Deep learning convolutional neural network in rainfall–runoff modelling
Song Pham Van, Hoang Minh Le, Dat Vi Thanh, Thanh Duc Dong, Ho Huu Loc and Duong Tran Anh

ABSTRACT
Rainfall–runoff modelling is complicated due to numerous complex interactions and feedback in the water cycle among precipitation and evapotranspiration processes, and also geophysical characteristics. Consequently, the lack of geophysical characteristics such as soil properties leads to difficulties in developing physical and analytical models when traditional statistical methods cannot simulate rainfall–runoff accurately. Machine learning techniques with data-driven methods, which can capture the nonlinear relationship between prediction and predictors, have been rapidly developed in the last decades and have many applications in the field of water resources. This study attempts to develop a novel 1D convolutional neural network (CNN), a deep learning technique, with a ReLU activation function for rainfall–runoff modelling. The modelling paradigm includes applying two convolutional filters in parallel to separate time series, which allows for the fast processing of data and the exploitation of the correlation structure between the multivariate time series. The developed modelling framework is evaluated with measured data at Chau Doc and Can Tho hydro-meteorological stations in the Vietnamese Mekong Delta. The proposed model results are compared with simulations of long short-term memory (LSTM) and traditional models. Both CNN and LSTM have better performance than the traditional models, and the statistical performance of the CNN model is slightly better than the LSTM results. We demonstrate that the convolutional network is suitable for regression-type problems and can effectively learn dependencies in and between the series without the need for a long historical time series, is a time-efficient and easy to implement alternative to recurrent-type networks and tends to outperform linear and recurrent models.

Key words | 1D CNN, deep learning, LSTM, Mekong Delta, rainfall–runoff

INTRODUCTION
Rainfall–runoff simulation is the fundamental technique of hydrology when the availability of surface and subsurface water is an indispensable input for various water resource studies. However, a proper understanding of rainfall–runoff relationships has been a long-term challenge to the hydrological community because of the complex interactions and feedback of soil characteristics, land use, and land cover dynamics and precipitation patterns (Kumar et al. 2005). Physically based and conceptual models require an in-depth knowledge and profound understanding of the water cycle. Moreover, building these models is time-consuming and laborious. These models also require detailed soil profiles of study areas which cannot be adequately provided with current survey and remote sensing techniques. In contrast, data-driven methods are often inexpensive, accurate, precise, and most importantly more flexible (Abrahart
Among sophisticated machine learning techniques, artificial neural network (ANN) has been applied widely in recent years in water resource assessments due to its significant capability in handling nonlinear and non-stationary problems.

Various ANN architectures have successfully been applied in simulating and predicting hydrological and hydraulic variables, such as rainfall and runoff and sediment loads. In many studies, ANN performed better than conventional statistical modelling techniques (Coulibaly et al. 2000; Dawson & Wilby 2001; Sudheer et al. 2002), and this network has also been used as an alternative for rainfall–runoff forecasting. A three-layer feed-forward ANN can primarily represent the rainfall–runoff process in Half et al. (1993) at first. The success of this model then stimulates afterward numerous studies to employ diverse ANN structures for rainfall–runoff prediction (e.g., Minns & Hall 1996; Shamseldin 1997; de Vos & Rientjes 2005). Hsu et al. (1995) propose a linear least squares simplex algorithm to train ANN models. The results showed a better representation of the rainfall–runoff relationships than other time-series models. Mason et al. (1996) use a radial basis function network for rainfall–runoff modelling, which provides faster training compared with the conventional back-propagation technique. Birikundavyi et al. (2002), again, investigate ANN models for daily streamflow prediction and conclude that ANN can provide better performance than other models such as deterministic models and classic autoregressive models. Toth & Brath (2007) and Duong et al. (2019) found that ANN is an excellent tool for rainfall–runoff simulations of continuous periods, provided that an extensive set of hydro-meteorological data was available for calibration purposes. Bai et al. (2016) forecast daily reservoir inflows by using deep belief networks (DBNs).

Most of the studies mentioned above have focused on the specific form of ANN called the multilayer feed-forward neural network (FNN), and only a limited number of studies applied recurrent neural networks (RNNs). Even though FNN has numerous advantages in simulating statistical data, there are still several difficulties such as the selection of optimal parameters for neural networks and the overfitting problem. Thus, the performance of ANN predictions is also significantly dependent on the user’s experience (Dawson & Wilby 2001; de Vos & Rientjes 2005; Manisha et al. 2008). Moreover, the FNN may not capture the distinctive features of data. To model time-series data, the FNN needs to include temporal information in input data. RNNs are specifically designed to overcome this problem.

There are several extensions of RNNs such as the Elman and Jordan network. These models attempt to improve the capacity of memory and the performance of RNN (Cruse 2006; Yu et al. 2017). However, these models suffer from the exploding and vanishing gradient problems. Subsequently, Hochreiter & Schmidhuber (1997) propose long short-term memory (LSTM) to overcome these problems. LSTM is a state-of-the-art model which has particular advances in deep learning to provide useful insights for tackling complex issues such as image captioning, language processing, and handwriting recognition (Sutskever et al. 2014; Donahue et al. 2015; Vinyals et al. 2015). The modern design of LSTM uses several gates with different functions to control the neurons and store information. LSTM memory cells can keep relevant information for a more extended period (Gers et al. 2000). This feature of holding information allows LSTM to perform well on processing or predicting a complex dynamic sequence (Yu et al. 2017). Hu et al. (2018) propose deep learning with LSTM for rainfall–runoff modelling and conclude that ANN and LSTM are both suitable for rainfall–runoff models and better than conceptual- and physical-based models. Kratzert et al. (2018) used LSTM for rainfall–runoff modelling for 241 catchments and demonstrates the potential of LSTM as a regional hydrological model in which one model predicts the discharge for a variety of catchments. Several other studies have shown that LSTM can achieve better performance than the Hidden Markov Model and other RNNs in capturing long-range dependencies and nonlinear dynamics (Baccouche et al. 2011; Graves 2013).

