Runoff Prediction Based on Deep Belief Networks

Tran Thi Truong Thi[1], Hieu N. Duong[2], Hien T. Nguyen[3], Tran Van Hoai[2]

[1] Hoa Sen University, Ho Chi Minh City, Vietnam
[2] Ho Chi Minh City University of Technology, Ho Chi Minh City, Vietnam
[3] Banking University of Ho Chi Minh City, Ho Chi Minh City, Vietnam

Correspondence: Hien T. Nguyen, nthien@buh.edu.vn
Communication: received 21 June, revised 9 July, accepted 30 July
Digital Object Identifier: 10.32913/mic-ict-research.v2021.n2.973

***Abstract*: Runoff prediction has recently become an essential task with respect to assessing the impact of climate change to people's livelihoods and production. However, the runoff time series always exhibits nonlinear and non-stationary features, which makes it very difficult to be accurately predicted. Machine learning has been recently proved to be a powerful tool in helping society adapt to a changing climate and its subfield, deep learning, showed the power in approximate nonlinear functions. In this study, we propose a method based on Deep Belief Networks (DBN) for runoff prediction. In order to evaluate the proposed method, we collected runoff datasets from Srepok and Dak Nong rivers located in the mountain regions of the Central Highland of Vietnam in the periods of 2001-2007 at Dak Nong hydrology station and 1990-2011 at Buon Don hydrology station, respectively. Experimental results show that DBN outperforms, respectively, LSTM, BiLSTM, Multi-Layer Perceptron (MLP) trained by Particle Swarm Optimization (PSO) and MLP trained by stochastic gradient descent (SGD) in which gradients are computed using the backpropagation (BP) procedure. The results also confirm that DBN is suitable to employ for the task of runoff prediction.**

Keywords: *Deep belief networks, runoff prediction, artificial neural networks, particle swarm optimization.*

I. Introduction

Water resources play a central role in people's livelihoods and production, and thus for any country in the world, water resource management has been considered as a high priority in national development plans. In the past few decades, runoff prediction has been studied by a large number of scientists in many fields [1–3]. One can classify approaches to runoff prediction in literature into two folds: physical-based and data-driven. The main difference between the two approaches is that the former needs a physical model beforehand and the latter uses training data to learn a model [4]. This paper presents a data-driven method proposed to predict runoff in short-term.

To date, several advanced data-driven methods have been proposed to predict and forecast runoff, such as Support Vector Machines (SVM) [5–8], ensemble learning paradigms [9–11], Artificial Neural Networks (ANN) [1, 5, 8, 12–16], and so on. Among them, ANN-based methods has proved its strength in simulating sophisticated relationships of many different kinds of complex data, especially hydrologic data. Until now, many researchers have taken into consideration improving ANN to enhance effectiveness of runoff prediction. To this end, attempts to hybridize ANN with other learning models and theories, such as Support Vector Regression (SVR), fuzzy theory, meta-heuristic algorithms, chaotic theory, etc. have demonstrated the effectiveness and efficiency in hydrology prediction [3, 17–20].

In general, machine learning have been recently proved to be a powerful tool in helping society adapt to a changing of climate [21] and its subfield, deep learning [22], have attracted numerous researchers and shown remarkable performance in various fields [23]. Deep learning has been employed for time series analysis and most experimental results indicated that deep learning outperforms conventional methods in time series classification, prediction, forecasting, and anomaly detection [24–29].

In this work, we employ DBN [30] to predict the runoff of Srepok river and Dak Nong river in the Central Highland of Vietnam. We choose DBN due to nonlinear and non-stationary of the runoffs of these rivers caused by their sloped terrain. Moreover, as pointed out in [29], among deep leaning techniques, DBN is suitable to learn robust representations for network flow prediction. We conduct experiments and compare the performance of DBN with that of LSTM [31], Bidirectional LSTM (BiLSTM), Multi-Layer Perceptron (MLP) [32] trained by SGD using BP

to calculate gradients (henceforth *MLP-SGD*) and MLP trained by Particle Swarm Optimization (PSO) [33] (henceforth *MLP-PSO*). In experiments, the implementation of LSTM published with [34][1] is revised for both LSTM and BiLSTM, and the implementations of ANN-BP and ANN-PSO in [35] are used for MLP-SGD and MLP-PSO, respectively.

The work that is most relevant to ours is presented in [26, 36, 37]. In [26], the authors employed a *so-called* DBN that is a stack of two Restricted Boltzmann Machines (RBM), each of which is trained separately and the higher layer takes as input the output of the lower layer. The obtained DBN model then is fine-tuned by the standard BP. They reported that DBN outperforms MLP-SGD and MLP-PSO, respectively. We argue that the model employed in [26] is actually a Deep Boltzmann Machine (DBM) [38] other than DBN. Indeed, Fig. 2 presented in [38] shows the difference between a Deep Belief Network and a DBM.

Both [36] and [37] proposed the same method that combines Variational Mode Decomposition (VMD) and DBN. VMD decomposes runoff series into multiple intrinsic mode functions to reduce their complexity. The main difference between VMD-DBN [36] and VMD-DBN-IPSO [37] is how to train DBN. The former uses the original DBN and the latter trains DBN using the social learning PSO algorithm presented in [39]. While [40–42] showed that performance of MLP is comparable with that of other learning models, the authors of [36] and [37] did not compare VMD-DBN and VMD-DBN-IPSO with MLP.

The contributions of this paper are as follows:

- First, we propose to employ the DBN for runoff prediction. In particular, visible and hidden units of a RBM in DBN are Gaussian and rectified linear units, respectively. And, the *tanh* activation function is used at the hidden layers during fine-tuning. By using *tanh*, the inputs of a layer are normalized to obtain the data with zero mean because *tanh* squashes the outputs of the previous layer so that they are on average close to zero [43].
- Second, we conduct experiments and compare the performance of DBN to that of LSTM, BiLSTM, MLP-SGD and MLP-PSO on the same real datasets collected from two rivers located in the central highland of Vietnam. The experimental results show that DBN outperforms the models used in experiments. The findings confirm that DBN is suitable to learn robust representations for runoff prediction.

The rest of this paper is organized as follows. Section 2 presents related work. Section 3 briefly presents MLP, PSO, DBN, and algorithms used in experiments. In Section 4, we depict experimental results and some findings will be pointed out. Finally, we draw conclusion in Section 5.

[1] https://github.com/kratzert/pangeo_lstm_example

II. Related work

Many machine learning approaches have been proposed for runoff prediction. Those approaches employed many different learning models range from classical to deep learning, among them, MLPs are overwhelming. In [3], the authors surveyed ANN-based methods for prediction of water resource variables in river systems. Among 2010 surveyed papers published from 1999 to 2007, 178 papers reported using MLPs. Another survey was presented in [19] where the authors surveyed on applications of hybrid wavelet-machine learning models in hydrology and showed that among 103 surveyed papers, 85 reports using ANN techniques, most of them are MLPs. MLPs were also investigated in [15, 16, 40, 41, 44]. Besides, SVR [6] and genetic programming [45] can also be successfully applied to monthly runoff forecasting.

However, the performance of MLPs reported in literature were inconsistent in both long-term and short-term runoff prediction. Indeed, the authors in [20] indicated that the performance of MLP and Recurrent Neural Network (RNN) is comparable. MLPs in [16] and in [42] achieve performance better than that of Multiple Linear Regression (MLR) for runoff prediction, but the MLP in [40] gets performance worst among MLR, K-nearest neighbors (KNN), and Adaptive Network-based Fuzzy Inference Systems (ANFIS) for monthly runoff forecasting. While the MLPs (using radial basis or sigmoid as the activation function) in [41] provides more accurate monthly runoff prediction than KNN, SVR, Time Delay Neural Network (TDNN), and RNN, an investigation in [5] showed that SVR method was superior to MLP in the river flow forecasting. Meanwhile, in [15], MLPs have superior performance among six selected data-driven models. Moreover, according to an investigation in [44], the performance of Gaussian Mixture Regression, SVR and MLP are comparable in middle and long-term runoff forecasting.

