Knowledge-Based Systems 254 (2022) 109608

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

A PLS-based pruning algorithm for simplified long–short term memory neural network in time series prediction

Wenjing Li *, Xiaoxiao Wang, Honggui Han, Junfei Qiao

Faculty of Information Technology, Beijing University of Technology, Beijing, 100124, China
Beijing Artificial Intelligence Institute, Beijing, 100124, China
Engineering Research Center of Intelligence Perception and Autonomous Control, Ministry of Education, Beijing, 100124, China
Beijing Key Laboratory of Computational Intelligence and Intelligent System, Beijing, 100124, China
Beijing Laboratory for Intelligent Environmental Protection, Beijing, 100124, China

ARTICLE INFO

Article history:
Received 17 December 2021
Received in revised form 3 August 2022
Accepted 3 August 2022
Available online 8 August 2022

Keywords:
Long–short term memory (LSTM)
Time series prediction
Internal structure simplification
Partial least squares (PLS) regression
Pruning algorithm
Hidden layer size

ABSTRACT

As an extensively used model for time series prediction, the Long–Short Term Memory (LSTM) neural network suffers from shortcomings such as high computational cost and large memory requirement, due to its complex structure. To address these problems, a PLS-based pruning algorithm is hereby proposed for a simplified LSTM (PSLSTM). First, a hybrid strategy is designed to simplify the internal structure of LSTM, which combines the structure simplification and parameter reduction for gates. Second, partial least squares (PLS) regression coefficients are used as the metric to evaluate the importance of the memory blocks, and the redundant hidden layer size is pruned by merging unimportant blocks with their most correlated ones. The Backpropagation Through Time (BPTT) algorithm is utilized as the learning algorithm to update the network parameters. Finally, several benchmark and practical datasets for time series prediction are used to evaluate the performance of the proposed PSLSTM. The experimental results demonstrate that the PLS-based pruning algorithm can achieve the trade-off between a good generalization ability and a compact network structure. The computational complexity is improved by the simple internal structure as well as the compact hidden layer size, without sacrificing prediction accuracy.

1. Introduction

A time series is a sequence of observed data that can be used to discover the behavior of a system [1]. An effective time series prediction method is critical in system modeling and behavior forecasting, and has been widely used in many fields, such as financial analysis [2], environmental monitoring [3] and translation [4]. Many methods have been developed for time series prediction, such as traditional statistical methods [5,6] as well as artificial neural networks (ANNs) [7,8]. Although traditional statistical approaches are often employed for time series prediction and exhibit good performance, they have difficulties in predicting time series with strong non-linearity [1]. In contrast, ANNs have been successfully employed in time series prediction, because of their powerful ability to identify non-linear relationships between variables [9,10].

Being an extension of the feedforward neural network (FNN), recurrent neural networks (RNNs) are widely used for time series prediction, because of their memory ability to capture the short/long-term dependency for sequential data [11,12]. However, RNNs may encounter the problems of gradient exploding or vanishing [13,14]. Aiming to solve these problems, the long–short term memory (LSTM) neural network is hereby proposed, which is composed of multiple memory blocks by introducing the memory cells to track the history information and gate architectures to control the information flow [15]. This model shows superior learning performance for time series prediction [16]. In recent years, researchers have endeavored to improve the LSTM model in terms of prediction accuracy. For instance, the bidirectional LSTM (BiLSTM) model was proposed to improve the performance of time series prediction [17,18]. Combined with a convolutional neural network (CNN), the prediction performance of the LSTM neural network was considerably improved [19,20]. Additionally, some investigators have integrated the attention-based mechanism to the design of LSTM neural network [21,22], improving its capacity in time series prediction. Literature studies have combined the time series decomposition methods, such as empirical model decomposition and empirical wavelet transformation, with the LSTM model [23–25] to improve the time series prediction accuracy. However, due to their highly complex structure, LSTM-derived models present weaknesses such as high computational cost and large memory requirement [26].

* Correspondence to: No. 100, Pingleyuan, Chaoyang District, Beijing, 100124, China.
E-mail address: wenjing.li@bjut.edu.cn (W. Li).

https://doi.org/10.1016/j.knosys.2022.109608
0950-7051/

To overcome these shortcomings brought by the complex structure of LSTM, researchers have proposed methods for model simplification or compression to improve the computational complexity of the LSTM neural network. In recent years, a variety of simplified models have been proposed by simplifying the gate architectures of the LSTM neural network, such as the gated recurrent unit (GRU) [27,28], the minimal gated unit (MGU) [29] and the shared weight LSTM (SWLSTM) network [30]. These models are characterized by improved computational complexity because of their simpler structures, and can achieve comparable prediction accuracy as LSTM. Moreover, several other methods, such as approximate computing [31] and parameter reduction [32,33], are proposed as alternative ways to simplify calculations in LSTM and generate an energy-efficient model.

Nevertheless, due to the large size of the hidden layer, these simplified LSTM models may present the problem of parameter redundancy. The hidden layer size is determined by the number of memory blocks, which in turn directly determines the number of network parameters and is related to the learning capacity of the model, thus affecting the network performance [34,35]. It has been established that the performance of a larger network would be superior, but the generalization performance would degenerate and the required training time would increase with the increasing network size [36]. Therefore, an appropriate hidden layer size is key in the construction of the LSTM neural network.

The most commonly used method for the hidden layer size selection is the traditional random search method [37] or the user-dependent manual selection method [38]. However, achieving a satisfactory performance with these methods depends on multiple trial-and-error tests, which is very time-consuming. In recent years, researchers have focused their efforts on the investigation of an effective way to determine the hidden layer size of the LSTM neural network. For example, evolutionary algorithms, such as the genetic algorithm (GA) and particle swarm optimization (PSO) algorithm, have been used to tune the hidden layer size for LSTM neural network [39–41]. Nevertheless, the use of such methods would incur higher computational cost. Furthermore, Li et al. proposed a probability-based exploration method to enhance the exploration process efficiency for hyperparameter tuning in LSTM neural networks [42], but results might be unsatisfactory if the computational resource is limited. In recent years, the pruning algorithm has provided an alternative way for structural design in ANNs [43,44]. The network begins with a larger size and is followed by the deletion of unimportant nodes or weights based on the network performance, finally generating a compact structure and improving the generalization performance.

Therefore, a pruning algorithm is proposed in this study for a novel simplified LSTM neural network, with the model finally applied to time series prediction. The main contributions of this study are summarized as follows:

(1) A novel simplified LSTM (SLSTM) neural network is designed using a hybrid strategy combining structure simplification with gate parameter reduction. Specifically, the computational complexity of the LSTM neural network is improved by removing the forget gate and the calculations are further simplified by removing the input matrix.

(2) A pruning algorithm is proposed based on partial least squares (PLS) regression to reduce the hidden layer size by pruning the redundant memory blocks in the SLSTM neural network. The generated model is named as PSLSTM. The regression coefficients of PLS are used as a metric to measure the importance of memory blocks. The blocks with small PLS regression coefficients are regarded as less important, and then are merged with the most correlated block.

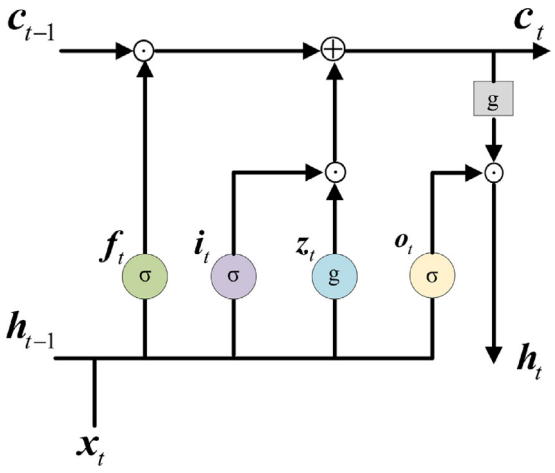

Fig. 1. The architecture of a memory block in LSTM.

(3) The stability of the SLSTM network based on the pruning algorithm (PSLSTM) is theoretically analyzed and further demonstrated by experiments in the training process.

The remainder of this paper is organized as follows. Basic concepts, including the LSTM network and PLS regression are briefly introduced in Section 2. The proposed PLSLTM is described in detail and its stability is analyzed theoretically in Section 3. Several benchmark and practical experiments are conducted in Section 4. The discussions and conclusions of this paper are given in Section 5.

2. Basic concepts

2.1. long–short term memory for time series prediction

The LSTM network with only one hidden layer and without peephole connections, which is the most widely used LSTM network, is utilized in the present study [45]. The LSTM is composed of a set of memory blocks, with each block incorporating a memory cell and three control gates, namely, the input gate, the forget gate and the output gate (Fig. 1). The memory cell maintains the cell state, $\boldsymbol{c}_t$, over time, with its information controlled by the three gates. Specifically, the input gate $\boldsymbol{i}_t$ controls the information to be input to the cell, the forget gate $\boldsymbol{f}_t$ controls the amount of previous information that is forgotten, and the output gate $\boldsymbol{o}_t$ controls the amount of block output.

