



Time Series Prediction Based on LSTM and High-Order Fuzzy Cognitive Map with Attention Mechanism

Yingzhi Teng¹ · Jing Liu¹ · Kai Wu²

Accepted: 3 June 2024 / Published online: 28 September 2024
© The Author(s) 2024

Abstract

Fuzzy cognitive map (FCM) has been successfully applied to time series prediction due to its powerful dynamic system modeling and inference ability. Although many FCM-based methods have been proposed, their performance is far from satisfactory. The existing FCM-based methods have two limitations: First, the feature extraction of some methods is unreasonable and even leads to overfitting. Second, most methods ignore the local temporal features of time series. In this work, we propose a novel framework for time series prediction based on long short-term memory (LSTM) and high-order FCMs (HFCM) with an attention mechanism, termed LSTM-HFCM_{AM}. To overcome the first limitation, different from other FCM-based methods that use Autoencoder to forcibly decompose each point of the time series into multiple points to represent the original sequences, we use a sliding window to preprocess the original time series and use the Encoder-Decoder framework to represent the sequences. This way makes the model has better generalization and interpretability. Then, HFCM is used to predict representations of sequences due to its powerful causal inference ability. Finally, self-attention is applied to restore the predicted sliding window data with focus, effectively improving the performance. To overcome the second limitation, we use the LSTM to learn the temporal features of time series fragments, thereby learning the local features of the whole time series. We validate the performance of LSTM-HFCM_{AM} on twelve benchmark datasets. Compared with current methods, LSTM-HFCM_{AM} has a maximum improvement of 51.02%. The experimental results demonstrate the effectiveness of LSTM-HFCM_{AM} and overcome the above limitations.

Keywords Time series prediction · High-order fuzzy cognitive maps (HFCM) · Long short-term memory (LSTM) · Attention mechanism

✉ Jing Liu
neouma@xidian.edu.cn

¹ Guangzhou Institute of Technology, Xidian University, Guangzhou 510555, China

² School of Artificial Intelligence, Xidian University, Xi'an 710126, China

1 Introduction

Fuzzy cognitive maps (FCMs) originated from fuzzy logic theory, neural networks, and soft computing techniques [1]. FCM [2] is a graph structure composed of vertices and directed weighted edges. Vertices represent abstract concepts in the real world, and directed weighted edges represent relationships between concepts. As a powerful tool for uncertainty expression and causal modeling [3–7], FCMs have been widely used in many fields, including time series prediction [8, 9], pattern recognition [10, 11], gene regulatory network reconstruction [12, 13] and medical diagnosis [14, 15]. In time series prediction, FCMs' ability of fuzzy modeling makes its prediction not only at the numerical level but also at the linguistic level [16]. The ability of FCMs to mine the potential causality of data improves the accuracy of prediction. These advantages make FCMs get more and more attention in time series prediction. FCM not only demonstrates good performance in numerical prediction but also contributes to modeling complex systems [12, 13].

In a general sense, a time series is a sequence observed in order from a system, denoted as $X = \{x_1, x_2, \dots, x_t\}$ [14]. Time series prediction is to predict the value at time $t + 1$ in the future based on the original time series X observed in history. The modeling process of the existing prediction methods based on FCM can be divided into three parts: feature extraction, FCM construction, and prediction output. First, the methods transform the original time series into multiple sequences by feature extraction. Then, FCMs utilize the interaction among concepts for inference and prediction. The process of feature extraction plays an important role in the prediction accuracy of FCM-based methods. Fuzzy C-means clustering [17], wavelet transform [18], kernel mapping [19], empirical mode decomposition (EMD) [20], sparse autoencoder (SAE) [21], or other methods have all been applied to extract features from original time series. Although these methods have been successful in time series prediction, there are still some limitations that affect the prediction performance.

Feature extraction in these methods has one or more limitations: First, some existing methods are unreasonable in feature extraction. The unreasonable method makes the acquired features not comprehensive, resulting in low accuracy of actual prediction, and even the risk of falling into over-fitting. For example, SAE has a strong representation ability of time series and performs better than other methods. SAE-HFCM [21] forcibly decomposes each point value of the time series into multiple points and uses each point to learn network structure. Even if a sparse structure is added, it will inevitably fall into overfitting. Meanwhile, the decomposition process of SAE for a single point value is less interpretable (Fig. 1b). Secondly, most of the existing methods cannot effectively use the local temporal features of the original time series. Most of them may only focus on extracting a certain aspect of the whole time series. For example, wavelet transform [18] only focuses on the global frequency and time domain information of the time series, but ignores the trend, state, and especially local feature representation of the time series. In practice, local temporal features are important.

To overcome the limitation of the FCM-based predictor mentioned above, we propose a novel framework, which is based on long short-term memory (LSTM) and high-order FCMs (HFCM) with attention mechanism (AM). The framework is termed LSTM-HFCM_{AM}. Existing methods based on FCM and autoencoder decompose individual time points, as illustrated in Fig. 1b. The rationale behind this approach (SAE-HFCM) is being questioned, as it fails to capture the trends and seasonality in the time series, thus unable to extract relationships between preceding and succeeding time points. Despite exhibiting good fitting performance for individual nodes, predictive performance on the test set is significantly diminished. To alleviate this overfitting phenomenon, as shown in Fig. 1a, we employ a sliding

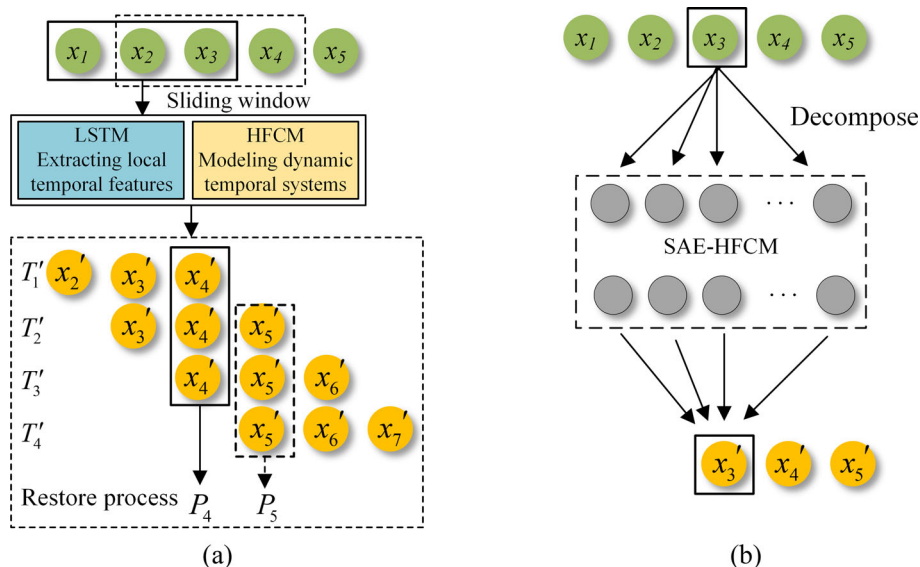


Fig. 1 Comparison of algorithm flowchart. **a** Our proposed method and **b** SAE-HFCM

window to partition the entire time series into contiguous sequence fragments. Compared to decomposing individual time point, the sliding window approach can to some extent preserve the integrity of temporal features. This facilitates subsequent feature extraction and enhances predictive performance on the test set. Subsequently, we utilize LSTM to extract local temporal features from these segments. This way of characteristic representation makes the model preserve local temporal and has better interpretability. Then, HFCM is applied to predict representations of time series due to its ability to mine hidden information and powerful causal inference ability. Since we use the sliding window segmentation sequence as input, the predicted sliding window data T'_i is obtained after the Encoder-Decoder framework. Then we need to restore the sliding window data to the actual predicted value. The restore process is shown in Fig. 1a, P_i is a vector composed of different X_i components, and we need to fit the prediction target value from P_i . We average each value of P_i vector or use Multi-layer Perceptron (MLP) to fit the results, but the performance is still not satisfactory. Since the restore process is sensitive to the contribution of this segment to the target value, and the position of this segment is constantly changing, how to restore the window sequence predicted by HFCM reasonably becomes the key to the problem.

Since the self-attention mechanism pays more attention to the importance degree inside the sequence and ignores the influence of position, we innovatively apply the self-attention mechanism to restore the predicted value of the corresponding position of the sliding window with focus. When the time series is non-stationary, complex, rapid, and changeable, the above mechanism has better generalization and interpretability in the actual prediction, which improves the prediction accuracy and effectively overcomes the first limitation. The Encoder-Decoder framework has been proven to be effective in feature extraction [22] and can extract meaningful representations in the process of decomposition and reconstruction, not limited to a single aspect. We add LSTM into the Encoder-Decoder framework to learn all sequence fragments. By utilizing the structural features of LSTM gates, we can effectively learn the

local temporal features of the whole time series, which overcomes the second limitation mentioned above. The main contributions of this work can be summarized as follows:

1. Innovatively apply the self-attention mechanism to the FCM-based model. The sliding window and self-attention mechanism are used to divide the original time series and restore the predicted results with focus, which effectively preserves local temporal, increases interpretability, and improves the accuracy of the model prediction.
2. We pay attention to the impact of local information of the original time series on the FCM-based prediction model and use the structure of LSTM gates to mine local temporal features of the original time series, thereby further improving the prediction accuracy of the model.
3. We validate the performance of LSTM-HFCM_{AM} with current methods on twelve benchmark datasets. The experimental results prove the effectiveness of LSTM-HFCM_{AM} on large-scale and non-stationary time series, which effectively overcomes the above limitations. Meanwhile, we also verify the influence of LSTM and the self-attention mechanism on the prediction accuracy of the model.