Even though an optimal ANN model can provide accurate forecasts for simple rainfall–runoff problems, it often yields sub-optimal solutions even with lagged inputs or tapped delay lines (Coulibaly et al. 2000). In general, rainfall and runoff have a quasi-periodic signal with frequently cyclical fluctuations and diverse noises at different levels (Wu et al. 2009). A standard ANN model is not well suited for complex temporal sequence processing owing to its static memory structure (Giles et al. 1997; Haykin 1999). Due to its seasonal nature and nonlinear characteristics, many...
hybrid methods have been developed to describe this relationship (Marquez et al. 2001; Hu et al. 2007; Wu et al. 2010; Wu & Chau 2011). However, there are still gaps that need to be addressed. For example, these models were unable to cope with peak values and fit time intervals successfully, and they usually underestimated the rainfall–runoff in extreme events.

Conventional neural network models only capture natural data in shallow forms without insightful information, whereas deep learning can be composed of multiple processing layers to learn representations of data with multiple levels of abstraction. It also helps to explore the insight structure of datasets. Two modern models used in deep learning are CNN and LSTM for modelling sequential data to enhance computer vision (Chen et al. 2018; Fischer & Krauss 2018). A convolutional neural network (CNN) is a biologically inspired type of deep neural network that has recently gained popularity due to its success in classification problems (e.g. image recognition (Krizhevsky et al. 2012) or time-series classification (Wang et al. 2017)). The CNN consists of a sequence of convolutional layers, the output of which is connected only to local regions in the input. This can be achieved by sliding a filter, or weight matrix, over inputs and at each point computing the dot product between the input and the filter. This structure allows the model to learn filters that can recognize specific patterns in the input data. Recent advances in the CNN for rainfall–runoff forecasting include Li et al. (2018) where the authors propose deep convolution belief neural network for rainfall–runoff modelling, and they conclude that the presented approach can accurately predict rainfall–runoff.

In general, the literature on rainfall–runoff with convolutional architectures is still scarce, as these types of networks are much more commonly applied in classification problems. Shen (2018) and Mosavi et al. (2019) also stated that the application of deep learning in earth system modelling is still limited. To the best of our knowledge, there are very few studies using deep learning in hydrology, especially applying deep learning of CNN and LSTM in rainfall–runoff modelling. Thus, in this study, we proposed a novel 1D CNN model for daily rainfall–runoff prediction. The modern CNN model with two-layer filters using Batch normalization, ReLU activation, and the max pooling technique is proposed for this study. The effectiveness and accuracy of these models were evaluated by comparison with a single LSTM model. To ensure wider applications of conclusions, two rain gauge stations and two discharge stations, namely Chau Doc and Can Tho on the Bassac River in the Vietnamese Mekong Delta (VMD), are investigated. This paper is structured in the following manner. Following the introduction, the study areas are described, and modelling methods are presented. The section ‘Methodology’ presents the methodology of this research. In the section ‘Model set-up’, the optimal model is identified, and the implementation of the CNN and LSTM models is described. In the section ‘Results and discussion’, the main results are shown along with discussions. The section ‘Conclusion’ summarizes the main conclusions in this study.

**STUDY AREA AND DATA**

Chau Doc and Can Tho, two long-term and continuous gauging stations (Figure 1) in the VMD, are considered for the purpose of this study. The daily rainfall and runoff data are measured at two meteorological and two hydrological stations with the same names located at the upstream and middle of the Bassac River. The data collected daily include rainfall and discharge data that are measured by the Southern Regional Hydro-Meteorological Center, and these data are also used in Dang et al. (2016, 2018). The data period measured at the Chau Doc station spans over 16 years from 1 January 1996 to 31 December 2011, and we also consider 12 years of data from 1 January 2000 to 31 December 2011 for the Can Tho station. The mean annual discharge at Chau Doc is approximately 3,200 m³/s, with an average annual rainfall of 1,700 mm. At Can Tho, the average discharge is about 9,200 m³/s, with an average annual rainfall of 1,500 mm. Figure 2 demonstrates the rainfall and runoff time series measured at the two stations. The data represent various types of hydrological conditions and flows range from low to very high. The input–output dataset in each station is randomly divided into three subsets, including a training set, cross-validation set and testing set (70% for training, 15% for cross-validation and 15% for testing). The training set serves the model training, and the testing set is used to evaluate the performance.
of models. The cross-validation set has two functions: one is to implement an early stopping approach, so we can avoid overfitting of the training data, and the second function is to select the best prediction from a large number of ANN’s runs. Moreover, the ANN employs the hyperbolic tangent function as transfer functions in both hidden and output layers. Table 1 presents statistical information on rainfall and streamflow data, including means ($\mu$), standard deviations ($S_x$), skewness coefficients ($C_s$), minimum ($X_{\text{min}}$), and maximum ($X_{\text{max}}$) values. We implemented this experiment with assumption that no prior knowledge about the study area is provided.

Figure 1 | Location of Chau Doc and Can Tho stations in VMD.
METHODOLOGY

Convolutional neural networks

CNNs are developed with the idea of local connectivity. The spatial extent of each connectivity is referred to as the receptive field of the node. The local connectivity is achieved by replacing weighted sums from the neural network with convolutions. In each layer of the CNN, the input is convolved with the weight matrix (the filter) to create a feature map. In other words, the weight matrix slides over the input and computes the dot product between the input and the weight matrix. The local connectivity and shared weights aspect of CNNs reduce the total number of learnable parameters resulting in more efficient training.