For time series prediction, given an input time series data $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t, \ldots, \boldsymbol{x}_T]$, with $\boldsymbol{x}_t$ $(t = 1, 2, \ldots, T)$ represented as $\boldsymbol{x}_t = [x_1(t), x_2(t), \ldots, x_N(t)]^{\mathrm{T}}$, the LSTM computes the block output $\boldsymbol{h}_t$ at time t according to:

$$\begin{cases} \boldsymbol{i}_t = \sigma\left(\boldsymbol{W}_i\boldsymbol{x}_t + \boldsymbol{U}_i\boldsymbol{h}_{t-1} + \boldsymbol{b}_i\right) \\ \boldsymbol{f}_t = \sigma\left(\boldsymbol{W}_f\boldsymbol{x}_t + \boldsymbol{U}_f\boldsymbol{h}_{t-1} + \boldsymbol{b}_f\right) \\ \boldsymbol{z}_t = \mathrm{g}\left(\boldsymbol{W}_z\boldsymbol{x}_t + \boldsymbol{U}_z\boldsymbol{h}_{t-1} + \boldsymbol{b}_z\right) \\ \boldsymbol{c}_t = \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \boldsymbol{z}_t \\ \boldsymbol{o}_t = \sigma\left(\boldsymbol{W}_o\boldsymbol{x}_t + \boldsymbol{U}_o\boldsymbol{h}_{t-1}^{+}\boldsymbol{b}_o\right) \\ \boldsymbol{h}_t = \boldsymbol{o}_t \odot \mathrm{g}\left(\boldsymbol{c}_t\right) \end{cases} \tag{1}$$

where subscripts i, f, z, and o of the matrices correspond to the input gate, forget gate, block input and output gate, respectively; The terms of $\boldsymbol{W}$, $\boldsymbol{U}$ and $\boldsymbol{b}$ denote the weight matrix, recurrent weight matrix and bias weight matrix; σ is the logistic sigmoid activation function ($\sigma(x) = 1/(1 + e^{-x})$) and g is the hyperbolic tangent activation function ($g(x) = \tanh(x)$); $\odot$ represents the operation of element-wise multiplication.

The final output of the LSTM is then calculated as:

$$\boldsymbol{y}_t = \boldsymbol{W}_y \boldsymbol{h}_t, \tag{2}$$

where $\boldsymbol{W}_y$ is the connection matrix between the block output and the final output of the LSTM.

2.2. Partial Least Squares (PLS) regression

PLS regression can be used to extract the information of the relationship between variables and has been widely used to study the interdependence between two groups of multiple variables that are related [46–48]. Given the standardized observations [$\boldsymbol{C}$, $\boldsymbol{Y}$] with the independent variable $\boldsymbol{C}$ and the dependent variable $\boldsymbol{Y}$, the algorithm is described in detail as follows.

Extracted from $\boldsymbol{C}$ and $\boldsymbol{Y}$, the first components t_1 and u_1 should have the maximal covariance, indicating that they contain as much information of the original data as possible and have the greatest correlation, resulting in the residual matrix from which the subsequent components are extracted. The extraction continues iteratively until the established PLS regression equation reaches the desired accuracy, and the variables $\boldsymbol{C}$ and $\boldsymbol{Y}$ are represented by:

$$\boldsymbol{C} = \boldsymbol{T}\boldsymbol{P}^{\mathrm{T}} + \boldsymbol{E} = \sum_{i=1}^{r} t_i p_i^{\mathrm{T}} + \boldsymbol{E} \tag{3}$$

$$\boldsymbol{Y} = \boldsymbol{U}\boldsymbol{Q}^{\mathrm{T}} + \boldsymbol{F} = \sum_{i=1}^{r} u_i q_i^{\mathrm{T}} + \boldsymbol{F}, \tag{4}$$

where $\boldsymbol{T}$, $\boldsymbol{P}$ and $\boldsymbol{E}$ are the score matrix, load matrix and residual matrix of the independent variable $\boldsymbol{C}$, respectively, $\boldsymbol{U}$, $\boldsymbol{Q}$ and $\boldsymbol{F}$ are the score matrix, load matrix and residual matrix of the dependent variable $\boldsymbol{Y}$, respectively and r is the number of components.

A linear relation between $\boldsymbol{T}$ and $\boldsymbol{U}$ exists, i.e., $\boldsymbol{U} = b\boldsymbol{T} + \boldsymbol{V}$, where b is a constant and $\boldsymbol{V}$ is the error term. Given the score matrix $\boldsymbol{T} = \boldsymbol{C}\boldsymbol{W}$ and ignoring the error term, the PLS regression equation between $\boldsymbol{C}$ and $\boldsymbol{Y}$ can be obtained as:

$$\boldsymbol{Y} = b\boldsymbol{C}\boldsymbol{R} + \boldsymbol{F}, \tag{5}$$

where $\boldsymbol{R} = \boldsymbol{W}\boldsymbol{Q}^{\mathrm{T}}$ is the regression coefficient and $\boldsymbol{W}$ can be obtained by the eigenvectors corresponding to the largest eigenvalue of the matrix $\boldsymbol{C}^T\boldsymbol{Y}\boldsymbol{Y}^T\boldsymbol{C}$.

In the present study, PLS regression is applied to analyze the relationship between the memory cell state and the final output of the LSTM, which is the basis for the design of pruning algorithm for SLSTM.

3. Pruning algorithm for simplified long–short term memory neural network (PSLSTM)

In the present study, a novel SLSTM neural network is designed to simplify the internal structure of LSTM. Additionally, a pruning algorithm based on PLS is proposed to reduce its redundant hidden layer size, generating a simple and compact structure for LSTM neural network.

3.1. Simplified long–short-term memory neural network

In this paper, a novel SLSTM neural network is built using a hybrid strategy, combining structure simplification and parameter reduction for gates. The following paragraphs detail the individual steps.

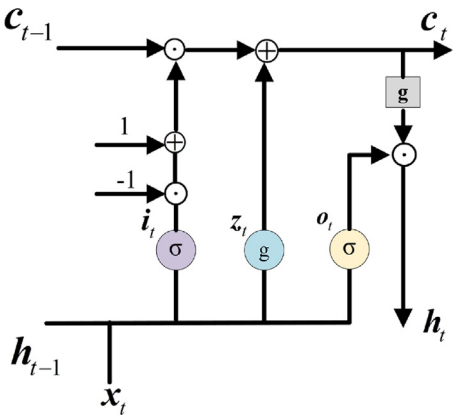

Fig. 2. The architecture of a memory block in SLSTM with the gate structure simplification.

3.1.1. Gate structure simplification

By removing the forget gate, the structure of LSTM is first simplified with the input and output gates as the two control gates, the architecture of which is shown in Fig. 2. The corresponding formulas are listed as:

$$\begin{cases} \boldsymbol{i}_t = \sigma\left(\boldsymbol{W}_i \boldsymbol{x}_t + \boldsymbol{U}_i \boldsymbol{h}_{t-1} + \boldsymbol{b}_i\right) \\ \boldsymbol{z}_t = \mathrm{g}\left(\boldsymbol{W}_z \boldsymbol{x}_t + \boldsymbol{U}_z \boldsymbol{h}_{t-1} + \boldsymbol{b}_z\right) \\ \boldsymbol{c}_t = (1 - \boldsymbol{i}_t) \odot \boldsymbol{c}_{t-1} + \boldsymbol{z}_t \\ \boldsymbol{o}_t = \sigma\left(\boldsymbol{W}_o \boldsymbol{x}_t + \boldsymbol{U}_o \boldsymbol{h}_{t-1}^{+} \boldsymbol{b}_o\right) \\ \boldsymbol{h}_t = \boldsymbol{o}_t \odot \mathrm{g}\left(\boldsymbol{c}_t\right) \\ \boldsymbol{y}_t = \boldsymbol{W}_y \boldsymbol{h}_t \end{cases} \tag{6}$$

From Fig. 2 and the above formulas, it is clear that the calculation is the same as in the standard LSTM except for the memory cell. The memory for the previous information is controlled by factor $(1 - \boldsymbol{i}_t)$ instead of the forget gate for the memory cell.

3.1.2. Gate parameter reduction

To further improve the computational complexity of LSTM, the calculation of LSTM is simultaneously simplified by gate parameter reduction. Specifically, considering the memory cell reflects the information of the current input, the input and output gates are computed using only the previous hidden layer output and the bias to minimize gate redundancy. The formulas are listed as follows:

$$\begin{cases} \boldsymbol{i}_t = \sigma\left(\boldsymbol{U}_i \boldsymbol{h}_{t-1} + \boldsymbol{b}_i\right) \\ \boldsymbol{z}_t = \mathrm{g}\left(\boldsymbol{W}_z \boldsymbol{x}_t + \boldsymbol{U}_z \boldsymbol{h}_{t-1} + \boldsymbol{b}_z\right) \\ \boldsymbol{c}_t = (1 - \boldsymbol{i}_t) \odot \boldsymbol{c}_{t-1} + \boldsymbol{z}_t \\ \boldsymbol{o}_t = \sigma\left(\boldsymbol{U}_o \boldsymbol{h}_{t-1}^{+} \boldsymbol{b}_o\right) \\ \boldsymbol{h}_t = \boldsymbol{o}_t \odot \mathrm{g}\left(\boldsymbol{c}_t\right) \\ \boldsymbol{y}_t = \boldsymbol{W}_y \boldsymbol{h}_t \end{cases} \tag{7}$$

Accordingly, the number of parameters updated in the training process can be reduced. By comparing Eq. (7) with Eq. (6), assuming that the input dimension is N and the hidden layer size is M, the number of updated parameters is reduced by $2NM$, when the connection matrices $\mathbf{W}_i$ and $\mathbf{W}_o$ are ignored.

3.2. Pruning algorithm for SLSTM

The present study proposes a pruning algorithm to further reduce the hidden layer size of SLSTM by merging the unimportant memory blocks with their most correlated block, resulting in a model named PSLSTM. The PLS-based pruning algorithm is first introduced in detail, followed by the analysis of the output invariance after pruning.

