Pol. J. Environ. Stud. Vol. 28, No. 2 (2019), 795-802
DOI: *10.15244/pjoes/81557* ONLINE PUBLICATION DATE: 2018-09-19

Original Research

Simulating and Predicting of Hydrological Time Series Based on TensorFlow Deep Learning

Jinbo Qin[1], Ji Liang[1]*, Tao Chen[2], Xiaohui Lei[3], Aiqing Kang[3]**

[1]Huazhong University of Science and Technology,Wuhan, China
[2]Electric Power Research Institute, Jilin Electric Power Co., Changcun, China
[3]State Key Laboratory of Simulation and Regulation of Water Cycle in River Basin and China Institute of Water Resources and Hydropower Research, Beijing, China

Received: 15 November 2017
Accepted: 23 December 2017

Abstract

Hydrological time series refers to the observation time point and the observed time value. The simulation and prediction of hydrological time series will greatly improve the predictability of hydrological time series, which is of great significance for hydrological forecasting. TensorFlow, the second generation of artificial intelligence learning system in Google, has been favored by a large number of researchers by virtue of its high flexibility, portability, multi-language support, and performance optimization. However, the application of deep learning in hydrology is less. Based on the TensorFlow framework, the AR model and the LSTM model are constructed in Python language. The hydrological time series is used as the input object, and the model is deeply studied and trained to simulate and predict the hydrological time series. The effect of the model was tested by fitting degrees and other indexes. The fitting degree of the AR model is 0.9551, and the fitting degree of the LSTM model is 0.8012, which shows the feasibility of the model for predicting the hydrological time series, and puts forward the solution for the limitation of the existing analysis results.

Keywords: TensorFlow, deep learning, AR model, LSTM model, hydrological time series

Introduction

Deep learning as an important part of machine learning, and there is a very wide range of application space. TensorFlow as the representative of deep learning the mainstream framework is also increasingly favored by the majority of research workers. The autoregressive model mainly uses the linear combination of stochastic variables at a specific time to describe the random variable at a certain time later [1]. With the development of time, many researchers have expanded the autoregressive model and applied it to many fields such as Bayesian, structural vector, spatial heterogeneous [2-4], and so on. LSTM is a long- and short-memory artificial neural network proposed by Hochreiter et al. and modified and promoted by Alex Graves [5]. LSTM can learn the short- and long-term dependent information of time series. Since the neural network contains a time memory unit, it is suitable for processing and predicting the interval and delay events in the time series [6-9]. In the prediction of hydrological time series, SENF C. and others [10]

*e-mail: larkwater1@163.com
**e-mail: lxh@iwhr.com

adopt the Bayesian model; while XU M. and others use the Echo state network [11]. In addition, several modern analysis techniques, such as neural network, wavelet analysis, support vector machine, self memory model, and chaos theory are also used in hydrological time series prediction [12-15]. However, the traditional hydrological model or statistical model has the problems of local optimization, over fitting, or low operational efficiency. It is difficult to predict and simulate the current large amount of data, large number of iterations, and harsh environment of parameters.

The AR model has a good autoregressive property. The model can push out the data at the front or rear of point N by the model, so its essence is similar to interpolation. Although its purpose is to increase the valid data, the AR model is a recursion from point N, and the interpolation points are derived from two points (or a few points). Therefore, the AR model is better than the traditional interpolation method. The LSTM model is different from the traditional neural network model in that it adds a practical processor that is set with three doors. The door can be judged according to the forgetting factor related to the model, to decide whether to leave the information, which will effectively avoid the cumulative error transmission and greatly improve the accuracy of forecasting and simulation. In view of the reasonable structure of the AR and LSTM models, it is suitable for dealing with the characteristics of time series. This paper attempts to apply deep learning technology to the LSTM and AR models, takes hydrological time series as the research object, and explores the application of deep learning in hydrology. And the ideal fitting effect is obtained.

