Contents lists available at ScienceDirect

Applied Soft Computing

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

Broad fuzzy cognitive map systems for time series classification

Kai Wu [a], Kaixin Yuan [b,*], Yingzhi Teng [b], Jing Liu [b], Licheng Jiao [a]

[a] *School of Artificial Intelligence, Xidian University, Xi'an, China*
[b] *Guangzhou Institute of Technology, Xidian University, Guangzhou, China*

A R T I C L E   I N F O

Article history:
Received 24 March 2022
Received in revised form 14 July 2022
Accepted 28 July 2022
Available online 11 August 2022

Keywords:
Broad learning systems
Fuzzy cognitive maps
Time series classification
Incremental learning
Sparse autoencoder

A B S T R A C T

Time series classification (TSC) is a crucial and challenging problem in sequential analysis. However, most of the existing best-performing methods are time-consuming, even if coping with small-scale datasets. Broad learning systems (BLS) have shown low time complexity and high accuracy in handling various tasks and have been applied to many fields. However, none of the BLS-based methods is suitable to tackle TSC due to (1) unsuitable structure in capturing temporal information of time series; (2) low interpretability in representing the evolving relations among states along time. Thus, this paper develops a broad fuzzy cognitive map system (BFCMS) to address time series classification efficiently, which consists of the sparse autoencoder (SAE) based feature extraction block, the high-order fuzzy cognitive map (HFCM) based spatiotemporal information aggregation block, and one multilayer perceptron (MLP) based prediction layer. The feature extraction block is designed to capture the underlying core evolving patterns, and the spatiotemporal information aggregation block is developed to model the underlying causal relationships and contextual dependencies. These two blocks are designed to overcome the limitations of BLS. MLP is applied to map the feature representation to the label of time series based on the aggregated feature representations from these two blocks. In addition, BFCMS develops three incremental learning strategies for fast updating in broad expansion without a retraining procedure if the model deems to be expanded. We compared BFCMS with other state-of-the-art baselines on 26 datasets. The experimental results demonstrate the superiority of BFCMS. Concretely, BFCMS achieves a lower training cost with on-par classification accuracy.

© 2022 Elsevier B.V. All rights reserved.

1. Introduction

Time series classification (TSC) has been viewed as one of the most challenging problems in sequential analysis [1–7]. Time series are generated and collected from many real-world scenarios [8]. Suppose that the univariate time series $\boldsymbol{x} = \{x_1, x_2, \dots, x_M\}$ consists of a sequence of observations ordered in time. A time series dataset $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_N\}$ includes N different univariate time series with $\boldsymbol{x}_i \in R^{1\times}M$. TSC aims to identify the corresponding category according to the observed data.

In recent years, many TSC approaches have been proposed [8–10]. These methods can be divided into two categories: traditional classifiers and deep learning-based classifiers. Traditional classifiers typically distinguish the label of time series based on the intuitive features. For example, the nearest neighbor classifier discriminates the label of a series as the label of its nearest neighbor [8,11]. Moreover, ensemble methods were proposed to further improve the classification accuracy, which are based on many classification algorithms, e.g., the decision tree, the shapelet classifier [12], or the dynamic time warping (DTW) classifier. Among them, the collective of transformation-based ensembles (COTE) is a representative with well classification performance, which is an ensemble of 35 classifiers [13]. Later, Lines et al. [14] extended COTE with a hierarchical vote system, called HIVE-COTE, which is currently considered to be the state-of-the-art method for TSC in traditional classifiers. However, HIVE-COTE is hugely computationally intensive and impractical to solve large-scale data mining problems [8]. To overcome this challenge, some researchers attempt to develop end-to-end classification methods based on deep neural networks [9], such as the convolutional neural network [15], the echo state network (ESN) [16,17], the fully convolutional neural networks [18], the deep residual network (ResNet) [19], and the recurrent neural networks (RNN) [20]. Although demonstrating more competitive performance than traditional classifiers, deep learning-based methods are typically time-consuming in the training process due to numerous hyperparameters and complex network structures. Besides, if the condition for developing models changes, e.g., the new sample is added, the values of all parameters need to be updated constantly, which is typically expensive. Moreover, deep learning-based methods usually fall short in interpretability.

* Corresponding author.
E-mail addresses: kwu@xidian.edu.cn (K. Wu), maisuiy@outlook.com (K. Yuan).

https://doi.org/10.1016/j.asoc.2022.109458
1568-4946/© 2022 Elsevier B.V. All rights reserved.

These pitfalls imply that there is a genuine demand for designing a fast TSC classifier with high accuracy.

Recently, a broad neural network, the broad learning system (BLS), was proposed by Chen and Liu [21] to provide an effective and efficient learning framework for various fields [22–26]. In contrast to the expensive training cost of deep-learning methods, BLS provides an efficient learning strategy for updating complex systems [27]. When adding new neuron nodes or inputs, the incremental learning of BLS could optimize the parameters without retraining. The prominent advantage of BLS in parameter learning attracts great attention from researchers in various fields. Xu et al. [28] proposed a recurrent BLS for time series prediction, which employed RNN to extract the temporal information of time series. However, this structure has low interpretability in representing the evolving relations of time series. Similarly, Du et al. [20] used RNNs and long short-term memory networks (LSTM) to cope with text classification and establish the recurrent and gated BLS. Han et al. [29] employed manifold learning and structured sparsity to handle non-uniform embedding of large-scale time series. Moreover, since noises commonly exist in dynamical systems, Feng et al. [30] proposed a robust manifold BLS to forecast large-scale noisy, chaotic time series. However, obtaining the evolution information of large-scale chaotic time series is still a problematic issue. Han et al. [31] proposed a maximum information exploitation BLS with the improved leaky integrator dynamical reservoir to cope with this issue. In addition, Liu et al. [32] employed the BLS to train traffic predictors and then demonstrated the effectiveness of BLS. Although several works have been proposed to handle time series prediction or dynamic modeling, none has been applied to the TSC task. Moreover, most existing studies focus on representing and recognizing representative patterns but ignore the changing transitional relations among them. Unlike time series prediction, we need to design a new structure to capture not only the long-term dependencies of time series with high interpretability but also label these series.

In this paper, a **B**road **F**uzzy **C**ognitive **M**ap **S**ystem, caked BFCMS, is proposed to overcome the limitations of BLS mentioned above, which contains three core components: the feature extraction block, the spatiotemporal information aggregation block, and the multilayer perceptron (MLP) layer. The feature extraction block consisting of sparse autoencoders (SAEs) [33,34] is designed to capture the underlying core evolving patterns, which are intended to obtain meaningful junior representations from the input. Inspired by the powerful ability of fuzzy cognitive map (FCM) [35–40] in modeling and flexibility, we propose a high-order FCM (HFCM) based spatiotemporal information aggregation block for capturing the casual relationships and spatiotemporal information among the obtained feature representations Besides, the evolving relations among states along time can be extracted simultaneously. Finally, MLP is applied to fuse both the aggregated temporal features of FCMs and the features automatically extracted by SAEs to predict the labels of time series. Moreover, three incremental learning strategies for fast updating BFCMS and improving their capability are proposed if the model deems to be expanded. The highlights of the proposed BFCMS are summarized as follows:

(1) Pretty good TSC methods are time-consuming in most cases. This paper aims to balance the time complexity and classification accuracy of the time series classifier. Since the BLS is fast in handling many tasks, we propose the BFCMS by extending the BLS to handle TSC and retaining its advantages. We compare BFCMS with the state-of-the-art TSC methods on 26 datasets. The experimental results demonstrate the lower training time of BFCMS.
(2) Applying BLS to handle TSC tasks directly is challenging due to the evolving time series patterns and low interpretability in contextual dependencies. To bridge this gap, we design the feature extraction block and the spatiotemporal information aggregation block to extract the long-term dependencies of time series with high interpretability. The results on 26 datasets indicate that BFCMS can overcome the limitations of the current BLS in terms of TSC tasks, thereby bringing great improvement in classification accuracy compared with BLS.
(3) To retain the advantages of BLS in flexibility, the incremental learning of the inputs and feature mapping nodes in SAEs and HFCMs is designed if the accuracy of BFCMS cannot reach the desired value. The corresponding experimental results demonstrate the effectiveness of the designed incremental learning strategies.

We organize the rest of this paper as follows: Section 2 introduces the background of the functional-link neural networks and FCMs. Section 3 presents the details of the proposed BFCMS. Section 4 presents the results to demonstrate the effectiveness of BFCMS. Section 5 concludes the work in this paper.

2. Background

2.1. Functional-link neural networks

For convenience, Table 1 briefly describes the main symbols in this paper. Chen et al. in [21] proposed a dynamic method to learn the system weights instantly for both a newly added feature vertex and a freshly added enhancement vertex, whose idea is shown in Fig. 1. Denote $\boldsymbol{A}$ be a $R^{N\times M}$ feature matrix and $\boldsymbol{Y} = \boldsymbol{AW}$, where N is the amount of the samples and M is the feature dimension. In the case of adding the new enhancement vector $\boldsymbol{a} \in R^{N\times M}$ shown in Fig. 1, the new feature matrix can be denoted by $\boldsymbol{A}^{+} = [\boldsymbol{A}|\boldsymbol{a}] \in R^{N\times(M+1)}$. The pseudoinverse of $\boldsymbol{A}^{+}$ can be calculated by:

$$\boldsymbol{A}^{+}_{inv} = \begin{bmatrix} \boldsymbol{A}_{inv} - \boldsymbol{db}^{T} \\ \boldsymbol{b}^{T} \end{bmatrix} \tag{1}$$

where "$\bullet_{inv}$" represents the pseudoinverse of "$\bullet$", $\boldsymbol{d} = \boldsymbol{A}_{inv}\boldsymbol{a}$,

$$\boldsymbol{b}^{T} = \begin{cases} \boldsymbol{c}_{inv} & if\ \boldsymbol{c} \neq 0 \\ (1+\boldsymbol{d}^{T}\boldsymbol{d})^{-1}\boldsymbol{d}^{T}\mathbf{A}_{inv} & if\ \boldsymbol{c} \neq 0 \end{cases} \tag{2}$$

and $\boldsymbol{c}$=$\boldsymbol{a}$−$\boldsymbol{Ad}$.

Therefore,

$$\boldsymbol{W}^{+} = \begin{bmatrix} \boldsymbol{W} - \boldsymbol{db}^{T}\boldsymbol{Y} \\ \boldsymbol{b}^{T}\boldsymbol{Y} \end{bmatrix} \tag{3}$$

where $\boldsymbol{W}$ is the system weight before adding the new enhancement vector, and $\boldsymbol{W}^{+}$ is a new weight after the new enhancement vector is added. According to (3), the new weights can be easily obtained without recomputing the pseudoinverse of the new pattern matrix.