Hybrid approaches were also proposed in literature. Considering a time series as a combination of linear and non-linear components, [17] employs ARIMA as linear modeling of the time series and SVR as nonlinear modeling of error series; then MLP was employed to learn a combination function of the output of linear and nonlinear modeling. Some other hybrid models were proposed in [46], [18], [47]. The hybrid model proposed in [46], namely Chaos-MGGP, integrates chaos theory into Multi-Gene Genetic Programming for runoff forecasting. The phase-space reconstruction, a basis for chaotic time series forecasts, is used to

extract hidden characteristics and information on dynamic systems from their time series. SVR-GANN proposed in [18] combines SVR and a geomorphologic ANN (GANN) that incorporates the geomorphologic characteristics of watershed directly to the model. It is trained by using a genetic algorithm.

Pre-processing of runoff data is also an important phase in approaches to runoff prediction. Indeed, in [11], the authors showed that runoff is pre-processed by using Ensemble Empirical Mode Decomposition (EEMD) leads to better performance and Elman neural networks coupled with EEMD outperforms MLP coupled with EEMD in runoff prediction. Wavelet transform [15] or EMD [7] is also used for pre-processing of input data. The findings in these work is that MLPs or SVM taking as input those pre-processed data improve the performance in comparison with that of methods without using the pre-processing. Besides, deep learning such as long-short term memory [47–50] or DBN [36, 37] have been also proposed for runoff prediction.

The most relevant to ours is the work presented in [36] and [37]. However, the performance of [36] and [37] has not been compared to that of MLPs yet. Basically, both methods proposed in [36] and [37] are the same and a kind of ensemble learning. They decompose the input time series into k components using VMD, then each component is used to train a DBN model. The structures of DBNs and the lengths of input sequences can be different. Given testing data, the final output is summing up the outputs of k trained models. The difference between two methods are training algorithms, the lengths of input sequences and normalizing the input data. In contrast, the proposed method in this paper does not exploit ensemble learning strategies. It does not need pre-process and normalize the input like the methods in [36, 37] but makes use of the *tanh* function to squash the output of each layer into the range of [-1,1] during fine-tuning. This trick was proved efficiency in training neural networks [43]. The experimental results show that DBN works well on runoff data and outperforms LSTM, BiLSTM, MLP-SGD, and MLP-PSO on real datasets collected from two rivers located in the central highland of Vietnam.

III. Methodology

1. Multi-layer Perceptron

MLP is a kind of ANN that stack many layers of neurons. The first layer is the input layer, the last layer is the output layer, and the others are hidden layers. Neurons in input, hidden, and output layers are called as input, hidden, and output units, respectively. There is not any connection between two units of the same layer. Units of two layers are fully connected. Fig. 1 presents a structure of MLP with $L+1$ layers, in which the 0th layer is the input layer, the output layer is the Lth layer, L is a certain constant, and $\hat{y}$ is the predicted value.

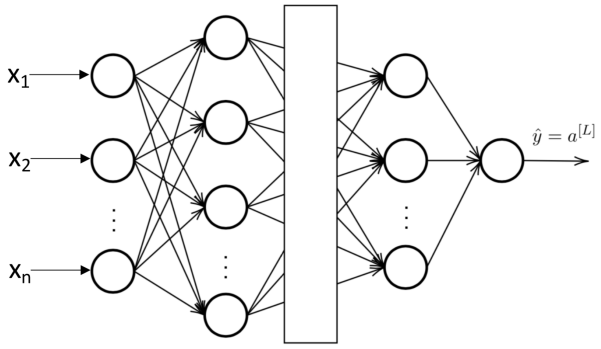


Figure 1: Multi-layer percetron with $L+1$ layers.

Let x be a training example, $W^{[\ell]}$ and $b^{[\ell]}$ be parameters at the layers ℓ $(0 < \ell \leq L)$, the hidden and output values of the ℓth layer are calculated as follows:

$$z^{[\ell]} = W^{[\ell]} a^{[\ell-1]} + b^{[\ell]}, \tag{1}$$

$$a^{[\ell]} = g^{[\ell]}(z^{[\ell]}) = g^{[\ell]}(W^{[\ell]} a^{[\ell-1]} + b^{[\ell]}) \tag{2}$$

At the input layer $a^{[0]} = x$ and at the output layer $a^{[L]} = z^{[L]}$. It means that we use the identity function as the activation function at the output layer. The function $g^{[\ell]}(.)$ in Eq. 2 is the activation function at the ℓth layer. In our experiments, we use *tanh* as the activation function at all hidden layers: $\tanh(z) = \frac{(e^z - e^{-z})}{(e^z + e^{-z})}$. Given a training set $D = \{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}) \ldots (x^{(m)}, y^{(m)})\}$, training MLP is to find the optimal values of parameters $\theta = \{W, b\}$ that minimizes the objective function $J(\theta)$. We employ the mean squared error as the objective function presented in Eq. 3, in which $f(x;\theta)$ is the learned model. The Algorithm 1 is used to train MLP.

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \left(y^{(i)} - f(x^{(i)};\theta)\right)^2 \tag{3}$$

2. PSO

PSO is a metaheuristic algorithm and was invented by Kennedy and Eberhart [33]. It attempts to simulate social behaviors of animals that fly, such as swarming behaviors observed from flocks of birds or swarms of bees. In a PSO system, a particle represents a potential solution of the optimization problem being solved. A basic principle is to randomly generate an initial swarm of particles and then

Algorithm 1: SGD with backpropagation

Inputs: Learning rate α, the number of mini-batches K, training set D.
Outputs: Parameters $W^{[\ell]}$, $b^{[\ell]}$ with $\ell \in \{1 \dots L\}$.
Initialization: $W^{[\ell]}$, $b^{[\ell]}$ `// we use Xavier initialization.`
for $i = 1$ *to* $epoch$ **do**
  for $t = 1$ *to* K **do**
    Sample a mini-batch of examples from the training set.
    `// Back-propagation:`
    Forward pass: Calculate $z^{[\ell]}$ and $a^{[\ell]}$ according to Eq. 1 and Eq. 2
    Backward pass: Calculate partial derivatives $dW^{[l]}$ and $db^{[l]}$ given $da^{[l]}$
      $dz^{[l]} = da^{[l]} g^{[l]\prime}(z^{[l]})$
      $dW^{[l]} = dz^{[l]} (a^{[l-1]})^T$
      $db^{[l]} = dz^{[l]}$
      $da^{[l-1]} = (W^{[l]})^T dz^{[l]}$
    `// Update parameters using Nesterov Momentum [51]`
      Update $W^{[l]}$
      Update $b^{[l]}$
  end
end

let them fly through the search space according to their flying experience. During flying, positions and velocity of particles are changed and the change to position of a particle is influenced by its own best position (denoted as *pbest*) and the global best position of the swarm (denoted as *gbest*).