The remainder of this paper is organized as follows. Section 2 presents the related work and the basic knowledge related to LSTM-HFCM_{AM}. Section 3 introduces the details of LSTM-HFCM_{AM}. Section 4 compares LSTM-HFCM_{AM} with state-of-the-art methods and discusses the experimental results. Section 5 concludes the work in this paper.

2 Related Work

In this section, we introduce the basic knowledge of FCM, LSTM, and the self-attention mechanism. The related work about the FCM-based time series prediction methods is given. For convenience, the symbols employed in this paper are described in Table 1.

2.1 Fuzzy Cognitive Maps

An FCM is a graph structure composed of concept nodes and directed weighted edges that represent the relationship between concept nodes [2]. An FCM consists of N_c concept nodes whose state values are denoted as vector C :

$$C = [C_1, C_2, C_3, \dots, C_{N_c}] \quad (1)$$

Each node has a state value of $[0, 1]$ (or $[-1, 1]$). The state value represents the degree of activation of each node. The directed edge between the concept nodes represents the causal relationship from the start node to the target node. The relationship between nodes can be expressed as an $N_c \times N_c$ weight matrix W :

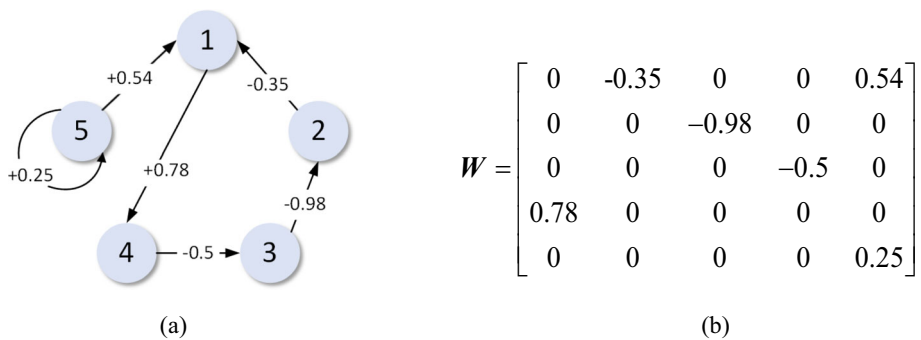
$$W = \begin{bmatrix} w_{11} & w_{12} & \dots & w_{1N_c} \\ w_{21} & w_{22} & \dots & w_{2N_c} \\ \vdots & \vdots & \ddots & \vdots \\ w_{N_c1} & w_{N_c2} & \dots & w_{N_cN_c} \end{bmatrix} \quad (2)$$

where the value range of w_{ij} is $[-1, 1]$, $i, j = 1, 2, 3, \dots, N$. w_{ij} represents the relationship from the j -th node pointing to the i -th node.

Figure 2a shows the topology of an FCM with 5 concept nodes, and the corresponding

Table 1 The symbols used in this paper

Parameters	Definition
X	The training data of the original time series;
N	The length of training data;
L	The length of the sliding window;
T	The data is processed after the sliding window;
D	The number of units in the LSTM layer;
h_i	The hidden states of LSTM at the time step i ;
W_{enc}	All the weights of LSTM in the encoder;
W_{dec}	All the weights of LSTM in the decoder;
d_{enc}	All the bias of LSTM in encoder;
d_{dec}	All the bias of LSTM in the decoder;
M	Features sequences generated from the Encoder-Decoder framework;
W	The weight matrix of HFCM;
N_c	The number of nodes in HFCM($N_c = D$);
k	The order of HFCM;
C	The output of HFCM;
ε	The regularization parameter;
T'	The predicted window sequence;
P	The input vector of the self-attention mechanism;
W_Q	The weights that correspond to the query;
W_K	The weights that correspond to the key;
W_V	The weights that correspond to the value;
A	The attention matrix;
$\hat{X}$	The prediction sequence of the LSTM-HFCM _{AM} framework;

**Fig. 2** An example FCM with 5 concept nodes. **a** Topological structure. **b** weight matrix

weight matrix is shown in Fig. 2b. For example, $w_{15} = 0.54$ denotes that node 5 has a promoting effect on node 1. $w_{45} = 0$ means that there is no interaction between nodes 4 and 5. $w_{55} = 0.25$ indicates node 5 has positive feedback on itself. The state vector $C(t + 1)$ at the $(t + 1)$ -th iteration is determined by the t -th iteration vector $C(t)$ and the weight matrix W of FCM, and its dynamic process can be expressed as:

$$C_i(t + 1) = f \left(\sum_{j=1}^{N_c} w_{ij} C_j(t) \right) \quad (3)$$

where $C_i(t)$ is the activation level of node i at the t -th iteration, and $f(\bullet)$ is a transfer function that is used to bound the activation degree to the range of $[-1, 1]$. The transfer function is defined as follows:

$$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (4)$$

In the traditional FCMs, the state value of each concept node at the $(t + 1)$ -th iteration is only affected by the previous iteration. A k -order HFCM [18] utilizes information from the previous k iteration and can be expressed as:

$$C_i(t + 1) = f \left(\sum_{j=1}^N w_{ij}^1 C_j(t) + w_{ij}^2 C_j(t - 1) + \dots + w_{ij}^k C_j(t - k + 1) + b \right) \quad (5)$$

where w_{ij}^k is the weight between nodes pointing from j -th node to i -th node at time step $t - k + 1$, and b is the constant bias.

2.2 LSTM.

The LSTM network is a special RNN [23]. Different from the common RNN structure, LSTM treats the hidden layer as a memory unit, which can obtain the long-term and short-term features of the sequence. The structure of the memory unit is shown in Fig. 3.

There are three gate structures in memory, which are input gate, the forget gate, and the output gate. The function of an input gate is to filter X_t inputs and bring more important

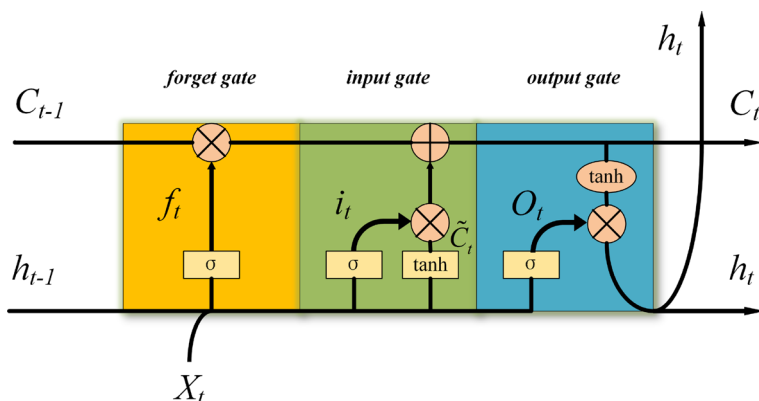


Fig. 3 The memory unit of LSTM

information into the structure. The function of the forgetting gate is to select the output state C_{t-1} of the previous node, which simply means to forget the unimportant information and remember the important ones. The output gate is used to control the output of the current memory unit. Two transfer functions, tanh, and sigmoid, are used in the LSTM memory unit, as shown in Eqs. (4) and (6).

$$\sigma(x) = \text{sigmoid}(x) = \frac{1}{1 + e^{-x}} \quad (6)$$

In Fig. 3, the three gates in the memory unit work as follows. The working process of the forget gate is shown in Eq. (7).

$$f_t = \sigma(W_f[h_{t-1}, X_t] + b_f) \quad (7)$$

where X_t represents the state of the input, and h_{t-1} represents the short-term memory generated by the previous memory unit of LSTM. W_f represents a coefficient matrix of the forget gate. b_f is the constant bias and f_t is the output of the forget gate. The principle of the input gate is expressed as

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \quad (8)$$

$$\tilde{C}_t = \tanh(W_i[h_{t-1}, x_t] + b_i) \quad (9)$$

$$C_t = f_t \times C_{t-1} + i_t \times \tilde{C}_t \quad (10)$$

where $\tilde{C}_t$ is a candidate vector for the update, and C_t is the new state of long-term memory. The output gate finally outputs the new state h_t , and the process is shown in Eqs. (11) and (12).

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \quad (11)$$

$$h_t = o_t \times \tanh(C_t) \quad (12)$$

where O_t is the switch of the output gate in the current state. Through the gate structure of LSTM, we can use the LSTM to obtain the temporal features of the time series.

2.3 Self-Attention

Self-attention mechanism has been successfully applied in natural language processing [24, 25], and has also shown great power in sequential modeling [26]. Self-attention is a special attention mechanism, which can find relationships within sequences without relying on the Encoder-Decoder framework.

