The GRU makes faster progress than LSTM during training, the RMSE value of the GRU is lower than LSTM during the first 30 epochs, but the final RMSE value of the GRU is slightly higher than LSTM. The quick learning curve of the GRU is due to its simplified architecture. For large-scale data, the model training can consume a significant amount of CPU time. Hence, the GRU's quicker convergence and comparable performance is a good alternative to LSTM. On the other hand, the GRU has fewer parameters. To make predictions in IoT, the controller has to calculate the future water level based on real-time water level data read from the sensor and the parameters of the prediction model. With fewer parameters, the hardware implementation and the real-time computing process of controllers could be simplified and accelerated.

4. Conclusion

In order to reduce pollutions caused by CSO, it is essential to predict the CSO event in advance. The prediction could enable better decision-making, greater automation, and higher efficiencies. Through a case study in Drammen, Norway, the present work compared the performance of four ANN on the multi-step-ahead prediction of the CSO water level collected by IoT. The preliminary conclusions summarized below.

The MLP only presented accurate predictions for the one- and two-step ahead prediction, but not for multi-step-ahead predictions. WNN can improve the multi-step ahead predictions. However, when checking the hydrography of observed and predicted water level, the WNN presented several problems such as time delay and strong fluctuation. With the gating mechanisms, the LSTM and GRU outperformed WNN and MLP. Both LSTM and GRU achieved satisfying results for multi-step-ahead prediction. The behavior of LSTM and GRU is quite similar. LSTM had marginally better performance than GRU, but the advantages of GRU are the quicker learning curve, fewer parameters, and simpler architecture. For practical problems with very large datasets, the GRU provides a trade-off between accuracy and training time.

IoT will become the backbone of urban success, devices are integrated together in order to enhance key 'smart' sectors such as transport, energy, water supply and waste treatment. On the other hand, advanced AI techniques such as the LSTM have shown their power of optimizing IoT. There are many possible scenarios to extend IoT for purposes of environmental monitoring and modeling applications.

The potential power of deep learning is fascinating, and it is worth to conduct further studies about adapting deep learning into water resource related fields. Studies about the performance of other deep learning methods such as deep belief nets, stacked deep encode and generative adversarial networks in water resource related fields, are other interesting research directions in the future.

Acknowledgements

This work has been supported by the Regnbyge-3M project (grant number 234974), which is granted by the Oslofjord Regional Research Fund. The authors would like to thank the engineers from Rosim AS for their supports.

References

Abadi M., Agarwal A., Barham P., Brevdo E., Chen Z., Citro C., et al., 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. arXiv preprint arXiv:160304467.
Abghari H., Nouri M., and Rezaeian Zadeh M., 2009. Reservoir dam inflow modeling using wavelet-base neural network and multilayer perceptron network. International Conference on Water Resources, Shahrood, Iran.
Abghari, H., Ahmadi, H., Besharat, S., Rezaverdinejad, V., 2012. Prediction of daily pan evaporation using wavelet neural networks. Water Resour. Manage. 26 (12), 3639–3652.

Alam, F., Mehmood, R., Katib, I., Albeshri, A., 2016. Analysis of eight data mining algorithms for smarter Internet of Things (IoT). Proc. Comput. Sci. 98, 437–442.

Alexandridis, A.K., Zapranis, A.D., 2013. Wavelet neural networks: a practical guide. Neural Networks 42, 1–27.

Atzori, L., Iera, A., Morabito, G., 2010. The internet of things: a survey. Comput. Networks 54 (15), 2787–2805.

Autixier, L., Mailhot, A., Bolduc, S., Madoux-Humery, A.S., Galarneau, M., Prévost, M., Dorner, S., 2014. Evaluating rain gardens as a method to reduce the impact of sewer overflows in sources of drinking water. Sci. Total Environ. 499, 238–247.

Barzegar, R., Fijani, E., Moghaddam, A.A., Tziritis, E., 2017. Forecasting of groundwater level fluctuations using ensemble hybrid multi-wavelet neural network-based models. Sci. Total Environ. 599, 20–31.

Chang, F.J., Chen, P.A., Lu, Y.R., Huang, E., Chang, K.Y., 2014. Real-time multi-step-ahead water level forecasting by recurrent neural networks for urban flood control. J. Hydrol. 517, 836–846.

Chen, Y., Yang, B., Dong, J., 2006. Time-series prediction using a local linear wavelet neural network. Neurocomputing 69 (4), 449–465.

Cho, K., Van Merriënboer, B., Bahdanau, D., and Bengio, Y., 2014. On the properties of neural machine translation: Encoder-decoder approaches. arXiv preprint arXiv:1409.1259.

Chollet F., 2015. Keras: Deep learning library for theano and tensorflow.

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y., 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555.

Darsono, S., Labadie, J.W., 2007. Neural-optimal control algorithm for real-time regulation of in-line storage in combined sewer systems. Environ. Model. Software 22 (9), 1349–1361.

Elman, J.L., 1990. Finding structure in time. Cogn. Sci. 14 (2), 179–211.