Let $x_i = \langle x_{i1}, x_{i2}, \dots, x_{id} \rangle$ and $v_i = \langle v_{i1}, v_{i2}, \dots, v_{id} \rangle$ be respectively the position and velocity of the ith particle in d-dimentional search space. Given a fitness function, at the time t, *pbest* corresponding to the best fitness value of the ith particle is $pbest_i = \langle pbest_{i1}, pbest_{i2}, \dots, pbest_{id} \rangle$ and $gbest_i = \langle gbest_1, gbest_2, \dots, gbest_d \rangle$ is the fittest value found so far in the swarm. The updating rules for the new position and velocity at the time $t+1$ of the ith particle are presented in Eq. 4 and Eq. 5 [52].

$$\begin{aligned} v_{ij}^{(t+1)} =& m * v_{ij}^{(t)} \\ &+ c_1 * rand * (pbest_{ij}^{(t)} - x_{ij}^{(t)}) \\ &+ c_2 * rand * (gbest_j^{(t)} - x_{ij}^{(t)}), \end{aligned} \tag{4}$$

$$x_{ij}^{(t+1)} = x_{ij}^{(t)} + v_{ij}^{(t+1)}, \tag{5}$$

where m, c_1, c_2 are configuring parameters and $rand$ is a random value representing the stochastic characteristic of PSO.

Generally, there are three ways to train MLP using PSO. The first way is to use PSO to find a combination of weights and biases which minimizes the objective function of MLP. The second way is to use PSO to find the optimal structure of MLP (*i.e.*, the number of layers and the number of neurons in each layer). The third way is to use PSO to tune hyper-parameters such as learning rate and momentum. In this work, we employ PSO following the first way. In particular, we employ the PSO-BP algorithm presented in [53] to train MLP.

3. Deep Belief Networks

Deep Belief Networks were initially invented by Hinton [54] and have been implemented successfully in feature learning. A DBN models the joint distribution between the observed vector and hidden layers. It is a stack of RBM to form a deep architecture. The Equation 6 represents a DBN with ℓ hidden layers and the observed vector x.

$$P(x, h^1, ..., h^\ell) = (\prod_{k=1}^{\ell-2} P(h^k | h^{k+1})) P(h^{\ell-1}, h^\ell) \tag{6}$$

where $x = h^0$, $P(h^{k-1}|h^k)$ is a conditional distribution for visible hidden units in a RBM associated with the level k of the DBN and $P(h^{\ell-1}, h^\ell)$ is the visible-hidden joint distribution in the top-level RBM.

RBM is a special type of generative-energy based models that can find out a probability distribution over its topology [30]. A RBM consists of two layers, a visible layer and a hidden layer, whose units undirectedly and symmetrically connect together and there is no connection among units in the same layer. We use the implementation of RBM in DeepLearning4j[2] using Gaussian visible units and hidden rectified linear units [55]. The energy of the joint configuration, $E(v, h)$, between visible and hidden units of a RBM is defined in Eq.7 [55].

$$E(v, h) = \sum_{i=1}^{n} \frac{(v_i - a_i)^2}{2\sigma_i^2} - \sum_{j=1}^{m} b_j h_j - \sum_{i,j=1}^{n,m} \frac{v_i}{\sigma_i} h_j w_{ij}, \tag{7}$$

where v_i and h_j are visible unit i and hidden unit j, respectively; a_i and b_j are the biases and w_{ij} is the weight between them, n and m are the numbers of visible and hidden units, respectively. This energy function is used to compute the probability assigned to each possible pair of visible and hidden units. The probability of a configuration is determined by the energy of the configuration as presented in [55]. Fig. 2 presents the structure of a DBN stacking three RBMs. A regression layer (fully-connected

[2]https://deeplearning4j.org

layer) is attached after a stack of multiple RBMs in which the hidden layer of every RBM is regarded as the visible layer of the next RBM.

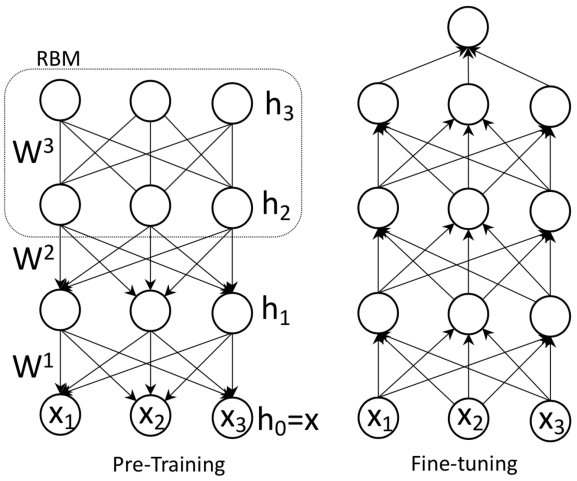


Figure 2: A Deep Belief Network with three hidden layers. **Left**, the Greedy Layer-wise Unsupervised Pre-training. **Right**, fine-tuning for smoothing of parameters.

Algorithm 2: Contrastive Divergence k=1

Inputs: The input v, learning rate α.
Outputs: Parameters W, a, b.
$v_0 \leftarrow v$
for ***all*** *hidden units* i **do**
  compute $Q(h_{0i} = 1|v_0) = ReLU(b_i + \sum_j W_{ij} v_{0j})$
  sample h_{0i} from $Q(h_{0i} = 1|v_0)$
end
for ***all*** *visible units* j **do**
  compute $P(v_{1j} = v_j|h_0) = \frac{1}{\sigma_j\sqrt{2\pi}} exp(-\frac{(v_j - a_j - \sigma_j \sum_i W_{ji} h_{0i})^2}{2\sigma_j^2})$
  sample v_{1j} from $P(v_{1j} = v_j|h_0)$
end
for ***all*** *hidden units* i **do**
  compute $Q(h_{1i} = 1|v_1) = ReLU(b_i + \sum_j W_{ij} v_{1j})$
end
$W \leftarrow W + \alpha(h_0 v_0^T - Q(h_{1.} = 1|v_1)v_1^T)$
$b \leftarrow b + \alpha(h_0 - Q(h_{1.} = 1|v_1))$
$a \leftarrow a + \alpha(v_0 - v_1)$

Training DBN typically consists of two major steps. In the first step, an unsupervised learning algorithm is performed to obtain near-optimal parameter values of the network [30]. This algorithm is a kind of greedy algorithms in which each RBM is separately and independently trained using unlabeled data by the Contrastive Divergence (CD) learning method [55] in consideration of reconstructing the input. In [56], the authors indicated that better initial values of parameters can be obtained by the layer-wise unsupervised training than those obtained by random initialization. In the second step, the entire network is then fine-tuned by BP with labeled data in a supervised manner. The Algorithm 2 implements CD algorithm represented in [57] in which k is set to one.

Algorithm 3: Greedy Layer-wise Unsupervised Pre-training of DBN

Inputs: The input training distribution for the network $\hat{p}$, learning rate α, the number of layers L, the numbers of units in layers $n = \{n^1, n^2, ..., n^L\}$.
Outputs: W^ℓ, b^ℓ with $\ell \in \{1 \dots L\}$.
Data: Training set X
Initialization: $W^{[\ell]}$, $b^{[\ell]}$
for $\ell = 1$ *to* L **do**
  $W^\ell = 0, b^\ell = 0$
  while *not stopping criterion* **do**
    sample $h^0 = x$ from $\hat{p}$
    for $i = 1$ *to* $\ell - 1$ **do**
      sample h^i from $Q(h^i|h^{i-1})$
    end
    call Algorithm2($h^{\ell-1}$, α, W^ℓ, b^ℓ, $b^{\ell-1}$)
      `// providing` $Q(h^i|h^{i-1})$ `for future use`
  end
end

Algorithm 4: Training DBN

Training set: X
Pre-training:
  $\{W, b\} \leftarrow pre - train(X)$ `// Algorithm 3`
Fine-tuning:
  $\{W, b\} \leftarrow fine - tune(X)$ `// Algorithm 1`