The details of self-attention mechanism are shown in Fig. 4, and the calculation process is similar to that in a database query operation, where the query statement is used to find the value that matches the key. Query (Q), key (K), and value (V) can be obtained by Eqs. (13), (14), and (15), where W_Q , W_K , and W_V are matrices that need to be optimized.

$$Q = X_{att} \cdot W_Q \quad (13)$$

$$K = X_{att} \cdot W_K \quad (14)$$

$$V = X_{att} \cdot W_V \quad (15)$$

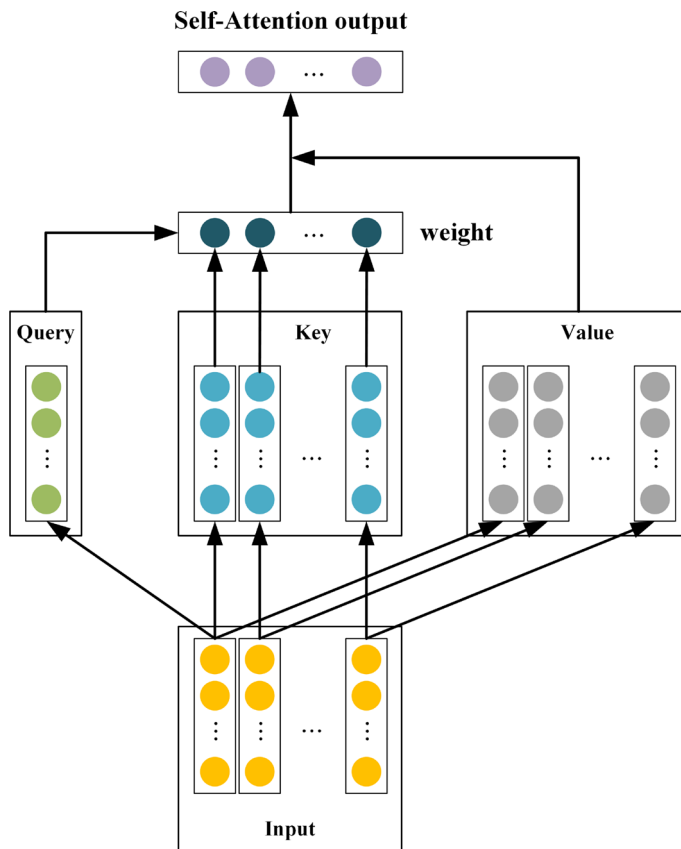


Fig. 4 Schematic diagram of self-attention

The degree of correlation between the vectors in self-attention is achieved through the following steps. Firstly, the similarity between query (Q) and key (K) is calculated by dot multiplication. After scaling the result, the weight of the self-attention mechanism is obtained, and then the final output of the self-attention mechanism is obtained by multiplying the weight by value (V). The whole process is shown in Eq. (16)

$$Attention(Q, K, V) = \text{softmax}\left(\frac{Q \cdot K^T}{\sqrt{d_k}}\right) \cdot V \quad (16)$$

where $\sqrt{d_k}$ is the scaling factor. The essence of self-attention mechanism is to calculate the correlation in a sequence and find out the parts of the sequence that deserve more attention.

2.4 Related Work to Time Series Prediction Based on FCM

In this section, we summarize the existing FCM-based methods for time series prediction and the limitations of these methods. Most FCM-based methods have similar frameworks. Firstly, the univariate time series is transformed into fuzzy time series by feature extraction.

Then, the fuzzy time series is used to model FCMs. Finally, the value at $t + 1$ is predicted by combining the output of FCMs and the output of the predictor.

Stach et al. [16] first proposed a prediction algorithm based on granular FCMs, in which the time series was transformed into granular time series, and then the granular time series was used to learn FCMs. This method can predict at both numerical and linguistic levels. The models Pedrycz et al. in [27, 28] are similar to [16]. This type of model represents the original sequence as a granular time series by using fuzzy clustering method. Then, FCM was developed from granular time series. These methods can capture the essential temporal features and have strong interpretability, but the accuracy of prediction is low. In addition, in these methods, feature extraction and FCM construction are optimized separately. The optimal solution for each part may not be the optimal solution for the whole model.

Other typical FCM-based time series prediction models include: Song et al. [29] proposed a method to combine the FCMs with automatically determined membership function, and used the fuzzy neural network to improve the learning ability of FCMs. Froelich et al. [30] designed a time series method based on evolutionary algorithm and FCM, and applied it to the prostate cancer problem. Salmeron et al. [31] proposed an algorithm of dynamic learning fuzzy cognitive map for time series prediction, which could select concepts before each prediction. Papageorgiou et al. [32] proposed a two-stage prediction method based on ANN and FCMs. These methods have made great progress in time series prediction, but their accuracy is far from satisfactory.

In recent years, with the focus on HFCM, prediction accuracy has been greatly improved. Yang et al. [18] proposed a method combining wavelet transform and HFCM. The wavelet method is used to extract the features of the original sequence, which greatly improves the prediction performance. Subsequently, Yuan et al. [19] proposed a time series prediction algorithm based on kernel mapping and HFCM, and used kernel mapping to select features of time series. Mohammadi et al. [8] used empirical wavelet transform and HFCM for time series prediction. Liu et al. [20] proposed a robust time series prediction algorithm based on EMD and HFCM. Xia et al. [35] utilized variational mode decomposition (VMD) and HFCM to achieve short-term accurate PV generation forecasting. Hu et al. [36] proposed a novel method that integrates VMD, SAE, and FCM and applies it to wind speed forecasting. Wu et al. [21] proposed a prediction algorithm based on SAE and HFCM, using SAE to extract the latent features of time series.

Although the prediction accuracy of methods in [18, 20] has been improved, feature extraction relies on traditional signal processing methods and only focuses on a certain aspect of the time series. The method in [21] used a machine learning algorithm to effectively mine potential features, but due to unreasonable feature decomposition, the method is prone to overfitting. Meanwhile, most of the above methods do not consider the local temporal features of time series. To solve these problems, we propose a framework of LSTM-HFCM_{AM} in this paper.

3 LSTM-HFCMAM

3.1 Overall Process

In this section, we introduce the details of the proposed LSTM-HFCM_{AM}. We divide the framework into three parts: feature extraction by LSTM, modeling and prediction by HFCM, and restoring data by self-attention, as shown in Fig. 5. The original time series can be

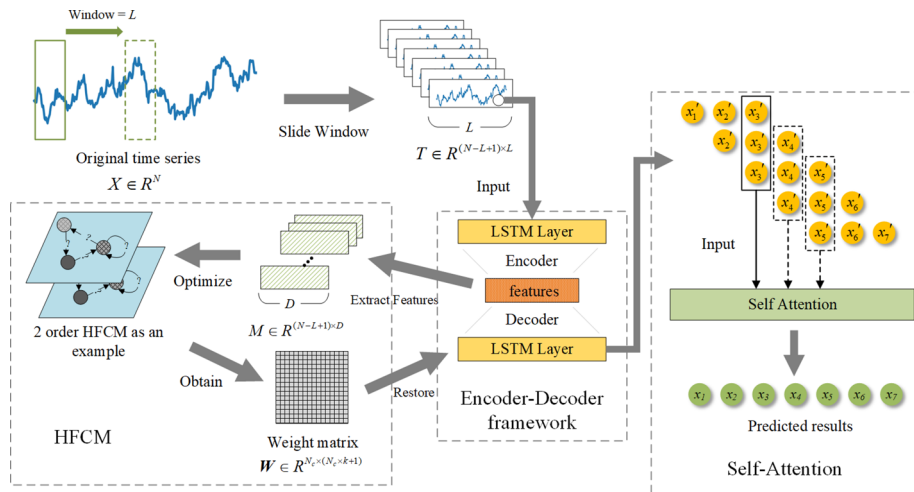


Fig. 5 The process of time series prediction by LSTM-HFCM_{AM}

expressed as $X = \{x_1, x_2, \dots, x_N\}$, where N is the total length of the original time series. After processing in the sliding window, the data can be expressed as $T \in R^{(N-L+1)L}$, where L is the length of the sliding window. First, we take T as the input and the target to train the Encoder-Decoder framework with LSTM, where the number of units in the LSTM layer is D . Through the Encoder-Decoder framework, we can get the features $M \in R^{(N-L+1) \times D}$ from the original time series. Then, we use features M to learn the weights W of HFCM and get the final output C of HFCM. Next, we input C into the decoder to obtain the predicted window sequence T' , which corresponds to T . Finally, we use the self-attention mechanism to restore the window sequences T' with focus to obtain the final predicted value. The details of LSTM-HFCM_{AM} are shown in the following subsections.

3.2 Local Temporal Feature Extraction by Encoder-Decoder Framework with LSTM

In this subsection, we apply Encoder-Decoder framework with LSTM to extract the local temporal features of the original time series X , which can extract the hidden features of the original information in the process of making the output equal to the input. Due to the sensitivity of LSTM gate structure to temporal features, the Encoder-Decoder framework is combined with the LSTM. To avoid model overfitting, we decompose the original sequence X by the sliding window to get $T \in R^{(N-L+1)L}$, and then take T as the input of LSTM cell in the encoder. The output of LSTM in the encoder is

$$h_i^E = \text{LSTM}_{enc}(h_{i-1}^E, T_i; W_{enc}, b_{enc}) \quad (17)$$

To facilitate the description of the model, the principle of LSTM is simplified as $\text{LSTM}(\cdot)$ in Eq. (17), where h_i^E denotes the hidden states at the time step i , $i = 1, 2, \dots, L$; LSTM_{enc} denotes LSTM cells that extract temporal sequence features. W_{enc} and b_{enc} denote the LSTM weight matrix and bias in the encoder. When we give the number of units D in the LSTM, we input a sequence of length L into the encoder, and get h_i^E with dimension D . Therefore, when

we input $N-L+1$ sequences into the encoder, we can obtain a matrix $M \in R^{(N-L+1) \times D}$ composed of feature sequences.

During the decoding process, features M is input into the decoder with LSTM to reconstruct the window sequence T . The output formula of the decoding process is shown as follows:

$$h_i^D = \text{LSTM}_{dec}(h_{i-1}^D, M_i; W_{dec}, b_{dec}) \quad (18)$$

where h_i^D denotes the hidden states of LSTM in the decoder. Corresponding to the structure in the encoder, the LSTM_{dec} reconstructs feature M to obtain $\hat{T}$. To optimize these weights and biases, the mean square error used as the cost function is shown in Eq. (19),

$$L(W_{lstm}, b_{lstm}) = \frac{1}{N-L+1} \sum_{s=1}^{N-L+1} \hat{T}_s - T_s^2 \quad (19)$$

where W_{lstm} and b_{lstm} represent all weights and biases in the Encoder-Decoder framework. We use adaptive moment estimation (Adam) [33] optimization technique to minimize $L(\cdot)$. The output M of the hidden layer is extracted by LSTM to represent the local temporal features of the input data.

