A Comparison Between Stacked Auto-Encoder and Deep Belief Network in River Run-Off Prediction

Bui Tan Kinh(✉), Duong Tuan Anh, and Duong Ngoc Hieu

Faculty of Computer Science and Engineering, HCM University of Technology, Ho Chi Minh, Vietnam
tankinhlk@gmail.com, dtanhcse@gmail.com, dnhieu@hcmut.edu.vn

Abstract. The application of deep neural networks in forecasting hydrological time series data is increasingly popular, aiming to improve prediction accuracy in this challenging problem. As for river runoff prediction, Deep Belief Network (DBN) and Stacked Autoencoder (SAE) are two kinds of deep neural networks which are commonly used for extracting meaningful features from the data before prediction. In this study, we aim to compare the prediction performance of SAE model with that of DBN model on the runoff data of Srepok River in Central Highlands of Vietnam. Experiments are conducted by using historical data of the Srepok River that were collected in 11 years. The experimental results in this case study show that SAE brings out better prediction accuracy than DBN in terms of three evaluation criteria: correlation, root mean square error, and mean absolute percentage error.

Keywords: Runoff prediction · Stack autoencoder · Deep belief network · Srepok river

1 Introduction

Time series analysis includes methods for analyzing time series data, from which meaningful statistical attributes and data characteristics can be extracted to serve some purposes such as: prediction, classification, clustering for time series. Time series prediction is the use of a model to predict time events based on known events in the past and thereby predicting data points before they occur (or are measured).

The prediction of river run-off is very important in water resource planning and management. In this research, we focus on river run-off prediction of Srepok River (Vietnamese: Sêrêpok) which is a major tributary of Mekong River in the Central Highlands of Vietnam.

In general, there are two main approaches to solving the river runoff prediction problem: physical-based model and data-driven model [1]. The main disadvantage of the physical-based method is that it requires diverse kinds of

P. C. Vinh and A. Rakib (Eds.): ICCASA 2020/ICTCC 2020, LNICST 343, pp. 65–81, 2021.
https://doi.org/10.1007/978-3-030-67101-3_6

data, ranging from climate, water resource to soil map data. In contrast, data-driven model requires little information, are easy to implement, and do not need experienced specialists.

Among several methods used to forecast hydrological time series data such as rainfall, reservoir inflow, and river runoff, Artificial Neural Networks (ANNs) have been applied commonly [2–5]. But forecasting in hydrological time series data is still a challenging task since forecasting power on this kind of time series with some proposed methods is limited. As for ANN, river runoff time series will increase the feature learning difficulties and the network computation complexities.

Deep neural network models, such as Deep Belief Networks (DBNs), Long Short Term Memory Networks (LSTMs) and Stacked Auto-encoders (SAEs), have recently attracted the interest of many researchers in some applications on big data analysis. For recent years, there have been several researches of applying deep neural networks to predict time series data in various fields. Particularly, some research works on forecasting rainfall and runoff time series can be listed as follows. Gope et al. (2016) [6] employed Stacked Autoencoder in rainfall prediction. Tri et al. (2016) applied Deep Belief Network in daily runoff prediction for a river in Vietnam [7]. Hernandez et al. (2016) applied stacked auto-encoder networks in rainfall prediction [8]. Li et al. in (2016) [9] applied Deep Belief Network in reservoir inflow forecasting. Kratzert et al. (2018) [10] employed Long Short-Term Memory (LSTM) network in rainfall-runoff modeling. Lee et al. (2020) applied Long Short Term Memory Network in runoff analysis for Red River in Vietnam [11]. From these above mentioned research works, DBN model and SAE model are the two kinds of deep neural networks which are commonly-used in river runoff prediction. However, so far there exists no work to answer the question that between DBN model and SAE model which one performs better in river runoff prediction.

In this study, we aim to compare the prediction performance of SAE model with that of Deep belief network (DBN) on the daily runoff data of Srepok River in Vietnam. Experiments are conducted by using historical data of the Srepok River that were collected in 11 years. The experimental results in this case study show that SAE brings out better prediction accuracy than DBN in terms of three evaluation criteria: coefficient of correlation (R), root mean square error (RMSE), mean absolute percentage error (MAPE).

The remainder of the paper is organized as follows. Section 2 provides some basic backgrounds about Deep Belief Network and Stacked Autoencoder. In Sect. 3, the proposed architectures of DBN and SAE for runoff prediction are introduced. Section 4 reports the experiments to compare the prediction accuracy of SAE and that of DBN model in runoff prediction. Finally, Sect. 5 gives some conclusions and future works.

2 Theoretical Background

2.1 Deep Belief Network

Deep Belief Networks have been proposed by Hinton [12] with remarkable success in image processing and AI areas. DBN models are based on stacking of Restricted Boltzmann Machines (RBMs) [13].

RBM is a kind of stochastic artificial neural network with two connected layers: a layer of binary visible units (v, whose states are observed) and a layer of binary hidden units (h, whose states cannot be observed). The hidden units act as latent variables (features) that allow the RBM to model probability distribution over state vectors (see Fig. 1). The hidden units are conditionally independent given visible units.