IV. Experimental Results

1. Study Areas

The study areas are Srepok River (Fig. 3) and Dak Nong River (Fig. 4). Dak Nong and Srepok are important rivers in the Central Highlands of Vietnam. Dak Nong basin a total area of about $292km^2$ and is located between latitudes 11.59^o N to 12.15^o N, longitudes 108.40^o E to 108.42^o E, and altitudes about $500m$ to $1500m$ whereas Srepok has a total area of about $18.200km^2$ and is located between latitudes 12.53^o N to 13.55^o N, longitudes 107.30^o E to 108.45^o E and altitudes about $200m$ to $2500m$. Both rivers are surrounded by multiple mountains and have sloped

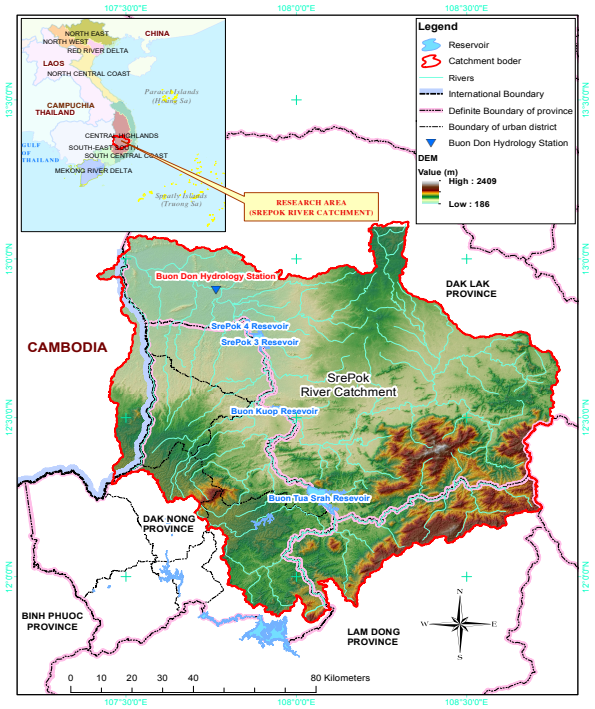


Figure 3: Srepok River.

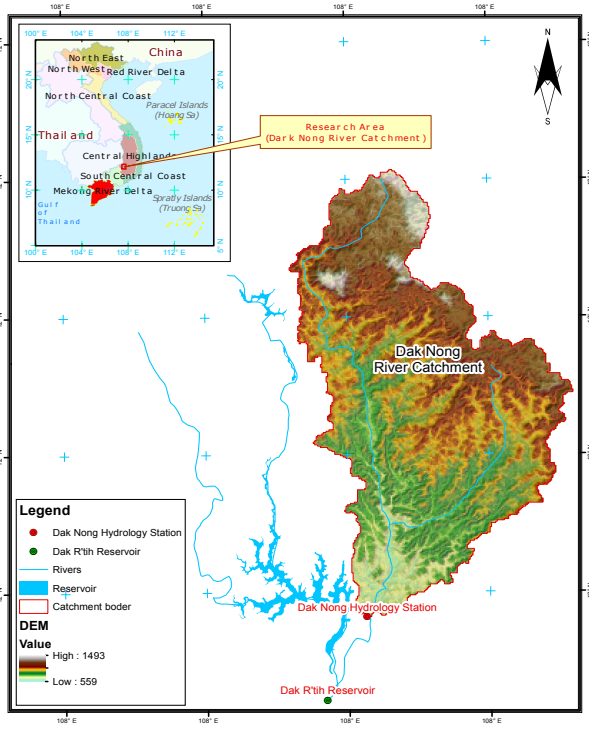


Figure 4: DakNong River.

terrain. The datasets collected contains runoff and rainfall data.

2. Datasets

The runoff of Srepok is collected at Buon Don Hydrology station as presented in Fig. 3 and the runoff of Dak Nong is collected at Dak Nong Hydrology station as presented in Fig. 4. Srepok runoff is divided into 1990-2009 for training and 2010-2011 for testing, while the Dak Nong runoff is divided into 2001-2005 for training and 2006-2007 for testing. Table I presents statistics of the datasets.

3. Data Pre-Processing

The original time series, denoted as $t = \{t_1, t_2, \ldots, t_n\}$, will be converted to S which consists of $s_i = \{t_i, t_{i+1}, \ldots, t_{i+(m-1)\times\tau}\}$, $i = 1, 2, \ldots, n - (m-1) \times \tau$. In which τ (lag) and m (window size) are parameters used for configuration settings, normally $\tau = 1$ and m is set by experience or experiments. For example, for data collected from Dak Nong in the range of Sep. 19, 2007 to Sep. 24, 2007, $t = \{43.1, 45.5, 84.2, 74.4, 68.1, 66.4\}$, with $m = 4$ and $\tau = 1$, we obtain S as follows, in which the i^{th} row is s_i:

$$\mathrm{S} = \begin{bmatrix} 43.1, 45.5, 84.2, 74.4 \\ 45.5, 84.2, 74.4, 68.1 \\ 84.2, 74.4, 68.1, 66.4 \end{bmatrix}$$

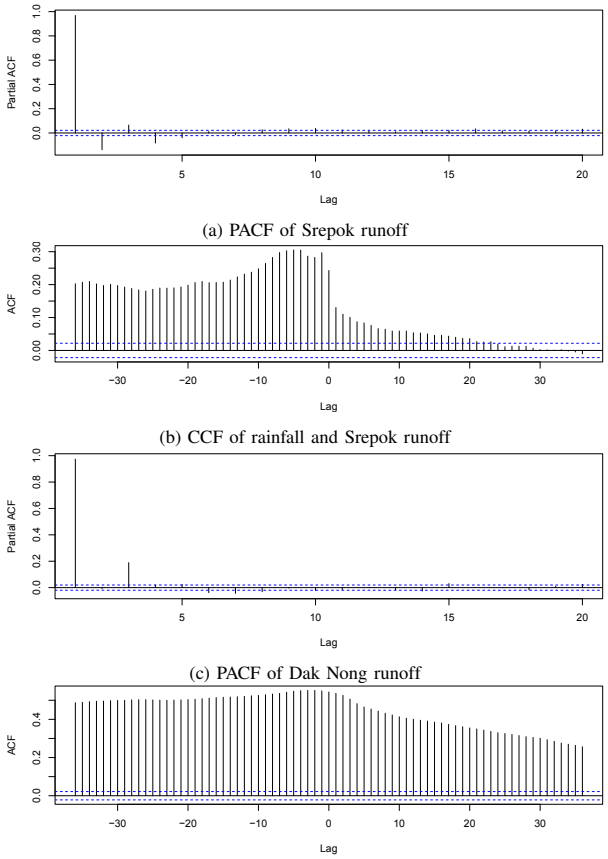
Then each s_i is used to generate a training example. Let $D = \{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})\}$ be the training set, given s_i and its length ℓ, the i^{th} training example $(x^{(i)}, y^{(i)}) = ([s_{i1}, s_{i2}, \ldots, s_{i(\ell-1)}], s_{i\ell})$. For instance, with the above-mentioned example, the training set is generated as follows:

$$\begin{bmatrix} (x^{(1)}, y^{(1)}) = ([43.1, 45.5, 84.2], 74.4) \\ (x^{(2)}, y^{(2)}) = ([45.5, 84.2, 74.4], 68.1) \\ (x^{(3)}, y^{(3)}) = ([84.2, 74.4, 68.1], 66.4) \end{bmatrix}$$

Partial Autocorrelation Functions (PACF) and Cross Correlation Functions (CCF) can be used to identify τ. Therefore, we analyze the relationship between rainfall and runoff data. Fig. 5 presents the partial autocorrelation values of Srepok runoff (Fig. 5a) and Dak Nong runoff (Fig. 5c) whereas Fig. 5b) and Fig. 5d) exhibit the cross correlation values of rainfall with Srepok runoff and rainfall with Dak Nong runoff, respectively. The results demonstrate that the Dak Nong runoff is autoregressive and lag 1 is the most appropriate setting whereas with lags set to the range of -4 to -1, the Dak Nong rainfall has highest cross correlations (approximately 0.55) with the Dak Nong runoff. The partial autocorrelation values indicate that the Srepok rainfall has weak cross correlations with the Srepok runoff (approximately 0.3 at lags of -4, -5 and -6). Naturally, rainfall data commonly have significant impact on runoff, thus the results of PACF are slightly extraordinary.