3.2.1. Design of the PLS-based pruning algorithm

Using PLS regression method, the importance of each LSTM block is represented by the corresponding regression coefficient of its internal memory cell. In brief, the relationship between the independent variables $\boldsymbol{c}_t$ and dependent variables $\boldsymbol{y}_t$ is as follows:

$$\boldsymbol{y}_t = a_1\boldsymbol{c}_{t,1} + \cdots + a_m\boldsymbol{c}_{t,m} + \cdots + a_{M(t)}\boldsymbol{c}_{t,M(t)}, \tag{8}$$

where a_m is PLS regression coefficient and $M(t)$ is the number of memory blocks at time t.

When the condition $a_m(t) < \theta \cdot M(t) \cdot \bar{a}$ is met, block m is identified as an unimportant block at time t. Here, $\bar{a}$ is the average of the PLS regression coefficients over all blocks, and θ is a predefined threshold set within the range of (0,0.1] by experience. The unimportant block is then merged with its most correlated block, with the correlation measured according to:

$$S_{mn} = \frac{\sum_{t=1}^{T}(\boldsymbol{c}_{t,m} - \bar{\boldsymbol{c}}_m)(\boldsymbol{c}_{t,n} - \bar{\boldsymbol{c}}_n)}{\sigma_m\sigma_n}, \tag{9}$$

where $\boldsymbol{c}_{t,m}$ and $\boldsymbol{c}_{t,n}$ are the cell states in blocks m and n at time t, respectively, $\bar{\boldsymbol{c}}_m$ and $\bar{\boldsymbol{c}}_n$ are their average over T time points, and σ_m and σ_n are their standard deviations, respectively. The block with the highest correlation coefficient is then selected as the most correlated block.

Let blocks m_1 and m_2 denote the unimportant block and its most correlated block to merged into a new block m_0. All input and recurrent weights as well as the bias to the block m_0 are first randomly assigned within the range of $[-1,1]$. The block output $\boldsymbol{h}_{t,m_0}$ is then calculated according to Eq. (7). Finally, the connection weight matrix from the block output to the final output of the PSLSTM is assigned as:

$$\boldsymbol{W}_{y,m_0} = \frac{\boldsymbol{W}_{y,m_1}\boldsymbol{h}_{t,m_1} + \boldsymbol{W}_{y,m_2}\boldsymbol{h}_{t,m_2}}{\boldsymbol{h}_{t,m_0}} \tag{10}$$

Remark. The threshold for unimportant block identification is adaptive to the number of blocks. In this study, the threshold is calculated by $\theta \cdot M(t) \cdot \bar{a}$, where $M(t)$ would decrease in the merging process. Therefore, the threshold is larger initially and becomes smaller later, which indicates the rate of neuron merging is faster at beginning and slower while the merging process progresses.

3.2.2. Analysis of the output invariance after pruning

In this section, the output invariance of the PSLSTM network after pruning is analyzed. The theorem and its proof are given as follows.

Theorem 1 (*Output Invariance in PSLSTM After Pruning*)**.** *The proposed PSLSTM can guarantee output invariance, if the unimportant block is merged with its most correlated block and its connection weights from the block output to the final output of the network are updated according to Eq. (10).*

Proof. Assuming that blocks m_1 and m_2 are merged into the block m_0, the output at time t is given by:

$$\boldsymbol{y}'_t = \sum_{m=1}^{M-1}\boldsymbol{W}_{y,m}\boldsymbol{h}_{t,m} = \sum_{m=1,m\neq m_0}^{M-1}\boldsymbol{W}_{y,m}\boldsymbol{h}_{t,m} + \boldsymbol{W}_{y,m_0}\boldsymbol{h}_{t,m_0} \tag{11}$$

By substituting Eq. (10) into Eq. (11):

$$\begin{aligned}\boldsymbol{y}'_t &= \sum_{m=1,m\neq m_0}^{M-1}\boldsymbol{W}_{y,m}\boldsymbol{h}_{t,m} + \frac{\boldsymbol{W}_{y,m_1}\boldsymbol{h}_{t,m_1} + \boldsymbol{W}_{y,m_2}\boldsymbol{h}_{t,m_2}}{\boldsymbol{h}_{t,m_0}}\cdot\boldsymbol{h}_{t,m_0}\\ &= \sum_{m=1,m\neq m_0}^{M-1}\boldsymbol{W}_{y,m}\boldsymbol{h}_{t,m} + \boldsymbol{W}_{y,m_1}\boldsymbol{h}_{t,m_1} + \boldsymbol{W}_{y,m_2}\boldsymbol{h}_{t,m_2},\\ &= \sum_{m=1}^{M}\boldsymbol{W}_{y,m}\boldsymbol{h}_{t,m} = \boldsymbol{y}_t\end{aligned} \tag{12}$$

where $\boldsymbol{y}_t$ and $\boldsymbol{y}'_t$ represent the actual network output before and after pruning, respectively. Eq. (12) shows that the network output does not change after the pruning, which ensures the network stability for PSLSTM after pruning.

3.3. Training algorithm

As a classic learning algorithm for RNN, Backpropagation Through Time (BPTT) [49,50] is used as the learning algorithm of the proposed model for parameter updating, and is briefly described as follows.

The cost function is defined as:

$$\mathbf{E}(t) = \frac{1}{2}\left(\hat{y}_t - y_t\right)^2 \tag{13}$$

where $\hat{\boldsymbol{y}}_t$ and $\boldsymbol{y}_t$ are the desired and actual outputs of PSLSTM at time t, respectively.

First, $\delta\boldsymbol{h}_t$ is computed as $\partial\boldsymbol{E}(t)/\partial\boldsymbol{h}_t$ in the last iteration. Otherwise, $\delta\boldsymbol{h}_t$ is calculated according to:

$$\delta\boldsymbol{h}_t = \delta\boldsymbol{z}_{t+1}\boldsymbol{U}_z + \delta\boldsymbol{i}_{t+1}\boldsymbol{U}_i + \delta\boldsymbol{o}_{t+1}\boldsymbol{U}_o \tag{14}$$

Second, the changes of the parameters related to the memory cell and the gates are calculated as:

$$\delta\boldsymbol{c}_t = \delta\boldsymbol{h}_t \odot \boldsymbol{o}_t \odot g'(\boldsymbol{c}_t) \tag{15}$$

$$\delta\boldsymbol{o}_t = \delta\boldsymbol{h}_t \odot g(\boldsymbol{c}_t) \odot \sigma'(\hat{\boldsymbol{o}}_t) \tag{16}$$

$$\delta\boldsymbol{i}_t = \delta\boldsymbol{c}_t \odot \boldsymbol{z}_t \odot \sigma'(\hat{\boldsymbol{i}}_t) \tag{17}$$

$$\delta\boldsymbol{z}_t = \delta\boldsymbol{c}_t \odot \boldsymbol{i}_t \odot g'(\hat{\boldsymbol{z}}_t), \tag{18}$$

where $\hat{\boldsymbol{o}}_t, \hat{\boldsymbol{i}}_t, \hat{\boldsymbol{z}}_t$ represent the raw values before transformation by the corresponding activation function for the output gate, input gate and block input, respectively.

Finally, the input weights, recurrent weights and bias at time t, respectively, are updated according to:

$$\boldsymbol{W}_{z,t} = \boldsymbol{W}_{z,t+1} - \eta \times \delta\boldsymbol{W}_{z,t} \tag{19}$$

$$\boldsymbol{U}_{\Omega,t} = \boldsymbol{U}_{\Omega,t+1} - \eta \times \delta\boldsymbol{U}_{\Omega,t} \tag{20}$$

$$\boldsymbol{b}_{\Omega,t} = \boldsymbol{b}_{\Omega,t+1} - \eta \times \delta\boldsymbol{b}_{\Omega,t}, \tag{21}$$

with their gradients calculated as:

$$\delta\boldsymbol{W}_{z,t} = \delta\boldsymbol{z}_t \otimes \boldsymbol{x}_t \tag{22}$$

$$\delta\boldsymbol{U}_{\Omega,t} = \delta\Omega_t \otimes \boldsymbol{h}_{t-1} \tag{23}$$

$$\delta\boldsymbol{b}_{\Omega,t} = \delta\Omega_t, \tag{24}$$

where $\otimes$ is a matrix cross product, Ω is one of $\{\boldsymbol{z}, \boldsymbol{i}, \boldsymbol{o}\}$ and η is the learning rate.

The training process continues until the training RMSE converges to the desired RMSE (E_d) or the training epochs reaches the maximum number of epochs ($I_{\max}$).

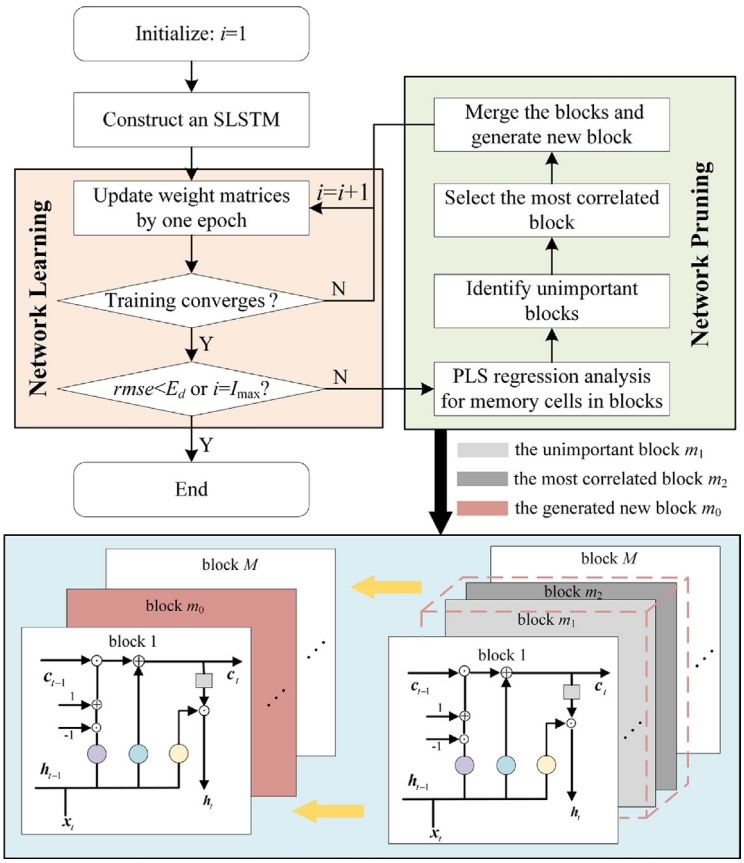

Fig. 3. The overall procedure for the PSLSTM design.