Model Method and Operating Environment

TensorFlow Frame and its Characteristics

TensorFlow was originally developed by the Google Brain Group, part of the Google Machine Intelligence Research Institute, and is used primarily for deep neural networks and machine learning [16]. Google believes that machine learning is a key part of the future of new products and new technologies. The research in this area is also global and evolving quickly, but lacks a standardized tool, for which Google will open TensorFlow to create an open standard that allows all users to brainstorm and promote machine learning[17].

TensorFlow has the following features[18-20]:

1) High degree of flexibility: TensorFlow supports user-defined data flow diagrams, the user can build a map, and describe the drive to calculate the internal period. By using the rich and practical toolkit provided by TensorFlow, users can assemble their own "subgraph" to achieve a special neural network function.
2) True portability: TensorFlow supports running on the CPU and GPU, which means that TensorFlow can not only run on a server with a special configuration, but for personal computer users can run unimpeded. TensorFlow can now scale on a number of CPU operations, and can be used in mobile apps.
3) Automatic differential design: a machine learning algorithm is based on the gradient of the calculation, and Tensorflow has the ability to automatically seek the differential for such operations. In use, it only needs to define the structure of the prediction model, this structure and the objective function together, and add data; Tensorflow will automatically calculate the relevant derivative for you.
4) Performance optimization: Tensorflow's full support for threads, queues, asynchronous, and other operations can all play the computing power of personal computing equipment potential. The user can freely assign the calculated elements in the Tensorflow diagram to different devices, and Tensorflow can help you manage these different copies.

TensorFlow in development did not consider the backward compatibility issues, so the compatibility between the various versions of the poor, the AR model, and LSTM model were implementated using TensorFlow 3.0 (released in July 2017). In this version of the introduction a TensorFlow Time Series module is used to simulate and predict time series.

Hydrological Time Series

In general, the time series consists of two parts [21-25]: the observed point in time and the observed actual value. In the field of hydrology, the observed time point is defined as the time point of the hydrological series. The specific time point is the daily date (d), and the observed actual value is defined as the daily flow (m^3/s).

Operating Environment

Currently TensorFlow has been running in the system environment for Windows 10 Professional (64bit). Python is an object-oriented interpreted computer programming language, with the advantages of simple and concise syntax. Both the AR model and the LSTM model use Python as the programming language, with version 3.5.4 (64-bit). PyCharm is a Python IDE with a set of tools that can help users improve their efficiency when using Python. Use PyCharm as the Python IDE compilation platform, version 2017.2.2 for 64-bit.

Establishing the Model

Establishing the AR Model

The autoregressive model (which can be referred to as AR model) is one of the basic methods for statistically processing time series models [26-28]. In the TensorFlow time series module, an autoregressive model has been implemented.

(1) Read hydrological time series data

Save the hydrological time series data in the CSV file by reading the tf.contrib package in the tensorflow framework and further calling the CSVReader command in the time series module to read the hydrological series data into the model. The code is:

```
csv_file_name = './data/period_trend6.csv'
reader = tf.contrib.timeseries.CSVReader(csv_file_name)
```

(2) Define the input of the model

Create a train_input_fn as the input variable for the model. Call the RandomWindowInputFn command in the time series module, and you need to set the parameters of batch_size and window_size. Where batch_size represents the number of batches in the model (which means the smallest batch of data within the model), window_size represents the length of each series within a batch whose value is equal to the sum of the input and output values. The code is:

```
train_input_fn=tf.contrib.timeseries.
RandomWindowInputFn
(reader, batch_size=9, window_size=30)
```