2.2. Fuzzy cognitive maps

The FCM [41–43] has been widely applied to time series analysis owing to its recurrent structure [35,37]. It can be regarded as a signed fuzzy network that is structured as c vertices. The state values of these vertices are usually represented as a vector $\boldsymbol{H}$:

$$\boldsymbol{H} = [H_1, H_2, \ldots, H_c] \tag{4}$$

where H_i takes the value in the range [0,1] (or [−1, 1]) and quantifies the activation strength of the corresponding vertex in

Table 1
The main symbols used in this paper.

Symbols	Definition
$\boldsymbol{X}$	The input training dataset;
$\boldsymbol{Y}$	The labels of the training dataset;
N	The total numbers of X;
M, D	The dimensions of each training sample and each label, respectively;
m, n	The number of HFCMs and SAEs in BFCMS, respectively;
$\boldsymbol{x}$	A training sample in X;
$\boldsymbol{z}^i$	The latent encoding representation of $\boldsymbol{x}$ obtained by ith SAE;
$\tilde{\boldsymbol{x}}$	The reconstruction representation of $\boldsymbol{x}$ obtained by ith SAE;
$\boldsymbol{W}^i$, $\hat{\boldsymbol{W}}^i$	The parameters of the ith SAE;
ρ	The penalty coefficient in (11);
$\boldsymbol{z}^i$	The latent encoding representation of uth training sample obtained by ith SAE;
$\boldsymbol{w}$, $\boldsymbol{o}$, $\boldsymbol{u}$	The iterative variables in solving (16) by ADMM;
S	The soft thresholding operator in (17);
$\boldsymbol{Z}^{[n]}$	The union encoding representation of $\boldsymbol{x}$ obtained by all n SAEs;
$\boldsymbol{H}^{[m]}$	The tensor representation of $\boldsymbol{x}$ obtained by all m HFCMs;
$\boldsymbol{H}^j(t)$	The output vector of $\boldsymbol{x}$ obtained by the jth HFCM at iteration t in (19);
$\boldsymbol{W}^{j,1}$, $\boldsymbol{W}^{j,2}$	The parameters of the jth HFCM;
$\boldsymbol{H}^j$	The output of the jth HFCM;
L_j	The order of the jth HFCM;
$\boldsymbol{P}$	The union tensor representation of $\boldsymbol{x}$ obtained by all n SAEs and m HFCMs in (19);
$\boldsymbol{v}$	The resized vector representation of $\boldsymbol{P}$;
W_{out}	The output weight matrix;
$\boldsymbol{V}$	The resized matrix of all samples in (21);
$\boldsymbol{H}^+$	The output of the newly added HFCM unit;
$\boldsymbol{z}^{++}$	The output of the newly added SAE unit;
$\boldsymbol{x}^{+++}$	The newly added training sample;
$\langle\bullet\rangle_u$	The feature "•" corresponding to the uth samples in Algorithm 1;

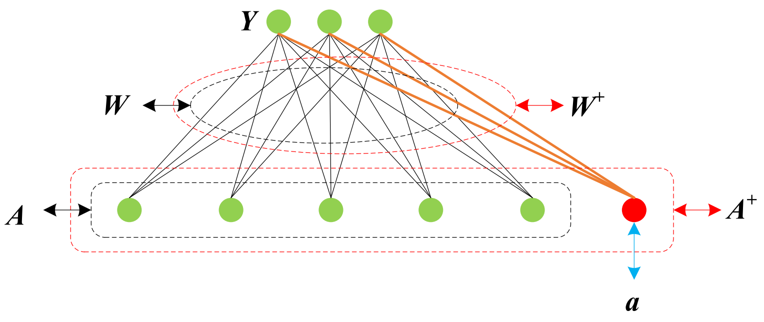


Fig. 1. Illustration of stepwise update algorithm.

the system at a time step. The correlations among vertices are denoted as an $R^{c\times c}$ weight matrix $\boldsymbol{W}$:

$$\boldsymbol{W} = \begin{bmatrix} w_{11} & \cdots & w_{1c} \\ \vdots & \ddots & \vdots \\ w_{c1} & \cdots & w_{cc} \end{bmatrix} \tag{5}$$

where w_{ij} ($w_{ij} \in [-1, 1]$) represents the directed relationship strength from vertex i to vertex j, and $i, j = 1, 2, \ldots, c$. e.g., $w_{ij} > 0$ indicates the promoting effect from vertex i to vertex j; $w_{ij} < 0$ implies the inhibiting influence; $w_{ij} = 0$ represents that vertex i shows no effect to vertex j. Fig. 2 describes a simple 2-order FCM example with four vertices.

Denote the state vector at iteration $t + 1$ as $\boldsymbol{H}(t + 1)$, the regression equation of the L-order HFCM is defined as follows,

$$H_i\,(t+1) = \varphi\left(\sum_{j=1}^{n}\left(w_{ji}^{1}H_j\,(t) + w_{ji}^{2}H_j\,(t-1) + \cdots + w_{ji}^{L}H_j\,(t-L+1) + b\right)\right) \tag{6}$$

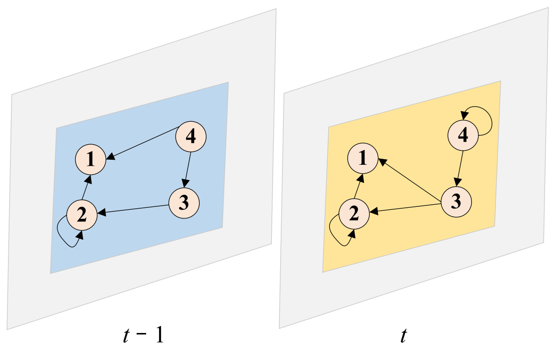


Fig. 2. An 2-order FCM example with four vertices.

where $H_i(t)$ denotes the activation strength of vertex i at iteration t, w_{ij}^{L} represents the edge relationship strength from vertex i to vertex j at time step $t-L+1$, and φ is a transform function bounding the activation degree to the interval [0,1] or [−1, 1]. Due to the advantages of the sigmoid function in [44,45], this

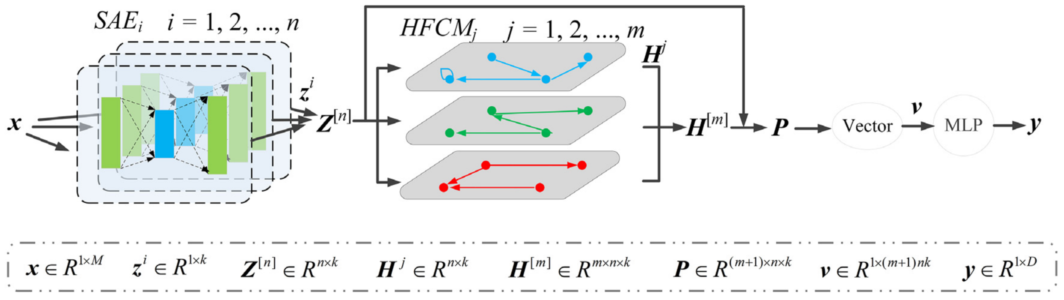

$SAE_i \quad i = 1, 2, ..., n$
$\boldsymbol{x}$
$\boldsymbol{z}^i$
$\boldsymbol{Z}^{[n]}$
$HFCM_j \quad j = 1, 2, ..., m$
$\boldsymbol{H}^j$
$\boldsymbol{H}^{[m]}$
$\boldsymbol{P}$
Vector
$\boldsymbol{v}$
MLP
$\boldsymbol{y}$
$\boldsymbol{x} \in R^{1\times M}$ $\boldsymbol{z}^i \in R^{1\times k}$ $\boldsymbol{Z}^{[n]} \in R^{n\times k}$ $\boldsymbol{H}^j \in R^{n\times k}$ $\boldsymbol{H}^{[m]} \in R^{m\times n\times k}$ $\boldsymbol{P} \in R^{(m+1)\times n\times k}$ $\boldsymbol{v} \in R^{1\times(m+1)nk}$ $\boldsymbol{y} \in R^{1\times D}$

Fig. 3. Structure of the proposed BFCMS.

paper employs the sigmoid function:

$$\varphi(x) = \frac{1}{1+e^{-lx}} \tag{7}$$

where $l = 1$ is a parameter [40].

3. The proposed framework

In this section, we propose a framework called BFCMS for time series classification. Given the training dataset $\{\boldsymbol{X} \in R^{N\times M}, \boldsymbol{Y} \in R^{N\times D}\}$, BFCMS aims to identify the data category efficiently, where $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$ and $\boldsymbol{Y} = \{\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_N\}$, M and D represent the input and output dimensions, respectively, and N denotes the number of training samples. Fig. 3 depicts the structure of the proposed BFCMS, which mainly consists of one feature extraction block, one spatiotemporal information aggregation block, and one MLP layer. The feature extraction block contains n SAE layers, which are dedicated to capturing multiple hidden feature sequences from input $\boldsymbol{x}$. The spatiotemporal information aggregation block contains m HFCM layers. They are designed to model the underlying dynamics of these sequence features, e.g., causal relationships or contextual dependencies. Based on the constructed feature representation from the first two blocks, the MLP layer is applied to predict the corresponding label $\boldsymbol{y}$. The details about BFCMS are illustrated in the following sections.

3.1. Feature extraction from time series

As an unsupervised learning algorithm, an autoencoder neural network has been applied to mining out the intrinsic correlations of the input variables. Inspired by the sparse coding, we employ the SAE to extract the feature sequence from the raw time series. The structure of an SAE is shown in Fig. 4, which contains the encoding and decoding layers. For ease of understanding, one input datum $\boldsymbol{x} \in R^{1\times M}$ and the ith SAE module are used to explain the pipeline of the feature extraction block. Specifically, the encoding layer of the ith SAE maps $\boldsymbol{x}$ to a hidden representation through a deterministic mapping, and the output of this hidden layer is

$$\boldsymbol{z}^i = f_{i,e}(\boldsymbol{x}) = \varphi\left(\boldsymbol{x}\boldsymbol{W}^i\right) \tag{8}$$

where $\boldsymbol{W}^i \in R^{M\times k}$ is the encoding parameter and $\varphi(\cdot)$ represents the sigmoid function. The resulting hidden representation $\boldsymbol{z}^i$ is then mapped back to a reconstructed vector by the same sigmoid function $\varphi(\cdot)$, whose mathematical form is as follows:

$$\widetilde{\boldsymbol{x}} = f_{i,d}\left(\boldsymbol{z}^i\right) = \varphi\left(\boldsymbol{z}^i\hat{\boldsymbol{W}}^i\right) \tag{9}$$

where $\hat{\boldsymbol{W}}^i \in R^{k\times M}$ is the decoding parameter. To optimize these weights, the objective function is designed based on the mean-square error between the input series and the reconstructed series for all N samples as follows:

$$J_E = \sum_{u=1}^{N} \|\widetilde{\mathbf{x}}_u - \mathbf{x}_u\|_2^2 + \rho(\left\|\boldsymbol{W}^i\right\|_1 + \left\|\hat{\boldsymbol{W}}^i\right\|_1) \tag{10}$$

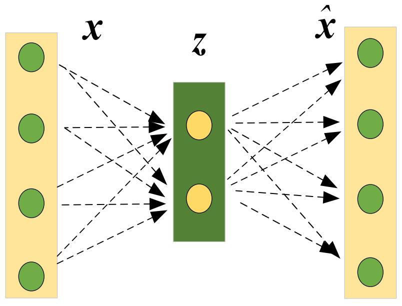


Fig. 4. Architecture of a SAE neural network.