Given an energy function *E(v, h)* on the whole set of visible and hidden units, the joint probability is given by:

$$p(v,h) = \frac{e^{-E(v,h)}}{Z} \tag{1}$$

where Z is a normalization partition function, which is obtained by summing up the energy of all possible *(v, h)* configurations.

$$Z = \sum_{v,h} e^{-E(v,h)} \tag{2}$$

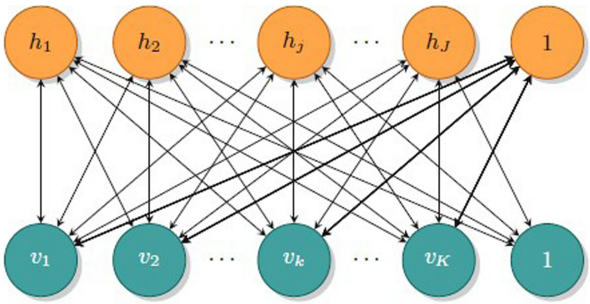


Fig. 1. Restricted Boltzmann Machine (RBM)

For the binary units $h_i \in \{0, 1\}$ and $v_i \in \{0, 1\}$, the energy function of the whole configuration is:

$$\begin{aligned} E(v,h) &= -cv^T - bh^T - hWv^T \\ &= -\sum_{k=1}^{K} c_k v_k - \sum_{j}^{J} b_j h_j - \sum_{j}^{J}\sum_{k=1}^{K} W_{jk} v_k hj \end{aligned} \tag{3}$$

where W is $J \times K$ matrix of RBM weights connecting hidden and visible units, $c = [c_1, c_2, ..., c_K]$ is the bias of the visible units and $b = [b_1, b_2, ..., b_J]$ is the bias of the hidden units. The marginal distribution over v is:

$$p(v) = \sum_h (v, h) \tag{4}$$

The posterior probability of one layer given the other is easy to compute by the two following equations:

$$\begin{aligned} p(h, v) &= \prod_j p(h_j = 1|v) \\ \text{where } p(h_j = 1|v) &= \sigma(\, b_j + \sum_k W_{jk} v_k) \end{aligned} \tag{5}$$

$$\begin{aligned} p(v, h) &= \prod_k p(v_k = 1|h) \\ \text{where } p(v_k = 1|h) &= \sigma(\, c_k + \sum_j W_{jk} h_j) \end{aligned} \tag{6}$$

Notice that σ is the sigmoid function. Inference of hidden factor **h** given the observed **v** can be done because **h** is conditionally independent given **v**.

A DBN is a generative model with an input layer and an output layer, separated by l layers of hidden stochastic units. This multilayer neural network can be efficiently trained by composing RBMs in such a way that the feature activations of one layer are used as the training data for the next layer.

An energy-based model of RBMs can be trained by performing gradient ascent on the log-likelihood of the training data with respect to the RBM parameters. This gradient is difficult to compute analytically. Gibbs sampling method is well-suited for RBMs. One iteration of the Gibbs sampling works well and corresponding to the following sampling procedure:

$$v_0 \xrightarrow{p(h_0|v_0)} h_0 \xrightarrow{p(v_1|h_0)} v_1 \xrightarrow{p(h_1|v_1)} h_1 ... h_{k-1} \xrightarrow{p(h_{k-1}|v_k)} v_k \tag{7}$$

where the sampling operations are schematically described. It is known that Gibbs sampling is very time-consuming. Rough estimation of the gradient using the above procedure is denoted by CD-k, where CD-k represents the Contrastive Divergence algorithm [12] for performing k iterations of Gibbs sampling up to v_k.

The weight parameter is updated with the rate of change as shown in the following formula:

$$\nabla W_{jk} = \eta(\langle v_k h_j \rangle_0 - \langle v_k h_j \rangle_k) \tag{8}$$

where η represents the learning rate.

2.2 Stacked Auto-Encoder

Autoencoder
Autoencoder (AE) is a deep learning neural network model, in which it tries to recreate the input. The Autoencoder basic network is an unsupervised one layered neural network where the input is $X = x_1, x_2, x_3, ..., x_n$. as a characteristic vector with n dimensions. The network output is calculated using the following formula:

$$h_{W,b}(X) = f(W^T X) = f(\sum_{i=1}^{n} W_i X_i + b) \tag{9}$$

where f: $\mathbb{R} \mapsto \mathbb{R}$ is the nonlinear conversion function, W and b are the weights and bias of the corresponding network. The goal is to try to approximate the function $h_{W,b}(X) \approx X$ in order to learn the characteristics of X and recreate it.