(3) Define the AR model

The AR model is defined. Call the ARRegressor command in the time series module, where you need to set periodicities, input_window_size, output_window_size, num_features, and loss five parameters. Where the periodicities represent the regularity of the series, this is 365 (yearly scale) or 30 (monthly scale). Output_window_size represents the value of each output of the model, and the sum of input_window_size and output_window_size is equal to the window_size value in the model input train_input_fn. Where window_size is 30, input_window_size is 20, and output_window_size is 10, which means that the length of each series within a batch is 30, where the first 20 numbers are treated as input values for the model and the last 10 numbers as input values the target output value. The last parameter loss specifies which loss to take. There are two kinds of loss to choose: NORMAL_LIKELIHOOD_LOSS and SQUARED_LOSS, for the hydrological time series, with the choice of NORMAL_LIKELIHOOD_LOSS making for a better calculation. The num_features parameter represents the dimension of the number observed at a point in time. Each step here is a separate value, so num_features is 1. The code is:

```
ar=tf.contrib.timeseries.ARRegressor
(periodicities=30, input_window_
size=20, output_window_size=10,
num_features=1,
loss=tf.contrib.timeseries.ARModel.
NORMAL_LIKELIHOOD_LOSS)
```

(4) Model training

For the defined AR model, call the train method for training, where steps for the training times can be set and where the training times is set to 2000 times. The code is:

```
ar.train(input_fn=train_input_fn, steps=2000)
```

(5) Model evaluation

The model evaluation indicates that the model is used to validate the fitting effect of the model by using the trained model to calculate it on the measured training set. The meaning of the AR model is to receive an input observation series of length 20 each time and output a prediction series of length 10. The entire training set (measured hydrological series) is a series of length 1000 (based on the length of the hydrological series, where the length is assumed to be 1000). The first 20 numbers are entered as an "initial observation series" into the model, and the predicted values of the last 10 steps can be calculated. The next step, continue to take 20 numbers to predict, then 10 of these 20 numbers are the predicted values for the previous step, so the number of 20 new results as the next input value, and so on. Since the first 20 numbers are unpredictable, we will eventually get 980 predictions (simulate values) and record 980 predictions (simulate values) in evaluation ['mean']. There are several other key values, such as evaluation ['loss'] for total loss, evaluation ['times'] for evaluation time ['mean']. The code is:

```
evaluation_input_fn = tf.contrib.timeseries.
WholeDatasetInputFn(reader)
evaluation = ar.evaluate(input_
fn=evaluation_input_fn, steps=1)
```

(6) Model prediction

For the established model, the value of the measured hydrological time series is predicted. The number of the predicted value is the prediction value of the model. When the step value is 31, it is the monthly prediction, and when the time value is 366, it is the annual prediction. The code here predicts 31 time points after 1000 steps (assuming an input series length of 1000). The corresponding value is stored in predictions ['mean']. The code is:

```
(predictions,) = tuple(ar.predict
(input_fn=tf.contrib.timeseries.
predict_continuation_input_fn
(evaluation, steps=31)))
```

(7) Output graphical simulation, prediction results

Data represents the measured value of the hydrological series, evaluates the model's simulation value, predictions represent the predicted value of the model and call the matplotlib toolkit to plot the measured hydrological series value, the simulated value of the model, and the predicted value of the model on a graph and save it as a "predict_result.pdf" pdf file.

The code for importing the matplotlib toolkit is:

```
import matplotlib
matplotlib.use('agg')
import matplotlib.pyplot as plt
```

Use the matplotlib toolkit in the drawing function and graphics file save function, the corresponding code:

```
plt.figure(figsize=(15, 5))
plt.plot(data['times'].reshape(-1), data['values'].
reshape(-1), label='origin')
plt.plot(evaluation['times'].reshape
(-1),evaluation['mean'].reshape(-1), label='evaluation')
plt.plot(predictions['times'].reshape
(-1),predictions['mean'].reshape(-1), label='prediction')
plt.xlabel('time_step')
plt.ylabel('values')
```

```
plt.legend(loc=4)
plt.savefig('predict_result.pdf')
```