To speed up the convergence of the learning algorithms, Chen et al. [21] randomly initialized the weight matrix $\boldsymbol{W}^i$, and then

$$\underset{\hat{\boldsymbol{w}}^i}{\arg\min}: \sum_{u=1}^{N} \left\|\varphi_i\left(\boldsymbol{z}_u^i\hat{\boldsymbol{W}}^i\right) - \mathbf{x}_u\right\|_2^2 + \rho\left\|\hat{\boldsymbol{W}}^i\right\|_1 \tag{11}$$

where $\hat{\boldsymbol{W}}^i$ is the learned weight of the decoding layer and $\boldsymbol{z}_u^i$ is the desired coding feature of the uth sample obtained by the ith SAE. For convenience, (11) is converted equivalently to (12) as follows:

$$\underset{\hat{W}^i}{\arg\min}: \sum_{u=1}^{N} \left\|\boldsymbol{z}_u^i\hat{\boldsymbol{W}}^i - \varphi^{-1}(\mathbf{x}_u)\right\|_2^2 + \rho\left\|\hat{\boldsymbol{W}}^i\right\|_1 \tag{12}$$

where $\varphi^{-1}(\cdot)$ is the inverse function of $\varphi(\cdot)$. Also, (12) can be expressed as the matrix multiplication form as follows:

$$\underset{\hat{\boldsymbol{w}}^i}{\arg\min}: \left\|\boldsymbol{Z}^i\hat{\boldsymbol{W}}^i - \varphi^i(\mathbf{X})\right\|_2^2 + \rho\left\|\hat{\boldsymbol{W}}^i\right\|_1 \tag{13}$$

where $\boldsymbol{z}^i = \left[\boldsymbol{z}_1^i, \boldsymbol{z}_2^i, \ldots, \boldsymbol{z}_N^i\right]^{\mathrm{T}} \in R^{N\times k}$ is the representation of all N samples obtained by the ith SAE.

Then, (13) can be equivalently considered as a general optimization problem expressed as follows:

$$\underset{\boldsymbol{w}}{\arg\min}: f(\boldsymbol{w}) + g(\boldsymbol{w}), \quad \boldsymbol{w} \in R^k \tag{14}$$

where $f(\boldsymbol{w}) = \left\|\boldsymbol{Z}^i\boldsymbol{w} - \varphi^{-1}(\boldsymbol{x})\right\|_2^2$ and $g(\boldsymbol{w}) = \rho\|\boldsymbol{w}\|_1$.

The above problem is a convex optimization problem and can be solved by many solvers. This paper uses the alternating direction method of multipliers (ADMM) [46] to learn the solution of the sparse autoencoder and fine-tune the parameters due to its great success in optimizing the l_1 norm involved problems. In ADMM form, the above formula can be reorganized as

$$\underset{\boldsymbol{w}}{\arg\min}: f(\boldsymbol{w}) + g(\boldsymbol{o}), \quad s.t.\ \boldsymbol{w} - \boldsymbol{o} = 0 \tag{15}$$

Therefore, we can attain the proximal solution via the following updating steps:

$$\begin{cases} \boldsymbol{w}_{k+1} := ((\boldsymbol{Z}^i)^T\boldsymbol{Z}^i + \rho\boldsymbol{I})^{-1}((\boldsymbol{Z}^i)^T\varphi^{-1}(\mathbf{x}) + \rho(\mathbf{o}^k - \mathbf{u}^k)) \\ \mathbf{o}_{k+1} := S_{\frac{\lambda}{\rho}}(\boldsymbol{w}_{k+1} + \mathbf{u}_k) \\ \mathbf{u}_{k+1} := \mathbf{u}_k + (\boldsymbol{w}_{k+1} - \mathbf{o}_{k+1}) \end{cases} \tag{16}$$

where $\rho > 0$ and S denotes the soft thresholding operator. Its details are shown as follows:

$$S_\kappa(a) = \begin{cases} a - \kappa, & a > \kappa \\ 0, & |a| \leq \kappa \\ a + \kappa, & a < -\kappa \end{cases} \tag{17}$$

Next, the linear inverse problem in (16) is applied to fine-tune the initial $\boldsymbol{W}^i$ to obtain better features [21]. The encoding representations of the hidden layer $\boldsymbol{z}^i$ ($i = 1, 2, \ldots, n$) are then concatenated to form $\boldsymbol{Z}^{[n]} = [\boldsymbol{z}^1, \boldsymbol{z}^2, \ldots, \boldsymbol{z}^n]^T \in R^{n\times k}$, which is the junior feature representation of $\boldsymbol{x}$ from the feature extraction block.

3.2. Representing evolving relationships

After capturing the junior feature sequence representation $\boldsymbol{Z}^{[n]} = [\boldsymbol{z}^1, \boldsymbol{z}^2, \ldots, \boldsymbol{z}^n]^T \in R^{n\times k}$ via the feature extraction block, the spatiotemporal information aggregation block consisting of multiple HFCMs with different orders is designed to deal with the various temporal dependencies to construct the new senior feature representation $\boldsymbol{H}^{[m]}$. The jth HFCM with order L_j is used to model the feature sequence representation $\boldsymbol{Z}^{[n]}$, where each feature series $\boldsymbol{z}^i$ is considered an observation generated by a vertex in the jth HFCM. Therefore, the number of vertices in an HFCM is n. Based on (6) described in Section 2, the state values of vertices in the jth HFCM ($j = 1, 2, \ldots, m$) at iteration t can be derived on the basis of the following equation:

$$\boldsymbol{H}^j(t) = \varphi\left(\boldsymbol{W}^{j,1}\boldsymbol{z}^{[n]}(t) + \boldsymbol{W}^{j,2}\left[\boldsymbol{z}^{[n]}(t-1)\,|\cdots|\boldsymbol{z}^{[n]}\left(t - L_j + 1\right)\right]\right) \tag{18}$$

where $[\cdot|\cdot]$ denotes a vertical vector concatenation operation, $\boldsymbol{H}^j(t) \in R^{n\times 1}$ is the output vector obtained by the jth HFCM at iteration t, $\boldsymbol{z}^{[n]}(t) \in R^{n\times 1}$ is the observation vector at iteration t, $t = 1, 2, \ldots, k$. $\boldsymbol{W}^{j,1} \in R^{n\times n}$, $\boldsymbol{W}^{j,2} \in R^{n\times(L_j-1)n}$ are two parameter matrices in HFCM. These weights matrices are randomly assigned before optimization. Define $\boldsymbol{H}^j = [\boldsymbol{H}^j(1), \boldsymbol{H}^j(2), \ldots, \boldsymbol{H}^j(k)] \in R^{n\times k}$, the joint senior feature representation $\boldsymbol{H}^{[m]}$ outputted from m HFCMs can be obtained as $\boldsymbol{H}^{[m]} = [\boldsymbol{H}^1, \boldsymbol{H}^2, \ldots, \boldsymbol{H}^m] \in R^{m\times n\times k}$. The HFCM with larger order L_j can capture the long-term causal dependency of the feature time series; When L_j is set to a small value, the HFCM is capable of modeling the short-term causal dependence. Therefore, the mixture of multiple HFCMs with various orders enables BFCMS to extract the underlying complicated temporal dynamics of the raw series.

After obtaining $\boldsymbol{Z}^{[n]}$ and $\boldsymbol{H}^{[m]}$, we concatenate them to construct the union tensor representation $\boldsymbol{P} \in R^{(m+1)\times n\times k}$ as follows:

$$\boldsymbol{P} = [\boldsymbol{Z}^{[n]}, \boldsymbol{H}^{[m]}] \tag{19}$$

and then, $\boldsymbol{P}$ is resized as a feature vector by the vector operation, which is the input of MLP for predicting the label $\boldsymbol{y}$. The mathematical formulation of MLP is shown as follows:

$$\boldsymbol{y} = \boldsymbol{v}\boldsymbol{W}_{out} \tag{20}$$

where $\boldsymbol{v} = \boldsymbol{Vector}(\boldsymbol{P}) \in R^{1\times(m+1)nk}$ is the resized feature vector representation and $\boldsymbol{Vector}(\cdot)$ is the vector operation.

For all N train samples,

$$\boldsymbol{Y} = \boldsymbol{V} \times \boldsymbol{W}_{out} \tag{21}$$

where $\boldsymbol{V} = [\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_N]^T \in R^{N\times(m+1)nk}$ is the input matrix of the MLP layer and $\boldsymbol{W}_{out} \in R^{(m+1)nk\times D}$ is the weight matrix.

Note that (21) is a linear equation. Thus, to avoid overfitting, $\boldsymbol{W}_{out}$ can be calculated by the ridge regression

$$\boldsymbol{W}_{out} = \left[\boldsymbol{V}^{\mathbf{T}}\boldsymbol{V} + \lambda\boldsymbol{I}\right]^{-1}\boldsymbol{V}^T\boldsymbol{Y} \tag{22}$$

where λ is a positive constant, and $\boldsymbol{I}$ is an identity matrix.

3.3. Increment of the HFCM units

Inspired by the incremental learning in BLS, we propose a new learning strategy for BFCMS by adding the HFCM to improve the performance of the algorithm, which optimizes the total structure by adding new units (e.g., HFCMs) instead of adjusting the parameters of the designed architecture by gradient descent. The increment learning strategy of additional HFCMs (IL-HFCM) is illustrated in Fig. 5. Compared with the deep learning methods, our newly designed learning strategy can avoid the gradient vanishing problem and gradient exploding problem. Specifically, if the classification accuracy of BFCMS cannot reach the desired value, the new HFCM can be added to optimize the structure, thereby improving the performance of the algorithm. The details of IL-HFCM are illustrated as follows:

Denote the output of the added HFCM unit is $\boldsymbol{H}^+$, and then the new union tensor representation can be represented as

$$\boldsymbol{P}^+ = [\boldsymbol{Z}^{[n]}, \boldsymbol{H}^{[m]}, \boldsymbol{H}^+] = [\boldsymbol{P}, \boldsymbol{H}^+] \tag{23}$$

Similar to (19) and (20), the mathematical formulation of the MLP layer is updated by

$$\boldsymbol{y} = \boldsymbol{v}^+\boldsymbol{W}^+_{out} \tag{24}$$

$$\begin{aligned} \boldsymbol{v}^+ &= \boldsymbol{Vector}(\boldsymbol{P}^+) = \boldsymbol{Vector}([\boldsymbol{P}, \boldsymbol{H}^+]) \\ &= [\boldsymbol{Vector}(\boldsymbol{P}), \boldsymbol{Vector}(\boldsymbol{H}^+)] = [\boldsymbol{v}, \boldsymbol{Vector}(\boldsymbol{H}^+)] \end{aligned} \tag{25}$$

where $\boldsymbol{Vector}(\boldsymbol{H}^+) \in R^{nk\times 1}$.