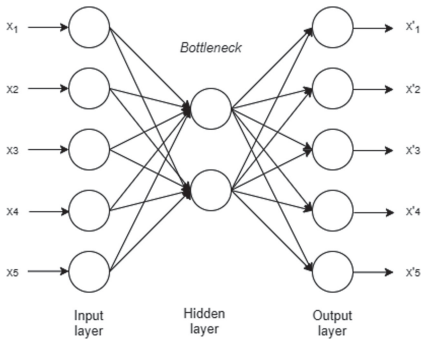


Fig. 2. Autoencoder neural network

Figure 2 describes the Autoencoder network which has two basic components:

- Encoder: The first layer is an encoder, which is simply a group of fully connected layers or convolution layers taking the input and compressing it into a smaller representation with a smaller size. The smaller representation of input is called the bottleneck. Here knowledge or features from the input will be compressed, retained important features and used to pass through the decoder.
- Decoder: The decoder also has the same architecture as the encoder, but the difference is that it takes the knowledge from the bottleneck then tries to reconstruct the input by using the fully connected layers or convolution layers.

The loss function of the Autoencoder network with a hidden layer is as follows:

$$J(W,b) = [\frac{1}{m}\sum_{i=1}^{m}\frac{1}{2}\|h_{W,b}(x_i) - x_i\|^2] + \frac{\lambda}{2}\sum_{i}^{n}\sum_{j}^{nhid}(W_{j,i})^2 \quad (10)$$

where m is the number of training examples, *nhid* considered only one hidden layer is the number of units in the hidden layer and λ is the weight decay parameter.

The goal of autoencoder network is to learn how to represent the features of a dataset, by training the network to reduce noise from input in order to retain valuable information and features from the input data. There are many variations of autoencoder networks such as: Denoising Autoencoder, Sparse Autoencoder, Stacked Autoencoder, Convolutional Autoencoder, etc. Depending on the different goals of the problem, we use different models to solve the problem.

Stacked Autoencoder

The Stacked Autoencoder (SAE) is formed by stacking autoencoder to make a deep neural network, where each autoencoder is independently trained, and the output of each hidden layer of autoencoder is connected to the input of the next hidden layer, Fig. 3 illustrates how to stack an autoencoder neural network.

The hidden layers are trained by an unsupervised algorithm and then refined by a supervised method. The Stack autoencoder network is trained as follows:

1. Train the first autoencoder with input data and obtain the feature vector.
2. Feature vector of the previous layer is used as input to the next layer and the process is repeated until the training is completed.
3. Use the output of the last layer as input for prediction layer.
4. After all the hidden layers have been trained, the backpropagation algorithm is used to minimize the cost function and updated the weights with the training set labeled to achieve fine-tuning in a supervised way.

Based on the advantages of the Stacked autoencoder network, the model can create new data with more compressed knowledge and features which are used to train a new model to classify or predict, to find greater accuracy than using the original data. There have been studies showing that the use of Stacked autoencoder network instead of Autoencoder in preprocessing data can improve the accuracy of the classification model [14].

3 Two Model Architectures

3.1 Deep Belief Network

The DBN model that will be used in this work has the structure given in Fig. 4. In Fig. 4 the DBN model consists of only two RBMs and one MLP but in general, we can add some more RBMs into the DBN model to make the model deeper. The number of RBMs in DBN model is one of the important hyper-parameters of this deep learning model. Notice that in DBN model, the hidden layer of one

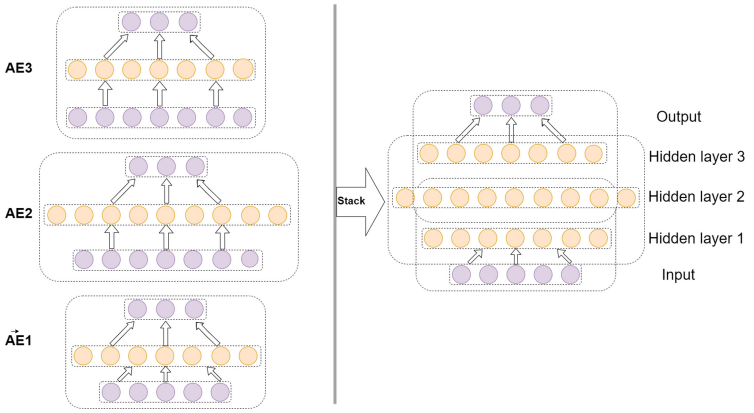


Fig. 3. How to stack autoencoder neural network

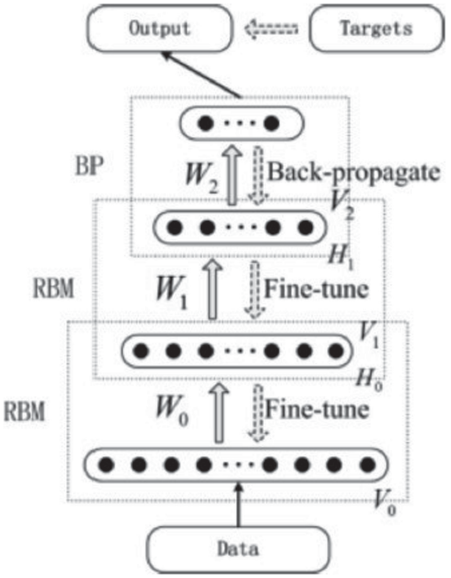


Fig. 4. The DBN model with two RBMs

RBM plays the same role of the visible layer of the RBM at the next (higher) level.