Remark. Each training algorithm suitable for RNN can be used for the proposed model as well.

3.4. Procedures for the design of PSLSTM

An SLSTM is first constructed by a hybrid strategy combining structure simplification and parameter reduction for gates. The pruning algorithm based on PLS regression is then used to merge the unimportant blocks with their most correlated block, resulting in the reduction of the redundant hidden layer size. The overall procedure for the design of PSLSTM is listed in Algorithm 1, and the schematic diagram is illustrated in Fig. 3.

4. Experiments and results analysis

Several experiments are conducted to evaluate the performance of PSLSTM on time series prediction, in terms of both prediction accuracy and computational complexity. The experimental datasets are first described in detail, and then numerous experimental results are presented in this section.

4.1. Experimental dataset

Four datasets, including (1) Mackey–Glass time series prediction, (2) Sunspot time series prediction, (3) PM2.5 prediction in Beijing, and (4) BOD concentration prediction in wastewater treatment process, are used to evaluate the performance of the proposed PSLSTM on time series prediction. While the Mackey–Glass time series prediction is a benchmark problem, the other three datasets are collected from real dynamic systems with strong non-linearity. The detailed information of these datasets is described as follows, with the hyperparameters set according to Table 1.

Experiment 1: Mackey–Glass time-series prediction

Mackey–Glass time series, which exhibits some periodicity and chaotic characteristics, is a benchmark time series and extensively used for model performance evaluation for time series prediction [51,52]. It is generated by the following discrete equations:

$$x(t+1) = (1-a)\,x(t) + \frac{b x(t-\tau)}{1 + x^{10}(t-\tau)} \tag{25}$$

where $a = 0.1$, $b = 0.2$, $\tau = 17$, $x(0) = 1.2$.

Input: $[x(t), x(t-6), x(t-12), x(t-18)]$; Output: $x(t+6)$.

Out of the total of 1000 samples generated, the first 500 groups are used as the training samples and the last 500 groups as the testing samples.

Experiment 2: Sunspot time series prediction

Being a basic phenomenon of solar activity, the prediction for the sunspot time series is of great practical significance and is widely used for the performance evaluation on a real dynamic system. The monthly smoothed sunspot time series is downloaded from the World Data Center (https://wwwbis.sidc.be/silso/), and the time series from November 1834 to June 2001 is selected in this experiment.

Input: $[x(t), x(t-2), x(t-4), x(t-6), x(t-8)]$; Output: $x(t+1)$.

A total of 2000 groups are generated with the first 1000 groups treated as training samples and the remaining groups as the testing samples.

Algorithm 1 Design of PSLSTM

Input: Training samples $\boldsymbol{X}$

Output: $\boldsymbol{Y}$

Procedure: Initialize the hidden layer size M, the learning rate η, the desired training RMSE (E_d), the maximum number of epochs ($I_{\max}$), and the parameter for the merging threshold θ.

Construct the SLSTM with the hidden layer size M.

For i = 1: $I_{\max}$ **Do:**

  For t=1:T **Do:**

    Calculate the output of each structure of the SLSTM network; % Eq. (7)

    Update all weight matrices by BPTT algorithm; % Eqs.(14) - (24)

  End For

  Calculate training RMSE (*rmse*);

  If *rmse* converges

    If $rmse \geqslant E_d$

      Calculate the PLS regression coefficient a_m for each block; % Eqs. (3) - (5)

      If $a_m < \theta \cdot M \cdot \bar{a}$

        Identify the unimportant blocks;

        Calculate the correlation coefficient between unimportant blocks and other blocks; % Eq. (9)

        Select the most correlated block;

        Merge the unimportant block with its most correlated block and assign the connection matrix and bias for the new block; % Eq. (10)

      End If

    Else

      break;

    End If

  End If

End For

Table 1
Setting of hyperparameters in the experiments.

	M	θ	η	E_d	$I_{\max}$
Experiment 1	20	0.06	0.01	0.0045	5000
Experiment 2	40	0.06	0.01	0.0080	2000
Experiment 3	40	0.06	0.01	0.0380	1000
Experiment 4	40	0.06	0.01	0.0200	1000

Experiment 3: PM2.5 time series prediction in Beijing

PM2.5 is an important and harmful air pollutant, which may cause damages to human health and the ecological environment. The accurate prediction for PM 2.5 is crucial but difficult due to the complex and non-linear process. The dataset for PM2.5 time series prediction is provided by UCI Repository of Machine Learning (http://archive.ics.uci.edu/ml/datasets.php). The samples are recorded from 0:00 on January 1, 2010 to 24:00 on December 31, 2011 with samples recorded each hour. The variables of Dew Point (*DP*), Temperature (T) and Combined Wind Direction (*CWD*) are selected as the variables with high correlation to PM2.5, identified by correlation analysis based on mutual information.

Input: *PM*2.5, *DP*, T and *CWD* at current time t and previous time $t-1$ and $t-2$; Output: *PM*2.5 at the next time $t+1$.

After deleting abnormal and missing data, a total of 16,120 samples are selected with the 8091 samples from year 2010 used as the training samples and the remaining 8029 samples from year 2011 as the testing samples.

Experiment 4: BOD prediction in wastewater treatment process

BOD is one of the main indicators for wastewater quality evaluation. Since the wastewater treatment process is a highly non-linear process with multiple variables, accurate prediction of BOD concentration is challenging. The dataset for biochemical oxygen demand (BOD) prediction includes data collected from a wastewater treatment plant in Beijing in 2011. The effluent total nitrogen (*ETN*), influent phosphate (*IP*) and influent chromaticity (*IC*) are selected as the variables with high correlation to BOD, identified by correlation analysis based on mutual information.

Table 2
Comparison results between PSLSTM and SLSTM with different hidden layer sizes.

Datasets	Models	Hidden layer size	Training RMSE	Testing RMSE
Experiment 1	PSLSTM	5	0.0045	0.0050
		20	0.0045	0.0055
		15	0.0045	0.0053
	SLSTM	10	0.0045	0.0052
		5	0.0045	0.0050
		2	0.0069	0.0073
Experiment 2	PSLSTM	8	0.0080	0.0096
		40	0.0080	0.0098
		30	0.0080	0.0096
	SLSTM	20	0.0080	0.0094
		10	0.0080	0.0097
		2	0.0085	0.0109
Experiment 3	PSLSTM	8	0.0380	0.0389
		40	0.0380	0.0520
		30	0.0380	0.0524
	SLSTM	20	0.0380	0.0442
		10	0.0380	0.0394
		2	0.0394	0.0609
Experiment 4	PSLSTM	15	0.0200	0.0247
		40	0.0200	0.0414
	SLSTM	30	0.0200	0.0447
		20	0.0200	0.0250
		10	0.0200	0.0297
		2	0.0200	0.0492

Input: *BOD*, *ETN*, *IP* and *IC* at current time t and previous time $t-1$ and $t-2$; Output: *BOD* at the next time $t+1$.

A total of 362 samples are acquired with the first 250 samples used for training and the last 112 samples for testing.

4.2. Experimental results

The effectiveness of the proposed PSLSTM is evaluated and further compared with other models. The experimental results are described in detail as follows. The experimental environment is Windows XP with a CPU (2.30 GHz) and RAM (4 GB) using MATLAB R2018b.

4.2.1. Performance evaluation of PSLSTM

In this subsection, the effectiveness of the pruning algorithm for PSLSTM is evaluated. The root mean square error (RMSE) is used to represent the accuracy for time series prediction, which is calculated as:

$$RMSE = \sqrt{\frac{\sum_{t=1}^{T} \left(\hat{\boldsymbol{y}}_t - \boldsymbol{y}_t\right)^2}{T}}, \tag{26}$$

where T is the number of time points. The hyperparameters for each experiment are set according to Table 1.

In Fig. 4, the subfigure (a) in each experiment illustrates the training RMSE curve in the learning process, showing that the training RMSE decreases and converges along the learning process in all these experiments. This demonstrates the convergence of the learning process. Subfigures (b) and (c) depict the fitting curve and error for the testing samples in each experiment, respectively. The fitting curves exhibit the satisfactory fitting results of the PSLSTM, and the fitting error is within an allowable range, demonstrating the effectiveness of the proposed PSLSTM in time series prediction.

To establish the structural pruning process, the changes of hidden layer size during the training process for PSLSTM are illustrated for the four experiments in Fig. 5. The hidden layer size is decreased by the proposed pruning algorithm, with a relatively sharp drop at the early stage followed by a slow pruning rate and then the convergence to a stable size.