Therefore, the new input of the output layer $\boldsymbol{V}^+ \in R^{N\times(m+2)nk}$ can be derived by

$$\boldsymbol{V}^+ = \left[\boldsymbol{V}, \boldsymbol{H}^a\right] \tag{26}$$

where $\boldsymbol{H}^a \in R^{N\times nk}$ the new incremental feature representation and

$$\boldsymbol{H}^a = \left[[\boldsymbol{Vector}(\boldsymbol{H}^+)]_1, [\boldsymbol{Vector}(\boldsymbol{H}^+)]_2, \ldots, [\boldsymbol{Vector}(\boldsymbol{H}^+)]_N\right]^T \tag{27}$$

As discussed in Section 2, the pseudoinverse of $\boldsymbol{V}^+$ can be deduced as

$$\boldsymbol{V}^+_{inv} = \begin{bmatrix} \boldsymbol{V}_{inv} - \boldsymbol{D}\boldsymbol{B}^T \\ \boldsymbol{B}^T \end{bmatrix} \tag{28}$$

where $\boldsymbol{V}^+_{inv}$ is the pseudoinverse of $\boldsymbol{V}^+$ and

$$\boldsymbol{D} = \boldsymbol{V}_{inv} \times \boldsymbol{H}^a \tag{29}$$

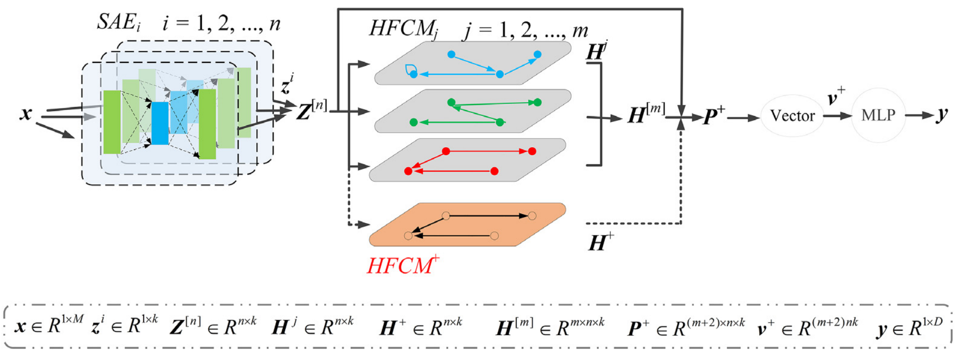
$$\boldsymbol{B}^T = \begin{cases} \boldsymbol{C}_{inv} & if\ \boldsymbol{C} \neq \mathbf{0} \\ (\mathbf{1} + \boldsymbol{D}^{\mathbf{T}}\boldsymbol{D})^{-1}\boldsymbol{D}^T\boldsymbol{V}_{inv} & if\ \boldsymbol{C} = \mathbf{0} \end{cases} \tag{30}$$

and $\boldsymbol{C} = \boldsymbol{H}^a - \boldsymbol{V} \times \boldsymbol{D}$.

Again, the new output weights $\boldsymbol{W}^+_{out}$ can be deduced by

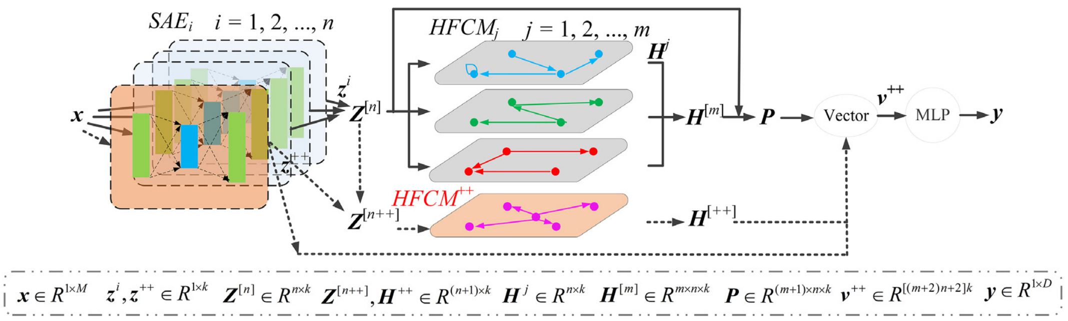
$$\boldsymbol{W}^+_{out} = \begin{bmatrix} \boldsymbol{W}_{out} - \boldsymbol{D}\boldsymbol{B}^T\boldsymbol{Y} \\ \boldsymbol{B}^T\boldsymbol{Y} \end{bmatrix} \tag{31}$$

That is, the new weights of BFCMS can be obtained based on the one-step update in (31), instead of multi-step gradient descent layer-by-layer.



$SAE_i \quad i = 1, 2, ..., n$
$HFCM_j \quad j = 1, 2, ..., m$
x
z^i
$Z^{[n]}$
H^j
$H^{[m]}$
P^+
Vector
v^+
MLP
y
H^+
$HFCM^+$
$x \in R^{1\times M}$ $z^i \in R^{1\times k}$ $Z^{[n]} \in R^{n\times k}$ $H^j \in R^{n\times k}$ $H^+ \in R^{n\times k}$ $H^{[m]} \in R^{m\times n\times k}$ $P^+ \in R^{(m+2)\times n\times k}$ $v^+ \in R^{(m+2)nk}$ $y \in R^{1\times D}$

Fig. 5. Broad learning strategy: increment of the additional HFCM.



$SAE_i \quad i = 1, 2, ..., n$
$HFCM_j \quad j = 1, 2, ..., m$
x
z^i
$Z^{[n]}$
H^j
$H^{[m]}$
P
Vector
v^{++}
MLP
y
z^{++}
$HFCM^{++}$
$Z^{[n++]}$
$H^{[++]}$
$x \in R^{1\times M}$ $z^i, z^{++} \in R^{1\times k}$ $Z^{[n]} \in R^{n\times k}$ $Z^{[n++]}, H^{++} \in R^{(n+1)\times k}$ $H^j \in R^{n\times k}$ $H^{[m]} \in R^{m\times n\times k}$ $P \in R^{(m+1)\times n\times k}$ $v^{++} \in R^{[(m+2)n+2]k}$ $y \in R^{1\times D}$

Fig. 6. Broad learning strategy: increment of Additional SAEs.

3.4. Increment of the SAE units

Although the BFCMS can be expanded with IL-HFCM, the performance may still be not wonderful enough due to the influence of insufficient feature mapping nodes. To overcome this limitation, new nodes and the corresponding feature series can be added to enhance the capacity of the feature extraction block. However, the increment of new SAE feature extractors brings difficulty in optimizing the parameters of BFCMS without retraining the whole network. To address it, we proposed the increment learning strategy of additional SAEs (IL-SAE), whose pipeline can be seen in Fig. 6. The details about IL-SAE are introduced in the following paragraphs.

Consider that a new SAE unit is added, whose feature representation is denoted as $\boldsymbol{z}^{++}$, and then

$$\boldsymbol{Z}^{[n++]} = [\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_n, \boldsymbol{z}^{++}]^T = [\boldsymbol{Z}^{[n]}, \boldsymbol{z}^{++}] \tag{32}$$

where $\boldsymbol{Z}^{[n++]} \in R^{(n+1)\times k}$ is the new feature representation after adding a new SAE unit. To capture the effect of $\boldsymbol{z}^{++}$, we add a new HFCM unit, i.e., HFCM^{++}, where the total number of vertices is one more than that of the initialized HFCM, e.g., HFCM$_1$ in Fig. 6. The regression formulation of HFCM^{++} is shown as follows,

$$\boldsymbol{H}^{++}(t) = \varphi\left(\boldsymbol{W}^{++,1}\boldsymbol{Z}^{[n++]}(t) + \boldsymbol{W}^{++,2}\right.$$
$$\left.\left[\boldsymbol{Z}^{[n++]}(t-1)\,|\cdots|\boldsymbol{Z}^{[n++]}(t-L_{++}+1)\right]\right) \tag{33}$$

where $\boldsymbol{W}^{++,1} \in R^{(n+1)\times(n+1)}$ and $\boldsymbol{W}^{++,2} \in R^{(n+1)\times(L_{++}-1)(n+1)}$ are two randomly generated parameter matrices, L_{++} denotes the order parameter corresponding to HFCM^{++}, and $\boldsymbol{H}^{++}(t) \in^{(n+1)\times 1}$ is the added feature representation obtained by HFCM^{++} at iteration t. Define $\boldsymbol{H}^{++} = [\boldsymbol{H}^{++}(1), \boldsymbol{H}^{++}(2), \ldots, \boldsymbol{H}^{++}(k)] \in R^{(n+1)\times k}$. From the pipeline shown in Fig. 6, we can get,

$$\begin{aligned}\boldsymbol{v}^{++} &= [\boldsymbol{Vector}(\boldsymbol{P}), \boldsymbol{Vector}(\boldsymbol{H}^{++}), \boldsymbol{Vector}(\boldsymbol{z}^{++})] \\ &= [\boldsymbol{v}, \boldsymbol{Vector}(\boldsymbol{H}^{++}), \boldsymbol{Vector}(\boldsymbol{z}^{++})]\end{aligned} \tag{34}$$

where $\boldsymbol{Vector}(\boldsymbol{H}^{++}) \in R^{(n+1)k\times 1}$, $\boldsymbol{Vector}(\boldsymbol{z}^{++}) \in R^k$, and $\boldsymbol{v}^{++} \in R^{[(m+2)n+2]k}$. Therefore, the new input of the output layer for all samples $\boldsymbol{V}^{++} \in R^{N\times[(m+2)n+2]k}$ can be derived by:

$$\boldsymbol{V}^{++} = \left[\boldsymbol{V}, \boldsymbol{H}^b\right] \tag{35}$$

where $\boldsymbol{H}^b \in R^{N\times(n+2)k}$ is the matrix representation of vector $[\boldsymbol{Vector}(\boldsymbol{H}^{++}), \boldsymbol{Vector}(\boldsymbol{z}^{++})]$ for all samples, which is shown as follows,

$$\boldsymbol{V}^{++} = \begin{bmatrix} [\boldsymbol{Vector}(\boldsymbol{H}^{++}), \boldsymbol{Vector}(\boldsymbol{z}^{++})]_1 \\ [\boldsymbol{Vector}(\boldsymbol{H}^{++}), \boldsymbol{Vector}(\boldsymbol{z}^{++})]_2 \\ \vdots \\ [\boldsymbol{Vector}(\boldsymbol{H}^{++}), \boldsymbol{Vector}(\boldsymbol{z}^{++})]_N \end{bmatrix} \tag{36}$$