The training algorithm for our proposed DBN consists of two stages: an unsupervised learning and a supervised learning.

The unsupervised learning stage is the Contrastive Divergence (CD) algorithm [12] used for training the RBM(s). The CD algorithm progresses on a layer-by-layer basis. First, a RBM is trained directly on the input data. Hence, the neurons in the hidden layer of the RBM can capture the important features of the input data. The activations of the trained features are then used as "input data" to train a second RBM.

The supervised learning stage is the back-propagation algorithm used for training the MLP.

3.2 Stacked Auto-Encoder

In this section, we describe the general architecture of our proposed SAE model which can predict runoff data of the river for the following days.

The architecture of the model consisting of some stacked autoencoders, which is built by stacking the autoencoders together in order to extract meaningful features from the data. How to design the SAE model architecture for prediction will be discussed in the next section. We will try different configurations, the number of different encoders, the size of the sliding window, and so on to find the model with the best performance. A block diagram of our SAE model is shown in Fig. 5

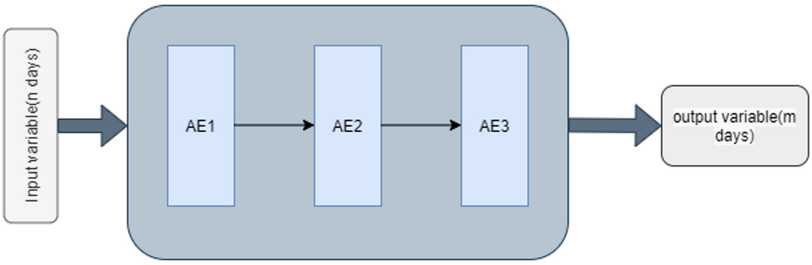


Fig. 5. Proposed stacked autoencoder architecture

In this work, we will test the number of input nodes with eight different values which will be reported later.

3.3 Sliding Window Technique

In order to preprocess the runoff data before entering a deep learning model, we use sliding window technique. From the initial time series $X = x_1, x_2, x_3, ..., x_n$ where n is the number of points (days), the data will be converted into $y_i = x_i, x_{i+1}, x_{i+2}, ..., x_{i+m}$ where m is the size of the sliding window. We will have a new input data is $Y = y_1, y_2, ..., y_n$. Then we apply Y to the predictor model.

3.4 Performance Evaluation

To evaluate the performance of a predictive model, the three statistical criteria applied are root mean square error (RMSE) and coefficient of correlation (R) and mean absolute percentage error (MAPE). The three criteria are computed as follows:

$$\begin{aligned} RMSE &= \sqrt{\frac{1}{n}\sum_{i=1}^{n}(O_i - P_i)^2} \\ R &= \frac{\sum_{i=1}^{n}(O_i - \overline{O})(P_i - \overline{P})}{\sqrt{\sum_{i=1}^{n}(O_i - \overline{O})^2(P_i - \overline{P})^2}} \\ MAPE &= \frac{100}{n}\sum_{i=1}^{n}\left|\frac{O_i - P_i}{O_i}\right| \end{aligned} \tag{11}$$

where n is the number of observations, O_i is the actual value at time point i, $\overline{O}$, is the average actual value and P_i is the forecast value for time point i and $\overline{P}$ is the average predicted value. Notice that R value is higher, the prediction accuracy is better. *RMSE* or *MAPE* value is lower, the prediction accuracy is better. RMSE is an absolute performance measure while MAPE is a relative measure.

4 Experimental and Evaluation

4.1 Study Area

For this study, the Srepok River in Viet Nam has been selected as the study region. The runoff data are collected from Cau14, BuonHo and DucXuyen measurement stations (Fig. 6). They are three measurement stations of the Srepok river in three different locations. A data point is representative of the flow collected at a fixed time of day in a given river basin. With Cau14 and BuonHo measurement stations, we collected 4000 data points corresponding to 4000 days (approximately 11 years) of runoff, while with DucXuyen measuring station we collected 3000 data points. The evaluation criteria of the models' performances are also given Sect. 3.4.

Srepok river [15] (the Khmer name Tonlé Srepok) is an important tributary of the Mekong river. The river originates in Dak Lak, then Dak Nong provinces of Vietnam, and flows through Ratanakiri and Stung Treng provinces, Cambodia. In Vietnam there is a river section called the Dak Krong River. Its length varies about 406 km in which the last 281 km course is in Cambodian territory. The Srepok River, in turn, has three main tributaries: the Krông Nô, Krông Ana, and Ea H'leo Rivers. Before joining the Mekong, the Srepok also merges with the Sesan River and Kong River in Stung Treng province.