TABLE I
THE STATISTICS OF RUNOFF TIME SERIES.

River	Period	#Days	Min (m3/s)	Max (m3/s)	Mean	Variance	Standard deviation	Skewness	Kurtosis
Srepok	1990-2011	8036	10.2	3220	294.57	80272.3	283.32	2.9	14.86
Dak Nong	2001-2007	2557	1.2	147	17.69	502.52	22.02	2.06	4.46



(a) PACF of Srepok runoff

(b) CCF of rainfall and Srepok runoff

(c) PACF of Dak Nong runoff

(d) CCF of rainfall and Dak Nong runoff

Figure 5: (a) and (c) exhibit the partial autocorrelation values of Srepok runoff and Dak Nong runoff, respectively whereas (b) and (d) exhibit the cross correlation values of rainfall with Srepok runoff and rainfall with Dak Nong runoff, respectively.

TABLE II
THE PERFORMANCE OF DBNS WITH DIFFERENT CONFIGURATIONS DURING TESTING PHASES TO THE SREPOK RIVER.

#hidden layers	#nodes per a hidden layer	MAPE	RMSE	MAE	R
2	10	16.459	78.672	45.771	0.950
	15	16.264	**78.460**	45.539	0.951
	20	16.412	79.479	**45.041**	**0.952**
	25	16.640	79.991	45.918	0.950
4	10	16.168	78.471	45.191	**0.952**
	15	**16.164**	78.470	45.241	**0.952**
	20	16.427	80.053	46.093	0.949
	25	16.401	79.266	45.718	0.950
6	10	17.050	81.209	46.900	0.948
	15	16.867	80.524	46.255	0.948
	20	16.802	80.151	46.060	0.949
	25	16.538	80.303	46.139	0.948
8	10	16.725	80.932	46.654	0.948
	15	16.357	80.543	46.064	0.948
	20	16.375	80.798	46.134	0.947
	25	16.372	80.542	46.134	0.948
10	10	16.624	80.880	46.479	0.948
	15	16.394	80.847	46.001	0.947
	20	16.325	80.739	46.110	0.948
	25	16.525	80.642	46.291	0.948

4. Results

We run DBN, LSTM, and BiLSTM using a personal computer equipped with CPU core I3 6100 and NVIDIA GTX 1080 8GB GDDR5X PCI Express 3.0 Graphics Card and run MLP-SGD and MLP-PSO on a server equipped with CPU Intel Xeon E5 2620 v4 8-Core 2.1GHz and 128GB RAM DDR4 2133MHz. The performance of trained models is assessed by using four standard statistical performance evaluation criteria. The statistical measures considered are coefficient of correlation (R), Root Mean Square Error (RMSE), Mean Absolute Percentage Error (MAPE), and Mean Absolute Error (MAE) as follows.

$$\mathrm{R} = \frac{\sum_{i=1}^{n}((O_i - \overline{O})(P_i - \overline{P}))}{\sqrt{\sum_{i=1}^{n}(O_i - \overline{O})^2}\sqrt{\sum_{i=1}^{n}(P_i - \overline{P})^2}}, \tag{8}$$

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(O_i - P_i)^2}, \tag{9}$$

$$\mathrm{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\frac{|O_i - P_i|}{O_i}, \tag{10}$$

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}|(O_i - P_i)|, \tag{11}$$

where O_i is the observed runoff at time i; $\overline{O}$ is the average observed runoff; P_i is the predicted runoff at time i; $\overline{P}$ is the average predicted runoff; and n is the number of observed runoff data.

To explore an optimal DBN structure, we use the *trial-and-error* strategy in which the numbers of hidden layers and the numbers of nodes per hidden layer are set in $\{2, 4, 6, 8, 10\}$ and $\{10, 15, 20, 25\}$, respectively. For each configuration of DBNs, learning coefficients including learning rate, epoch, etc. are set as in the training process

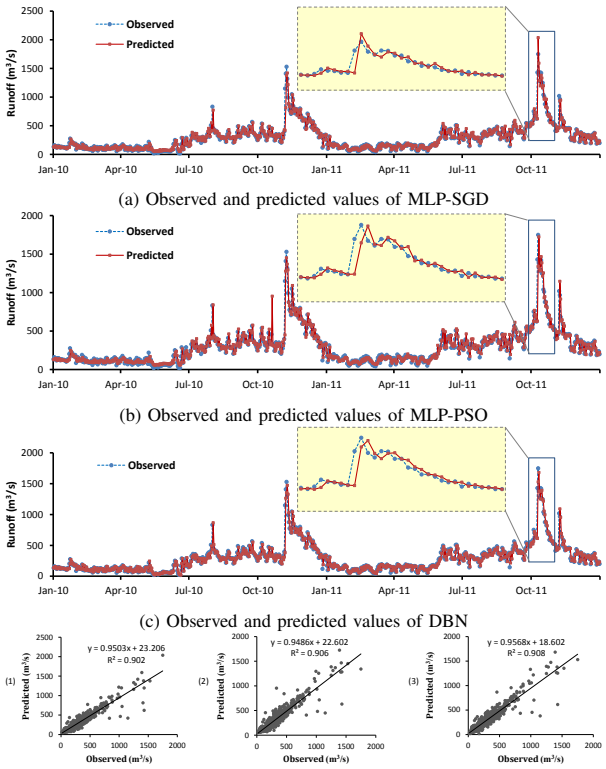
TABLE III
THE PERFORMANCE OF MLP-SGD, MLP-PSO AND DBN DURING TRAINING AND TESTING PHASES ON THE SREPOK RIVER

	Training				**Testing**			
	MAPE	RMSE	MAE	R	MAPE	RMSE	MAE	R
LSTM	7,791	74,530	27,655	0.961	24,253	89,393	53,987	0.930
Bi-LSTM	8,520	73,871	28,683	0.962	26,526	90.664	55,947	0.926
MLP-SGD	9,891	63,870	22,781	0.975	16,922	83,084	47,014	0.950
MLP-PSO	8,429	63,387	22,298	0.976	17,537	81,948	47,228	0.951
DBN	**6,951**	**51,901**	**21,750**	**0.977**	**16,164**	**78,470**	**45,241**	**0.952**

of MLP-SGD. Note that we set up instances of MLP in both models, MLP-SGD and MLP-PSO, consisting of two hidden layers and 20 hidden nodes per layer. Epoch, learning rate values are set to 100, 000, 0.01, respectively, when training MLP-SGD. Table II shows the performance of DBNs with various configurations on the testing runoff of Srepok River. The experimental results indicate that two configurations, in which the numbers of hidden layers and the numbers of nodes per hidden layer are set to 2 and 20 or 4 and 15, give nearly identical and superior performance. With the Dak Nong River, the combination of 4 hidden layers and 25 hidden nodes per hidden layer gave the best performance since the values of MAPE, RMSE, MAE and R are 10.408, 5.038, 2.037 and 0.986, respectively.