The deep CNN can be broadly segregated into two major parts as shown in Figure 3, the first part contains the sequence of two 1D convolutional blocks with a convolutional 1D layer of 32 and 64 channels for the first and second blocks, respectively, Batch norm layer, ReLU activation functions, and a max pooling 1D layer, and another part contain the sequence of fully connected layers. Two main convolutional blocks encode the input signal by reducing its length and increasing the number of channels. The

Figure 2 | Daily rainfall–runoff time series (a) Chau Doc and (b) Can Tho.
output of the second convolutional block is concatenated with the input signal using a residual skip connection. This identity shortcut connection does not add extra parameters and computation complexity to the whole network, but it can help the network retain information from input at the deeper layers (He et al. 2018). After concatenating the input signal and the output of convolutional blocks, the fully connected layer is used for the last decision layer, which generates the output.

The output value of the Conv1D layer with input size \((N, C_{\text{in}}, L)\) and output \((N, C_{\text{out}}, L_{\text{out}})\):

\[
\text{out}(N_{i}, C_{\text{out}j}) = \text{bias}(C_{\text{out}j}) + \sum_{k=0}^{C_{\text{out}j}-1} \text{weight}(C_{\text{out}j}, k) \ast \text{input}(N_{i}, k)
\]

(1)

where \(\ast\) is the valid cross-correlation operator (in this case, it is a convolutional operator), \(N_{i}\) is a batch-size \(i\)th, \(C_{\text{out}j}\) is a channel \(j\)th, \(L\) is the length of signal sequence (if the input is image, width and height should be used instead of length).

And the length of output signal sequence can be calculated by using the following formula:

\[
L_{\text{out}} = \left\lfloor \frac{L_{\text{in}} + 2 \times \text{padding} - \text{dilation} \times (\text{kernelsize} - 1) - 1 + 1}{\text{stride}} \right\rfloor + 1
\]

(2)

where:

- \(\text{stride}\) is the stride of the cross-correlation
- \(\text{padding}\) is the amount of zero-paddings on both sides
- \(\text{dilation}\) is the spacing between the kernel elements
- \(\text{kernelsize}\) is the size of the convolution kernel

### Table 1 | Statistical information on rainfall and streamflow data

<table>
<thead>
<tr>
<th>Hydrological stations and datasets</th>
<th>Statistical parameters</th>
<th>(\mu)</th>
<th>(S_x)</th>
<th>(C_s)</th>
<th>(X_{\text{min}})</th>
<th>(X_{\text{max}})</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Chau Doc</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rainfall (mm)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Original data</td>
<td>3.741</td>
<td>10.825</td>
<td>7.354</td>
<td>0</td>
<td>104.5</td>
<td></td>
</tr>
<tr>
<td>Training</td>
<td>3.746</td>
<td>11.084</td>
<td>8.260</td>
<td>0</td>
<td>104.5</td>
<td></td>
</tr>
<tr>
<td>Cross-validation</td>
<td>4.231</td>
<td>11.162</td>
<td>4.231</td>
<td>0</td>
<td>104.5</td>
<td></td>
</tr>
<tr>
<td>Testing</td>
<td>3.055</td>
<td>9.092</td>
<td>5.027</td>
<td>0</td>
<td>105.8</td>
<td></td>
</tr>
<tr>
<td>Runoff (m³/s)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Original data</td>
<td>2.583</td>
<td>2.146</td>
<td>0.649</td>
<td>133</td>
<td>8.210</td>
<td></td>
</tr>
<tr>
<td>Training</td>
<td>2.570</td>
<td>2.153</td>
<td>0.658</td>
<td>133</td>
<td>8.150</td>
<td></td>
</tr>
<tr>
<td>Cross-validation</td>
<td>2.361</td>
<td>1.901</td>
<td>0.607</td>
<td>214</td>
<td>6.420</td>
<td></td>
</tr>
<tr>
<td>Testing</td>
<td>2.868</td>
<td>2.312</td>
<td>0.543</td>
<td>238</td>
<td>8.210</td>
<td></td>
</tr>
<tr>
<td><strong>Can Tho</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rainfall (mm)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Original data</td>
<td>4.254</td>
<td>10.908</td>
<td>5.769</td>
<td>0</td>
<td>230.4</td>
<td></td>
</tr>
<tr>
<td>Training</td>
<td>4.281</td>
<td>11.213</td>
<td>6.139</td>
<td>0</td>
<td>230.4</td>
<td></td>
</tr>
<tr>
<td>Cross-validation</td>
<td>3.801</td>
<td>8.763</td>
<td>3.103</td>
<td>0</td>
<td>60.90</td>
<td></td>
</tr>
<tr>
<td>Testing</td>
<td>4.232</td>
<td>10.975</td>
<td>4.872</td>
<td>0</td>
<td>109.0</td>
<td></td>
</tr>
<tr>
<td>Runoff (m³/s)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Original data</td>
<td>6.371</td>
<td>4.928</td>
<td>0.592</td>
<td>0</td>
<td>34.190</td>
<td></td>
</tr>
<tr>
<td>Training</td>
<td>6.165</td>
<td>4.856</td>
<td>0.637</td>
<td>0</td>
<td>34.190</td>
<td></td>
</tr>
<tr>
<td>Cross-validation</td>
<td>6.968</td>
<td>4.582</td>
<td>0.288</td>
<td>0</td>
<td>16.600</td>
<td></td>
</tr>
<tr>
<td>Testing</td>
<td>6.736</td>
<td>5.581</td>
<td>0.601</td>
<td>0</td>
<td>19.600</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 3** | CNN with two BatchNorm + ReLU and max pooling, and fully connected.
For the Max Pooling 1D, the output value with the input size \((N, C, L)\) and output \((N, C, L_{out})\) could be described as:

\[
\text{out}(N_i, C_{out}) = \max_{m=0, \ldots, \text{kernel\_size}-1} \text{input}(N_i, C_j, \text{stride} \times k + m)
\]

(3)

where \(N_i\) is the input \(i\)th; \(C_j\) is the channel \(j\)th.