Establishing the LSTM Model

Long short term term (LSTM) is a time-recurrent neural network suitable for dealing with and predicting the relatively long events in the time series [29-31]. LSTM is different from recurrent neural network (RNN), mainly because it adds a "processor" to the algorithm that is useful or not. The structure of this processor is called cell. A cell that was placed three doors, respectively, called the input door, forget the door, and the output door. A message into the LSTM network, according to the rules to determine whether it is useful. Only to meet the algorithm authentication information will be left, do not match the information through the forgotten door was forgotten. That is, into the work of two out of the principle, you can solve the neural network in the repeated operation of long-term problems. It has been proven that LSTM is an effective technique for solving long-term dependency problems, and that the universality of this technology is very high, resulting in a great deal of change. Researchers have made their own variable versions based on LSTM, which allows LSTM to handle the ever-changing problems.

(1) Read hydrological time series data:

Save the hydrological time series data in the CSV file by calling the tf.contrib package in the tensorflow framework and further calling the CSVReader command in the timeseries module to read the hydrological series data into the model. The code is:

```
csv_file_name = path.join("./data/eriod_trend_lstm.csv")
reader = tf.contrib.timeseries.CSVReader(csv_file_name,
        column_names=((tf.contrib.timeseries.
        TrainEvalFeatures.TIMES,)
  + (tf.contrib.timeseries.TrainEvalFeatures.VALUES,)))
```

(2) Define the input of the model

Create a train_input_fn as the input variable for the model. Call the RandomWindowInputFn command in the timeseries module, and you need to set the parameters of batch_size and window_size. Where batch_size represents the number of hydrological seriess randomly selected in a batch, which means the smallest batch of data within the model, and window_size represents the length of each series within a batch. The code is:

```
train_input_fn = tf.contrib.timeseries.
RandomWindowInputFn(reader, batch_
        size=17, window_size=30)
```

(3) Define the LSTM model

The LSTM model is defined. Call the timeseries module _TimeSeriesRegressor command, here need to set num_features, num_units, learning rate three parameters. Where num_features is 1, and its meaning is a univariate time series, that is, the amount observed on each point in time is only a single value; num_units is 128, which means that the LSTM model with a hidden layer size of 128 is used. Using the AdamOptimizer optimization algorithm provided by the train module in the TensorFlow framework to control the learning speed, AdamOptimizer achieves a learning rate of 0.001 by using momentum, that is, the moving average of the parameters to improve the traditional gradient drop and to promote hyperparameter dynamic adjustment. The code is:

```
estimator = ts_estimators._TimeSeriesRegressor(
model=_LSTMModel(num_features=1, num_units=128),
        optimizer=tf.train.AdamOptimizer(0.001))
```

(4) Model training

On the definition of the LSTM model variable estimators, call train method for training, where steps for the training times can be set. The training times here is set to 2000 times. The code is:

```
estimator.train(input_fn=train_input_fn, steps=2000)
```

(5) Model evaluation

Through the use of trained models in the measured training set on the calculation, the model of the fitting effect. The simulate values are recorded in the "evaluated" of the variables. Evaluated_times represents the corresponding point in time. Observed indicates the measured value, observed_times represents the corresponding point in time. The code is:

```
evaluation_input_fn = tf.contrib.timeseries.
        WholeDatasetInputFn(reader)
evaluation = estimator.evaluate(input_
        fn=evaluation_input_fn, steps=1)
```

(6) Model prediction

For the established model, the value of the measured hydrological series is predicted and the number of the predicted value is the prediction value of the model. When the step value is 31, it is the monthly prediction, and when the time value is 366, it is the annual prediction. The "predicted" value is recorded in the variable. predicted_times represents the corresponding point in time. The code is:

```
(predictions,) = tuple(estimator.predict(
        input_fn=tf.contrib.timeseries.predict_
    continuation_input_fn(evaluation, steps=31)))
```