3.3 Modeling Features by HFCM

After the Encoder-Decoder framework with LSTM, we can obtain local temporal features M from the original time series. Then, we used the powerful causal inference ability of HFCM as a predictor and used features M to model HFCM. M is composed of D components, each of which corresponds to a concept node in HFCM. The state values of concept nodes update can be denoted by Eq. (5). which is nonlinear and can be obtained by linear transformation as follows:

$$f^{-1}(C_i(t+1)) = \sum_{j=1}^{N_c} (w_{ij}^1 C_j(t) + w_{ij}^2 C_j(t-1) + \dots + w_{ij}^k C_j(t-k+1) + b) \quad (20)$$

where N_c is the concept node number, k is the order of HFCM, and $f^{-1}(\cdot)$ is the inverse function of the transfer function $f(\cdot)$ in (4). Equation (20) can be written as follows:

$$F_i = C W_i \quad (21)$$

where F_i is a vector consisting of $f^{-1}(C_i(t+1))$ at different iteration, C is a matrix composed of the state vectors of all nodes in different iterations, and W_i is the weight vector between the i -th node and all nodes. Take the 2-order HFCM as an example. The variables are expressed as follows:

$$F_i = \begin{bmatrix} f^{-1}(C_i(3)) \\ f^{-1}(C_i(4)) \\ \vdots \\ f^{-1}(C_i(N)) \end{bmatrix} \quad (22)$$

$$C = \begin{bmatrix} C_1(2) & C_1(1) & C_2(2) & C_2(1) & \cdots & C_{N_c}(2) & C_{N_c}(1) & 1 \\ C_1(3) & C_1(2) & C_2(3) & C_2(2) & \cdots & C_{N_c}(3) & C_{N_c}(2) & 1 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ C_1(N-1) & C_1(N-2) & C_2(N-1) & C_2(N-2) & \cdots & C_{N_c}(N-1) & C_{N_c}(N-2) & 1 \end{bmatrix} \quad (23)$$

$$W_i^T = [w_{i1}^1, w_{i1}^2, w_{i2}^1, w_{i2}^2, \dots, w_{iN_c}^1, w_{iN_c}^2, w_{i0}] \quad (24)$$

Yang et al. [18] applied ridge regression to optimize HFCM and achieved good performance while avoiding the problem of model overfitting. Local connections for each node are optimized as follows:

$$\min \left\{ \frac{1}{2N} \|F_i - C W_i\|_2^2 + \varepsilon \|W_i\|_2 \right\} \varepsilon > 0 \quad (25)$$

where ε is the regularization parameter.

Wu et al. [12, 34] proposed that the process of learning FCM matrix could be decomposed into optimizing the weight vectors of each concept point, and finally, all the weight vectors could be combined. The i -th node can get its corresponding weight vector $W_i \in R^{N_c \times k+1}$ by optimizing (20). Repeat this process for all N_c nodes to produce the weight matrix $W \in R^{N_c \times (N_c \times k+1)}$ of HFCM, and we can obtain the output C of HFCM.

Algorithm 1 Training of the LSTM-HFCMAM model

Input:

X : The training data of the original sequence;
 L : The length of the sliding window;
 D : The number of hidden units in the LSTM layer;
 N_c : The number of nodes in HFCM ($N_c = D$);
 k : The order of HFCM;

Output: W_{enc} , W_{dec} , W , W_Q , W_K , W_V , b_{enc} , and b_{dec} .

- 1: The original sequence X is processed by the sliding window to obtain T ;
 - 2: Obtain features M using Encoder-Decoder framework with LSTM with the input data T , as in (17);
 - 3: Learn HFCM with M as input data. Calculate W by ridge regression optimization Eq. (25);
 - 4: The output C of HFCM can be obtained by (5). Then, C is input into the decoder to obtain the window sequence T' ;
 - 5: Using the Adam method to optimize Encoder-Decoder framework with LSTM to get W_{enc} , W_{dec} , b_{enc} , and b_{dec} as in (19);
 - 6: Randomly initialize W_Q , W_K , W_V , and obtain the attention matrix A by (13)-(16);
 - 7: The self-attention model is optimized by (27) to obtain W_Q , W_K , W_V ;
 - 8: **return** W_{enc} , W_{dec} , W , W_Q , W_K , W_V , b_{enc} , and b_{dec} .
-

3.4 Restore the Window Sequence by Self-attention

In the previous step, features M is used to model HFCM, and the output C of HFCM is obtained. In this subsection, we need to restore the predicted state C to achieve the purpose of

the original time series prediction. We input C into the decoder to obtain the predicted window sequence T' . The corresponding relationship between the predicted T' and the original time series X is shown in Fig. 1. Since the restore process is sensitive to the contribution of this segment to the target value, and the position of this segment is constantly changing, we chose the self-attention mechanism to reconstruct the predicted value.

We can divide T' according to the window length into the vector P_i shown in Fig. 1. First, we take P_i as input X_{att} in Eqs. (13), (14), and (15). After randomly initializing W_Q , W_K , and W_V , we can get Q , K , and V . Then, we get the attention matrix A according to Eq. (16). Finally, the output result of the attention mechanism is shown as follows:

$$\hat{X}_i = P_i A \quad (26)$$

Optimize W_Q , W_K , and W_V by minimizing the loss function, which is defined as follows:

$$Loss = \sum_{i=1}^N (X - \hat{X})^2 \quad (27)$$

where $\hat{X}$ is the final output of the LSTM-HFCM_{AM}. The detailed steps of the training phase are shown in Algorithm 1.

4 Experiments and Discussion

In this section, we compare the LSTM-HFCM_{AM} with state-of-the-art methods on twelve benchmark datasets to demonstrate its performance. Then, we validate and analyze the effects of LSTM, the self-attention mechanism, and hyper-parameters on the framework. The source code and dataset are available at https://github.com/IngeTeng/LSTM-HFCM_AM.

4.1 Data Sets and Experiment Setup

We use twelve benchmark datasets with different characteristics to demonstrate the effectiveness of LSTM-HFCM_{AM}. Datasets I–X are the same as those used by these methods [18–21]. We added datasets XI and XII to verify the performance of LSTM-HFCM_{AM} on large-scale datasets. The description of these datasets is shown in Table 2. For ease of comparison, the division of the dataset is consistent with [18]–[21], as shown in Table 3. In this experiment, the MG time series dataset is obtained by solving (28) using the fourth-order Runge–Kutta algorithm.

$$\dot{x}(t) = \frac{0.2 \times (t - \tau)}{1 + x^{10}(t - \tau)} - 0.1 \times (t) \quad (28)$$

where the initial condition $x(0)$ is set to 1.2 and τ is set to 17. The transfer function of HFCM is tanh, which can map the value into the interval of $[-1, 1]$. Thus, we can use the following equation to map the training and test set values to $[-1, 1]$, respectively.

$$\bar{x} = \frac{x - x_{\min}}{x_{\max} - x_{\min}}(x_h - x_l) + x_l \quad (29)$$

where $x_{\max}$ and $x_{\min}$ represent the maximum and minimum values of the original time series, and x_h and x_l represent the maximum and minimum values of the normalized range.

Table 2 The descriptions of twelve benchmarks in this paper

ID	Dataset	Description
I	Sunspot [37]	From 1700 to 1987, the annual number of sunspots, with a total of 288 records;
II	MG [38]	By using the fourth-order Runge–Kutta algorithm to solve formula (28), a total of 1000 points of data from $t = 124$ to $t = 1123$ are obtained
III	Dow-Jones [39]	From Aug. 1968 to Aug. 1981, the monthly closing of the Dow-Jones industrial index, with a total of 291 records;
IV	S&P500 [40]	from Jun. 1, 2016, to Jun. 1, 2017, the daily open price of S&P 500 stock index, with a total of 251 records;
V	Radio [41]	From May. 1934 to Apr. 1954, the highest radio frequencies available for broadcast in Washington, D.C., containing a total of 240 records;
VI	Lake [42]	From 1921 to 1970, monthly Lake Erie level, containing 600 records;
VII	TNX	From Jan. 1962 to Sept. 2019, the monthly Treasury yield 10 year time series, a total of 694 records;
VIII	RUT	From Sept. 2016 to Sept. 2019, the weekly Russell 2000 time series, a total of 753 records;
IX	N225	From Jan. 1965 to Sept. 2019, the monthly Nikkei 225 time series, a total of 657 records;
X	Istanbul stock	Records the daily returns of the Istanbul stock exchange national 100 index from Jun. 5. 2009, to Feb. 22, 2011, totaling 536 records;
XI	ACEG [43]	From Jan. 1 to Jan. 31, 2020, the total actual consumption of electricity from the German grid, a total of 2976 records;
XII	PGEM [44]	From Jan. 1 to Jan. 31, 2020, the day-ahead prices of the German electricity market, a total of 1440 records;

The root means square error (RMSE) is used to evaluate the performance of the prediction model and is defined as follows:

$$RMSE = \sqrt{\frac{1}{N} \sum_{n=1}^N (x_n - \hat{x}_n)^2} \quad (30)$$

where x_n and $\hat{x}_n$ represent the original time series and predicted time series, and N is the length of time series.