- **kernelsize** is the size of the window for taking the max over.
- **stride** is the stride of the window.
- **padding** is the amount of zero to be added on both sides.
- **dilation** is a parameter that controls the stride of elements in the window.

The length of output signal sequence for the max pooling 1D layer can be calculated using the similar formula in the Conv1D layer.

**LSTM recurrent neural network**

Although RNNs have proved successful in tasks such as speech recognition (Vinyals et al. 2015) and text generation (Sutskever et al. 2011), it can be difficult to train them to learn long-term dynamics, partially due to the vanishing and exploding gradient problems (Hochreiter & Schmidhuber 1997) that can result from propagating the gradients down through the many layers of the recurrent network, each corresponding to a particular time step. LSTM provides a solution by incorporating memory units that allow networks to learn when to forget previously hidden states and when to update hidden states given new information (Figure 4).

LSTM extends the RNN with memory cells, instead of recurrent units, to store an output information, easing the learning of temporal relationships on long time scales. The major innovation of LSTM is its memory cell which essentially acts as an accumulator of the state information. LSTM makes use of the concept of gating – a mechanism based on the component-wise multiplication of the input, which defines the behaviour of each memory cell. LSTM updates cell states according to the activation of the gates. One advantage of using the memory cell and gates to control information flow is that the gradient will be trapped in the cell and be prevented from vanishing too quickly, a critical problem for the vanilla RNN model (Hochreiter & Schmidhuber 1997; Pascanu et al. 2013). The input provided to LSTM is fed into different gates when operation is performed on the cell memory: write (input gate), read (output gate), or reset (forget gate). The activation of LSTM units is calculated as in the RNN. The computation of a hidden value \(h_t\) of an LSTM cell is updated at every time step \(t\). The vector representation (vectors denoting all units in a layer) of the update of an LSTM layer is denoted as an input gate \(i_t\), a forget gate \(f_t\), an output gate \(o_t\), a memory cell \(c_t\), and a hidden state \(h_t\).

As research on LSTM has progressed, hidden units with varying connections within the memory unit have been proposed. We use the LSTM unit as described in Figure 2, which is described in detail in Graves & Jaitly (2014). Letting the sigmoid nonlinearity which squashes real-valued inputs to a \([0; 1]\) range, and letting the hyperbolic tangent...
nonlinearity, similarly squashing its inputs to a $[-1; 1]$ range. LSTM updates for time step $t$ given inputs $x_t$, $h_{t-1}$, and $c_{t-1}$ are:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i)$$
$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f)$$
$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o)$$
$$g_t = \phi(W_{xc}x_t + W_{hc}h_{t-1} + b_c)$$
$$c_t = f_t \odot c_{t-1} + i_t \odot g_t$$
$$h_t = o_t \odot \phi(c_t)$$

where $i, f, o, c,$ and $g$ are, respectively, the input gate, forget gate, output gate, cell activation, and input modulation gate vectors. All gate vectors are the same size as the vector $h$ that defines the hidden value. Terms represent an element-wise application of the sigmoid (logistic) function. The term $x_t$ is the input to the memory cell layer at time $t$; $W_{xi}$, $W_{xf}$, $W_{xo}$, $W_{xc}$, $W_{hi}$, $W_{hf}$, $W_{ho}$, $W_{hc}$ are weight matrices, with subscripts representing from-to relationships (the input–input gate matrix, the hidden–input gate matrix, etc.). $b_i$, $b_f$, $b_o$, $b_c$ are bias vectors; $\phi$ stands for an element-wise application of the $\tanh$ function; $\odot$ denotes element-wise multiplication.

The Adam optimizer is applied for training the LSTM model. This algorithm is widely used for deep learning models that need the first-order, gradient-based descent with small memory and a computer adaptive learning rate for different parameters (Jangid & Srivastava 2018). This method is easy to implement and computationally efficient and has proved better than the RMSprop and Rprop optimizers (He et al. 2018). The rescaling process of the gradient is dependent on the magnitudes of parameter updates. The Adam optimizer does not need a stationary object and works with limited gradients. We compute the decaying averages of past and past squared gradients $m_t$ and $v_t$, respectively, as follows:

$$m_t = \beta_1m_{t-1} + (1 - \beta_1)g_t$$
$$v_t = \beta_2v_{t-1} + (1 - \beta_2)g_t^2$$

$m_t$ and $v_t$ are estimates of the first moment (the mean) and the second moment (the uncentred variance) of the gradients, respectively. $m_t$ and $v_t$ are initialized as vectors of 0’s, the authors of Adam observe that they are biased towards zero, especially during the initial time steps, and especially when the decay rates are small (i.e. $\beta_1$ and $\beta_2$ are close to 1). They counteract these biases by computing bias-corrected first- and second-moment estimates:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$
$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

They then use these to update the parameters:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon_1} \hat{m}_t$$

In this study, we use the default value: $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$, and the learning rate $\eta = 0.001$. More detail about this method is available in Kingma & Ba (2014).

**MODEL SET-UP**

**Potential input variables**

Screening possible variables for model inputs in the neural network method is an important step to select an optimal architecture of models. The causal variables in the rainfall–runoff relationship may include rainfall, evaporation, and temperature. The number of different variables depends on the availability of data and the objectives of the studies. Most studies applied rainfall and previous discharges with different time steps and combinations as inputs (Sivapragasam et al. 2001; Xu & Li 2002; Jeong & Kim 2005; Kumar et al. 2005), while other studies attempted to apply other factors such as temperature or evapotranspiration, or relative humidity (Coulibaly et al. 2000; Abebe & Price 2003; Solomatine & Dulal 2003; Wilby et al. 2003; Hu et al. 2007; Toth & Brath 2007; Solomatine & Shrestha 2009; Solomatine et al. 2009). However, some studies pointed out that
evaporation or temperature as an input variable seemed unnecessary and may lead to chaos and noises during the training process (Abrahart et al. 2001; Ancil et al. 2004; Toth & Brath 2007). Ancil et al. (2004) pointed out that potential evapotranspiration did not contribute to improving the ANN performance of rainfall–runoff models. Toth & Brath (2007) also concluded that considering potential evapotranspiration data did not enhance model performance and may yield poorer results in comparison with the non-use of these data in the models. These results may be explained by the fact that the addition of evapotranspiration or temperature input nodes increases the network complexity and therefore the risk of overfitting (Wu & Chau 2011). Thus, in this study, we use rainfall and streamflow as input variables in model development.