Furthermore, the results of PSLSTM are compared with those from the SLSTM to validate the performance of PSLSTM. The structure of SLSTM is predefined and fixed during the learning process. The predefined hidden layer size is set from 2 to 20 for experiment 1 (i.e., the benchmark problem) and to 40 for experiments 2 – 4 (i.e., the practical problems). The curves for the testing RMSE along with the hidden layer size for SLSTM are plotted in Fig. 6, with the red asterisk representing the results from PSLSTM. The curves show that the testing RMSE first decreases with the increasing hidden layer size, followed by a relatively flat valley and a late increasing trend. The experimental results from both PSLSTM and SLSTM are listed in detail in Table 2, for different hidden layer sizes. These results indicate that a larger hidden layer size might cause a relatively larger testing RMSE due to over-fitting, while a smaller hidden layer size would degenerate the modeling accuracy due to under-fitting. This demonstrates the significance of finding a suitable structure for a satisfying modeling accuracy, which is the aim of the present study. The PSLSTM experimental results show that PSLSTM can achieve a relatively small testing RMSE with a compact structure, that is, the trade-off between a good generalization performance and a compact structure.

4.2.2. Comparison results with other models

To further prove the superior performance of PSLSTM, the proposed model is compared with other models, including SLSTM-PSO, PLSTM, SLSTM, GRU and LSTM. Detailed information on these models is reported in Table 3. The hidden layer sizes of SLSTM, GRU and LSTM are set the same as the initial hidden layer size of PSLSTM and PLSTM. Furthermore, the hyperparameters for learning, including E_d, $I_{\max}$ and η, are set the same for these models. In addition to RMSE, the mean absolute percentage error (MAPE) and mean absolute error (MAE) are used to evaluate the accuracy for time series prediction, and are calculated by Eqs. (27)–(28), respectively.

$$MAPE = \frac{1}{T}\sum_{t=1}^{T}\left|\frac{\hat{\boldsymbol{y}}_t - \boldsymbol{y}_t}{\boldsymbol{y}_t}\right| \tag{27}$$

$$MAE = \frac{1}{T}\sum_{t=1}^{T}\left|\hat{\boldsymbol{y}}_t - \boldsymbol{y}_t\right| \tag{28}$$

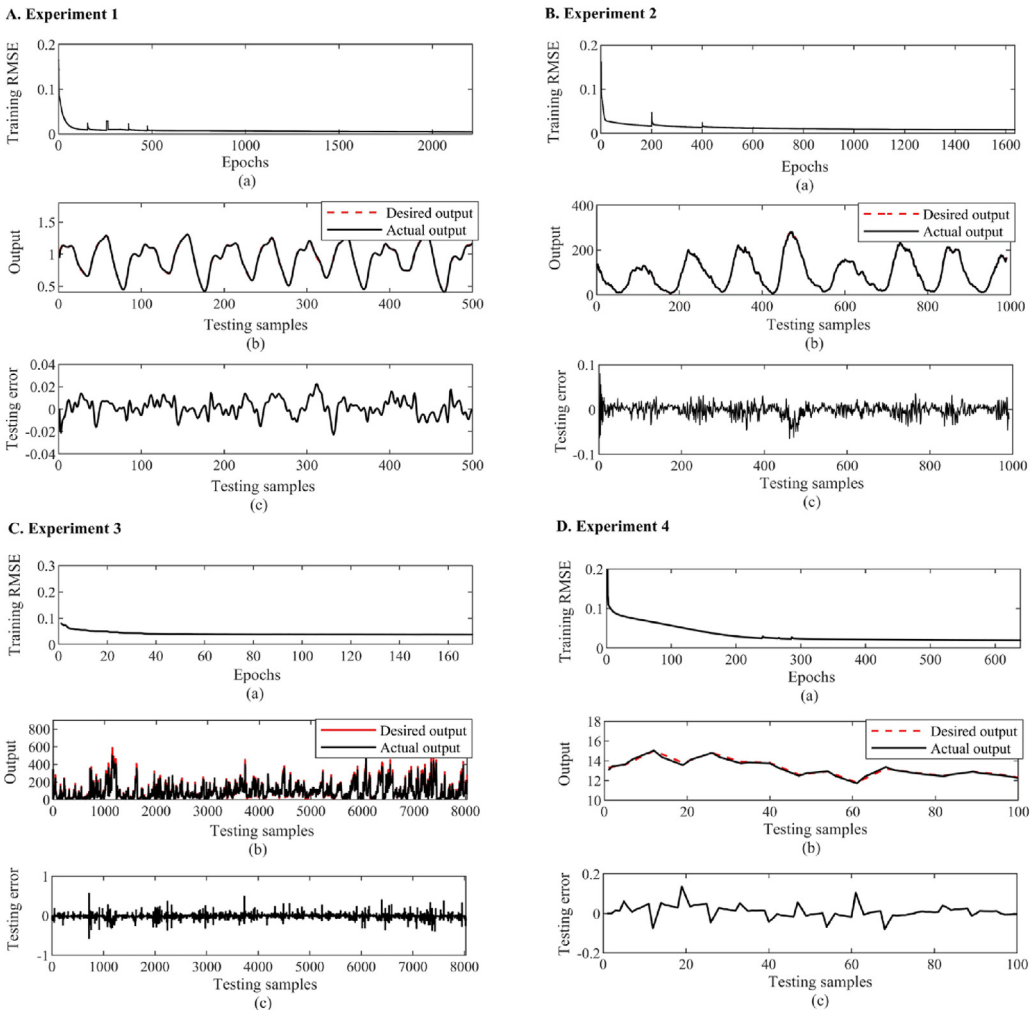

Fig. 4. The training and testing results for PSLSTM in the four experiments. (a) The training RMSE curve converges during the learning process. (b) The fitting curve for testing samples. (c) The fitting error for testing samples.

Table 3
Description of characteristics of the models used for comparison.

Models	Description
SLSTM-PSO	A simplified LSTM with structure optimized by PSO algorithm
PLSTM	A standard LSTM pruned based on PLS regression analysis
SLSTM	An LSTM simplified by the hybrid strategy proposed in this paper
GRU	The gated recurrent unit [27]
LSTM	A standard LSTM

The parameter quantity reduction ratio and the testing time are used as evaluation indicators of the improved computational complexity of these models relative to the standard LSTM. All experiments are run 20 times independently and the average results are calculated for evaluation as listed in Table 4.

Table 4 shows the satisfactory performance of PSLSTM in prediction accuracy and generalization ability, especially for experiments 3 and 4, which produces the highest prediction accuracy for all evaluation indicators. Application of the proposed pruning algorithm to the LSTM neural network (PSLSTM and PLSTM), makes the network structure simpler and thus the computational complexity is improved and the testing time is shortened. Comparing PSLSTM with PLSTM, although PSLSTM does not always have the smaller hidden layer size than PLSTM, namely in experiments 3 and 4, its parameter quantity reduction ratio is higher and its testing time is the shortest in all experiments. This indicates that the hybrid simplification strategy and the pruning algorithm collaborate to achieve a simpler structure. Although the structure of LSTM can be optimized by the PSO algorithm (SLSTM-PSO), a much higher parameter quantity reduction ratio is required compared with the pruning algorithm. Additionally, the comparison results between SLSTM and GRU show that the hybrid simplification strategy can obtain a higher reduction ratio of the parameter quantity, while keeping high prediction accuracy, when assigned the same hidden layer size. The SLSTM, GRU and LSTM in experiments 3 and 4 and LSTM in experiment 1 encounter the overfitting problem caused by the redundant network structure, indicating the necessity of the application of the pruning algorithm as well.

4.2.3. Impact of θ for PSLSTM

In the present study, the hidden layer size is reduced by merging unimportant blocks with their most correlated ones, and the extent of the pruning is determined by the pruning threshold, which is directly affected by parameter θ. Theoretically, if

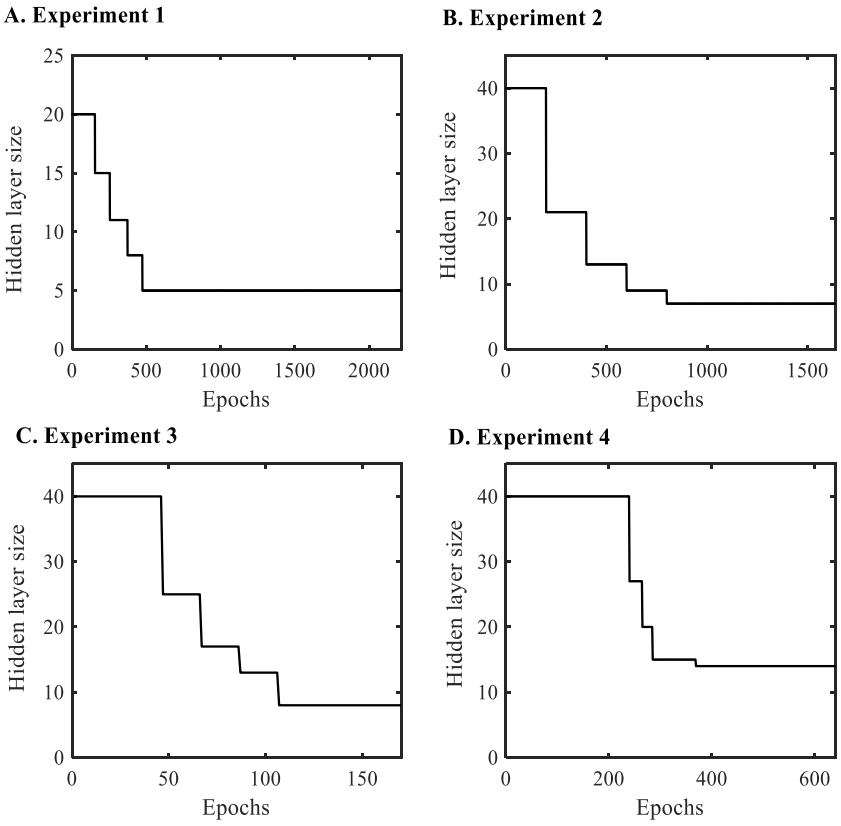

Fig. 5. The pruning process of the hidden layer size in PSLSTM during learning in the four experiments.