The data collected will be divided into two subsets which are training and testing sets. We normalized the time series values to between zero and one interval in order to avoid effects of scale to our deep learning architecture. Data

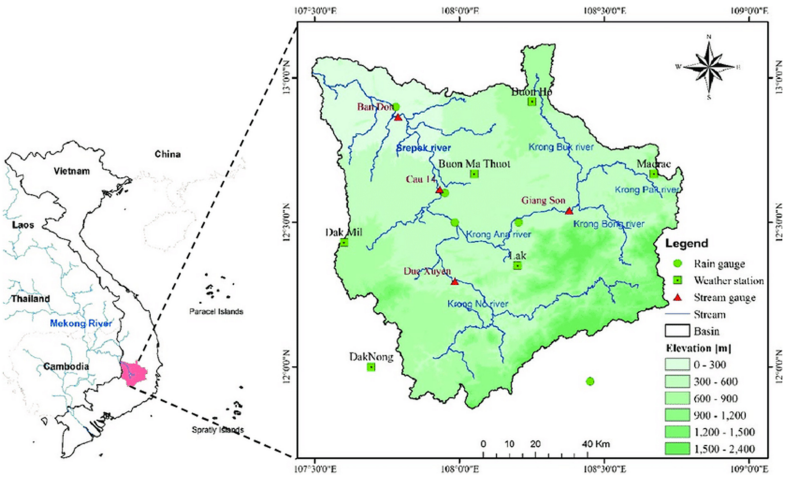


Fig. 6. The location of study area [20]

normalization method is shown in Eq. 12, where x_i represents a value to normalize at the i_{th} time point, min_{x_i} is the minimum value for the time series in the training set/testing set and max_{x_i} is the maximum value for the time series in the training set/testing set. We will normalize separately for training set and testing set.

$$x_i = \frac{x_i - min_{x_i}}{max_{x_i} - min_{x_i}} \tag{12}$$

4.2 Experimental Results

In this section, we will evaluate the performance of SAE model in river runoff prediction by comparing its performance with that of DBN model on the same dataset. The evaluation criteria given in Sect. 3.4 will be used to measure prediction performance.

We will build DBN model based on the best hyper-parameters selected for the DBN prediction model reported in the previous work by Tri et al. [7]. In sum, the good DBN model consists of 4 hidden layers and 5 neurons per layer.

We will not compare the two deep neural network models with ANN since through experiment some previous works [7,9] pointed out that DBN model always outperforms ANN in runoff prediction.

The two comparative deep learning models were implemented in Python with open-source framework Keras [16]. The experiments on four time series datasets were conducted on the PC with CPU Intel(R) Core(TM) i7-8750H CPU @ 2.20 GHz, 16 GB RAM, NVIDIA GeFore GTX 1050Ti. To evaluate the performance of the models and their configurations, we used Cau14 dataset as the standard dataset. The dataset is divided into two parts, the first 10 years (3672 days) for training and the remaining year (328 days) for testing.

To assess the impact of sliding window size on the overall performance of the prediction model, we applied the manual trial-and-error strategy to find out which window size brings out the best performance [17]. For the parameter in the SAE model, the number of epochs was set to 1000 times for sufficient learning of the neural network. Three AE layers are used to extract features and for each AE layer, the corresponding number of nodes in the hidden layer is in turn set to 200, 400, 200. Finally, to consider the temporal continuity of the runoff on Srepok River, the sequence length was changed via eight cases: 2 days, 5 days, 7 days, 8 days, 14 days, 15 days, 20 days and 28 days to investigate the reproducibility of the prediction results according to the length change of the sliding window. Table 1 summarizes the evaluation results in varying the sliding window size m. The lowest error result of RMSE is 45,401 ($m = 7$) and MAPE is 13,897 ($m = 14$). The highest correlation R is 0.9771 ($m = 5$, $m = 7$).

Table 1. Experimental results of SAE model in varying sliding window size.

	Size of sliding window	RMSE	R	MAPE
1	2	48.345	0.9760	14.932
2	5	52.993	**0.9771**	16.208
3	7	**45.401**	0.9771	**14.039**
4	8	49.467	0.9766	15.417
5	14	47.525	0.9763	13.897
6	15	50.060	0.9753	15.600
7	20	50.347	0.9756	14.549
8	28	55.711	0.9732	17.232

We found out that the best case for the length of sliding window is $m = 7$. From the analysis on Table 1, the larger sliding window size has an effect on the prediction performance of the model but it does not mean that the larger the window slide, the higher the prediction performance.

In addition to the effect of sliding window size on prediction, selecting the network size also affects to the predictive performance greatly. This is a typical problem for all neural network design. Tables 2, 3 and 4 show the results of changing the number of hidden nodes per layer and the number of AEs for predictions. Table 2 shows the results of varying the number of hidden nodes per layer when sliding windows with a length of 7 and three AEs are used in this experiment. From Table 2 we can discover the best results in terms of RMSE, R, MAPE measures. In this experiment two sets of values (200, 400, 200) and (200, 600, 200) are the optimal choices. Therefore, we will pick (200, 400, 200) for the number of hidden nodes per layer in our SAE model.