Model development

This study developed a rainfall–runoff relationship with the CNN and LSTM models for the two hydrological stations on the Bassac River. The general representative data-driven model can be defined as:

\[
\hat{Q}_t = f(X_t)
\]

\[
= f(Q_{t-1}, Q_{t-2}, Q_{t-n}, \ldots, R_{t-1}, R_{t-2}, R_{t-m})
\]

(10)

where \(\hat{Q}_t\) stands for the predicted flow at time instance \(t\); \(Q_{t-1}, Q_{t-2}, Q_{t-3}\) are the antecedent flow (up to \(t-1, t-2, t-n\) time steps); \(R_{t-1}, R_{t-2}, R_{t-m}\) are the antecedent rainfall \((t-1, t-2, t-m\) time steps). The predictability of future behaviours is a consequence of the correct identification of the system transfer function of \(f(.)\). We test three different correlation types including Kendall, Pearson, and Spearman to analyse the correlation between \(Q\) and \(Q_{t-1}, Q_{t-2}, Q_{t-3}\), and correlation between \(Q\) with \(R_{t-1}, R_{t-2}, R_{t-3}, R_{t-4}\).

From Table 2, the correlations for discharge and rainfall and the correlation between \(Q\) and \(Q_{t-1}, Q_{t-2}, Q_{t-3}\) are still high, while the autocorrelations between \(Q\) and \(R_{t-1}, R_{t-2}, R_{t-3}\) reduce significantly, meaning the later antecedent rainfall from \(t - 4\) time steps does not contribute considerably to the forecast performance (autocorrelation for 4 lag day <0.1 for rainfall data). Therefore, we consider the antecedent flow and rainfall values from \(t\) to \(t - 3\) time steps.

Since the appropriate number of hidden layers and dependent nodes for the models is unknown, a trial-and-error method was used to find the best network’s configuration. An optimal architecture was determined by changing the number of the channel from 8, 16, 32, and 64 for CNN and 10, 15, 20, 25, and 30 memory blocks for LSTM, and was based upon minimizing the difference among the neural network predicted values and the desired outputs. The total architectures of both models are 30 obtained from four different channels and five numbers of memory blocks and six input combinations. The training of the neural network models was stopped when either the acceptable level of errors was achieved, or the number of iterations exceeded a prescribed value. The neural network model configuration that minimized the mean absolute error (MAE) and root mean square error (RMSE) and optimized the \(R\) was selected as the optimum and the whole analysis was repeated several times. The CNN and LSTM architectures were modified by changing the number of

<table>
<thead>
<tr>
<th>Correlations</th>
<th>Discharge</th>
<th></th>
<th></th>
<th></th>
<th>Rainfall</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(Q_t)</td>
<td>(Q_{t-1})</td>
<td>(Q_{t-2})</td>
<td>(Q_{t-3})</td>
<td>(Q_t)</td>
<td>(R_t)</td>
<td>(R_{t-1})</td>
<td>(R_{t-2})</td>
<td>(R_{t-3})</td>
</tr>
<tr>
<td>Kendall</td>
<td>Chau Doc</td>
<td>0.9594</td>
<td>0.9382</td>
<td>0.9267</td>
<td>0.1997</td>
<td>0.205</td>
<td>0.2103</td>
<td>0.2153</td>
<td>0.2203</td>
</tr>
<tr>
<td></td>
<td>Can Tho</td>
<td>0.9054</td>
<td>0.8663</td>
<td>0.8352</td>
<td>0.3243</td>
<td>0.3253</td>
<td>0.3282</td>
<td>0.3309</td>
<td>0.3317</td>
</tr>
<tr>
<td>Pearson</td>
<td>Chau Doc</td>
<td>0.9990</td>
<td>0.9974</td>
<td>0.9953</td>
<td>0.1609</td>
<td>0.1626</td>
<td>0.164</td>
<td>0.1656</td>
<td>0.1675</td>
</tr>
<tr>
<td></td>
<td>Can Tho</td>
<td>0.9851</td>
<td>0.9781</td>
<td>0.9701</td>
<td>0.2027</td>
<td>0.2038</td>
<td>0.2054</td>
<td>0.2053</td>
<td>0.2060</td>
</tr>
<tr>
<td>Spearman</td>
<td>Chau Doc</td>
<td>0.9962</td>
<td>0.9925</td>
<td>0.9888</td>
<td>0.2683</td>
<td>0.2754</td>
<td>0.2827</td>
<td>0.2895</td>
<td>0.2963</td>
</tr>
<tr>
<td></td>
<td>Can Tho</td>
<td>0.9854</td>
<td>0.9738</td>
<td>0.9629</td>
<td>0.4413</td>
<td>0.4426</td>
<td>0.4458</td>
<td>0.4494</td>
<td>0.4507</td>
</tr>
</tbody>
</table>

Table 2 | The Kendall, Pearson, and Spearman correlations between \(Q\) and \(R\) for all data at Chau Doc and Can Tho stations.
hidden layers and its neurons, of the initial weights, as well as the type of input and output functions. Each modification was tested with 50 trials, which served as the basis for the performance assessment of mean values.