Garofalo, G., Giordano, A., Piro, P., Spezzano, G., Vinci, A., 2017. A distributed real-time approach for mitigating CSO and flooding in urban drainage systems. J. Network Comput. Appl. 78, 30–42.

Gers, F., 2001. Long short-term memory in recurrent neural networks, (Dissertation), Universität Hannover.

Gers, F.A., Schmidhuber, J., Cummins, F., 2000. Learning to forget: continual prediction with LSTM. Neural Comput. 12 (10), 2451–2471.

Giusto, D., 2010. A. Lera, G. Morabito, I. Atzori (Eds.) The Internet of Things. Springer, 2010. ISBN: 978-1-4419-1673-0.

Google, 2016. https://research.googleblog.com/2016/09/a-neural-network-for-machine.html. (Accessed 27 April 2017).

Grum, M., Thornberg, D., Christensen, M. L., Shididi, S. A., and Thirsing, C. (2011). Full-scale real time control demonstration project in Copenhagen's largest urban drainage catchments. In: Proceedings of the 12th international conference on urban drainage, Porto Alegre.

Hinton, G.E., Osindero, S., Teh, Y.W., 2006. A fast learning algorithm for deep belief nets. Neural Comput. 18 (7), 1527–1554.

Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. Neural Comput. 9 (8), 1735–1780.

Ishak, S., Kotha, P., Alecsandru, C., 2003. Optimization of dynamic neural network performance for short-term traffic prediction. Trans. Res. Rec.: J. Trans. Res. Board 1836, 45–56.

Kingma, D., Ba, J., 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Kurth, A., Saul, A., Mounce, S., Shepherd, W., and Hanson, D., 2008. Application of Artificial Neural Networks (ANNs) for the prediction of CSO discharges. In: 11th International Conference on Urban Drainage.

Lipton, Z. C., Berkowitz, J., and Elkan, C., 2015. A critical review of recurrent neural networks for sequence learning. arXiv preprint arXiv:1506.00019.

Lucas, W.C., Sample, D.J., 2015. Reducing combined sewer overflows by using outlet controls for Green Stormwater Infrastructure: case study in Richmond, Virginia. J. Hydrol. 520, 473–488.

Ma, X., Tao, Z., Wang, Y., Yu, H., Wang, Y., 2015. Long short-term memory neural network for traffic speed prediction using remote microwave sensor data. Trans. Res. Part C: Emerg. Technol. 54, 187–197.

Majeed, K., Masood, Z., Samar, R., Raja, M.A.Z., 2017. A genetic algorithm optimized morlet wavelet artificial neural network to study the dynamics of nonlinear troech's system. Appl. Soft Comput.

Marçais, J., de Dreuzy, J.R., 2017. Prospective interest of deep learning for hydrological inference. *Groundwater*.

Montserrat, A., Bosch, L., Kiser, M.A., Poch, M., Corominas, L., 2015. Using data from monitoring combined sewer overflows to assess, improve, and maintain combined sewer systems. Sci. Total Environ. 505, 1053–1061.

Mounce, S.R., Shepherd, W., Sailor, G., Shucksmith, J., Saul, A.J., 2014. Predicting combined sewer overflows chamber depth using artificial neural networks with rainfall radar data. Water Sci. Technol. 69 (6), 1326–1333.

Nourani, V., Baghanam, A.H., Adamowski, J., Kisi, O., 2014. Applications of hybrid wavelet–Artificial Intelligence models in hydrology: a review. J. Hydrol. 514, 358–377.

Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., Van Den Driessche, G., Dieleman, S., 2016. Mastering the game of Go with deep neural networks and tree search. Nature 529 (7587), 484–489.

Tang, D., Qin, B., Liu, T., 2015. Document Modeling with Gated Recurrent Neural Network for Sentiment Classification. In EMNLP, pp. 1422–1432.

Wang, W., Ding, J., 2003. Wavelet network model and its application to the prediction of hydrology. Nat. Sci. 1 (1), 67–71.

Wu, Z., King, S., 2016. Investigating gated recurrent networks for speech synthesis. In: Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on, pp. 5140–5144. IEEE.

Xingjian, S.H.I., Chen, Z., Wang, H., Yeung, D.Y., Wong, W.K., Woo, W.C., 2015. Convolutional LSTM network: a machine learning approach for precipitation nowcasting. Adv. Neural Inf. Proc. Syst., 802–810

Zaytar, M.A., Amrani, C.E., 2016. Sequence to sequence weather forecasting with long short-term memory recurrent neural networks. Int. J. Comput. Appl. 143, 7–11.

Zhang, M., Zhang, Y., Vo, D.T., 2016. Gated Neural Networks for Targeted Sentiment Analysis. In AAAI pp. 3087–3093.

Xu, L., Liu, S., 2013. Study of short-term water quality prediction model based on wavelet neural network. Math. Comput. Model. 58 (3), 807–813.

Chitsaz, H., Amjady, N., Zareipour, H., 2015. Wind power forecast using wavelet neural network trained by improved Clonal selection algorithm. Energy Convers. Manage. 89, 588–598.