Tables 3 and 4 show the analysis results when changing the number of AEs, we find that the number of AEs has an impact on the prediction performance and the large number of AEs does not mean that the prediction performance

Table 2. Experimental results in varying the number of nodes in one layer of SAE model (Cau14 dataset).

	Nodes on one layer	RMSE	R	MAPE
1	200, 200, 200	54.289	0.9765	16.383
2	400, 400, 400	46.052	0.9766	13.863
3	400, 200, 400	46.158	0.9760	13.954
4	200, 600, 200	46.036	0.9761	**13.634**
5	200, 400, 200	**45.401**	**0.9769**	14.039

will be high. We select the number of AEs and the number of hidden nodes per layer to give the best predictive performance. From Table 3, with the number of hidden nodes per layer of 400, the best ratings are RMSE = 45,789 (Number of AE layers = 2), R = 0.9766 (Number of AE layers = 3), MAPE = 13.863% (Number of AE layers = 3). From Table 4 with a number of hidden nodes per layer of 200, the best indicators are RMSE = 47.454 (Number of AE layers = 2), R = 0.9766 (Number of AE layers = 4), MAPE = 14,390% (Number of AE layers = 4).

Table 3. Performance results in varying the number of AE layers (each AE layer has 400 nodes per layer) in SAE model.

	Number of AE layers	RMSE	R	MAPE
1	2	**45.789**	0.9761	13.916
2	3	46.052	**0.9766**	**13.863**
3	4	53.078	0.9718	16.625

Table 4. Performance results in varying the number of AE layers (each AE layer has 200 nodes per layer) of SAE model.

	Number of AE layers	RMSE	R	MAPE
1	2	**47.454**	0.9756	14.956
2	3	57.271	0.9763	16.825
3	4	48.471	**0.9766**	**14.390**

To evaluate the prediction performance of the SAE model in river runoff prediction, the Deep Belief Network model as in [7] is built as a comparative model. For SAE model, the number of input nodes is also determined by the experiment results given in Table 1 which shows that the best value for this

Table 5. Comparison of the performance results for two different models on the three datasets with sliding window size 2 (for DBN), 7 (for SAE)

Input	Model	RMSE	R	MAPE
Cau14	DBN	52.783	0.9711	14.236
	SAE	**45.401**	**0.9770**	**14.040**
BuonHo	DBN	**94.158**	**0.9266**	19.861
	SAE	99.536	0.9241	**19.835**
DucXuyen	DBN	**54.303**	0.8495	86.138
	SAE	55.798	**0.8695**	**83.483**

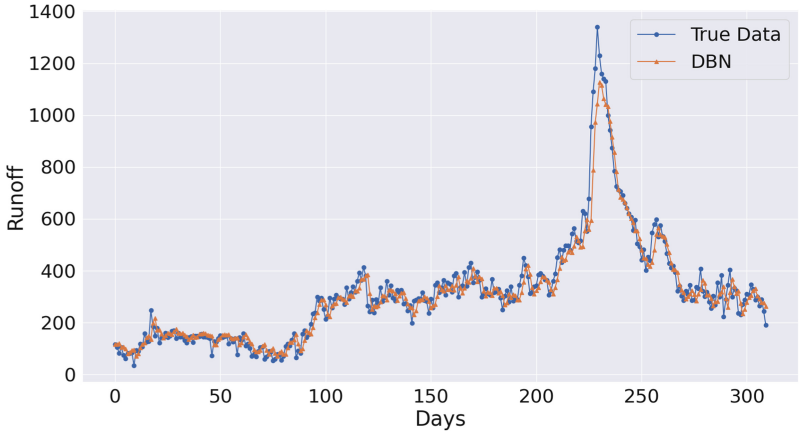
parameter is 7. From the experimental results in Tables 3 and 4, the number of AE layers is set to three, and the number of hidden nodes per layer used in SAE model is (200, 400, and 200). Besides, we reuse a selected model for SAE which is based on Cau14 dataset to experiment the runoff prediction on the two datasets at DucXuyen and BuonHo measurement station. To train the SAE model, we applied back-propagation algorithm. To train the DBN model, we apply Contrastive Divergence (CD-k) algorithm in pre-training stage and back-propagation algorithm in fine-tuning stage.

As shown in Table 5, on the Cau14 dataset, the SAE model outperforms DBN model in all three RMSE, R, MAPE evaluation measures. Furthermore, on the DucXuyen and BuonHo datasets, SAE model is also slightly better. These results prove that the ability of feature extraction in predicting the future runoff value of the SAE model is more powerful than DBN.