The LSTM rainfall–runoff model was developed based on the recurrent neural network, but the structure of network is more complicated with input, output, and forget gates in memory blocks. The input units are fully connected to a hidden layer consisting of memory blocks with one cell each. The cell outputs are fully connected to the cell inputs, to all gates, and to the output units. All gates, the cell itself, and the output unit are biased. Bias weights to input and output gates are initialized block-wise: −0.5 for the first block, −1.0 for the second, −0.5 for the third, and so forth. Forget gates are initialized with symmetric positive values: +0.5 for the first block, +1 for the second block, etc. These are standard values that we use for all experiments. All other weights are randomly initialized in the range [−0.1; 0.1]. The cell’s input squashing function g is a sigmoid function with the range [−1.0; 1.0]. The squashing function of the output unit is the identity function.

A critical concern in the CNN and LSTM application is how to select the best model structure from the possible input variables and to define the number of hidden nodes, but there is no general rule to deal with this problem. Therefore, the trial-and-error procedure is a unique technique to handle this obstacle. To select the input variables of CNN and LSTM, we propose the input combination based on correlation and lag analysis and the candidate input variables as rainfall and runoff at different time steps. There are six selected combinations of input variables for model training and the construction of model structure:

C1: \( R(t - 1), Q(t - 1) \)
C2: \( R(t - 1), Q(t - 1), Q(t - 2) \)
C3: \( R(t - 1), R(t - 2), Q(t - 1), Q(t - 2) \)
C4: \( R(t - 1), R(t - 2), Q(t - 1) \)
C5: \( R(t - 1), Q(t - 1), Q(t - 2), Q(t - 3) \)
C6: \( R(t - 1), R(t - 2), Q(t - 1), Q(t - 2), Q(t - 3) \)

**Evaluation of model performance**

The evaluation of model performance is based on the statistical properties of model outputs. Legates & McCabe (1999) concluded that only one statistical index as the correlation coefficient \( R \) is an inappropriate measure in hydrologic model evaluation. Ritter & Muñoz-Carpena (2013) recommended that a combination of graphical results, absolute value error statistics (i.e., RMSE), and normalized goodness-of-fit statistics is applied. Moreover, Moriasi et al. (2007) also recommended that three quantitative statistics (Nash–Sutcliffe, percent bias, and the ratio of the RMSE) should be used to evaluate the model efficiency. Therefore, we applied the three different indices for presenting goodness of fit, including the RMSE, MAE, and \( R \). To better compare the performance of different model architectures, the present study additionally uses another statistical index, mean absolute percentage error (MAPE). The MAPE is a statistical measure of predictive accuracy expressed as a percentage. The MAPE is useful for evaluating the performance of predictive models due to its relative values. The MAPE effectively reflects relative differences between models because it is unaffected by the size or unit of actual and predicted values (Kaveh et al. 2017). Four measures are, therefore, used in this study and are listed below:

\[
R = \frac{n \sum_{i=1}^{n} (Q_i - \hat{Q}_i) - (\sum_{i=1}^{n} Q_i)(\sum_{i=1}^{n} \hat{Q}_i)}{\sqrt{n \sum_{i=1}^{n} Q_i^2 - (\sum_{i=1}^{n} Q_i)^2} \times \left(n \sum_{i=1}^{n} \hat{Q}_i^2 - (\sum_{i=1}^{n} \hat{Q}_i)^2\right)} \tag{11}
\]

\[
\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (Q_i - \hat{Q}_i)^2} \tag{12}
\]

\[
\text{MAE} = \frac{1}{n} \sum_{i=1}^{n} |Q_i - \hat{Q}_i| \tag{13}
\]

\[
\text{MAPE} = \frac{1}{n} \sum_{i=1}^{n} \left| \frac{Q_i - \hat{Q}_i}{|Q_i|}\right| \times 100\% \tag{14}
\]

where \( n \) is the number of observations, \( \hat{Q}_i \) is the predicted flow, \( Q_i \) represents the observed river flow.
RESULTS AND DISCUSSION

The predictions of daily runoff were modelled by 24 different architectures of CNN and 30 topologies of LSTM for the two hydrological stations and six input combinations based on the testing dataset. Tables 3 and 4 present respective obtained results for the CNN and LSTM models. In Table 3, the CNN model using input data of the combination C5 provides the best result for Chau Doc and Can Tho stations in the testing period. In this combination, the CNN structure consists of 32 channels at the layer 1 and 64 channels at the layer 2 for both Chau Doc and Can Tho stations.

According to Table 4, at the Chau Doc station, the LSTM model, trained with 30 memory blocks and 50,000 loops, provides the best efficiency using the combination C2 with a high value of $R = 0.993$ and the lowest RMSE = 187.221 m$^3$/s and MAE = 148.475 m$^3$/s in the testing phase. From this table, it is also seen that for the Can Tho station, the LSTM using the input combination C5 performs

<table>
<thead>
<tr>
<th>Table 3</th>
<th>Performance of the CNN model for discharge estimation in both stations (testing dataset)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Combination</strong></td>
<td><strong>C1</strong></td>
</tr>
<tr>
<td>Station</td>
<td>Chau Doc</td>
</tr>
<tr>
<td>Layer 1 out channel</td>
<td>8</td>
</tr>
<tr>
<td>Layer 2 out channel</td>
<td>16</td>
</tr>
<tr>
<td>$R$</td>
<td>0.9992</td>
</tr>
<tr>
<td>RMSE</td>
<td>104.907</td>
</tr>
<tr>
<td>MAE</td>
<td>80.535</td>
</tr>
<tr>
<td>Station</td>
<td>Can Tho</td>
</tr>
<tr>
<td>Layer 1 out channel</td>
<td>16</td>
</tr>
<tr>
<td>Layer 2 out channel</td>
<td>32</td>
</tr>
<tr>
<td>$R$</td>
<td>0.955</td>
</tr>
<tr>
<td>RMSE</td>
<td>1,187.327</td>
</tr>
<tr>
<td>MAE</td>
<td>822.854</td>
</tr>
</tbody>
</table>

Bold values indicate the best performance evaluation metrics.