According to the discussion in Section 2, the pseudoinverse of $\boldsymbol{V}^{++}$ can be deduced as

$$\boldsymbol{V}_{inv}^{++} = \begin{bmatrix} \boldsymbol{V}_{inv} - \boldsymbol{D}\boldsymbol{B}^T \\ \boldsymbol{B}^T \end{bmatrix} \tag{37}$$

where $\boldsymbol{V}_{inv}$ is the pseudoinverse of $\boldsymbol{V}$,

$$\boldsymbol{D} = \boldsymbol{V}_{inv} \times \boldsymbol{H}^b \tag{38}$$

$$\boldsymbol{B}^T = \begin{cases} \boldsymbol{C}_{inv} & \text{if } \boldsymbol{C} \neq 0 \\ (1 + \boldsymbol{D}^{\mathbf{T}}\boldsymbol{D})^{-1}\boldsymbol{D}^T\boldsymbol{V}_{inv} & \text{if } \boldsymbol{C} = 0 \end{cases} \tag{39}$$

and $\boldsymbol{C} = \boldsymbol{H}^b - \boldsymbol{V} \times \boldsymbol{D}$.

Again, the new output weights $\boldsymbol{W}_{out}^{++}$ can be derived by

$$\boldsymbol{W}_{out}^{+} = \begin{bmatrix} \boldsymbol{W}_{out} - \boldsymbol{D}\boldsymbol{B}^T\boldsymbol{Y} \\ \boldsymbol{B}^T\boldsymbol{Y} \end{bmatrix} \tag{40}$$

Based on the two incremental learning strategies illustrated above, we propose a novel learning algorithm for BFCMS, whose details are listed in Algorithm 1.

3.5. Increment of input samples

In online scenes, the new training samples are typically obtained one by one or bulk by bulk, and we hope the trained BFCMS can still be further adjusted and optimized according to these newly added samples. Therefore, the new strategy is designed to update the weights easily without an entire training cycle.

Denote $\mathbf{x}^{+++}$ is a newly added sample, and the new input of the output layer $\boldsymbol{V}^{+++} \in R^{(N+1)\times(m+1)nk}$ can be derived by:

$$\boldsymbol{V}^{+++} = \begin{bmatrix} \boldsymbol{V} \\ \boldsymbol{v}^{+++} \end{bmatrix} \tag{41}$$

where $\boldsymbol{v}^{+++}$ is the newly generated feature representation vector corresponding to $\boldsymbol{x}^{+++}$. Similarly, the pseudoinverse of $\boldsymbol{V}^{+++}$ can be derived as

$$\boldsymbol{V}_{inv}^{+++} = \left[\boldsymbol{V}_{inv} - \boldsymbol{BD}^{T}, \boldsymbol{B}\right] \tag{42}$$

where

$$\boldsymbol{D} = \boldsymbol{v}^{+++} \times \boldsymbol{V}_{inv} \tag{43}$$

$$\boldsymbol{B}^{T} = \begin{cases} \boldsymbol{C}_{inv} & if\ \boldsymbol{C} \neq \boldsymbol{0} \\ (\boldsymbol{1} + \boldsymbol{D}^{T}\boldsymbol{D})^{-\boldsymbol{1}}\boldsymbol{D}^{T}\boldsymbol{V}_{inv} & if\ \boldsymbol{C} = \boldsymbol{0} \end{cases} \tag{44}$$

and $\boldsymbol{C} = \boldsymbol{v}^{+++} - \boldsymbol{D}^{T} \times \boldsymbol{V}$.

Then, the new output weights $\boldsymbol{W}_{out}^{+++}$ can be derived by

$$\boldsymbol{W}_{out}^{+++} = \boldsymbol{W}_{out} + (\boldsymbol{y}^{+++} - \mathbf{x}^{+++}\boldsymbol{W}_{out})\boldsymbol{B} \tag{45}$$

3.6. Computation complexity analysis

According to the previous definition, N is the number of samples, M is the dimension of each series, k is the encoding length of the hidden feature representation in the SAE unit. As shown in Fig. 3, the pipeline of BFCMS contains the following three modules: the feature extraction block based on n SAE units, the spatiotemporal information aggregation block based on m HFCM units, and one MLP layer. The complexity in calculating hidden encoding representation is $O\,(NMnk)$, the time complexity for calculating the senior feature representation is $O\,(Nnk^2)$, and the time complexity for calculating the $\boldsymbol{W}_{out}$ of the MLP layer is $O\left(2N\,((m+1)\,nk)^2 + ((m+1)\,nk)^3 + N\,((m+1)\,nk)\,D\right)$. Therefore, the computation complexity of BFCMS is,

$$\max\left\{O\,(NMnk)\,,\,O\left(Nnk^2\right),\,O\left(2N\,((m+1)\,nk)^2 + ((m+1)\,nk)^3 + N\,((m+1)\,nk)\,D\right)\right\} \tag{46}$$

4. Experiments

4.1. Experimental setup

Datasets. UCR archive [47], firstly introduced by Keogh & Folias in 2002, has become an important resource in time series data mining, which contains hundreds of time series datasets. One intuitive thought is to compare BFCMS with other state-of-the-art methods in all datasets. However, as mentioned in previous work [8], conducting experiments on all datasets is extremely time-consuming. Therefore, this paper shows a comparison experiment on 26 representative datasets.[1] Their characteristics also vary in different perspectives, e.g., the size, the training-test partition, and the number of labels. Moreover, the motivation of BFCMS is to develop a novel efficient TSC classifier with a lower training cost and competitive classification performance, instead of designing the method with further improvements in accuracy. Therefore, the selected 26 datasets with various characteristics are enough to validate the effectiveness of BFCMS. Among these datasets, several are large with thousands of time series, whereas several are small with no more than 100 samples. Besides, proportions between the training and test datasets are typically dissimilar. For example, the Meat dataset contains 60 training and 60 test samples. In contrast, the training dataset of ECG5000 consists of 500 samples, and the test dataset includes 4500 samples. The length of the time series varies as well. The shortest time series in the Chinatown dataset comprises 24 data points. The longest time series in the Rock dataset consists of 2844 observations. A comprehensive description of these datasets is on the referenced web page. One can easily retrieve a related introduction by searching for the name of some specific time series on the web page. Owing to space limitations, we do not reiterate it here. Note that the data partition is consistent with the previous work and keeps the same as the default partition in the UCR archive for a fair comparison.

Performance Measurement. In classification analysis, accuracy is a popular measure for evaluating the performance of the classification algorithms. As such, we employ accuracy to assess the performance of the proposed approach. Accuracy can be calculated as

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{47}$$

where TP and TN denote the number of the true-positive samples, and the true-negative samples, respectively. FP and FN represent the amount of false-positive and false-negative samples, respectively.

Algorithms. We compare BFCMS with the following four categories of TSC algorithms:

(1) Broad neural network: BLS [21] is employed to show that BFCMS can overcome its limitations and keep its advantages in terms of training time and flexibility.

(2) Deep neural networks: ResNet [19] and InceptionTime [48]. They are two state-of-the-art deep learning methods for TSC tasks.

(3) Traditional methods: The bag of SFA symbols (BOSS) [49], Catch22 forest [50], word extraction for time series classification (WEASEL) [51], time series forest (TSF) [52], random interval spectral ensemble (RISE) [14], and proximity forest (PF) [53], which are popular and competitive in these years.

(4) FCM-based TSC method. Time series classification using FCMs (TSC-FCM) [54] is a representative method in this part.

Among these baselines, the implementation of BLS is obtained from the original paper, and the codes of deep neural networks and traditional methods are obtained from the website[1] and the sktime package.[2] Since no code is provided in the original TSC-FCM paper, we attempt to implement it for comparison by ourselves, and the corresponding parameters are set to the recommended values in the paper.

Other settings. In this paper, the grid-search operator is applied to determine the key parameters of BFCMS, in which the sets for determining k and n are both set to [5, 6, …, 25], the set for selecting m is [1, 2, …, 8], and the set [1, 2, …, 5] is applied as the search space of L. Besides, λ is a penalty norm and is not sensitive to different tasks. Therefore, it is defaulted to 2e−30 [21]. Table 2 shows the corresponding hyperparameter settings chosen by a grid search to obtain the best performance. Also, the python implementation of BFCMS is accessed in: https://github.com/maisuiqianxun/BFCMS.

[1] These datasets are available from http://timeseriesclassification.com.

[2] https://github.com/alan-turing-institute/sktime.

Algorithm 1 The Learning Algorithm of BFCMS

Input: $\{X, Y\}$: the training data including N samples;
Output: W_3.

```
1:  for $u$=0; $u \le N$ do
2:      for $i$=0; $i \le n$ do
3:          Random generate initial $W^i$;
4:          Obtain the representation $z^i$ using the $i$-th SAE with the $u$-th input $<x>_u$
            as in (9);
5:      end
6:      $<Z^{[n]}>_u = [z^1, z^2, \ldots, z^n]^T \in R^{n \times k}$;
7:      for $j$=0; $j \le m$ do
8:          Random generate initial $W^{j,1}$, $W^{j,2}$;
9:          Obtain the feature representation $<H^j>_u$ using the $j$-th HFCM with the
            $u$-th input $<Z^{[n]}>_u$ via (19);
10:     end
11:     $<H^{[m]}>_u = [<H^1>_u, <H^2>_u, \ldots, <H^m>_u] \in R^{m \times n \times k}$;
12:     Construct the final tensor representation $<P>_u$ via (20);
13:     Obtained feature vector representation via $v_u = Vector(P) \in R^{1 \times (m+1)nk}$;
14: end
15: $V = [v_1, v_2, \ldots, v_N]^T \in R^{N \times (m+1)nk}$;
16: Calculate $W_{out}$ with (23);
17: while $training\ accuracy < \varepsilon$ do
18:     if new HFCMs are added do
19:         Recalculate $V^+$ with (27), and Update $W_{out}$ with (32);
20:     end
21:     if new SAEs are added do
22:         Recalculate $V^{++}$ with (36), and Update $W_{out}$ with (41);
23:     end
24: end while
25: Output $W_{out}$.
```

Table 2
The parameters of BFCMS on twenty five datasets.