Table 4
Comparison results among different models in the four experiments.

	Models	Hidden layer size	Training RMSE	Testing RMSE	MAE	MAPE	Time (s)	Parameter quantity reduction ratio (%)
Experiment 1	PSLSTM	**5**	**0.0045**	0.0050	**0.0055**	0.0063	**0.0275**	**94.50**
	SLSTM- PSO	11	**0.0045**	**0.0049**	0.0058	0.0068	0.0362	78.00
	PLSTM	6	**0.0045**	0.0059	0.0062	0.0067	0.0547	86.80
	SLSTM	20	**0.0045**	0.0051	0.0059	0.0069	0.0531	33.00
	GRU	20	**0.0045**	0.0051	0.0056	**0.0061**	0.0509	25.00
	LSTM	20	**0.0045**	0.0071	0.0076	0.0086	0.0643	–
Experiment 2	PSLSTM	**8**	**0.0080**	0.0096	**0.0098**	**0.0575**	**0.0310**	**96.52**
	SLSTM- PSO	21	0.0081	**0.0095**	0.0098	0.0578	0.0507	81.52
	PLSTM	**8**	**0.0080**	0.0106	0.0109	0.0658	0.0576	93.91
	SLSTM	40	**0.0080**	0.0097	0.0099	**0.0575**	0.0569	30.43
	GRU	40	**0.0080**	0.0109	0.0115	0.0633	0.0521	25.00
	LSTM	40	**0.0080**	0.0107	0.0112	0.0611	0.0776	–
Experiment 3	PSLSTM	8	**0.0380**	**0.0389**	**0.0387**	**0.1087**	**0.1356**	**96.32**
	SLSTM- PSO	20	**0.0380**	0.0397	0.0389	0.1218	0.2718	82.31
	PLSTM	**6**	**0.0380**	0.0400	0.0395	0.1148	0.2971	94.62
	SLSTM	40	**0.0380**	0.0475	0.0489	0.1159	0.3056	36.32
	GRU	40	**0.0380**	0.0502	0.0496	0.1371	0.3183	25.00
	LSTM	40	**0.0380**	0.0498	0.0478	0.1278	0.3859	–
Experiment 4	PSLSTM	15	**0.0200**	**0.0247**	**0.0257**	**0.2169**	**0.0040**	**89.39**
	SLSTM- PSO	16	**0.0200**	0.0318	0.0301	0.2345	0.0071	88.11
	PLSTM	**13**	**0.0200**	0.0289	0.0300	0.2353	0.0171	84.06
	SLSTM	40	**0.0200**	0.0414	0.0401	0.3564	0.0139	36.32
	GRU	40	**0.0200**	0.0432	0.0374	0.3779	0.0154	25.00
	LSTM	40	**0.0200**	0.0403	0.0398	0.3579	0.0197	–

Bold font indicates the best performance.

θ is smaller, the pruning threshold is accordingly smaller, and less blocks would be identified as the unimportant blocks to be merged. Thus, this would lead to a larger hidden layer size and vice versa.

To determine the effect of parameter θ, the changes of the testing RMSE and the hidden layer size are plotted for increasing θ with the step of 0.01 between [0.01,0.1], as shown in Figs. 7 and 8, respectively. The curves show that parameter θ affects

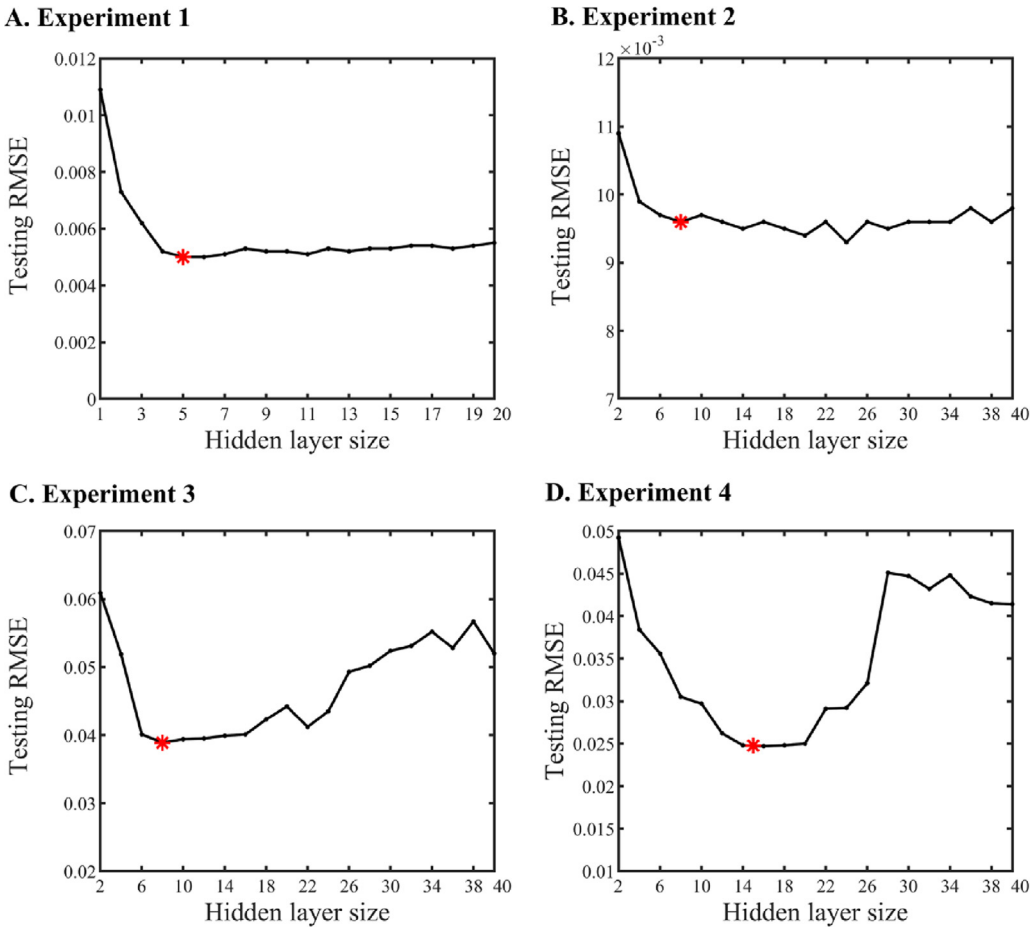

Fig. 6. The curves for testing RMSE along with the increasing hidden layer size in SLSTM, with the red asterisk representing the results from PSLSTM.

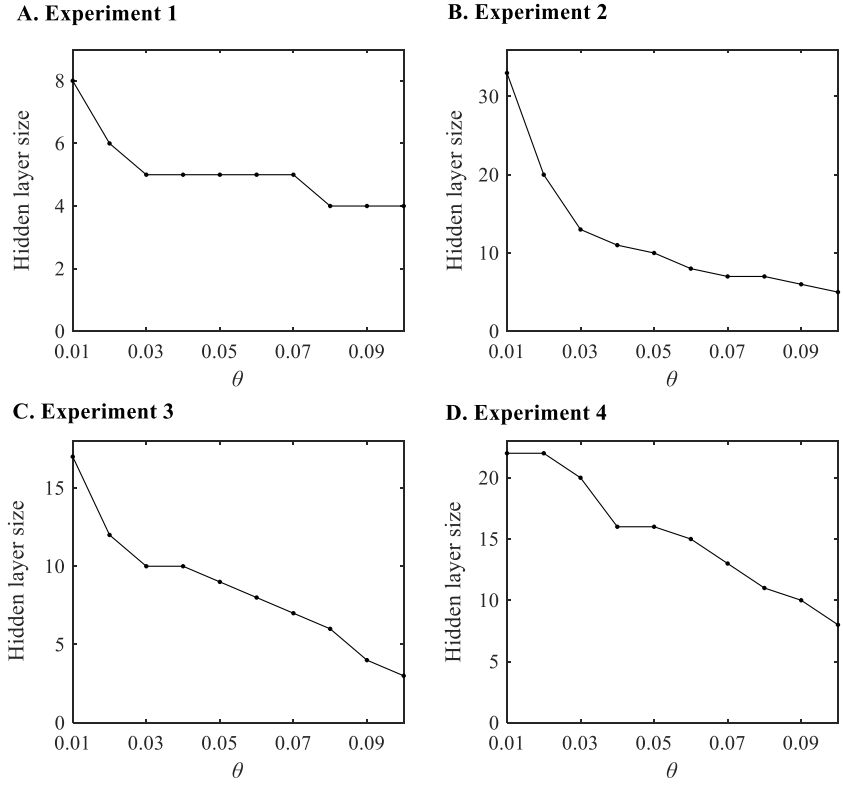

Fig. 7. The hidden layer size decreases with the increasing θ in the four experiments.