In our experiment, we compare LSTM-HFCM_{AM} with the FCM-based method [18]–[21], the classical neural network method [23, 45], and the traditional time series prediction method [47–49]. Hyper-parameters play an important role in classical neural network, and

Table 3 Size division of each dataset

Dataset	Total length	Training	Validation	Test
Sunspot	288	177	44	67
MG	1000	400	100	500
Dow-Jones	291	175	43	73
S&P500	251	120	30	101
Radio	240	184	36	20
Lake	600	368	92	140
TNX	694	388	97	209
RUT	753	422	105	226
N225	657	368	91	198
Istanbul stock	536	343	85	108
ACEG	2976	1667	417	892
PGEM	1440	806	202	432

determining the appropriate hyper-parameters is a challenging task. We apply grid search in LSTM, GRU, and CNN to determine the appropriate hyper-parameters. The mainly considers hyper-parameters include the number of layers, the length of the input, and the number of neurons in the hidden layer. The number of layers is selected from [1, 3]. The length of the input is selected from {5, 10, 20, 40}. The number of neurons is selected from {10, 20, 40, 60}. In the process of training LSTM, GRU, and CNN models, the maximum number of epochs is set to 1000.

4.2 Comparison of LSTM-HFCM_{AM} with Other Methods

In this subsection, we validate the performance of LSTM-HFCM_{AM} on benchmark datasets, and the original and the predicted time series by LSTM-HFCM_{AM} are shown in Fig. 6. The red lines represent the original time series, and the blue ones represent the predicted time series. According to Fig. 6, it can be seen that the predictive ability of LSTM-HFCM_{AM} is excellent. To numerically evaluate the predictive performance of our proposed framework, LSTM-HFCM_{AM} is compared SAE-HFCM [21], EMD-HFCM [20], Wavelet-HFCM [18], LSTM [23], GRU [45], CNN, ANFIS [46], MAR [47], AR [48], and ANN [49] in terms of RMSE. From Table 4, our method shows an improvement of about 50% over state-of-art results in 7 out of 12 cases.

The optimal parameters of LSTM-HFCM_{AM} on each dataset are given in Table 5 and the RMSE of LSTM-HFCM_{AM} on different subsets are given in Table 6. From Table 4, LSTM-HFCM_{AM} outperforms the other methods in 10 out of 12 cases. SAE-HFCM innovatively combines Autoencoder with FCM, and shows very good performance. Compared with SAE-HFCM, our proposed framework outperforms it in 11 out of 12 cases. It can be seen that on large-scale dataset ACEG, LSTM-HFCM_{AM} performs significantly better than other algorithms. Although our proposed framework does not perform as well as SAE-HFCM on the MG time series, the gap between the two is relatively small, with an RMSE difference of only 0.0003. Datasets similar to MG, such as Sunspot, Radio, and Lake, Istanbul stock,

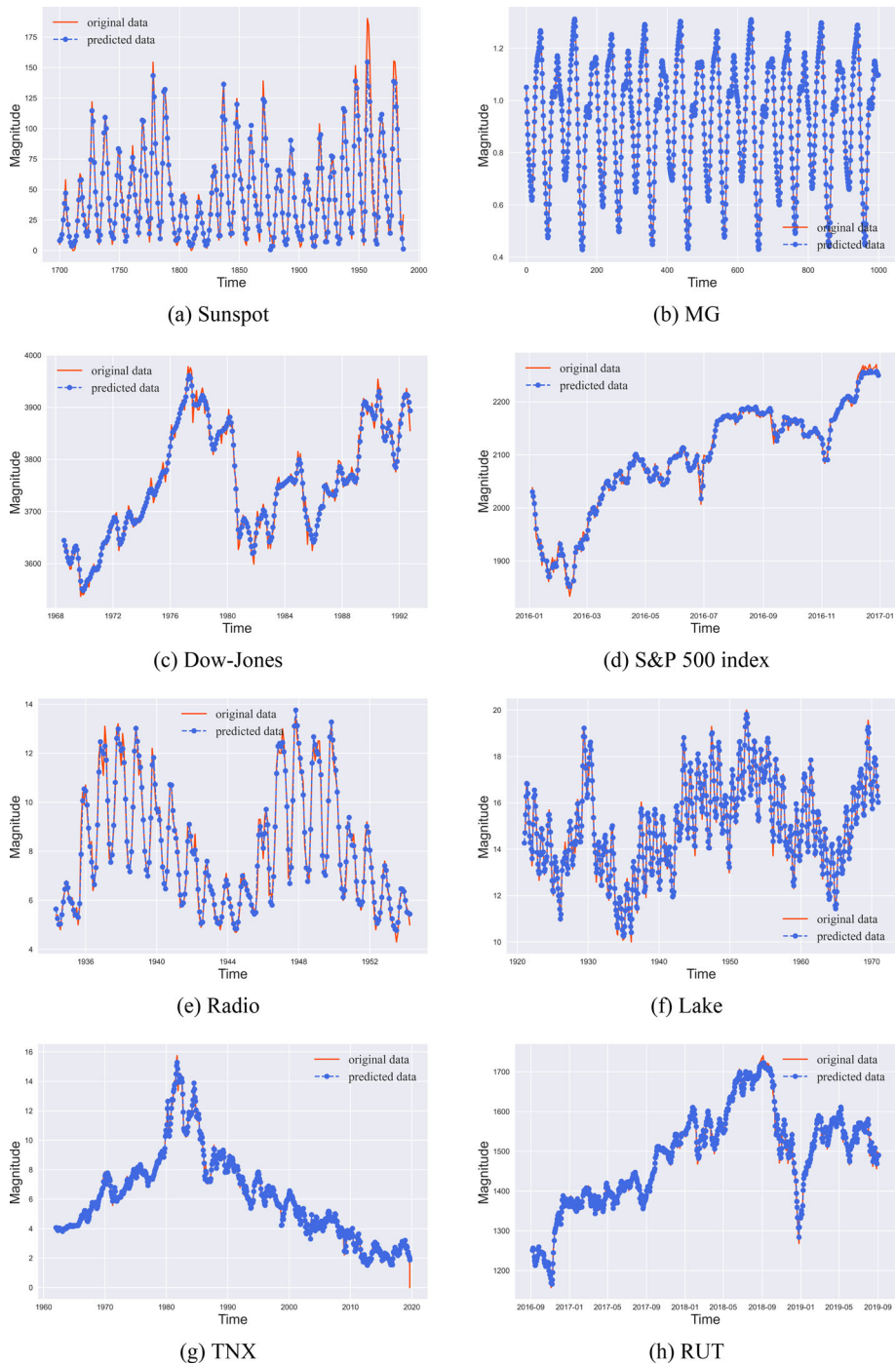


Fig. 6 Original time series and predicted time series by LSTM-HFCM_{AM} of **a** Sunspot, **b** MG, **c** Dow-Jones, **d** S&P 500 index, **e** Radio, **f** Lake, **g** TNX, **h** RUT, **i** N225, **j** Istanbul stock, **k** ACEG, and **l** PGEM

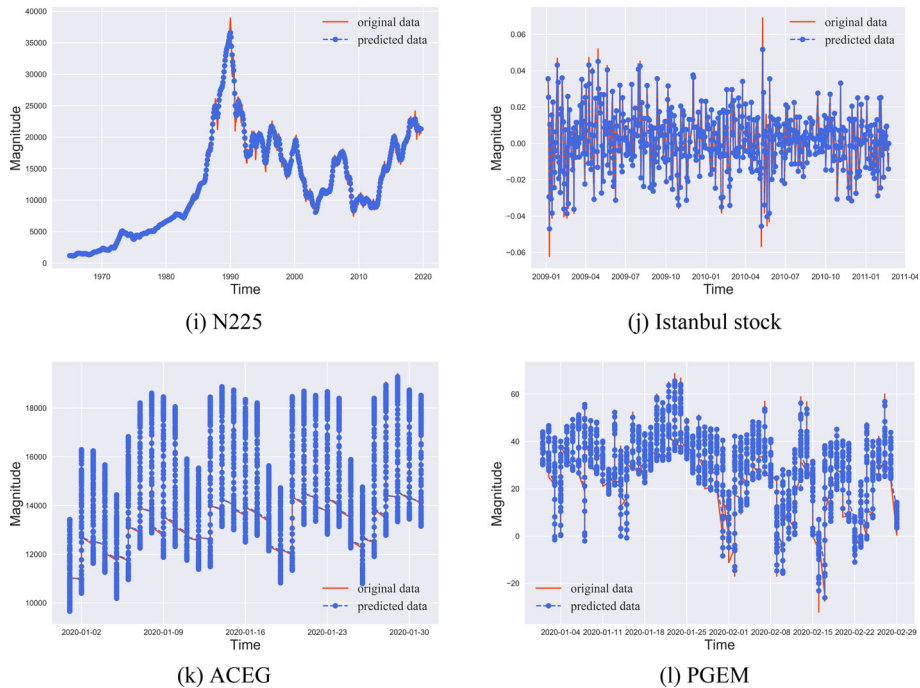


Fig. 6 continued

ACEG, PGEM, all exhibit strong local periodic features. Our method demonstrates significant performance across these datasets. However, the S&P 500 dataset lacks distinct periodic features, resulting in the slightly weaker performance of our method on it.

Compared to classical neural network algorithms (i.e., LSTM, GRU, CNN), our proposed method demonstrates superior performance. This is because LSTM-HFCM_{AM} combines neural networks with FCM, where FCM exhibits strong capabilities in modeling complex systems. LSTM-HFCM_{AM} is not only for time series prediction; it can also model complex systems that generate time series, facilitating predictions of the system's next state. Therefore, our proposed method combines the advantages of neural networks and FCM, and the specific roles of each component can be validated through subsequent ablation experiments.

To verify that our method effectively alleviates overfitting issues, we compared the RMSE of LSTM-HFCM_{AM} and SAE-HFCM on both training and testing sets across two group datasets. In Fig. 7, SAE-HFCM demonstrates testing errors greater than training errors on both group datasets, indicating a clear overfitting problem. Particularly noteworthy is the significant disparity observed in the Lake and Radio datasets, indicating a limitation in the generalization capability of SAE-HFCM. In Fig. 7, in comparison, our proposed method (LSTM-HFCM_{AM}) demonstrates testing errors smaller than training errors on the majority of datasets, with only the RUT and Sunspot datasets being exceptions. However, the performance metrics on these two datasets are significantly superior to those of SAE-HFCM. This demonstrates that our algorithm effectively mitigated overfitting issues.