<table>
<thead>
<tr>
<th>Table 4</th>
<th>Performance of the LSTM model for discharge estimation in both stations (testing dataset)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Combination</strong></td>
<td><strong>C1</strong></td>
</tr>
<tr>
<td>Station</td>
<td>Chau Doc</td>
</tr>
<tr>
<td>LSTM: memory blocks</td>
<td>30</td>
</tr>
<tr>
<td>Number of loops</td>
<td>10,000</td>
</tr>
<tr>
<td>$R$</td>
<td>0.98</td>
</tr>
<tr>
<td>RMSE</td>
<td>329.675</td>
</tr>
<tr>
<td>MAE</td>
<td>225.602</td>
</tr>
<tr>
<td>Station</td>
<td>Can Tho</td>
</tr>
<tr>
<td>LSTM: memory blocks</td>
<td>20</td>
</tr>
<tr>
<td>Number of loops</td>
<td>10,000</td>
</tr>
<tr>
<td>$R$</td>
<td>0.971</td>
</tr>
<tr>
<td>RMSE</td>
<td>2,084.928</td>
</tr>
<tr>
<td>MAE</td>
<td>991.933</td>
</tr>
</tbody>
</table>

Bold values indicate the best performance evaluation metrics.
better than the models using other combinations. This model uses 15 memory blocks and 10,000 loops.

Tables 3 and 4 also show that the CNN model can significantly improve the prediction efficiency in the testing period at Chau Doc and Can Tho stations. The best CNN model improves the RMSE, MAE, and $R^2$ values from 89.571, 66.348, and 0.9994 for Chau Doc and 834.01, 652.742, and 0.978 for Can Tho, respectively.

The temporal variations in the observed and predicted discharges using both models and the best input combinations (C5 and C5 for CNN, and C2 and C5 for LSTMs) for Chau Doc and Can Tho stations are, respectively, illustrated in Figures 5 and 6, which shows that the predicted discharges are plotted against observed discharges.

To assess the model efficiency for improving the forecasting accuracy, some researchers carried out runoff predictions using ANN with two different inputs: inputs with previously observed runoffs only and inputs with both previous rainfalls and runoffs. Only a few researchers applied the pre-processing technique to improve the ANN model ability for time-series prediction. For example, Antar et al. (2006) used rainfall and runoff as an input for ANN model training, and the results were compared with distributed rainfall–runoff models. The results obtained from ANN show that the ANN technique has great potential in simulating the rainfall–runoff process adequately. Tokar & Johnson (1999) also investigated different ANN architectures for runoff prediction using daily precipitation, temperature, and snowmelt as the model inputs. Nine models were built to test the effect of a number of input variables, and the ratio of the standard error to the standard deviation of runoff used as goodness-of-fit indices indicated that the highest values were in a range of 0.7–0.82 for training and testing. Sivapragasam et al. (2007) applied ANN combined with genetic programming to forecast flows using both rainfall and runoff data. Results indicated that the model with rainfall and flow data as inputs made a more accurate prediction than that with only a flow input. Furthermore, Wu & Chau (2011) carried out runoff prediction using ANN coupled with singular spectrum analysis (SSA) as a pre-processing technique. The results show that the coefficient of efficiency (CE) varies in a range of 0.74–0.89 for both using rainfall and flow as model input variables without using SSA and the CE varies from 0.87 to 0.94 for the case using SSA. From the statistical performance evaluation, it is clear that our study used CNN and LSTM models.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure5}
\caption{Predicted discharge for the Can Tho station in the testing period (a) CNN-C5 and (b) LSTM-C5.}
\end{figure}
only without the pre-processing technique, but the model performances are better than the above-mentioned models in terms of model efficiency.

From Table 2, it can be concluded that rainfall did not significantly contribute to the runoff prediction because the most important factor to CNN and LSTM models is previous flows. In general, the inclusion of rainfall in the input could be helpful in improving the accuracy of predictions; and adding local rainfall help capturing climatic variability of the studied watershed (Wu & Chau 2011).

As illustrated in Figure 5, the CNN model yields better results for discharge prediction than those predicted by the LSTM model. Both models underestimate the discharge peaks. However, in this instance, the CNN model performs better than the LSTM model, and the results obtained by the CNN model are closer to the 45° straight line in the scatter plots. This point is also obvious from the temporal plot where the CNN model demonstrates an improved agreement with the observed time series at the peaks than the LSTM model.

Figure 6 proves that the best results obtained by the CNN and LSTM models are very close to the observed data and the differences between their prediction results are insignificant. This point makes the graphical comparison between these models difficult. As a consequence, the statistical performance presented in Tables 3 and 4 provides statistical indices that show better efficiency comparison.

Figure 7 shows the performance index MAPE of the CNN and LSTM models for the two stations and all different input combinations. As can be observed, the CNN model performs better than the LSTM model for all the input combinations at the Chau Doc station. The CNN model shows the lower MAPE values with all combinations for Chau Doc and Can Tho stations, except for C2 at the Can Tho station. The differences between the two values for both models are significant. This proves that the CNN model can work efficiently to predict rainfall–runoff.

Tables 3 and 4 present a comparison of runoff predictions using CNN and LSTM with rainfall and flow rates as input variables including different previous days of past rainfall and flow as input variables. It can be observed that, for the case study of Chau Doc and Can Tho, the inclusion of one previous rainfall (combination C5) in input results in the improvement of model performance of CNN. While for the case of Chau Doc, the inclusion of two previous flow and one previous rainfall as input variables (combination C2) can result in the highest LSTM model efficiency. However, the LSTM model can simulate runoff

Figure 6 | Predicted discharge for the Chau Doc station in the testing period (a) CNN-C5 and (b) LSTM-C5.
with the best efficiency falling into the combination C5 with one previous rainfall and three previous flows. Results indicate that the architectures of the LSTM model are strongly influenced by the quality of input data (e.g., length, magnitude, and noise).