(7) Output graphical simulation, prediction results

"Observed" represents the measured hydrological sequence values, "evaluated" represents the simulation values of the model, and "predicted" represents the prediction values of the model. By calling the matplotlib toolkit, the measured hydrological series values, the model's simulation values and predictions are plotted on a graph. And save it as a "predict_result.pdf" pdf file. Import the matplotlib toolkit corresponding code:

```
import matplotlib
matplotlib.use("agg")
import matplotlib.pyplot as plt
```

Use the matplotlib toolkit in the drawing function and graphics save function, the corresponding code:

```
plt.figure(figsize=(15, 5))
plt.axvline(181, linestyle="dotted",
        linewidth=4, color='r')
observed_lines = plt.plot(observed_times,
observed, label="observation", color="k")
```

```
evaluation_lines = plt.plot (evaluated_times,
evaluation, label = “evaluation”, color = “g”)
predicted_lines = plt.plot(predicted_times,
predicted, label=”prediction”, color=”r”)
plt.legend(handles=[observed_lines[0],
evaluated_lines[0], predicted_lines[0]],
loc=”upper left”)
plt.savefig(‘predict_result.pdf’)
```

Results and Discussion

Model Effect Indicators

There are three statistical indicators used to evaluate the effect of the model: root mean square error (RMSE), the coefficient of determination (also known as coefficient of determination (R^2)), and mean absolute error (MAE). RMSE is used to evaluate the residuals between the measured and predicted values and can be expressed as:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(y_0 - y_f)^2}{N}} \quad (1)$$

…where N is the total number of all values, y_0 is the observed value, and y_f is a prediction value or an simulate value.

R^2 is used to illustrate the degree to which the model's predictive series interprets the measured hydrological series, which can be expressed as:

$$R^2 = \left| \frac{\sum_{i=1}^{N}\left(y_0 - \overline{y_0}\right)\left(y_f - \overline{y_f}\right)}{\sqrt{\sum_{i=1}^{N}\left(y_0 - \overline{y_0}\right)^2 \sum_{i=1}^{N}\left(y_f - \overline{y_f}\right)^2}} \right|^2 \quad (2)$$

… where N is the total number of all values, y_0 is the observed value, y_f is a prediction value or an simulate value, $\overline{y_0}$ is the average of the observed values, and $\overline{y_f}$ is the average of the predicted or simulated values.

MAE represents the absolute mean error between the measured value and the predicted value, which can be expressed as:

$$MAE = \frac{\sum_{i=1}^{N}\left|y_0 - y_f\right|}{N} \quad (3)$$

…where N is the total number of all values, y_0 is the observed value, and y_f is a prediction value or an simulate value.

For the three effect indicators, the smaller the MAE and RMSE, the higher the accuracy of the model; the closer the R^2 is to 1, the better the model works.

Analysis of Model Operation Results

In this paper, the model data of the Hanjiang River Basin are used to calculate the model. The measured data of the AR model are from July 2015 to June 2016, and test data for prediction of measured data in July 2016. The simulation uses data from January 2016 to June 2016, using the July 2016 flow data as simulation data. For the LSTM model, the prediction is based on the flow data from January 2016 to June 2016, which is used to evaluate the measured data in July 2016. The flow data from January 2006 to December 2015 are used as measured data. In January 2016 to December 2016 flow data as simulation evaluation data.

The operating parameters of the model are determined by the empirical method. Due to the large number of parameters, the parameters or variables of AR model parameter calibration are: input and output value, period, training times, and time scale; LSTM model parameter calibration of the parameters or variables: number of batch (batch_size), hidden layer size (num_units), period, training times, and time scales.

Table 1. AR model prediction/simulation parameters.

	Input and output values	Period (days)	Training times	Time Scale
Prediction	Input 20, output 10	30	2000	12 months-1 month
	RMSE= 110.79	R^2= 0.5080	MAE= 104.47	
Simulation	Input 40, output 10	30	2000	6 months-1 month
	RMSE= 29.05	R^2= 0.9551	MAE= 13.30	

Table 2. LSTM model prediction/simulation parameters and results.