Figs. 6 shows the hydrograph and scatter plots of both the observed data and the predicted data obtained by using the MLP-SGD, MLP-PSO and DBN models on both training and testing runoffs. It can be observed from the hydrographs and scatter plots that the predicted results of DBN are closer to the corresponding observed runoff values than those of the other models. As observed from the fit line equations (under the formula of $y = ax + b$) in the scatter plots, the a and b coefficients of DBN are, respectively, 0.9568 and 18.602 leading to a higher R^2 value of 0.908 than those of MLP-SGD and MLP-PSO models. In the testing phase, the Srepok runoff has min, max and average values of 19.9, 1, 750 and 303.85, respectively; and the Dak Nong runoff has min, max and average values of 1.26, 147 and 22.07, respectively. Therefore, MAE values 45.241 and 2.037 corresponding to running of DBN on testing runoffs of Srepok and Dak Nong Rivers indicates that DBN is significantly acceptable to predicting the daily river runoff in short term. In addition, the training cost of DBN is the shortest.

Table III and IV present the comparison of performance of LSTM, BiLSTM, MLP-SGD, MLP-PSO and DBN on training and testing runoff, respectively. The experimental results indicate that DBN significantly outperforms LSTM, BiLSTM, MLP-SGD and MLP-PSO. The results also show that BiLSTM is the worst model and both MLP-SGD



(a) Observed and predicted values of MLP-SGD

(b) Observed and predicted values of MLP-PSO

(c) Observed and predicted values of DBN

(d) Scatter plots of MLP-SGD, MLP-PSO and DBN corresponding to (1), (2) and (3), respectively

Figure 6: The hydrograph and scatter plots of observed and predicted values obtained by using MLP-SGD, MLP-PSO and DBN on the testing dataset of the Srepok River.

and MLP-PSO outperfoms LSTM and BiLSTM. In this study, the population of each swarm is set to 100 and the evolutionary process is iterated for 500 times. LSTM and BiLSTM are trained in 20,000 epoches in which the batch size is set to 365 and Adam is used as the optimizer.

5. Peak analysis

Detecting peaks of any rivers is essential because it can give helpful warnings to timely decision-making. Outliers are detected by using boxplot as an instance presented in Fig 7 where the outliers of Srepok runoff are shown by year. Following a propose in [58], the peaks are outliers whose

TABLE IV
THE PERFORMANCE OF MLP-SGD, MLP-PSO AND DBN DURING TRAINING AND TESTING PHASES ON THE DAK NONG RIVER.

	Training				Testing			
	MAPE	RMSE	MAE	R	MAPE	RMSE	MAE	R
LSTM	8,707	2,986	1,407	0.987	15,934	7,203	3,296	0.966
Bi-LSTM	7,901	2,471	1,156	0.991	16,842	7,693	3,423	0.962
MLP-SGD	14,367	3,570	2,690	0.977	10,542	5,160	2,104	0.984
MLP-PSO	18,766	5,430	3,226	0.978	15,056	6,131	4,139	0.973
DBN	11,031	3,274	1,419	0.981	**10,408**	**5,038**	**2,037**	**0.986**

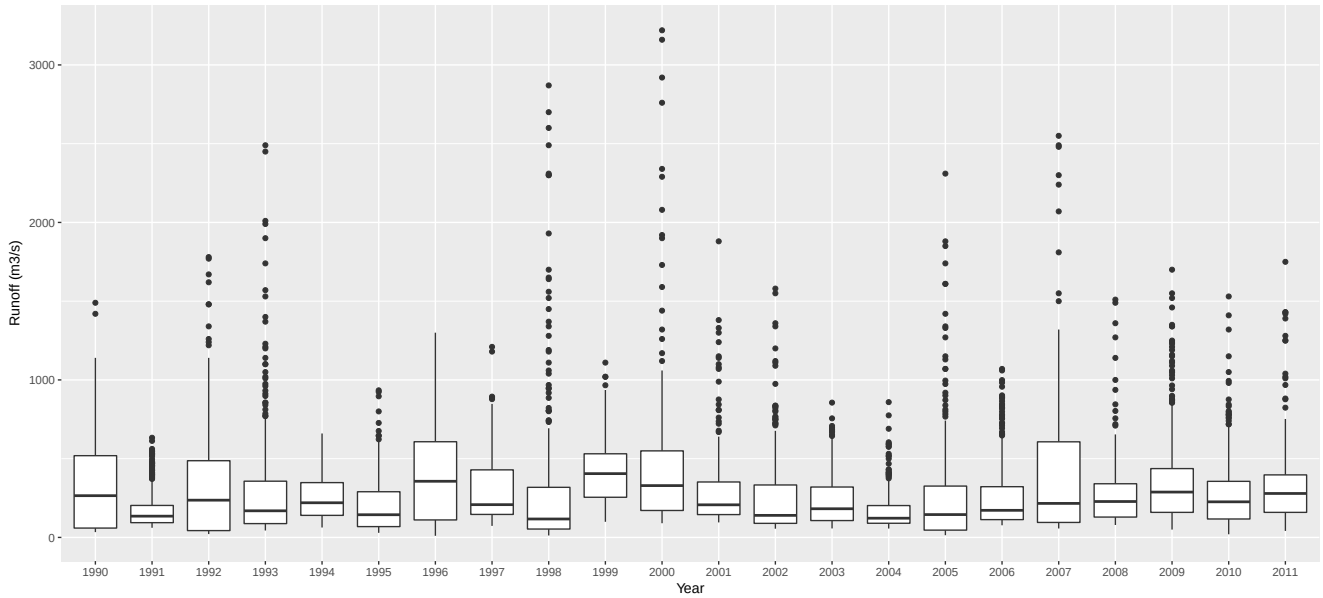


Figure 7: The boxplot of the Srepok runoff by year.

TABLE V
THE PERFORMANCE OF MLP-SGD, MLP-PSO AND DBN IN TERMS OF CAPTURING PEAKS

	Srepok			Dak Nong		
	MAPE	RMSE	MAE	MAPE	RMSE	MAE
MLP-SGD	15,650	268,145	173,631	16,754	27,520	19,136
MLP-PSO	**15,298**	**256,311**	**167,550**	**15,961**	**22,712**	**17,406**
DBN	16,188	261,889	175,440	21,249	38,476	23,755

values higher than 1.5×IQR. In the testing data, there are 35 outliers in Srepok runoff and 25 outliers in Dak Nong runoff, respectively. These outliers are considered as peaks.

Table V shows that the performance of MLP-PSO is superior to MLP-SGD and DBN when capturing the peaks of both Srepok and Dak Nong rivers. While the average peaks of Srepok and Dak Nong rivers are 1041.2 and 108.18, respectively, the best MAE values are 167.550 and 17.406 corresponding to these average peaks. Such predicted results are very far lower than the real values. Therefore, in order to predict peaks of runoff more accuracy, three models MLP-SGD, MLP-SPO, and DBN investigated in this article need to be improved.

V. Conclusion

In this research, we propose a method for river runoff prediction based on Deep Belief Networks. In order to evaluate the method, we conduct experiments on datasets collected from two rivers in Vietnam. The experimental results demonstrate that DBN is an effective method for short-term river runoff prediction. For capturing the peaks of rivers, among methods employed in experiments, MLP-PSO outperforms both MLP-SGD and DBN. Overall, the results confirm that DBN is suitable to employ for the task of runoff prediction.