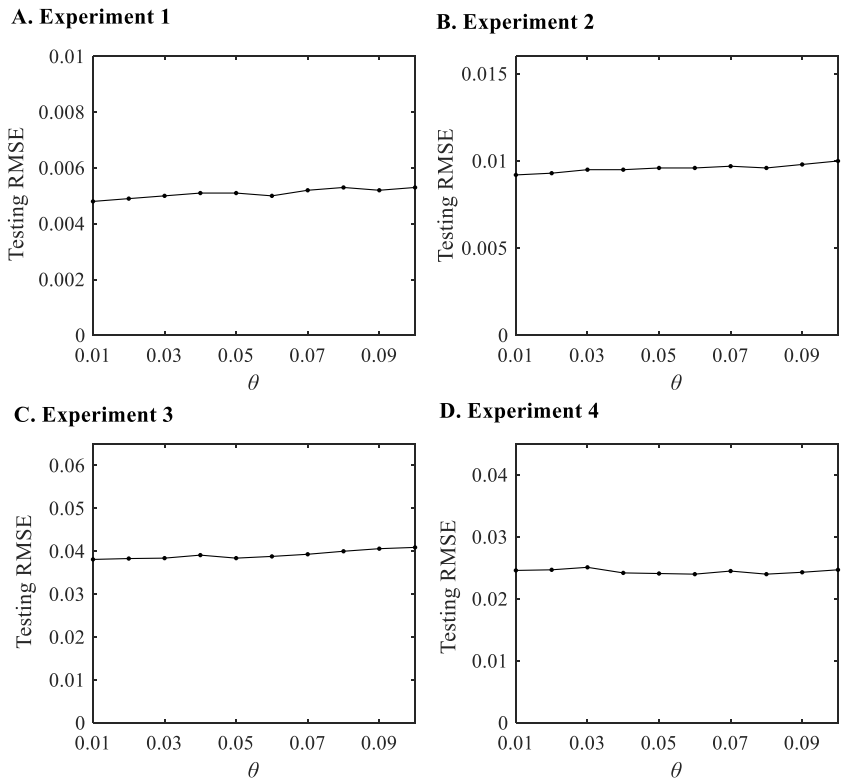

Fig. 8. The testing RMSE remains relatively stable, followed by a slightly increasing trend with increasing θ in the four experiments.

the performance of the PSLSTM. The hidden layer size decreases with increasing θ as expected. These results indicate that larger θ values can reduce the redundant hidden layer size with a larger extent. Although the testing RMSE remains relatively stable, a slightly increasing trend is observed when parameter θ increases, which might be caused by insufficient learning with an over-pruned network structure. Therefore, aiming to achieve a satisfactory prediction accuracy with a compact structure, a moderate value is set for parameter θ, namely 0.06 for all experiments in this study.

5. Discussion and conclusion

A pruning algorithm based on PLS regression is hereby proposed for a novel simplified LSTM neural network (PSLSTM), which is applied to time series prediction. First, a hybrid simplification strategy is proposed to simplify the internal structure of LSTM. Second, a pruning algorithm based on PLS regression analysis is designed to prune the redundant hidden layer size in SLSTM, by merging unimportant blocks with their most correlated blocks. Finally, the performance of the proposed PSLSTM is tested on several benchmark and practical datasets, demonstrating that PSLSTM can achieve satisfactory prediction accuracy with a compact structure.

The characteristics and superior performance of the proposed PSLSTM are summarized as follows:

(1) A simplified internal structure by a hybrid simplification strategy

A hybrid strategy combining the structure simplification and parameter reduction for gates is first developed to simplify the internal structure of the LSTM network, generating the SLSTM network. By comparing SLSTM with both GRU and LSTM, the SLSTM has a higher reduction ratio of the parameter quantity but does not degenerate the prediction accuracy when given the same hidden layer size.

(2) A compact hidden layer size by a PLS-based pruning algorithm

The redundant hidden layer size is pruned by merging unimportant blocks using a PLS-based pruning algorithm. The unimportant blocks below the adaptive pruning threshold are merged with their most correlated blocks. Consequently, the pruning algorithm can determine a compact hidden layer size as demonstrated by experiments, with a sharp drop at the early stage followed by a slow pruning rate and then convergence to a stable size.

(3) Improved computational complexity with a satisfactory prediction performance

By simplifying the internal structure and pruning the redundant hidden layer size, the proposed PSLSTM produces a simpler network structure, which is evaluated using the index of parameter quantity reduction ratio and thus improves computational complexity. Meanwhile, the prediction accuracy and generalization ability remain high after pruning (i.e., either comparable to or better than the models without pruning) and can overcome overfitting.

Despite the considerably superiority of the proposed PSLSTM, some limitations still exist. An important hyperparameter used to determine the pruning threshold is θ, which in this study is selected by experience. Although the impact of θ on the performance of PSLSTM is investigated and its setting is suggested in the experiments, a self-adjusted θ is preferred and will be studied in future work.

CRediT authorship contribution statement

Wenjing Li: Conceptualization, Methodology, Formal analysis, Writing, Supervision, Funding acquisition. **Xiaoxiao Wang:** Investigation, Software, Validation, Writing. **Honggui Han:** Resources, Supervision, Project administration. **Junfei Qiao:** Resources, Supervision, Project administration.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the National Key Research and Development Program of China (2021ZD0112301), National Natural Science Foundation of China (Grant Nos. 62173008, 62021003, 61890930-5, and 62125301), and National Key Research and Development Project [No. 2018YFC1900800-5].

References

[1] J. Hu, X. Wang, Y. Zhang, D. Zhang, M. Zhang, J. Xue, Time series prediction method based on variant LSTM recurrent neural network, Neural Process. Lett. 52 (2020) 1485–1500, http://dx.doi.org/10.1007/s11063-020-10319-3.

[2] H. Yan, H. Ouyang, Financial time series prediction based on deep learning, Wirel. Pers. Commun. 102 (2018) 683–700, http://dx.doi.org/10.1007/s11277-017-5086-2.

[3] J. Zhao, F. Deng, Y. Cai, J. Chen, Long short-term memory - Fully connected (LSTM-FC) neural network for PM2.5 concentration prediction, Chemosphere 220 (2019) 486–492, http://dx.doi.org/10.1016/j.chemosphere.2018.12.128.

[4] R. Qing-dao-er ji, Y. La Su, W.W. Liu, Research on the LSTM mongolian and Chinese machine translation based on morpheme encoding, Neural Comput. Appl. 32 (2020) 41–49, http://dx.doi.org/10.1007/s00521-018-3741-5.

[5] L. Zhu, Y. Wang, Q. Fan, MODWT-ARMA model for time series prediction, Appl. Math. Model. 38 (2014) 1859–1865, http://dx.doi.org/10.1016/j.apm.2013.10.002.

[6] M.S. Omar, H. Kawamukai, Prediction of NDVI using the Holt-Winters model in high and low vegetation regions: A case study of East Africa, Sci. Afr. 14 (2021) e01020, http://dx.doi.org/10.1016/j.sciaf.2021.e01020.

[7] H. Zhang, B. Hu, X. Wang, J. Xu, L. Wang, Q. Sun, Z. Wang, Self-organizing deep belief modular echo state network for time series prediction, Knowl.-Based Syst. 222 (2021) http://dx.doi.org/10.1016/j.knosys.2021.107007.

[8] R. Chandra, M. Zhang, Cooperative coevolution of elman recurrent neural networks for chaotic time series prediction, Neurocomputing 86 (2012) 116–123, http://dx.doi.org/10.1016/j.neucom.2012.01.014.

[9] A. Sagheer, M. Kotb, Time series forecasting of petroleum production using deep LSTM recurrent networks, Neurocomputing 323 (2019) 203–213, http://dx.doi.org/10.1016/j.neucom.2018.09.082.

[10] H. Wang, Z. Yang, Q. Yu, T. Hong, X. Lin, Online reliability time series prediction via convolutional neural network and long short term memory for service-oriented systems, Knowl.-Based Syst. 159 (2018) 132–147, http://dx.doi.org/10.1016/j.knosys.2018.07.006.

[11] Y. Huang, J.J.C. Ying, V.S. Tseng, Spatio-attention embedded recurrent neural network for air quality prediction, Knowl.-Based Syst. 233 (2021) 107416, http://dx.doi.org/10.1016/j.knosys.2021.107416.

[12] Y. Liu, C. Gong, L. Yang, Y. Chen, DSTP-RNN: A dual-stage two-phase attention-based recurrent neural network for long-term and multivariate time series prediction, Expert Syst. Appl. 143 (2020) 113082, http://dx.doi.org/10.1016/j.eswa.2019.113082.

[13] Y. Bengio, P. Simard, P. Frasconi, Learning long-term dependencies with gradient descent is difficult, IEEE Trans. Neural Netw. 5 (1994) 157–166, http://dx.doi.org/10.1109/72.279181.

[14] A. Rehmer, A. Kroll, On the vanishing and exploding gradient problem in gated recurrent units, IFAC-PapersOnLine 53 (2020) 1243–1248, http://dx.doi.org/10.1016/j.ifacol.2020.12.1342.

[15] S. Hochreiter, Long Short-Term Memory, 1780 (1997) 1735–1780.

[16] T. Wang, H. Leung, J. Zhao, W. Wang, Multiseries featural LSTM for partial periodic time-series prediction: A case study for steel industry, IEEE Trans. Instrum. Meas. 69 (2020) 5994–6003, http://dx.doi.org/10.1109/TIM.2020.2967247.

[17] A. Graves, J. Schmidhuber, Framewise phoneme classification with bidirectional LSTM and other neural network architectures, Neural Netw. 18 (2005) 602–610, http://dx.doi.org/10.1016/j.neunet.2005.06.042.