	Batch_size	Num_units	Period (days)	Training times	Time Scale
Prediction	20	200	30	3000	6 months-1 month
	RMSE= 122.01	R^2= 0.4543	MAE= 115.82		
Simulation	20	200	30	2000	10 years-1 year
	RMSE= 84.89	R^2= 0.8012	MAE= 52.1062		

RMSE, goodness of fit (R^2) and MAE were used to evaluate the effect of the two models. In the range of partial parameters, the corresponding parameters and the evaluation indexes are shown in Tables 1 and 2 when the optimal effect is reached.

From the simulation results we can see that the two models are better. Goodness of fit (R^2) of the AR model is 0.9551, and the RMSE and MAE are small. In contrast, the LSTM model has a goodness of fit (R^2) of 0.8012, and its simulation effect is within the qualified range, and its RMSE and MAE model error is greater. On the whole, the simulation results of the two models are above the qualified level, which shows that the two models are more feasible for the prediction of hydrological time series and have further research value.

Conclusion

Figs 1 and 2 show the comparison between the measured values and the simulated values. In terms of the prediction results, there are some limitations in the predicting abilities of the two models, which may be highly nonlinear and uncertain to the hydrological series [32] and so on. Compared with the LSTM model, the goodness of the AR model (R^2) is better, and the RMSE and MAE are smaller, indicating that the residual between the measured value and the predicted value poor and absolute mean errors are smaller. On the whole, the prediction results of the two models under the existing parameters are not good, but the simulation results are better, indicating that the two models are available and there is still much room for improvement.

It is worth noting that the LSTM model has many parameters, and the learning rate and the optimization algorithm to improve the traditional gradient descent are also adjustable. If more comprehensive examination and comprehensive determination of each variable parameter or variable, its prediction effect will have more room for improvement. And there is a great difference between the maximum and the minimum values of the hydrological time series in the Hanjiang River Basin. If the non-stationary nature of the hydrological time series is reasonably analyzed and its stabilization is carried out, its simulation and prediction results will be greatly improved.

Deep learning is currently a popular and widely used technology, its application in the hydrological area also needs further exploration, if the model of learning, training, prediction, and parameter automatic rate is linked to the use of optimization algorithm for automatic parameter calibration. And even that can be combined with TensorFlow's distributed features, in the GPU to

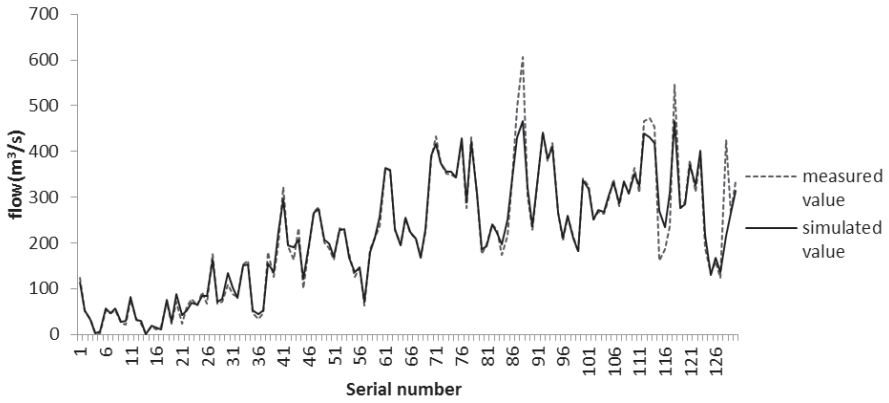

Fig. 1. AR model measured values and simulated values.