However, the Contrastive Divergence algorithm used to train DBN has biases and needs to be improved. Moreover, unsupervised representation learning have recently been successful in natural language processing and some breakthrough ideas in that field have been borrowed for time series prediction such as methods based on CBOW [59], sequence-to-sequence model [60] or transformers [61]. Such pre-trained models can be exploited to transfer learnt knowledge to the task of runoff prediction. In addition, deep learning has shown remarkable performance in various fields [23] and can be adapted to runoff prediction. We put those lines of research in future.

REFERENCES

[1] H. Delafrouz, A. Ghaheri, and M. A. Ghorbani, "A novel hybrid neural network based on phase space reconstruction technique for daily river flow prediction," *Soft Computing*, vol. 22, no. 7, pp. 2205–2215, 2018.

[2] E. Gusev, G. Ayzel, and O. Nasonova, "Runoff evaluation for ungauged watersheds by swap model. 1. application of artificial neural networks," *Water Resources*, vol. 44, no. 2, pp. 169–179, 2017.

[3] H. R. Maier, A. Jainb, G. C. Dandya, and K. Sudheer, "Methods used for the development of neural networks for the prediction of water resource variables in river systems: Current status and future directions," *Journal of Hydrology*, vol. 25, pp. 891–909, 2010.

[4] D. An, N. H. Kim, and J.-H. Choi, "Practical options for selecting data-driven or physics-based prognostics algorithms with reviews," *Reliability Engineering & System Safety*, vol. 133, pp. 223–236, 2015.

[5] Z. He, X. Wen, H. Liu, and J. Du, "A comparative study of artificial neural network, adaptive neuro fuzzy inference system and support vector machine for forecasting river flow in the semiarid mountain region," *Journal of Hydrology*, vol. 509, pp. 379–386, 2014.

[6] Z. Liang, Y. Li, Y. Hu, B. Li, and J. Wang, "A data-driven svr model for long-term runoff prediction and uncertainty analysis based on the bayesian framework," *Theoretical and applied climatology*, vol. 133, no. 1-2, pp. 137–149, 2018.

[7] E. Meng, S. Huang, Q. Huang, W. Fang, L. Wu, and L. Wang, "A robust method for non-stationary streamflow prediction based on improved emd-svm model," *Journal of hydrology*, vol. 568, pp. 462–478, 2019.

[8] A. P. Piotrowski and J. J. Napiorkowski, "Monthly river flow forecasting using artificial neural network and support vector regression models coupled with wavelet transform," *Computers & Geosciences*, vol. 54, pp. 1–8, 2013.

[9] H. I. Erdal and O. Karakurt, "Advancing monthly streamflow prediction accuracy of cart models using ensemble learning paradigms," *Journal of Hydrology*, vol. 477, pp. 119–128, 2013.

[10] X. Wen, Q. Feng, R. C. Deo, M. Wu, Z. Yin, L. Yang, and V. P. Singh, "Two-phase extreme learning machines integrated with the complete ensemble empirical mode decomposition with adaptive noise algorithm for multi-scale runoff prediction problems," *Journal of hydrology*, vol. 570, pp. 167–184, 2019.

[11] X. Zhang, Q. Zhang, G. Zhang, Z. Nie, and Z. Gui, "A hybrid model for annual runoff time series forecasting using elman neural network with ensemble empirical mode decomposition," *Water*, vol. 10, no. 4, p. 416, 2018.

[12] P. K. d. M. M. Freire, C. A. G. Santos, and G. B. L. da Silva, "Analysis of the use of discrete wavelet transforms coupled with ann for short-term streamflow forecasting," *Applied Soft Computing*, vol. 80, pp. 494–505, 2019.

[13] M. Khashei and M. Bijari, "A novel hybridization of artificial neural networks and arima models for time series forecasting," *Applied Soft Computing*, vol. 11, no. 2, pp. 2664–2675, 2011.

[14] F.-F. Li, Z.-Y. Wang, and J. Qiu, "Long-term streamflow forecasting using artificial neural network based on preprocessing technique," *Journal of Forecasting*, vol. 38, no. 3, pp. 192–206, 2019.

[15] M. Shoaib, A. Y. Shamseldin, S. Khan, M. M. Khan, Z. M. Khan, T. Sultan, and B. W. Melville, "A comparative study of various hybrid wavelet feedforward neural network models for runoff forecasting," *Water resources management*, vol. 32, no. 1, pp. 83–103, 2018.

[16] M. Zounemat-Kermani, O. Kisi, and T. Rajaee, "Performance of radial basis and lm-feed forward artificial neural networks for predicting daily watershed runoff," *Applied Soft Computing*, vol. 13, no. 12, pp. 4633–4644, 2013.

[17] S. d. O. Domingos, J. F. de Oliveira, and P. S. de Mattos Neto, "An intelligent hybridization of arima with machine learning models for time series forecasting," *Knowledge-Based Systems*, vol. 175, pp. 72–86, 2019.

[18] S. M. Hosseini and N. Mahjouri, "Integrating support vector regression and a geomorphologic artificial neural network for daily rainfall-runoff modeling," *Applied Soft Computing*, vol. 38, pp. 329–345, 2016.

[19] V. Nourani, A. H. Baghanam, J. Adamowski, and O. Kisi, "Applications of hybrid wavelet–artificial intelligence models in hydrology: a review," *Journal of Hydrology*, vol. 514, pp. 358–377, 2014.

[20] M. Shoaib, A. Y. Shamseldin, B. W. Melville, and M. M. Khan, "A comparison between wavelet based static and dynamic neural network approaches for runoff prediction," *Journal of Hydrology*, vol. 535, pp. 211–225, 2016.

[21] D. Rolnick, P. L. Donti, L. H. Kaack, K. Kochanski, A. Lacoste, K. Sankaran, A. S. Ross, N. Milojevic-Dupont, N. Jaques, A. Waldman-Brown *et al.*, "Tackling climate change with machine learning," *arXiv preprint arXiv:1906.05433*, 2019.

[22] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[23] S. Pouyanfar, S. Sadiq, Y. Yan, H. Tian, Y. Tao, M. P. Reyes, M.-L. Shyu, S.-C. Chen, and S. Iyengar, "A survey on deep learning: Algorithms, techniques, and applications," *ACM Computing Surveys (CSUR)*, vol. 51, no. 5, p. 92, 2018.

[24] Y. Chen, Y. Kang, Y. Chen, and Z. Wang, "Probabilistic forecasting with temporal convolutional neural network," *In KDD 2019, Workshop on Mining and Learning from Time Series*, 2019.

[25] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Deep learning for time series classification: a review," *Data Mining and Knowledge Discovery*, pp. 1–47, 2019.

[26] T. Kuremoto, S. Kimura, K. Kobayashi, and M. Obayashi, "Time series forecasting using a deep belief network with restricted boltzmann machines," *Neurocomputing*, vol. 137, pp. 47–56, 2014.

[27] M. Längkvist, L. Karlsson, and A. Loutfi, "A review of unsupervised feature learning and deep learning for time-series modeling," *Pattern Recognition Letters*, vol. 42, pp. 11–24, 2014.

[28] S.-Y. Shih, F.-K. Sun, and H.-y. Lee, "Temporal pattern attention for multivariate time series forecasting," *Machine Learning*, 2019. [Online]. Available:

https://doi.org/10.1007/s10994-019-05815-0

[29] C. Zhang, P. Patras, and H. Haddadi, "Deep learning in mobile and wireless networking: A survey," *IEEE Communications Surveys and Tutorials*, 2019.

[30] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[31] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[32] Y. Bengio *et al.*, "Learning deep architectures for ai," *Foundations and trends® in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[33] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. of the IEEE International Conference on Neural Networks*. Perth, Australia: IEEE, 1995, pp. 1942–1945.