Dataset	[k, n, m, L, λ]	Dataset	[k, n, m, L, λ]
Beef	[6, 25, 1, 3, 2e−30]	Herring	[9, 12, 8, 3, 2e−30]
BeetleFly	[5, 15, 6, 3, 2e−30]	InsectWingbeatSound	[5, 9, 1, 3, 2e−30]
BirdChicken	[7, 5, 3, 3, 2e−30]	ItalyPowerDemand	[8, 19, 4, 3, 2e−30]
Chinatown	[14, 10, 3, 1, 2e−30]	Meat	[5, 8, 1, 3, 2e−30]
ChlorineConcentration	[22, 30, 1, 3, 2e−30]	MoteStrain	[18, 20, 1, 1, 2e−30]
Coffee	[5, 5, 7, 1, 2e−30]	MiddlePhalanxOutlineAgeGroup	[14, 14, 2, 3, 2e−30]
DiatomSizeReduction	[6, 14, 3, 1, 2e−30]	MiddlePhalanxTW	[5, 15, 7, 3, 2e−30]
DistalPhalanxTW	[5, 9, 1, 3, 2e−30]	OliveOil	[5, 9, 1, 3, 2e−30]
Earthquakes	[5, 8, 6, 2, 2e−30]	Plane	[6, 30, 1, 3, 2e−30]
ECG200	[9, 18, 8, 1, 2e−30]	PowerCons	[6, 17, 1, 3, 2e−30]
ECG5000	[5, 8, 1, 3, 2e−30]	ProximalPhalanxOutlineAgeGroup	[5, 12, 3, 3, 2e−30]
ECGFiveDays	[10, 19, 1, 3, 2e−30]	ProximalPhalanxTW	[5, 13, 4, 3, 2e−30]
Haptics	[5, 16, 1, 3, 2e−30]	Rock	[5, 16, 2, 3, 2e−30]

4.2. Results

This section compares the BFCMS model with the above baselines on 26 time series datasets. The experimental results are reported in Table 3. We can see that: (1) BFCMS outperforms TSC-FCM in 22 out of 26 cases. The latter is a representative of FCM-based TSC algorithms, which applied the fuzzy c-means to capture the features of the original time series, and the results may be worse due to the improper initialization of cluster centers. (2) The achieved accuracies of BFCMS are higher than those of BLS in 24 datasets of all 26 datasets. In BLS, both the feature mapping nodes and the enhancement nodes cannot extract the temporal dependency of the original time series, which limits the classification ability of BLS. While in BFCMS, the spatiotemporal information aggregation block containing HFCMs units with different orders is specially designed for modeling the dynamic information hidden in the obtained junior feature representations. Therefore, BFCMS could make significant improvements

Table 3
The experimental results of BFCMS on twenty six datasets in terms of accuracy.

Dataset	BFCMS	BLS	TSC-FCM	BOSS	Catch22	WEASEL	TSF	RISE	PF	ResNet	InTime
Beef	**0.967**	0.733	0.8	0.612	0.8	0.74	0.688	0.742	0.594	0.676	0.682
BeetleFly	**1**	0.75	0.5	0.943	0.5	0.886	0.833	0.871	0.86	0.853	0.893
BirdChicken	0.950	0.5	0.5	**0.983**	0.5	0.864	0.815	0.868	0.903	0.945	0.951
Chinatown	**0.985**	0.967	0.778	0.877	0.778	0.957	0.952	0.888	0.948	0.970	0.964
ChlorineConcentration	**0.878**	0.875	0.473	0.658	0.642	0.755	0.723	0.764	0.631	0.841	0.864
Coffee	**1**	**1**	0.642	0.985	0.699	0.989	0.986	0.984	0.991	0.996	0.998
DiatomSizeReduction	**0.980**	0.960	0.699	0.945	0.675	0.908	0.941	0.932	0.956	0.306	0.950
DistalPhalanxTW	0.705	0.561	0.675	0.671	**0.748**	0.678	0.691	0.694	0.692	0.667	0.665
Earthquakes	**0.777**	0.424	0.748	0.746	0.77	0.747	0.747	0.748	0.749	0.717	0.732
ECG200	0.870	0.870	0.770	0.878	**0.923**	0.858	0.860	0.850	0.873	0.883	0.895
ECG5000	0.927	0.881	0.923	0.940	0.738	0.942	**0.943**	0.936	0.939	0.937	0.942
ECGFiveDays	0.978	0.981	0.738	0.992	0.792	0.993	0.951	0.972	0.882	0.951	**0.995**
Haptics	0.468	0.357	**0.792**	0.467	0.593	0.449	0.465	0.478	0.458	0.496	0.536
Herring	0.734	0.578	0.593	0.595	**0.903**	0.602	0.604	0.598	0.574	0.596	0.625
InsectWingbeatSound	0.581	0.416	**0.903**	0.511	0.552	0.619	0.603	0.636	0.606	0.491	0.627
ItalyPowerDemand	**0.959**	0.904	0.552	0.870	0.666	0.946	**0.959**	0.944	0.956	0.957	**0.960**
Meat	**1**	0.833	0.666	0.980	0.629	0.976	0.983	0.986	0.987	0.993	0.983
MoteStrain	0.884	0.851	0.530	0.844	0.643	0.905	0.855	0.878	**0.915**	0.903	0.880
MiddlePhalanxOAG	0.649	0.441	0.629	0.656	0.833	0.660	0.659	**0.699**	0.658	0.596	0.594
MiddlePhalanxTW	0.610	0.428	**0.643**	0.532	0.738	0.554	0.568	0.585	0.549	0.531	0.526
OliveOil	**0.967**	0.933	0.833	0.875	0.834	0.913	0.893	0.893	0.878	0.862	0.874
Plane	0.991	0.971	0.943	0.998	0.819	0.995	0.996	0.997	**1**	**1**	0.997
PowerCons	0.928	0.9	0.738	0.890	0.58	0.919	**0.993**	0.957	0.987	0.886	0.986
ProximalPhalanxOAG	**0.878**	0.629	0.834	0.828	0.813	0.844	0.845	0.857	0.840	0.817	0.822
ProximalPhalanxTW	0.776	0.585	**0.819**	0.768	0.82	0.801	0.801	0.813	0.790	0.789	0.780
Rock	**0.940**	0.88	0.58	0.802	0.705	0.854	0.76	0.781	0.774	0.416	0.627
wins\total	\	24\26	22\26	16\26	19\26	18\26	19\26	19\26	17\26	20\26	16\26
Average	0.861	0.739	0.704	0.802	0.719	0.821	0.812	0.821	0.807	0.772	0.821
(Improvement)	\	16.52%	22.30%	7.37%	19.73%	4.81%	6.01%	4.83%	6.63%	11.49%	4.84%

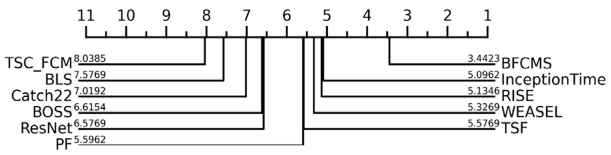


Fig. 7. Critical difference diagram of 11 algorithms on all used datasets.

compared with BLS. (3) BFCMS achieves better performance than ResNet and InceptionTime in most databases. Note that the number of training samples in many datasets is relatively small (the largest Chinatown only contains 1194 training samples), and the deep learning-based models may be overfitting, thereby resulting in a not competitive performance in TSC tasks. Although this phenomenon is not consistent with their outstanding performance expressed in large-scale image tasks, it is still reasonable and acceptable due to the unique characteristics of time series data. (4) BFCMS has an advantage over several traditional TSC models, e.g., the dictionary-based methods (BOSS and WEASEL), the interval-based method (TSF), the frequency-based method (RISE), the shapelet-based method (STC), and the distance-based method (PF). The results indicate that BFCMS has an advantage over these traditional TSC models regarding accuracy.

To compare the overall classification performance of different algorithms on all datasets, Fig. 7 shows the corresponding critical difference diagrams[3] with cliques formed with the Wilcoxon–Holm signed-rank test [17]. We can see that: BFCMS obtains the highest rank with an average value of almost 3.44 with respect to the other baselines. The top clique is (BFCMS, InceptionTime, RISE, WEASEL, and TSF), and the top classifier is slightly more accurate than the baselines TSC-FCM, BLS, Catch22, BOSS, and ResNet. Compared with the top clique, BFCMS outperforms InceptionTime, RISE, WEASEL, and TSF in most cases.

Note that the motivation of this paper is to develop an efficient TSC algorithm. Thus, the time cost of training the classifier is also our focus. Fig. 8 depicts how the training time changes as we adjust the ratio of the training set on the Haptics and the InsectWingbeatSound datasets. All experiments are tested on the Pycharm software platform under a desktop equipped with Intel-i7 3.70 GHz CPU, and 32 GB memory, where Catch22, TSF, IndividualBOSS, and WEASEL are simulated on the sktime package[4]. Note that TSC-FCM and deep learning methods are far slower than the visualized baselines in Fig. 8. Therefore, we do not plot their training time curves for clear resolution. It can be seen that: BFCMS and BLS achieve a smaller "derivative" of training time w.r.t the ratio than other baselines, which means that BFCMS and BLS will be more competitive in handling large-scale TSC tasks due to lower training time.

Also, in Fig. 9, we visualize a 2-order HFCM structure obtained from the Haptics dataset. In the first-order HFCM shown in Fig. 9(a), the effect of vertex "4" on vertex "1" is significant, while for the second-order HFCM shown in Fig. 9(b), the impact of vertex "4" on vertex "6" is more critical than other vertices, which indicates that the HFCMs can catch the different temporal dependencies hidden in time series data.

4.3. The effect of IL-HFCM on BFCMS

This section demonstrates how the increment of additional HFCMs affects the classification accuracy, taking the Haptics and the InsectWingbeatSound datasets as instances. The initial number of SAEs n is fixed as 8, k is set to 5, m is set to 1, L is fixed as 3, and λ is initialized as 2e−30. Then, the number of added HFCMs m_a is gradually added from 1 to 10 in steps of 1 in each cycle. Fig. 10 shows the performance of the above dynamic constructions for these two datasets. In Fig. 10, we can see that the accuracy fluctuates with more HFCMs added to BFCMS. Since

[3] https://github.com/hfawaz/cd-diagram.

[4] https://github.com/alan-turing-institute/sktime.