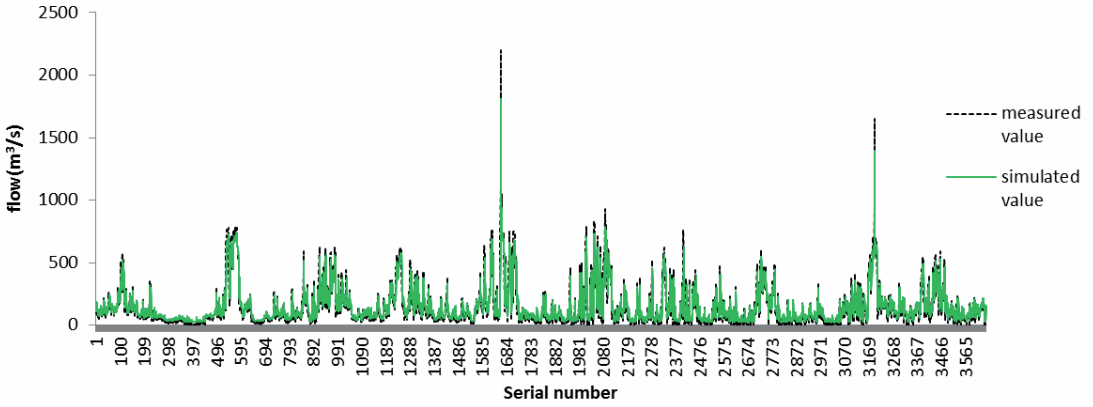

Fig. 2. LSTM model measured values and simulated values.

achieve distributed and efficient operation, which is the future depth of the field of learning a major trend, but also to achieve hydrological information, efficient, and correct choices.

Acknowledgements

This research is supported by the 2017 Independent Innovation Fund-General Program (2017KFYXJJ202) and National Key R&D Program of China (2017YFC0405900).

Conflict of Interest

The authors declare no conflict of interest.