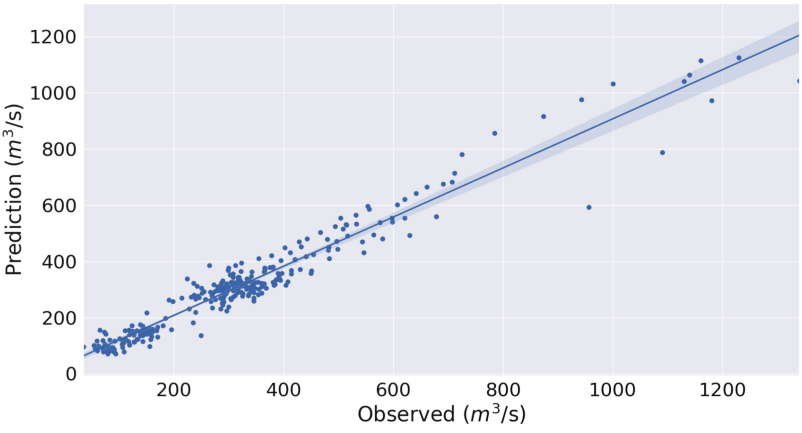
To study the relation between the observed and predicted data, the scatter plots are generated with line plot in Fig. 7 and Fig. 8. From the visualization results in Fig. 7a, DBN captures all the change over time of the runoff river, but there are also some small amounts of predicted values to move down, it means that the predicted value is smaller than the observed value. Looking at the Fig. 7a, we see that the points with predictive value move down, they are far from the ideal fit (Fig. 7b). Comparing the Fig. 7a with Fig. 8a, we see that the predicted results of the SAE model are closer to the observed data and compared to the forecast points that have the downward movement of the DBN model, it improves better. So they are closer to the ideal fit (Fig. 8b).

Table 6. Comparison of the training time and testing time (in seconds) of the two models on Cau14 dataset

Times	SAE	DBN
Training	**520.499**	585.914
Testing	**0.0589**	0.134



(a)



(b)

Fig. 7. (a) Observed (blue) and predicted (red) runoff by DBN in the testing phase. (b) The degree correlation between observed and predicted runoff by DBN in the testing phase. (Color figure online)

We also measured the training times and testing times of the two deep learning forecasting models on the runoff dataset at the Cau14 station and report the results (in seconds) in Table 6. The results in Table 6 indicate that the training/testing time of SAE is slightly shorter than that of DBN.

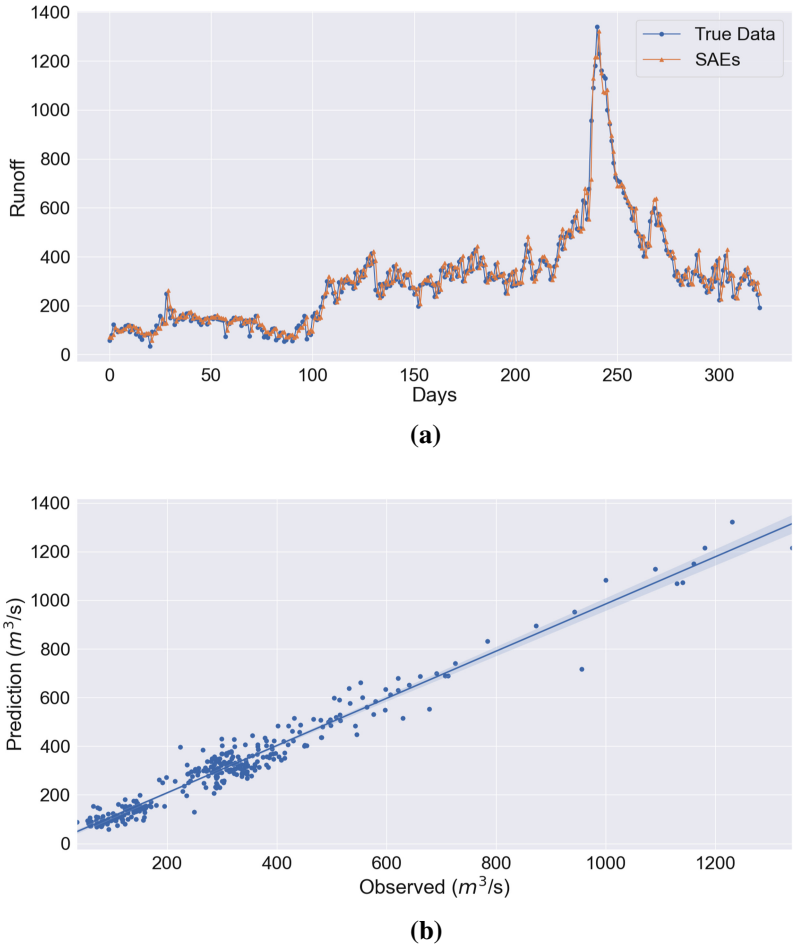


Fig. 8. (a) Observed (blue) and predicted (red) runoff by SAE in the testing phase. (b) The degree correlation between observed and predicted runoff by SAE in the testing phase. (Color figure online)

5 Conclusion and Future Work

River runoff prediction is a challenging forecasting problem. In this paper, we compare the two deep neural network models (SAE and DBN) in river runoff prediction in terms of prediction accuracy. The experimental results show that, SAE brings out better prediction accuracy than DBN on three runoff datasets from Srepok river in Highlands of Vietnam. The results of this work confirm that stacked autoencoder is a promising deep learning approach for runoff time series forecasting. Stacked autoencoder shows the power of feature extraction and gets high accuracy when predicting. Besides, the impact of size of window slides and the architecture of SAE model were also taken into account.