[34] F. Kratzert, D. Klotz, C. Brenner, K. Schulz, and M. Herrnegger, "Rainfall–runoff modelling using long short-term memory (lstm) networks," *Hydrol. Earth Syst. Sci*, vol. 22, no. 11, pp. 6005–6022, 2018.

[35] T. T. Tran, N. N. Giang, H. N. Duong, H. T. Nguyen, T. Van Hoai, and V. Van Nghi, "A comprehensive study on predicting river runoff," in *2017 9th International Conference on Knowledge and Systems Engineering (KSE)*. IEEE, 2017, pp. 251–256.

[36] X. He, J. Luo, G. Zuo, and J. Xie, "Daily runoff forecasting using a hybrid model based on variational mode decomposition and deep neural networks," *Water Resources Management*, vol. 33, no. 4, pp. 1571–1590, 2019.

[37] T. Xie, G. Zhang, J. Hou, J. Xie, M. Lv, and F. Liu, "Hybrid forecasting model for non-stationary daily runoff series: A case study in the han river basin, china," *Journal of Hydrology*, p. 123915, 2019.

[38] R. Salakhutdinov and G. Hinton, "Deep boltzmann machines," in *Artificial intelligence and statistics*, 2009, pp. 448–455.

[39] R. Cheng and Y. Jin, "A social learning particle swarm optimization algorithm for scalable optimization," *Information Sciences*, vol. 291, pp. 43–60, 2015.

[40] A. Ahani, M. Shourian, and P. R. Rad, "Performance assessment of the linear, nonlinear and nonparametric data driven models in river flow forecasting," *Water resources management*, vol. 32, no. 2, pp. 383–399, 2018.

[41] Z. Alizadeh, J. Yazdi, J. Kim, and A. Al-Shamiri, "Assessment of machine learning techniques for monthly flow prediction," *Water*, vol. 10, no. 11, p. 1676, 2018.

[42] A. K. Poul, M. Shourian, and H. Ebrahimi, "A comparative study of mlr, knn, ann and anfis models with wavelet transform in monthly stream flow prediction," *Water Resources Management*, pp. 1–17, 2019.

[43] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient backprop," in *Neural networks: Tricks of the trade*. Springer, 2012, pp. 9–48.

[44] Y. Liu, L. Ye, H. Qin, S. Ouyang, Z. Zhang, and J. Zhou, "Middle and long-term runoff probabilistic forecasting based on gaussian mixture regression," *Water Resources Management*, vol. 33, no. 5, pp. 1785–1799, 2019.

[45] Ö. Terzi, "A genetic programming approach to river flow modeling," *Journal of Intelligent & Fuzzy Systems*, vol. 27, no. 5, pp. 2211–2219, 2014.

[46] M. A. Ghorbani, R. Khatibi, A. D. Mehr, and H. Asadi, "Chaos-based multigene genetic programming: A new hybrid strategy for river flow forecasting," *Journal of hydrology*, vol. 562, pp. 455–467, 2018.

[47] X. Yuan, C. Chen, X. Lei, Y. Yuan, and R. M. Adnan, "Monthly runoff forecasting based on lstm–alo model," *Stochastic environmental research and risk assessment*, vol. 32, no. 8, pp. 2199–2212, 2018.

[48] S. Mouatadid, J. F. Adamowski, M. K. Tiwari, and J. M. Quilty, "Coupling the maximum overlap discrete wavelet transform and long short-term memory networks for irrigation flow forecasting," *Agricultural Water Management*, vol. 219, pp. 72–85, 2019.

[49] R. Feng, G. Fan, J. Lin, B. Yao, and Q. Guo, "Enhanced long short-term memory model for runoff prediction," *Journal of Hydrologic Engineering*, vol. 26, no. 2, p. 04020063, 2021.

[50] M. Gauch, F. Kratzert, D. Klotz, G. Nearing, J. Lin, and S. Hochreiter, "Rainfall–runoff prediction at multiple timescales with a single long short-term memory network," *Hydrology and Earth System Sciences*, vol. 25, no. 4, pp. 2045–2062, 2021.

[51] Y. Bengio, N. Boulanger-Lewandowski, and R. Pascanu, "Advances in optimizing recurrent networks," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013, pp. 8624–8628.

[52] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *1998 IEEE international conference on evolutionary computation proceedings. IEEE world congress on computational intelligence (Cat. No. 98TH8360)*. IEEE, 1998, pp. 69–73.

[53] J.-R. Zhang, J. Zhang, T.-M. Lok, and M. R. Lyu, "A hybrid particle swarm optimization–back-propagation algorithm for feedforward neural network training," *Applied mathematics and computation*, vol. 185, no. 2, pp. 1026–1037, 2007.

[54] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[55] G. E. Hinton, "A practical guide to training restricted boltzmann machines," in *Neural Networks: Tricks of the Trade: Second Edition*, G. Montavon, G. B. Orr, and K.-R. Müller, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 599–619.

[56] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Advances in neural information processing systems*, 2007, pp. 153–160.

[57] A. Fischer and C. Igel, "Training restricted boltzmann machines: An introduction," *Pattern Recognition*, vol. 47, no. 1, pp. 25–39, 2014.

[58] R. McGill, J. W. Tukey, and W. A. Larsen, "Variations of box plots," *The American Statistician*, vol. 32, no. 1, pp. 12–16, 1978.

[59] J.-Y. Franceschi, A. Dieuleveut, and M. Jaggi, "Unsupervised scalable representation learning for multivariate time series," in *Advances in Neural Information Processing Systems*, 2019, pp. 4652–4663.

[60] X. Lyu, M. Hueser, S. L. Hyland, G. Zerveas, and G. Rätsch, "Improving clinical predictions through unsupervised time series representation learning," *Machine Learning for Health (ML4H) Workshop at NeurIPS 2018*, 2018.

[61] N. Wu, B. Green, X. Ben, and S. O'Banion, "Deep transformer models for time series forecasting: The influenza prevalence case," *arXiv preprint arXiv:2001.08317*, 2020.

Tran Thi Truong Thi is lecturer of Hoa Sen University, Ho Chi Minh City, Vietnam. She received her M.S from Ho Chi Minh City University of Technology, Ho Chi Minh City, Vietnam in 2005. Her research interests are data science, visual question answering (VQA).
Email: thi.tranthitruong@hoasen.edu.vn

Hieu N. Duong is deputy director of the Center of Computer Engineering, Ho Chi Minh City University of Technology. Before playing the role of deputy director, he used to be a lecturer at the Faculty of Computer Science and Engineering, Ho Chi Minh City University of Technology from 2002-2018. He received his PhD from VSB-Technical University of Ostrava, Czech Republic in 2016. His research interests are Internet of Things, data science.
Email: dnhieu@hcmut.edu.vn

Hien T. Nguyen is an Associate Professor at the Banking University of Ho Chi Minh City (BUH). He received his Ph.D. degree in Computer Science from Ho Chi Minh City University of Technology (HCMUT) in 2011. His research interests include Natural Language Processing, Machine Learning, Social Network Analysis, and Time Series. He has published more than 60 peer-reviewed scientific papers in international journals and conferences. Before joining BUH, he served as Dean of Faculty of Information Technology, Ton Duc Thang University. He is presently an Associate Editor of Computational Social Networks journal indexed in Scopus.
Email: nthien@buh.edu.vn

Van Hoai Tran is an associate professor at Ho Chi Minh University of Technology. He received his PhD degree of applied mathematics from Heidelberg University in 2005. His theoretical background is combinatorial optimization. His research interests are metagenomic binning in bioinformatics, convexity in computational geometry and practical optimization problems.
Email: hoai@hcmut.edu.vn