References

1. TERASVIRTA T. Estimation and evaluation of smooth transition autoregressive models. Journal of the American Statistical Association, **89** (425), 208, **1994**.
2. HAN X., HSIEH C.S., LEE L.F. Estimation and model selection of higher-order spatial autoregressive model: an efficient bayesian approach. Regional Science & Urban Economics, **63**, 97, **2017**.
3. TSIONAS M.G., CHEN Z., WANKE P.A. Structural vector autoregressive model of technical efficiency and delays with an application to chinese airlines. Transportation Research Part A Policy & Practice, **101**, 1, **2017**.
4. HWANG E., DONG W.S. Stationary bootstrapping for structural break tests for a heterogeneous autoregressive model. Communications for Statistical Applications and Methods, **24** (4), 367, **2017**.
5. RIVEST F., KALASKA J.F., BENGIO Y. Conditioning and time representation in long short-term memory networks. Biological Cybernetics, **108** (1), 23, **2014**.
6. CAI M., LIU J., CAI M., LIU J., CAI M., LIU J. Maxout neurons for deep convolutional and lstm neural networks in speech recognition. Speech Communication, **77**, 53, **2016**.
7. HANSSON M. On stock return prediction with lstm networks. Asaio Journal, **40** (4), 928, **2017**.
8. HANSON J., YANG Y., PALIWAL K., ZHOU Y. Improving protein disorder prediction by deep bidirectional long short-term memory recurrent neural networks. Bioinformatics, **33** (5), 685, **2017**.
9. JIA G., LU Y., LU W., SHI Y., YANG J. Verification method for chinese aviation radiotelephony readbacks based on lstm-rnn. Electronics Letters, **53** (6), 401, **2017**.
10. SENF C., PFLUGMACHER D., HEURICH M., KRUEGER T. A bayesian hierarchical model for estimating spatial and temporal variation in vegetation phenology from landsat time series. Remote Sensing of Environment, **194**, 155, **2017**.
11. XU M., HAN M. Adaptive elastic echo state network for multivariate time series prediction. IEEE Transactions on Cybernetics, **46** (10), 2173, **2017**.
12. CHAYAMA M., HIRATA Y. When univariate model-free time series prediction is better than multivariate. Physics Letters A, **380** (31-32), 2359, **2016**.
13. REICH N.G., LESSLER J., SAKREJDA K., LAUER S.A., IAMSIRITHAWORN S., CUMMINGS D.A.T. Case study in evaluating time series prediction models using the relative mean absolute error. American Statistician, **70** (3), 285, **2016**.
14. WU X., ZHAN F.B., ZHANG K., DENG Q. Application of a two-step cluster analysis and the apriori algorithm to classify the deformation states of two typical colluvial landslides in the three gorges, china. Environmental Earth Sciences, **75** (2), 146, **2016**.
15. GHORBANI M.A., ZADEH H.A., ISAZADEH M., TERZI O.A. Comparative study of artificial neural network (mlp, rbf) and support vector machine models for river flow prediction. Environmental Earth Sciences, **75** (6), 1, **2016**.
16. ZHANG M., XU H., WANG X., ZHOU M., HONG S. Application of Google TensorFlow machine learning framework . Microcomputer and Application, **36** (10), 58, **2017** [In Chinese].
17. ABADI M. Tensorflow: learning functions at scale. Acm Sigplan Notices, **1**, 1, **2016**.
18. BAEK Y.T., LEE S.H., KIM J.S. Intelligent missing persons index system implementation based on the opencv image processing and tensorflow deep-running image processing. Journal of the Korea Society of Computer and Information, **22** (1), 15, **2017**.
19. QUAN D., SON T.C., CHAUDRI J. Classification of asthma severity and medication using tensorflow and multilevel databases. Procedia Computer Science, **113**, 344, **2017**.
20. XIA X.L., XU C., NAN B. Facial expression recognition based on tensorflow platform. ITM Web of Conferences, **12**, 01005, **2017**.
21. BAGNALL A., LINES J., BOSTROM A., LARGE J., KEOGH E. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. Data Mining & Knowledge Discovery, **31** (3), 606, **2017**.
22. XIAO W., HU Y., ZHAO X., CHEN Y., LI R. Time series based urban air quality predication. Big Data & Information Analytics, **1** (2/3), 171, **2017**.
23. LINDEN A., YARNOLD P.R. Using machine learning to identify structural breaks in single-group interrupted time series designs. Journal of Evaluation in Clinical Practice, **22** (6), 851, **2016**.
24. HAN H., LINTON O., OKA T., WHANG Y.J. The cross-quantilogram: measuring quantile dependence and testing directional predictability between time series. Journal of Econometrics, **193** (1), 251, **2016**.
25. GAO Z.K., SMALL M., KURTHS J. Complex network analysis of time series. Epl, **116** (5), 50001, **2016**.
26. TSAI L.C., MACALALAD E.P., LIU C.H. Taiwan ionospheric model (twim) prediction based on time series autoregressive analysis. Radio Science, **49** (10), 977, **2016**.
27. LIU X., XIAO H., CHEN R. Convolutional autoregressive models for functional time series. Journal of Econometrics, **194** (2), 263, **2016**.
28. CHEN C.W.S., LEE S. Generalized poisson autoregressive models for time series of counts. Computational Statistics & Data Analysis, **99**, 51, **2016**.
29. ZHAO Z., CHEN W., WU X., CHEN P.C.Y., LIU J. Lstm network: a deep learning approach for short-term traffic forecast. Iet Intelligent Transport Systems, **11** (2), 68, **2017**.
30. GAO L., GUO Z., ZHANG H., XU X., SHEN H.T. Video captioning with attention-based lstm and semantic

consistency. IEEE Transactions on Multimedia, **19** (9), 2045, **2017**.
31. CHEN H., CHEN J., HU R., CHEN C., WANG Z. Action recognition with temporal scale-invariant deep learning framework. China communications (English Edition), **14** (2), 163, **2017**.
32. CAO D., XU S.A Fast&Accurate Image Classifier Based on the Pre-training Models of TensorFlow. Journal of Hanjiang Normal University, **37** (03), 27, **2017** [In Chinese].