[18] T. Peng, C. Zhang, J. Zhou, M.S. Nazir, An integrated framework of Bi-directional long-short term memory (BiLSTM) based on sine cosine algorithm for hourly solar radiation forecasting, Energy 221 (2021) 119887, http://dx.doi.org/10.1016/j.energy.2021.119887.

[19] A.U. Rehman, A.K. Malik, B. Raza, W. Ali, A hybrid CNN-LSTM model for improving accuracy of movie reviews sentiment analysis, Multimedia Tools Appl. 78 (2019) 26597–26613, http://dx.doi.org/10.1007/s11042-019-07788-7.

[20] I.E. Livieris, E. Pintelas, P. Pintelas, A CNN–LSTM model for gold price time-series forecasting, Neural Comput. Appl. 32 (2020) 17351–17360, http://dx.doi.org/10.1007/s00521-020-04867-x.

[21] J. Qin, Y. Zhang, S. Fan, X. Hu, Y. Huang, Z. Lu, Y. Liu, Multi-task short-term reactive and active load forecasting method based on attention-LSTM model, Int. J. Electr. Power Energy Syst. 135 (2022) 107517, http://dx.doi.org/10.1016/j.ijepes.2021.107517.

[22] H. Abbasimehr, R. Paki, Improving time series forecasting using LSTM and attention models, J. Ambient Intell. Humaniz. Comput. 13 (2022) 673–691, http://dx.doi.org/10.1007/s12652-020-02761-x.

[23] I. Karijadi, S.Y. Chou, A hybrid RF-LSTM based on CEEMDAN for improving the accuracy of building energy consumption prediction, Energy Build. 259 (2022) 111908, http://dx.doi.org/10.1016/j.enbuild.2022.111908.

[24] G. Shi, C. Qin, J. Tao, C. Liu, A VMD-EWT-LSTM-based multi-step prediction approach for shield tunneling machine cutterhead torque, Knowl.-Based Syst. 228 (2021) 107213, http://dx.doi.org/10.1016/j.knosys.2021.107213.

[25] L. Peng, L. Wang, D. Xia, Q. Gao, Effective energy consumption forecasting using empirical wavelet transform and long short-term memory, Energy 238 (2022) 121756, http://dx.doi.org/10.1016/j.energy.2021.121756.

[26] Z. Wang, J. Lin, Z. Wang, Accelerating recurrent neural networks: A memory-efficient approach, IEEE Trans. Very Large Scale Integr. Syst. 25 (2017) 2763–2775, http://dx.doi.org/10.1109/TVLSI.2017.2717950.

[27] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning phrase representations using RNN encoder–decoder for statistical machine translation, in: EMNLP 2014-2014 Conf. Empir. Methods Nat. Lang. Process. Proc. Conf., 2014, pp. 1724–1734, http://dx.doi.org/10.3115/v1/d14-1179.

[28] X. Li, X. Ma, F. Xiao, C. Xiao, F. Wang, S. Zhang, Time-series production forecasting method based on the integration of Bidirectional Gated Recurrent Unit (Bi-GRU) network and Sparrow Search Algorithm (SSA), J. Pet. Sci. Eng. 208 (2022) 109309, http://dx.doi.org/10.1016/j.petrol.2021.109309.

[29] G.B. Zhou, J. Wu, C.L. Zhang, Z.H. Zhou, Minimal gated unit for recurrent neural networks, Int. J. Autom. Comput. 13 (2016) 226–234, http://dx.doi.org/10.1007/s11633-016-1006-2.

[30] Z. Zhang, L. Ye, H. Qin, Y. Liu, C. Wang, X. Yu, X. Yin, J. Li, Wind speed prediction method using shared weight long short-term memory network and Gaussian process regression, Appl. Energy 247 (2019) 270–284, http://dx.doi.org/10.1016/j.apenergy.2019.04.047.

[31] S. Sen, A. Raghunathan, Approximate computing for Long Short Term Memory (LSTM) neural networks, IEEE Trans. Comput. Des. Integr. Circuits Syst. 37 (2018) 2266–2276, http://dx.doi.org/10.1109/TCAD.2018.2858362.

[32] T. Ergen, A.H. Mirza, S.S. Kozat, Energy-efficient LSTM networks for online learning, IEEE Trans. Neural Netw. Learn. Syst. 31 (2020) 3114–3126, http://dx.doi.org/10.1109/TNNLS.2019.2935796.

[33] Y. Lu, F.M. Salem, Simplified gating in long short-term memory (LSTM) recurrent neural networks, in: Midwest Symp. Circuits Syst. 2017-Augus, 2017, pp. 1601–1604, http://dx.doi.org/10.1109/MWSCAS.2017.8053244.

[34] C. Zheng, S. Wang, Y. Liu, C. Liu, W. Xie, C. Fang, S. Liu, A novel equivalent model of active distribution networks based on LSTM, IEEE Trans. Neural Netw. Learn. Syst. 30 (2019) 2611–2624, http://dx.doi.org/10.1109/TNNLS.2018.2885219.

[35] H. Abbasimehr, M. Shabani, M. Yousefi, An optimized model using LSTM network for demand forecasting, Comput. Ind. Eng. 143 (2020) 106435, http://dx.doi.org/10.1016/j.cie.2020.106435.

[36] K. Greff, R.K. Srivastava, J. Koutnik, B.R. Steunebrink, J. Schmidhuber, LSTM: A search space odyssey, IEEE Trans. Neural Netw. Learn. Syst. 28 (2017) 2222–2232, http://dx.doi.org/10.1109/TNNLS.2016.2582924.

[37] J. Bergstra, Y. Bengio, Random search for hyper-parameter optimization, J. Mach. Learn. Res. 13 (2012) 281–305.

[38] Y. Li, Z. Zhu, D. Kong, H. Han, Y. Zhao, EA-LSTM: Evolutionary attention-based LSTM for time series prediction, Knowl.-Based Syst. 181 (2019) 104785, http://dx.doi.org/10.1016/j.knosys.2019.05.028.

[39] Y. Yu, M. Zhang, Control chart recognition based on the parallel model of CNN and LSTM with GA optimization, Expert Syst. Appl. 185 (2021) 115689, http://dx.doi.org/10.1016/j.eswa.2021.115689.

[40] J. Sun, X. Meng, J. Qiao, Prediction of oxygen content using weighted PCA and improved LSTM network in MSWI process, IEEE Trans. Instrum. Meas. 70 (2021) http://dx.doi.org/10.1109/TIM.2021.3058367.

[41] L. Peng, Q. Zhu, S.X. Lv, L. Wang, Effective long short-term memory with fruit fly optimization algorithm for time series forecasting, Soft Comput. 24 (2020) 15059–15079, http://dx.doi.org/10.1007/s00500-020-04855-2.

[42] W. Li, W.W. Wing, T. Wang, M. Pelillo, S. Kwong, HELP: An LSTM-based approach to hyperparameter exploration in neural network learning, Neurocomputing 442 (2021) 161–172, http://dx.doi.org/10.1016/j.neucom.2020.12.133.

[43] F.E. Fernandes, G.G. Yen, Automatic searching and pruning of deep neural networks for medical imaging diagnostic, IEEE Trans. Neural Netw. Learn. Syst. 32 (2020) 5664–5674, http://dx.doi.org/10.1109/TNNLS.2020.3027308.
[44] H. Zhou, Y. Zhang, W. Duan, H. Zhao, Nonlinear systems modelling based on self-organizing fuzzy neural network with hierarchical pruning scheme, Appl. Soft Comput. J. 95 (2020) 106516, http://dx.doi.org/10.1016/j.asoc.2020.106516.
[45] T. Ergen, S.S. Kozat, S. Member, Based on LSTM neural networks, IEEE Trans. Neural Netw. Learn. Syst. Effic. 29 (2017) 1–12.
[46] N.A. Ablitt, J. Gao, J. Keegan, L. Stegger, D.N. Firmin, G.Z. Yang, Predictive cardiac motion modeling and correction with partial least squares regression, IEEE Trans. Med. Imaging 23 (2004) 1315–1324, http://dx.doi.org/10.1109/TMI.2004.834622.
[47] H. Chen, Y. Sun, J. Gao, Y. Hu, B. Yin, via Manifold Optimization Approaches, 30 (2019) 588–600.
[48] X. Yang, X. Liu, C. Xu, Robust mixture probabilistic partial least squares model for soft sensing with multivariate Laplace distribution, IEEE Trans. Instrum. Meas. 70 (2021) http://dx.doi.org/10.1109/TIM.2020.3009354.
[49] M. Hermans, J. Dambre, P. Bienstman, Optoelectronic systems trained with backpropagation through time, IEEE Trans. Neural Netw. Learn. Syst. 26 (2015) 1545–1550, http://dx.doi.org/10.1109/TNNLS.2014.2344002.
[50] H.G. Han, Z.Y. Chen, H.X. Liu, J.F. Qiao, A self-organizing interval Type-2 fuzzy-neural-network for modeling nonlinear systems, Neurocomputing 290 (2018) 196–207, http://dx.doi.org/10.1016/j.neucom.2018.02.049.
[51] A. Safari, R. Hosseini, M. Mazinani, A novel deep interval type-2 fuzzy LSTM (DIT2FLSTM) model applied to COVID-19 pandemic time-series prediction, J. Biomed. Inform. 123 (2021) 103920, http://dx.doi.org/10.1016/j.jbi.2021.103920.
[52] Z. Chen, C. Yang, J. Qiao, The Optimal Design and Application of LSTM Neural Network Based on the Hybrid Coding PSO Algorithm, Springer US, 2022, http://dx.doi.org/10.1007/s11227-021-04142-3.