Figures 8–11 are the scatter plots, showing the correlation between observed and predicted discharge time series for the six combinations at Chau Doc and Can Tho stations. Both of the LSTM and CNN prediction results exhibit that if we adopt equalled or more input variables from rainfall data compared to discharge data (combinations C1, C3, and C4), the goodness-of-fit statistics is reduced. This also reveals that the impact of upstream inflows contributes more significantly to the flow in the delta compared to rainfall. In Li et al. (2018), the authors entered the same number of discharge and rainfall inputs for the model (the number of considered days for rainfall and discharge data is similar), but this may ignore the fact that soil layers delay runoff generation. Water, basically, can be absorbed into soil owing to the infiltration and percolation processes (Hu et al. 2018), and soil layers then release water later in the form of baseflow when saturated. As a result, when Hu et al. (2018) increase the number of days (N) considered, the model yields a more accurate prediction. The lack of model parameter information is the main barrier of traditional physical-based and conceptual hydrological models (Kratzert et al. 2018). Although deep learning models are normally considered as ‘black box’ as the
nature of nodes and their weights are unknown, these advance techniques can actually solve the problem of the lack of observation data of the conventional models. However, we suggest feeding the LSTM and CNN models with input variables for rainfall–runoff prediction at different time steps.

CNN and LSTM seem also successfully capturing both seasonal and daily flow fluctuations. The flow in the Mekong Basin mostly comes from rainfall in the lower basin, and the amount of rainfall fluctuates. Higher flows observed in the rainy season are due to the development of tropical typhoons and depression on the
Vietnamese East Sea during the monsoonal season. However, due to the uneven distribution of rainfall in space and time, the flows at the two gauged stations are different over time. Historical data exhibit that local rainfall contributes an important amount during the late stage of the wet season in the basin and in the dry season. In both models (CNN and LSTM), the first combination (C1) and the fourth combination (C4) have lower performance during the peak flow period, especially at Can Tho. These characteristics confirm the influence of upstream flows on these stations during the wet season. In the low-flow period, the prediction is quite accurate for all the combinations, which suggests a stable increase/decrease in flows.

It is also worth noticing that CNN performs better curve fitting than LSTM at Chau Doc, while at Can Tho, there was an opposite trend. This is, however, related to the hydrological characteristics of the study area. Dang et al. (2018), modelling the VMD with a hydrodynamic model, concluded that Can Tho is slightly influenced by tide originated from the East Sea. Subsequently, the changes in discharge at Can Tho is more drastic than at Chau Doc. LSTM is an augmented form of RNNs which mostly deal with a sequence of values (Graves & Jaitly 2014) and are more sensitive to both distant and recent events. In the case of Chau Doc, the CNN likely provides more accurate prediction with high consistent inputs.

Finally, we compared the performance of deep learning (CNN and LSTM) with traditional methods such as ANN, GA-SA, SARIMA, and ARIMA which were often carried out for tasks like rainfall–runoff modelling. Table 5 shows the statistical performance of the traditional models at two gauged stations (Chau Doc and Can Tho) on the mainstream of the Mekong River. Figures 12–15 are scatterplots exhibiting the relationship between the observed and predicted data at the stations. These results demonstrated that both CNN and LSTM have the ability to outperform linear and recurrent benchmarked models. In other words, CNN and LSTM are more suitable for rainfall–runoff modelling than the traditional models.

In the Mekong basin, although dozens of dams have been installed recently for electricity generation, the impact of dams on the water cycle in the VMD is still limited (Dang et al. 2016), and the river flow is still stable. Consequently, the CNN is effective for modelling. Nevertheless, the number of dams will increase dramatically in the next decades to fulfil the thirst for energy of surrounding economies (Hecht et al. 2019). More studies will be very much needed to understand if deep machine learning can capture regulated behaviours of river flows.
CONCLUSION

An attempt was made in this paper to investigate the use of the CNN and LSTM models for predicting daily rainfall–runoff at Chau Doc and Can Tho stations, the VMD. Both the CNN and LSTM models have a high potential for predicting daily rainfall–runoff, so as the CNN and LSTM models were assessed in this study with a Python.
Figure 12 | Scatterplot for ANN simulations (Can Tho: top panel, Chau Doc: bottom panel).

Figure 13 | Scatterplot for GA-SA simulations (Can Tho: top panel, Chau Doc: bottom panel).

Figure 14 | Scatterplot for SARIMA simulations (Can Tho: top panel, Chau Doc: bottom panel).
script. The CNN model provided better results for discharge prediction than those predicted by the LSTM model at the Can Tho station, especially for the peaks. For the high discharge values at both stations, the results obtained by the CNN model were closer to the 45° straight line in the scatter plots. At the Chau Doc station, the predicted results of the two models were close to each other, and the CNN model provided slightly better predictions. While both CNN and LSTM are superior to traditional methods as shown in this study, it can be concluded that both the proposed models can be used as alternatives to improve the prediction of hydrological variables. More opportunities exist for deep learning to advance our knowledge in earth system sciences. Since upstream flows have been increasingly regulated in the basin, studies on using deep learning to predict regulated flows should be devoted, so as policymakers could be more proactive in proposing adaptation measures.

ACKNOWLEDGEMENTS

The first author acknowledges the financial support from the Vietnamese-German University. Special thanks to Mr. Tung Kieu – Department of Computer Science, Aalborg University, Denmark in collaborating to build the LSTM model. We also especially thank Mr. Ta Huu Chinh – National Meteorological Center and Dr. Nguyen Mai Dang – Thuy Loi University for providing the daily rainfall and runoff data used in this study.

REFERENCES


First received 13 May 2019; accepted in revised form 25 March 2020. Available online 17 April 2020