In future research, we intend to combine SAE with Long Short Term Memory (LSTM) network in order to improve further river runoff prediction as suggested in ([18,19]).

Acknowledgement. This research is funded by Vietnam National University Ho Chi Minh City (VNU-HCM) under grant number C2019-20-17.

References

1. Thi, T.T.T., Ngo, N.H.G., Hieu, D.N., Hien, N.T., Hoai, T.V., Nghi, V.V.: A comprehensive study on predicting river runoff. In: The 9th International Conference on Knowledge and Systems Engineering (KSE), Hue, pp. 251–256 (2017)
2. Ngoc Duong, H., Nguyen, Q.N.T., Ta Bui, L., Nguyen, H.T., Snášel, V.: Applying recurrent fuzzy neural network to predict the runoff of Srepok river. In: Saeed, K., Snášel, V. (eds.) CISIM 2014. LNCS, vol. 8838, pp. 55–66. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45237-0_7
3. Guo, W., Wang, H., Xu, J., Zhang, Y.: RBF neural net-work model based on improved PSO for predicting river runoff. In: International Conference on Intelligent Computation Technology and Automation, pp. 968–971 (2010)
4. Piotrowski, A.P., Napiorkowski, J.J.: Optimizing neural networks for river flow forecasting - evolutionary computation methods versus the Levenberg-Marquardt approach. J. Hydrol. **407**, 12–27 (2011)
5. Demirela, M.C., Venancio, A., Kahya, E.: Flow forecast by SWAT model and ANN in Pracana basin Portugal. J. Adv. Eng. Softw. **40**, 467–473 (2009)
6. Gope, S., Sarkar, S., Mitra, P., Ghosh, S.: Early prediction of extreme rainfall events: a deep learning approach. In: Perner, P. (ed.) ICDM 2016. LNCS (LNAI), vol. 9728, pp. 154–167. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-41561-1_12
7. Tri N. C., Hieu D.N., Hoai, T. V., and Snasel, V.: Predicting daily river runoff by deep belief networks. In: International Conference on Information and Convergence Technology for Smart Society, 19–21 January 2016 in Ho Chi Minh, Vietnam (2016)
8. Hernández, E., Sanchez-Anguix, V., Julian, V., Palanca, J., Duque, N.: Rainfall prediction: a deep learning approach. In: Martínez-Álvarez, F., Troncoso, A., Quintián, H., Corchado, E. (eds.) HAIS 2016. LNCS (LNAI), vol. 9648, pp. 151–162. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-32034-2_13
9. Li, C., Bai, Y., Zeng, B.: Deep learning architecture for daily reservoir inflow forecasting. Water Resour. Manage. **30**, 5145–5161 (2016)
10. Kratzert, F., Klotz, D., Brenner, C., Schulz, K., Herrnegger, M.: Rainfall-runoff modeling using long short-term memory (LSTM) networks. Hydrol. Earth Syst. Sci. **22**, 6005–6022 (2018)
11. Lee, D.E., Lee, G., Kim, S., Jung, S.: Future runoff analysis in the Mekong river basin under a climate change scenario using deep learning. Water **12**, 1556 (2020)
12. Hinton, G.E., Osindero, S., The, Y.W.: A fast learning algorithm for deep belief nets. Neural Comput. **18**, 1527–1554 (2006)
13. Bengio, Y.: Learning deep architectures for AI". Found. Trends Mach. Learn. **2**(1), 1–127 (2009)
14. Katuwal, R., Suganthan, P.N.: Stacked autoencoder based deep random vector functional link neural network for classification. Appl. Soft Comput. **85**, 105854 (2019)

15. Wikipedia. https://vi.wikipedia.org/wiki/SôngSêrêpok. Accessed 10 July 2020
16. Cholett, F., Keras. http://keras.io. Accessed June 2020
17. Azlan, A., Yusof, Y., Mohamad, M., Mohamad, F.: Determining the impact of window length on time series forecasting using deep learning. Int. J. Adv. Comput. Res. **9**, 260–267 (2019)
18. Li, Z., Peng, F., Niu, B., Li, G., Wu, J., Miao, Z.: Water quality prediction model combining sparce auto-encoder and LSTM network. IFAC PapersOnLine **51–17**, 831–836 (2018)
19. Jaseena, K.U., Kovoor, B.C.: A hybrid wind speed forecasting model using stacked autoencoder and LSTM. J. Renew. Substain. Energy **12**, 023302 (2020)
20. Sam, T.T., et al.: Impact of climate change on meteorological, hydrological, and agricultural droughts in the Lower Mekong River Basin: a case study of the Srepok Basin Vietnam. Water Environ. J. (2018). https://doi.org/10.1111/wej.12424