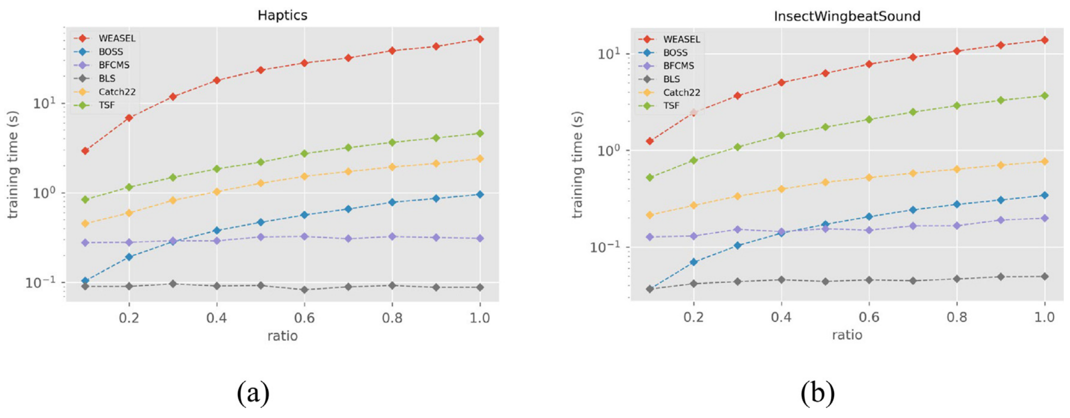


Fig. 8. Training time as a function of the ratio of the training set size for (a) the Haptics and (b) InsectWingbeatSound datasets.

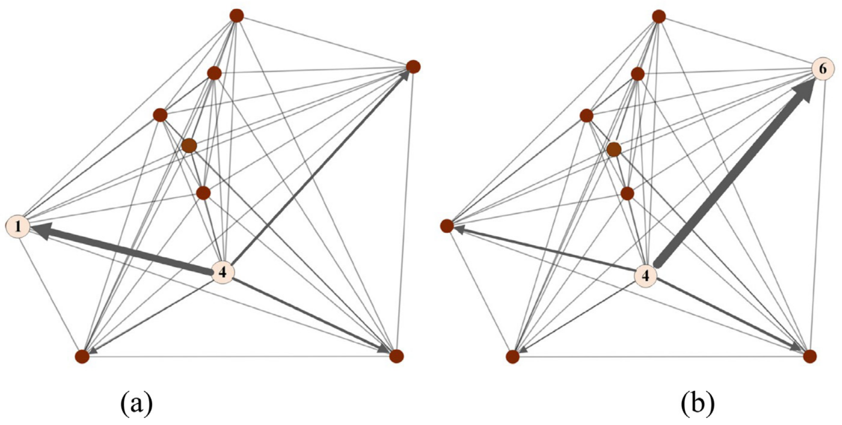


Fig. 9. The structure visualization of a 2-order HFCMs with ten vertices on (a) the first-order HFCM (b) the second-order HFCM, where the width of the edge is proportional to its weights.

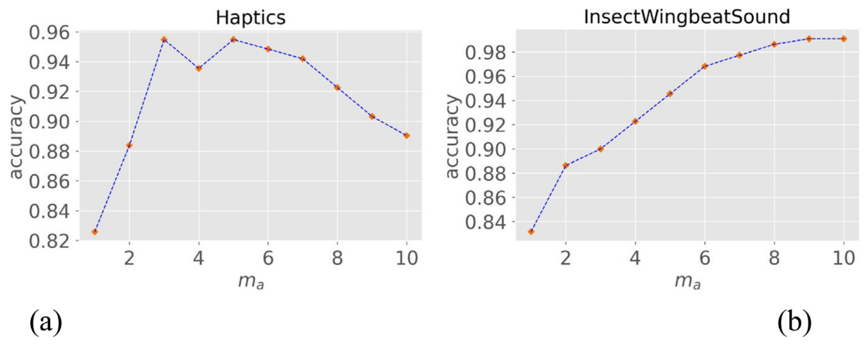


Fig. 10. The training accuracy curves with varying increment of HFCMs on (a) the Haptics and (b) the InsectWingbeatSound datasets.

the new HFCMs may not capture the representative information of the time series, there is no guarantee that the accuracy will improve. However, the BFCMS can still be optimized with the greedy strategy illustrated in lines 17–24 of Algorithm 1.

4.4. The effect of IL-SAE on BFCMS

This section demonstrates how the increment of additional SAEs affects the classification accuracy, also taking the Haptics and the InsectWingbeatSound datasets as examples. The initial parameters of the BFCMS are consistent with the setting in Section 4.3. Then the number of the SAEs n_a is dynamically added from 1 to 10 in steps of 1 in each update. Fig. 11 depicts the performance of the above dynamic constructions for the Haptics and the InsectWingbeatSound datasets. From Fig. 11, we can see that the accuracy of these two datasets shows an upward trend overall when the new SAEs are constantly being added. On some occasions (e.g., adding two SAE units on the DistalPhalanxTW dataset and four SAE units on Earthquakes datasets), the accuracy worsens with new SAEs added. This phenomenon may be caused by the fact that the newly added SAEs capture the redundant feature of the original time series, which limits the performance

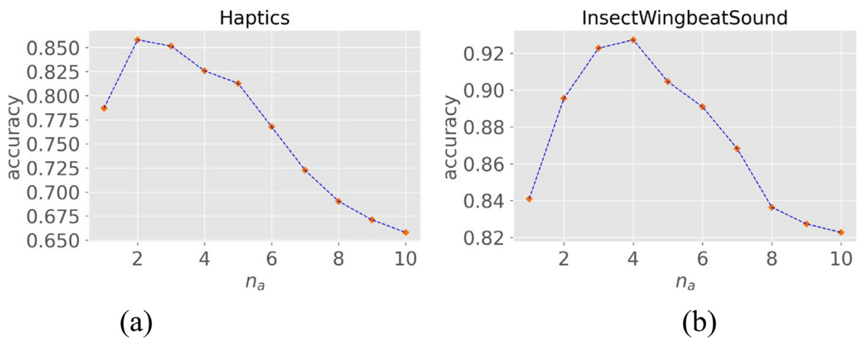


Fig. 11. The training accuracy curves with varying increment of SAEs on (a) the Haptics and (b) InsectWingbeatSound datasets.

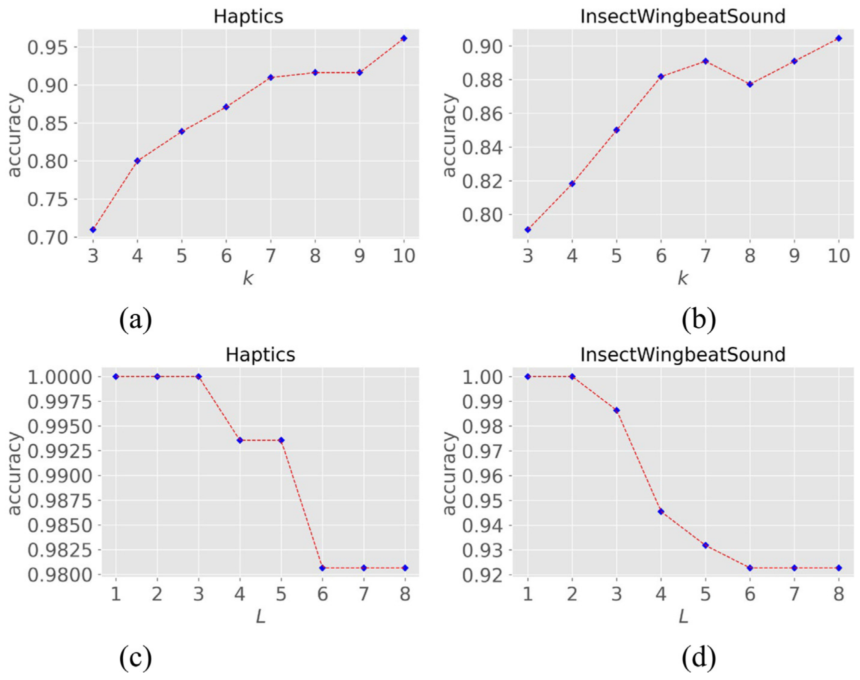


Fig. 12. The training accuracy versus varying k of the BFCMS on (a) the Haptics dataset and (b) the InsectWingbeatSound dataset. The training accuracy versus varying L of the BFCMS on (c) the Haptics dataset and (d) the InsectWingbeatSound dataset.

of the BFCMS. Nevertheless, we can still optimize the BFCMS via the greedy-based strategy shown in lines 17–24 of Algorithm 1.

4.5. Sensitivities analysis in BFCMS

This section studies several primary hyperparameters in BFCMS and takes the Haptics and the InsectWingbeatSound datasets as examples. There are four key parameters: k, n, m, and L, where n, and m are adjusted and optimized by adding extended HFCM and SAE units shown in Algorithm 1, and their effects are shown in Figs. 10 and 11 indirectly. Therefore, we mainly study the sensitivities of the other two parameters, k and L. When analyzing the effect of one hyperparameter, other hyperparameters remain fixed. Fig. 12(a) and (b) show the accuracy of BFCMS versus varying k with initial [k, n, m, L] is set to [1,8,8,34] on two selected examples, from which we can see that a smaller k value is not proper for extracting the features of the original time series, and we conclude that k may be more proper to set a larger value without a priori in the new situation. Fig. 12(c) and (d) show the accuracy of BFCMS versus varying L in the condition of initial [k, n, m, L] is set to [1,1,34,45]. When L increases gradually, the training accuracy of these two datasets becomes lower. Therefore, we advise that the order of the HFCMs can be initialized as a smaller value such as 2 or 3.

5. Conclusion

In this paper, a novel broad FCM system is designed to solve TSC problems with low training time and relatively high accuracy, which is critical for handling TSC tasks in real-world scenarios, e.g., classifying time series with concept drifting requires the analytical model can be expanded quickly. BFCMS mainly comprises the SAE-based feature extraction block, the HFCM-based spatiotemporal information aggregation block, and the MLP classification layer. In this way, BFCMS overcomes the limitations of BLS by capturing temporal information of time series, representing the evolving relations among states along time, and retaining the time complexity advantages of BLS. We conduct extensive experiments on dozens of datasets to test the performance of BFCMS.

The obtained results demonstrate the prominence of BFCMS in classification accuracy and the training time cost. Besides, three incremental learning strategies are developed to update BFCMS with excellent efficiency. We also validate the superiority of incremental learning strategies in model optimization.

However, how to enable the BFCMS to optimize itself through removing some redundant nodes, i.e., decreasing learning strategy, and how to provide the BFCMS to forget some poisoned data, i.e., machine unlearning in BFCMS, are still not discussed and addressed in this paper, which limits the generalization of BFCMS to some extent. In the future, we will focus on developing decreased learning algorithms to update the BFCMS and develop particular machine unlearning algorithms for broad neural networks.

CRediT authorship contribution statement

Kai Wu: Conceptualization, Methodology, Writing – original draft. **Kaixin Yuan:** Conceptualization, Methodology, Software, Validation, Formal analysis, Writing – original draft, Writing – review & editing. **Yingzhi Teng:** Writing – review & editing. **Jing Liu:** Writing – review & editing, Supervision, Funding acquisition. **Licheng Jiao:** Writing – review & editing.