To validate the effect of LSTM on our proposed framework, AE-HFCM_{AM} is used as a control experiment. To control variables, the sliding window and self-attention mechanism

Table 4 Comparison in terms of RMSE between LSTM-HFCM_{AM} and state-of-the-art methods

Dataset	LSTM - HFCM _{AM}	AE- HFCM _{AM}	SAE - HFCM	EMD - HFCM	Wavelet - HFCM	LSTM	GRU	CNN	ANFIS	MAR	AR	ANN
Sunspot	10.071	16.104	17.390	17.216	18.916	18.695	18.7863	18.171	22.753	19.186	35.262	19.901
MG	0.0012	0.0021	0.0009	0.009	0.004	0.018	0.010	0.003	0.001	0.002	0.035	0.005
Dow-Jones	11.873	15.056	21.335	18.875	23.159	26.922	25.861	26.473	27.526	26.733	29.822	28.520
S&P500	10.642	11.390	11.893	7.139	16.105	16.163	12.213	12.901	14.935	16.041	17.897	17.696
Radio	0.240	0.414	0.490	0.872	0.547	0.744	0.742	0.554	0.651	0.662	0.902	0.652
Lake	0.128	0.183	0.359	0.436	0.377	0.451	0.423	0.450	0.458	0.638	0.390	0.402
TNX	0.166	0.260	0.289	0.477	0.499	0.694	0.347	0.674	1.233	0.276	0.331	0.325
RUT	9.442	14.150	18.266	19.320	21.573	21.980	18.242	21.703	18.626	16.533	22.392	18.711
N225	474.408	630.244	1965.326	1679.977	2876.472	1177.066	1026.276	882.884	769.377	887.546	928.373	928.926
Istanbul stock	0.0009	0.013	0.013	0.011	0.015	0.013	0.013	0.014	0.014	0.015	0.018	0.015
ACEG	46.226	50.685	101.4182	677.143	116.293	107.588	96.741	168.023	95.126	95.762	206.179	197.758
PGEM	2.417	2.888	4.559	2.617	3.668	4.726	4.667	4.986	4.774	4.659	6.081	5.471
Win rate	0.833	0.000	0.083	0.083	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Bold indicates the optimal value

Table 5 Optimal parameters of LSTM-HFCM_{AM} on each dataset

Dataset	L	N_c	k	ε
Sunspot	4	6	5	1e-09
MG	6	6	4	1e-12
Dow-Jones	6	6	4	1e-18
S&P500	2	10	4	1e-05
Radio	4	8	5	1e-05
Lake	4	4	5	1e-16
TNX	8	8	5	1e-07
RUT	4	5	6	1e-13
N225	4	4	12	1e-05
Istanbul stock	2	10	6	1e-05
ACEG	2	30	4	1e-07
PGEM	2	10	4	1e-13

Table 6 RMSE of LSTM-HFCM_{AM} on different subsets

Dataset	Training	Validation	Test
Sunspot	5.813	6.621	10.071
MG	0.0013	0.0014	0.0012
Dow-Jones	13.583	11.645	11.873
S&P500	8.513	8.207	10.642
Radio	0.331	0.418	0.240
Lake	0.172	0.231	0.128
TNX	0.179	0.134	0.166
RUT	6.296	7.477	9.442
N225	548.104	586.868	474.408
Istanbul stock	0.0016	0.0027	0.0009
ACEG	39.412	42.980	46.226
PGEM	1.808	1.850	2.417

in AE-HFCM_{AM} are not changed. From Table 4, the performance of AE-HFCM_{AM} is inferior to LSTM-HFCM_{AM} to varying degrees on all datasets. Therefore, it can be concluded that adding LSTM into the time series feature extraction can effectively improve prediction accuracy. AE-HFCM_{AM} performs better than SAE-HFCM in 10 of 12 cases and performs the same in Istanbul stock data. It can be seen from the result that the sliding window and self-attention mechanism in our proposed framework play an important role in improving the prediction accuracy of the model. In the following part, we discuss the effect of the self-attention mechanism on the framework.

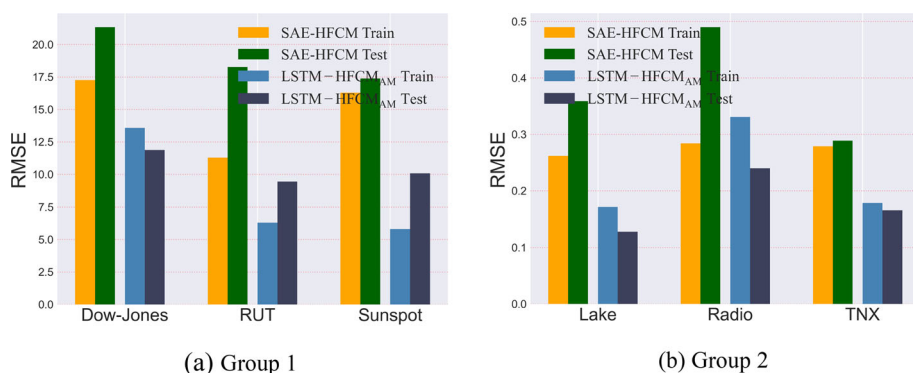


Fig. 7 Comparison of LSTM-HFCM_{AM} and SAE-HFCM on RMSE

4.3 Effects of Self-Attention on LSTM-HFCM_{AM}

In this section, we discuss the effect of the self-attention mechanism on our proposed framework. We used the method of averaging the values in vector P_i in Fig. 1 or the method of MLP fitting the final result to restore the window sequence, but the results were not satisfactory. We divide the numerically similar datasets into two groups to experiment with the self-attention mechanism, MLP, and averaging method. It is worth noting that, in the comparison, we replaced the self-attention mechanism in the LSTM-HFCM_{AM} framework with MLP and averaging methods for the purpose of ablation experiments, to validate the effectiveness of self-attention. The results are shown in Fig. 8, in which the bar chart represents the mean value of 30 experiments on the dataset, and the error bar in red represents the standard deviation. From the error bar in Fig. 8, although the averaging method used in the LSTM-HFCM framework performs better on some datasets, it is sensitive to predicting outliers, which leads to discrete predictions and poor robustness.

The time series predicted by the self-attention mechanism and the MLP are compared on four datasets including Sunspot, Radio, Lake, and Istanbul stock, as shown in Fig. 9. We find that the time series predicted by the MLP differs greatly from the real value in the peak

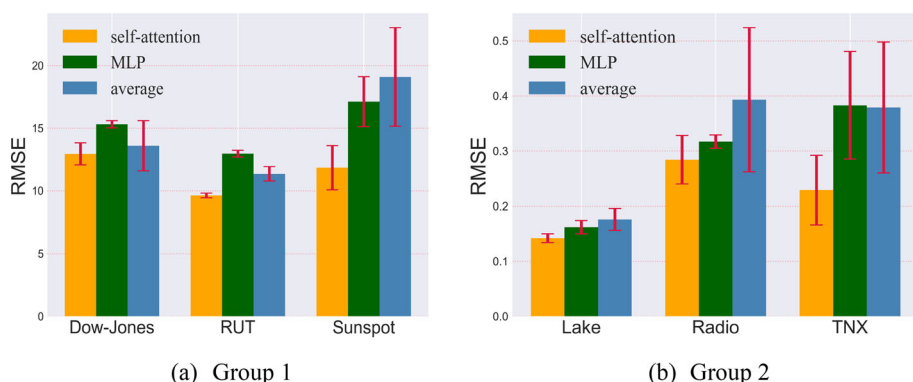


Fig. 8 RMSE versus different data restoration methods throughout the two groups of datasets

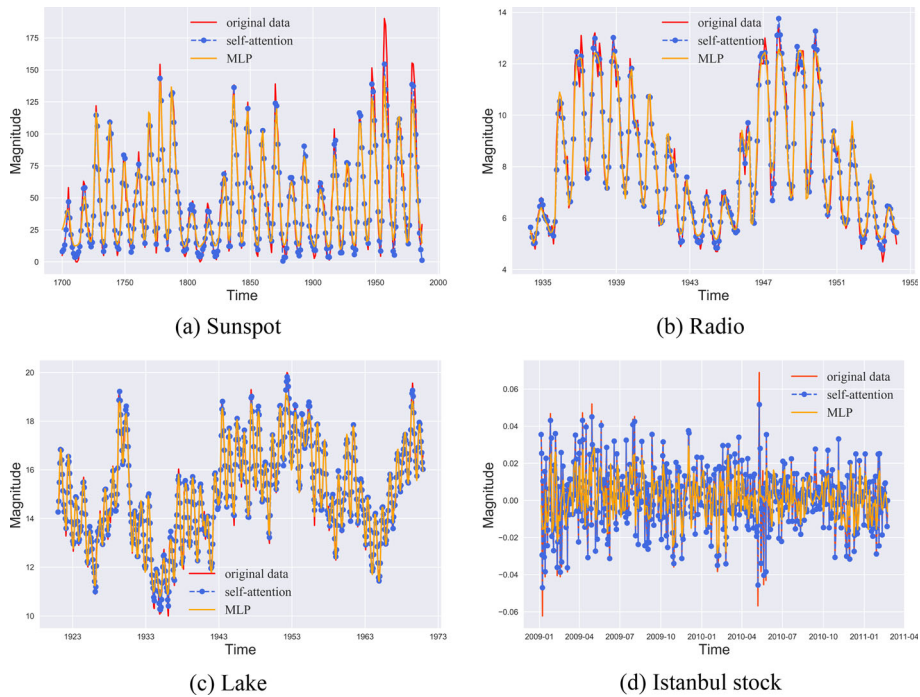


Fig. 9 The original time series, the time series predicted by the self-attention and the time series predicted by the MLP are on four datasets including **a** Sunspot, **b** Radio, **c** Lake, and **d** Istanbul stock

and trough position, which may be caused by the continuous change of the position of the important value in the sliding window, leading to the attenuation of the important value. The MLP does not pay attention to the relationship between the current value and other values in the vector during the process of restoring the window sequence. It can also be seen from Fig. 9 that MLP has high robustness, but the prediction results are not satisfactory. In contrast, the self-attention mechanism is most suitable for the framework we proposed to restore the window sequence, which has low RMSE and acceptable robustness.