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.

Data availability

Data will be made available on request.

Acknowledgments

This work was supported by the Key Project of Science and Technology Innovation 2030 supported by the Ministry of Science and Technology of China under Grant 2018AAA0101302.

References

[1] Q. Yang, X. Wu, 10 Challenging problems in data mining research, Int. J. Inf. Technol. Decis. Mak. 5 (4) (2006) 597–604.
[2] P. Esling, C. Agon, Time-series data mining, ACM Comput. Surv. 45 (1) (2012) 1–34.
[3] W.X. Cheng, P.N. Suganthan, R. Katuwal, Time series classification using diversified ensemble deep random vector functional link and resnet features, Appl. Soft Comput. 112 (2021) 107826.
[4] H. Zhu, J. Zhang, H. Cui, K. Wang, Q. Tang, TCRAN: Multivariate time series classification using residual channel attention networks with time correction, Appl. Soft Comput. 114 (2022) 108117.
[5] Z. Huang, et al., Functional deep echo state network improved by a bi-level optimization approach for multivariate time series classification, Appl. Soft Comput. 106 (2021) 107314.
[6] R. Khasha, M.M. Sepehri, N. Taherkhani, Detecting asthma control level using feature-based time series classification, Appl. Soft Comput. 111 (2021) 107694.
[7] S. Hassona, W. Marszalek, J. Sadecki, Time series classification and creation of 2D bifurcation diagrams in nonlinear dynamical systems using supervised machine learning methods, Appl. Soft Comput. 113 (2021) 107874.
[8] A. Bagnall, J. Lines, A. Bostrom, J. Large, E. Keogh, The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances, Data Min. Knowl. Discov. 31 (3) (2017) 606–660.
[9] H.I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, P.A. Muller, Deep learning for time series classification: a review, Data Min. Knowl. Discov. 33 (4) (2019) 917–963.
[10] A.P. Ruiz, M. Flynn, J. Large, M. Middlehurst, A. Bagnall, The great multivariate time series classification bake off: a review and experimental evaluation of recent algorithmic advances, Data Min. Knowl. Discov. 35 (2) (2021) 401–449.
[11] J. Lines, A. Bagnall, Time series classification with ensembles of elastic distance measures, Data Min. Knowl. Discov. 29 (3) (2015) 565–592.
[12] L. Ye, E. Keogh, Time series shapelets: a new primitive for data mining, in: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2009, pp. 947–956.
[13] A. Bagnall, J. Lines, J. Hills, A. Bostrom, Time-series classification with COTE: The collective of transformation-based ensembles, in: International Conference on Data Engineering, 2016, pp. 1548–1549.
[14] J. Lines, S. Taylor, A. Bagnall, Time series classification with HIVE-COTE: The hierarchical vote collective of transformation-based ensembles, ACM Trans. Knowl. Discov. Data 12 (5) (2018) 1–35.
[15] C.L. Liu, W.H. Hsaio, Y.C. Tu, Time series classification with multivariate convolutional neural network, IEEE Trans. Ind. Electron. 66 (6) (2018) 4788–4797.
[16] Q. Ma, E. Chen, Z. Lin, J. Yan, Z. Yu, W.W. Ng, Convolutional multitimescale echo state network, IEEE Trans. Cybern. 51 (3) (2021) 1613–1625.
[17] Z. Gong, H. Chen, B. Yuan, X. Yao, Multiobjective learning in the model space for time series classification, IEEE Trans. Cybern. 49 (3) (2018) 918–932.
[18] F. Karim, S. Majumdar, H. Darabi, Insights into LSTM fully convolutional networks for time series classification, IEEE Access 7 (2019) 67718–67725.
[19] Z. Wang, W. Yan, T. Oates, Time series classification from scratch with deep neural networks: A strong baseline, in: 2017 International Joint Conference on Neural Networks (IJCNN), 2017, pp. 1578–1585.
[20] J. Du, C.M. Vong, C.P. Chen, Novel efficient RNN and LSTM-like architectures: Recurrent and gated broad learning systems and their applications for text classification, IEEE Trans. Cybern. 51 (3) (2020) 1586–1597.
[21] C.L.P. Chen, Z. Liu, Broad learning system: An effective and efficient incremental learning system without the need for deep architecture, IEEE Trans. Neural Netw. Learn. Syst. 29 (1) (2018) 10–24.
[22] S. Feng, C.L.P. Chen, Fuzzy broad learning system: A novel neuro-fuzzy model for regression and classification, IEEE Trans. Cybern. 50 (2) (2018) 414–424.
[23] X. Gong, T. Zhang, C.L.P. Chen, Z. Liu, Research review for broad learning system: Algorithms, theory, and applications, IEEE Trans. Cybern. (2021) http://dx.doi.org/10.1109/TCYB.2021.3061094.
[24] R. Han, Z. Liu, C.L.P. Chen, Multi-scale 3D convolution feature-based broad learning system for alzheimer's disease diagnosis via MRI images, Appl. Soft Comput. 120 (2022) 108660.
[25] P. Chang, L. Zhao, F. Meng, Y. Xu, Soft measurement of effluent index in sewage treatment process based on overcomplete broad learning system, Appl. Soft Comput. 115 (2022) 108235.
[26] J. Guo, L. Wang, K. Fan, B. Yang, An efficient model for predicting setting time of cement based on broad learning system, Appl. Soft Comput. 96 (2020) 106698.
[27] C.P. Chen, Z. Liu, S. Feng, Universal approximation capability of broad learning system and its structural variations, IEEE Trans. Neural Netw. Learn. Syst. 30 (4) (2018) 1191–1204.
[28] M. Xu, M. Han, C.P. Chen, T. Qiu, Recurrent broad learning systems for time series prediction, IEEE Trans. Cybern. 50 (4) (2018) 1405–1417.
[29] M. Han, S. Feng, C.L.P. Chen, M. Xu, T. Qiu, Structured manifold broad learning system: A manifold perspective for large-scale chaotic time series analysis and prediction, IEEE Trans. Knowl. Data Eng. 31 (9) (2019) 1809–1821.
[30] S. Feng, W. Ren, M. Han, Y.W. Chen, Robust manifold broad learning system for large-scale noisy chaotic time series prediction: A perturbation perspective, Neural Netw. 117 (2019) 179–190.
[31] M. Han, W. Li, S. Feng, T. Qiu, C.P. Chen, Maximum information exploitation using broad learning system for large-scale chaotic time-series prediction, IEEE Trans. Neural Netw. Learn. Syst. 32 (6) (2020) 2320–2329.
[32] D. Liu, S. Baldi, W. Yu, J. Cao, W. Huang, On training traffic predictors via broad learning structures: A benchmark study, IEEE Trans. Syst. Man Cybern.: Syst. 52 (2) (2022) 749–758.
[33] A. Ng, Sparse Autoencoder, in: CS294A Lecture Notes, vol. 72, Stanford Univ., Stanford, CA, USA, 2011.
[34] P. Vincent, H. Larochelle, Y. Bengio, P.-A. Manzagol, Extracting and composing robust features with denoising autoencoders, in: Proc. 25th Int. Conf. Mach. Learn., 2008, pp. 1096–1103.
[35] B. Kosko, Fuzzy cognitive maps, Int. J. Man-Mach. Stud. 24 (1) (1986) 65–75.
[36] K. Wu, J. Liu, P. Liu, F. Shen, Online fuzzy cognitive map learning, IEEE Trans. Fuzzy Syst. 29 (7) (2020) 1885–1898.
[37] E.I. Papageorgiou, J.L. Salmeron, A review of fuzzy cognitive maps research during the last decade, IEEE Trans. Fuzzy Syst. 21 (1) (2012) 66–79.
[38] K. Wu, J. Liu, Robust learning of large-scale fuzzy cognitive maps via the lasso from noisy time series, Knowl. Based Syst. 113 (2016) 23–38.
[39] K. Wu, J. Liu, Learning large-scale fuzzy cognitive maps based on compressed sensing and application in reconstructing gene regulatory networks, IEEE Trans. Fuzzy Syst. 25 (6) (2017) 1546–1560.

[40] K. Wu, J. Liu, P. Liu, S. Yang, Time series prediction using sparse autoencoder and high-order fuzzy cognitive maps, IEEE Trans. Fuzzy Syst. 28 (12) (2020) 3110–3121.
[41] E. Puerto, J. Aguilar, C. López, D. Chávez, Using multilayer fuzzy cognitive maps to diagnose autism spectrum disorder, Appl. Soft Comput. 75 (2019) 58–71.
[42] F. Vanhoenshoven, G. Nápoles, W. Froelich, J.L. Salmeron, K. Vanhoof, Pseudoinverse learning of fuzzy cognitive maps for multivariate time series forecasting, Appl. Soft Comput. 95 (2020) 106461.
[43] A. Nair, D. Reckien, M. Maarseveen, A generalised fuzzy cognitive mapping approach for modelling complex systems, Appl. Soft Comput. 84 (2019) 105754.
[44] K. Wu, J. Liu, Y. Chi, Wavelet fuzzy cognitive maps, Neurocomputing 232 (2017) 94–103.
[45] K. Tsadiras, Comparing the inference capabilities of binary, trivalent and sigmoid fuzzy cognitive maps, Inform. Sci. 178 (2008) 3880–3894.
[46] S. Boyd, N. Parikh, E. Chu, Distributed optimization and statistical learning via the alternating direction method of multipliers, Found. Trends Mach. Learn. 3 (1) (2011) 1–122.
[47] Y. Chen, et al., The UCR time series classification archive, IEEE/CAA J. Autom. Sin. 6 (6) (2019) 1293–1305.
[48] H.I. Fawaz, et al., Inceptiontime: Finding alexnet for time series classification, Data Min. Knowl. Discov. 34 (6) (2020) 1936–1962.
[49] P. Schäfer, The BOSS is concerned with time series classification in the presence of noise, Data Min. Knowl. Discov. 29 (6) (2015) 1505–1530.
[50] C.H. Lubba, S.S. Sethi, P. Knaute, S.R. Schultz, B.D. Fulcher, N.S. Jones, Catch22: Canonical time-series characteristics, Data Min. Knowl. Discov. 33 (6) (2019) 1821–1852.
[51] P. Schäfer, U. Leser, Fast and accurate time series classification with weasel, in: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, 2017, pp. 637–646.
[52] H. Deng, G. Runger, E. Tuv, M. Vladimir, A time series forest for classification and feature extraction, Inform. Sci. 239 (2013) 142–153.
[53] B. Lucas, et al., Proximity forest: an effective and scalable distance-based classifier for time series, Data Min. Knowl. Discov. 33 (3) (2019) 607–635.
[54] W. Homenda, A. Jastrzebska, Time-series classification using fuzzy cognitive maps, IEEE Trans. Fuzzy Syst. 28 (7) (2020) 1383–1394.