4.4 Effects of Hyper-Parameters on LSTM-HFCM_{AM}

In this section, we analyze the effects of three hyper-parameters on our proposed framework. The details are given as follows.

1. Effect of N_c and k on LSTM-HFCM_{AM}: We take Sunspot and RUT time series as examples to study the influence of N_c and k on the prediction accuracy of the proposed framework. Figure 10 shows RMSE versus N_c throughout these two datasets. It can be seen from Fig. 10a – f that the prediction accuracy of LSTM-HFCM_{AM} is highly sensitive to the selection of the number of nodes N_c . For the Sunspot time series, when the value of N_c is 6, the RMSE of prediction is minimized in different order k of HFCM. We can observe from the Sunspot time series that when the order k of HFCM is 4 or 5, the RMSE of the LSTM-HFCM_{AM} is lower. RMSE does not change regularly with the increase of order k . A similar situation can be observed in the RUT time series. The

specific difference is that for the RUT time series with varying orders, the optimal N_c is 5. Therefore, we can see that N_c has a great impact on the performance of LSTM-HFCM_{AM}. Before further detailed parameter optimization, we can set N_c to 5 or 6 as the default value.

2. Effect of L on LSTM-HFCM_{AM}: To keep consistent with the above experiments, the Sunspot and RUT time series are still used for the experiment. Before verifying the effect of L , we select the best parameters in Table 5 to fix the other parameters except for L . It can be seen from Fig. 11 that L has a great effect on the prediction performance of LSTM-HFCM_{AM}. In Fig. 11a, when L is small, even though the RMSE on the training set is the smallest, the performance on the test set is very poor, and the model falls into overfitting, leading to poor prediction accuracy. When L is too small, the prediction accuracy difference between the training set and test set is too large. With the increase of L , the problem of overfitting is alleviated. When L exceeds the optimal value, the overall prediction accuracy of the model will gradually decrease. Therefore, L has an important influence on avoiding model overfitting and improving prediction accuracy.

5 Conclusions

In this work, we propose a time series prediction framework based on LSTM, HFCM, and self-attention mechanism. First, the sliding window and self-attention mechanism are applied to our framework. They can divide the original time series and restore the predicted results with focus, which effectively preserves local timing, increases interpretability, and improves the accuracy of the model prediction. Second, LSTM is added to the framework to learn the local temporal features. We use the representation and reasoning ability of the Encoder-decoder framework and HFCM, combined with the time series mining ability of LSTM, to achieve high-precision prediction.

In this work, the prediction framework is described in detail and its performance is verified by experiments. The experimental results show that LSTM-HFCM_{AM} can effectively improve prediction accuracy and overcome the two limitations of the existing methods. Through experiments, we verify that LSTM and self-attention mechanism can improve the overall prediction accuracy of the framework. We analyze the effect of hyper-parameters on the prediction results and conclude that N_c and L have a great effect on the prediction performance. Although our proposed framework overcomes some limitations of existing methods, it is currently only applied to univariate time series and supports only single-step prediction. Our future work will focus on applying the framework to multivariate time series and enabling multi-step or long-term prediction.

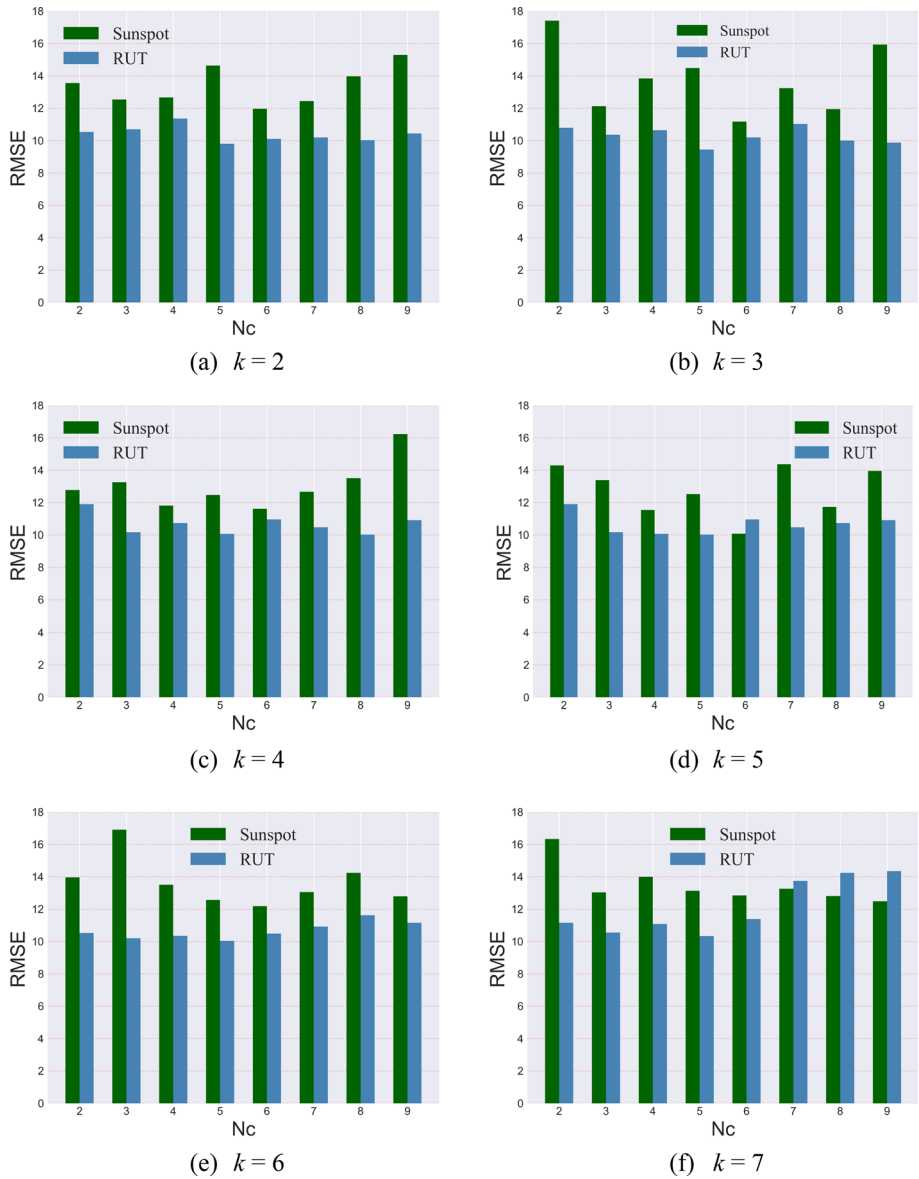


Fig. 10 RMSE versus varying number of nodes N_c of LSTM-HFCM_{AM} with **a** $k = 2$, **b** $k = 3$, **c** $k = 4$, **d** $k = 5$, **e** $k = 6$, and **f** $k = 7$

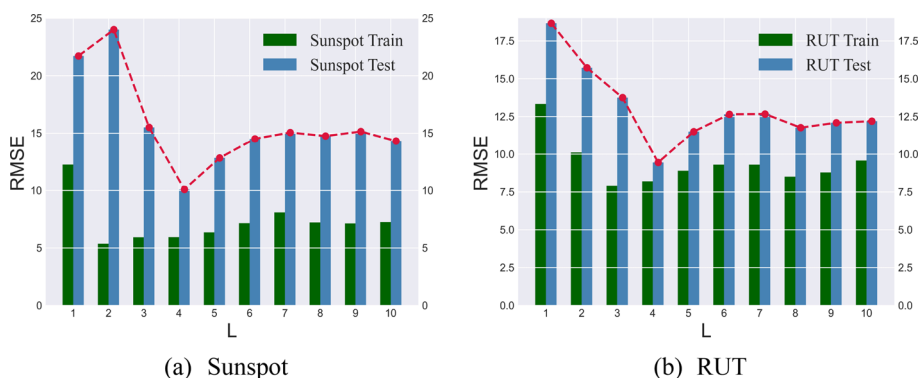


Fig. 11 The RMSE of LSTM-HFCM_{AM} for varying sliding window length L is compared on the two datasets including **a** Sunspot and **b** RUT

Author's Contribution YT: Conceptualization, Methodology, Software, Validation, Formal analysis, Writing—original draft, Writing—review and editing. JL: Writing—review and editing, Supervision, Funding acquisition. KW: Writing—review and editing.

Funding This work was supported in part by the National Natural Science Foundation of China under Grant 62206205, in part by the Guangdong High-level Innovation Research Institution Project under Grant 2021B0909050008, and in part by the Guangzhou Key Research and Development Program under Grant 202206030003.

Data Availability The datasets generated during and/or analyzed during the current study are available from the corresponding author on reasonable request.

Declarations

Conflict of 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.

Ethical Approval and Informed Consent This article does not contain any studies with human participants performed by any of the authors.

Open Access This article is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License, which permits any non-commercial use, sharing, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if you modified the licensed material. You do not have permission under this licence to share adapted material derived from this article or parts of it. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by-nc-nd/4.0/>.

References

1. Papageorgiou EI, Stylios CD (2008) Fuzzy Cognitive Maps. John Wiley & Sons
2. Kosko B (1986) Fuzzy cognitive maps. *Int J Man Mach Stud* 24(1):65–75
3. Hoyos W, Aguilar J, Toro M (2023) PRV-FCM: an extension of fuzzy cognitive maps for prescriptive modeling. *Exp Syst Appl*, pp 120729

4. Nápoles G, Grau I, Concepción L, Koumeri L, Papa J (2022) Modeling implicit bias with fuzzy cognitive maps. *Neurocomputing*
5. Borrero-Domínguez C, Escobar-Rodríguez T (2023) Decision support systems in crowdfunding: a fuzzy cognitive maps (FCM) approach. *Decis Support Syst*, pp 114000
6. Concepción L, Nápoles G, Falcon R, Vanhoof K, Bello R (2021) Unveiling the dynamic behavior of fuzzy cognitive maps. *IEEE Trans Fuzzy Syst* 29(5):1252–1261
7. M. Malakoutikhah, M. Alimohammadlou, M. Jahangiri, H. Rabiei, S. Faghihi, and M. Kamalinia, “Modeling the factors affecting unsafe behaviors using the fuzzy best–worst method and fuzzy cognitive map,” *Applied Soft Computing*, vol. 114, 2022.
8. Mohammadi HA, Ghofrani S, Nikseresh A (2023) Using empirical wavelet transform and high-order fuzzy cognitive maps for time series forecasting. *Appl Soft Comput* 135:109990
9. Yolcu OC, Egrioglu EB, Yolcu U (2022) Multivariate intuitionistic fuzzy inference system for stock market prediction: the cases of Istanbul and Taiwan. *Appl Soft Comput*, 116
10. Homenda W, Jastrzebska A (2019) Time-series classification using fuzzy cognitive maps. *IEEE Trans Fuzzy Syst* 28(7):1383–1394
11. Szwed P (2021) Classification and feature transformation with fuzzy cognitive maps. *Appl Soft Comput*, 105
12. Wu K, Liu J (2017) Learning large-scale fuzzy cognitive maps based on compressed sensing and application in reconstructing gene regulatory networks. *IEEE Trans Fuzzy Syst* 25(6):1546–1560
13. Shen F, Liu J, Wu K (2020) A preference-based evolutionary biobjective approach for learning large-scale fuzzy cognitive maps: an application to gene regulatory network reconstruction. *IEEE Trans Fuzzy Syst* 28(6):1035–1049
14. F. Shen, J. Liu, and K. Wu, “Multivariate time series forecasting based on elastic net and high-order fuzzy cognitive maps: A case study on human action prediction through EEG signals,” *IEEE Transactions on Fuzzy Systems*, vol. 29, no. 8, 2020.
15. Hoyos W, Aguilar J, Toro M (2023) Federated learning approaches for fuzzy cognitive maps to support clinical decision-making in dengue. *Eng Appl Artif Intell* 123:106371
16. Stach W, Kurgan LA, Pedrycz W (2008) Numerical and linguistic prediction of time series with the use of fuzzy cognitive maps. *IEEE Trans Fuzzy Syst* 16(1):61–72
17. Czerwinski D, Kiersztyn A, Łopucki R, Karczmarek P, Kitowski I, Zbyryt A (2020) An application of fuzzy C-means, fuzzy cognitive maps, and fuzzy rules to forecasting first arrival date of avian spring migrants. In: *IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pp 1–6
18. Yang S, Liu J (2018) Time-series forecasting based on high-order fuzzy cognitive maps and wavelet transform. *IEEE Trans Fuzzy Syst* 26(6):3391–3402
19. Yuan K, Liu J (2020) Time series forecasting based on kernel mapping and high-order fuzzy cognitive maps. *Knowl Based Syst*, 206
20. Liu Z, Liu J (2020) A robust time series prediction method based on empirical mode decomposition and high-order fuzzy cognitive maps. *Knowl Based Syst*, 203
21. Wu K, Liu J, Liu P, Yang S (2020) Time series prediction using sparse autoencoder and high-order fuzzy cognitive maps. *IEEE Trans Fuzzy Syst* 28(12):3110–3121
22. Xu M, Han M, Chen CLP, Qiu T (2020) Recurrent broad learning systems for time series prediction. *IEEE Transactions on Cybernetics* 50(4):1405–1417
23. Hochreiter S, Schmidhuber J (1997) Long short-term memory. *Neural Comput* 9(8):1735–1780
24. Shen T, Zhou T, Long G, Jiang J, Pan S, Zhang C (2018) Disan: directional self-attention network for rnn/cnn-free language understanding. In: *Proceedings of the AAAI conference on artificial intelligence*, 32(1)
25. Al-Rfou R, Choe D, Constant N, Guo M, Jones L (2019) Character-level language modeling with deeper self-attention. In: *Proceedings of the AAAI conference on artificial intelligence*, 33(01):3159–3166
26. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez A, Kaiser L, Polosukhin I (2017) Attention is all you need. *Adv Neural Inf Process Syst*
27. Pedrycz W, Jastrzebska A, Homenda W (2016) Design of fuzzy cognitive maps for modeling time series. *IEEE Trans Fuzzy Syst* 24(1):120–130
28. Froelich W, Pedrycz W (2017) Fuzzy cognitive maps in the modeling of granular time series. *Knowl-Based Syst* 115:110–122
29. Song HJ, Miao C, Roel W, Shen ZQ, Catthoor F (2010) Implementation of fuzzy cognitive maps based on fuzzy neural network and application in prediction of time series. *IEEE Trans Fuzzy Syst* 18(2):233–250
30. Froelich W, Papageorgiou EI, Samarinasc M, Skriapasc K (2012) Application of evolutionary fuzzy cognitive maps to the long-term prediction of prostate cancer. *Appl Soft Comput* 12:3810–3817
31. Salmeron JL, Froelich W (2016) Dynamic optimization of fuzzy cognitive maps for time series forecasting. *Knowl-Based Syst* 105:29–37

32. Papageorgiou EI, Pocz K et al (2017) A two-stage model for time series prediction based on fuzzy cognitive maps and neural networks. *Neurocomputing* 232:113–121
33. Kingma DP, Ba J (2014) Adam: a method for stochastic optimization. arXiv preprint: [arXiv:1412.6980](https://arxiv.org/abs/1412.6980)
34. Wu K, Liu J, Liu P, Shen F (2020) Online fuzzy cognitive map learning. *IEEE Trans Fuzzy Syst* 29(7):1885–1898
35. Xia Y, Wang J, Zhang Z, Wei D, Yin L (2023) Short-term PV power forecasting based on time series expansion and high-order fuzzy cognitive maps. *Appl Soft Comput* 135:110037
36. Hu Y, Guo Y, Fu R (2023) A novel wind speed forecasting combined model using variational mode decomposition, sparse auto-encoder and optimized fuzzy cognitive mapping network. *Energy* 278:127926
37. Soltani S (2002) On the use of the wavelet decomposition for time series prediction. *Neurocomputing* 48(1):267–277
38. Mackey MC, Glass L (1977) Oscillation and chaos in physiological control systems. *Science* 197(4300):287–289
39. Monthly closings of the dow-jones industrial index, Aug. 1968-Aug. (1981). Dec. 19, 2017. [Online]. Available: <https://datamarket.com/data/set/22v9/monthly-closings-of-the-dow-jones-industrial-index-aug-1968-aug-1981#!ds=22v9&display=line>
40. “GSPC historical prices | S&P 500 stock, Yahoo!” Jun. 14, 2017. [Online]. Available: <https://finance.yahoo.com/quote/%5EGSPC/history?p=%5EGSPC>.
41. “Monthly critical radio frequencies in Washington, D.C., May 1934-April 1954,” Dec. 19, 2017. [Online]. Available: <https://datamarket.com/data/set/22u2/monthly-critical-radio-frequencies-in-washington-dc-may-1934-april-1954-these-frequencies-reflect-the-highest-radio-frequency-that-can-be-used-for-broadcasting#!ds=22u2&display=line>
42. “Monthly Lake Erie levels 1921–1970,” Dec. 19, 2017. [Online]. Available: <https://datamarket.com/data/set/22pw/monthly-lake-erie-levels-1921-1970#!ds=22pw&display=line>
43. “Actual consumption of electricity from the German grid in January 2021,” Jun. 20, 2021. [Online]. Available: <https://www.smard.de/en/downloadcenter/download-market-data>
44. “The day-ahead prices of the German electricity market in January 2021,” Jun. 20, 2021. [Online]. Available: <https://www.smard.de/en/downloadcenter/download-market-data>
45. Chung J, Gulcehre C, Cho KH, Bengio Y (2014) Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint: [arXiv:1412.3555](https://arxiv.org/abs/1412.3555)
46. Jang J-S (1993) ANFIS: adaptive-network-based fuzzy inference system. *IEEE Trans Syst Man Cybern* 23(3):665–685
47. Renaud O, Starck J-L, Murtagh F (2005) “Wavelet-based combined signal filtering and prediction. *IEEE Trans Syst Man Cybernet Part B (Cybernetics)* 35(6):1241–1251
48. Zheng G, Starck J, Campbell J, Murtagh F (1999) Multiscale transforms for filtering financial data streams. *J Comput Intell Finance* 7:18–35
49. Geva AB (1998) Scalenet-multiscale neural-network architecture for time series prediction. *IEEE Trans Neural Networks* 9(6):1471–1482

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.