Information Sciences 571 (2021) 65–86

Contents lists available at ScienceDirect

Information Sciences

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

RTFN: A robust temporal feature network for time series classification

Zhiwen Xiao [a], Xin Xu [b], Huanlai Xing [a,*], Shouxi Luo [a], Penglin Dai [a], Dawei Zhan [a]

[a] School of Computing and Artificial Intelligence, Southwest Jiaotong University, Chengdu, China
[b] China University of Mining and Technology, Xuzhou, China

ARTICLE INFO

Article history:
Received 13 October 2020
Received in revised form 9 April 2021
Accepted 11 April 2021
Available online 20 April 2021

Keywords:
Attention mechanism
Convolutional neural network
Data mining
LSTM
Time series classification

ABSTRACT

Time series data usually contains local and global patterns. Most of the existing feature networks focus on local features rather than the relationships among them. The latter is also essential, yet more difficult to explore because it is challenging to obtain sufficient representations using a feature network. To this end, we propose a novel robust temporal feature network (RTFN) for feature extraction in time series classification, containing a temporal feature network (TFN) and a long short-term memory (LSTM)-based attention network (LSTMaN). TFN is a residual structure with multiple convolutional layers, and functions as a local-feature extraction network to mine sufficient local features from data. LSTMaN is composed of two identical layers, where attention and LSTM networks are hybridized. This network acts as a relation extraction network to discover the intrinsic relationships among the features extracted from different data positions. In experiments, we embed the RTFN into supervised and unsupervised structures as a feature extractor and encoder, respectively. The results show that the RTFN-based structures achieve excellent supervised and unsupervised performances on a large number of UCR2018 and UEA2018 datasets.

1. Introduction

Time series data has been used in various domains, such as, energy detection [8], earthquake detection [47], and arrhythmia detection [48]. Making full use of the data in real-world applications is crucial, but is dependent on how well the features are extracted. A time series is a sequence of time-ordered data points recording certain processes, which is different from other types of data, such as ImageNet[1] for image classification, SemEval-2014[2] for sentiment recognition, and ICDAR2019[3] for natural scene text processing [12]. In time series data, local patterns are local temporal features, while global patterns are relationships among local ones. Recently, effective feature and relation extraction has become a critical challenge, which is also a basis for time series classification [11,27].

Traditional and deep learning algorithms are the two main approaches for addressing the challenge above [11]. Traditional approaches aim at mining features and regularizations from data by revealing the significant differences and

* Corresponding author.
E-mail address: hxx@home.swjtu.edu.cn (H. Xing).

[1] http://www.image-net.org/challenges/LSVRC/2012/.
[2] http://alt.qcri.org/semeval2014/task4/.
[3] https://rrc.cvc.uab.es/?com=introduction.

https://doi.org/10.1016/j.ins.2021.04.053
0020-0255/

connections within the data. These approaches are mainly distance-based and feature-based. Distance-based approaches address a classification task by measuring the similarities between the spatial features of data. Combining the nearest neighbor (NN) and dynamic time warping (DTW) has been widely researched [24], such as, DD_{DTW} [28], DTD_C [28], DTW_I [9], DTW_D [32], and DTW_A [27]. In addition, there are a number of NN-DTW-based ensemble approaches. For example, the elastic ensemble (EE) uses 11 1-NN-based elastic distance measured to achieve decent performance on various time series datasets [24]. The collective of transformation ensembles (COTE) considered over 35 weighted classifiers [11]. Besides, the hierarchical vote collective of transformation-based ensembles (HIVE-COTE) [25] and the local cascade ensemble (LCE) [9] are two representative algorithms found in the literature.

Feature-based approaches pay more attention to exploring representative features from time series data. For example, Baydogan et al. [4] introduced a bag-of-features framework for feature extraction via a truncated discrete Fourier transform. The hidden state conditional random field used hidden variables to build the latent structure of the input [33]. The learned pattern similarity (LPS) [3], bag of SFA symbols (BOSS) [21], and time series forest (TSF) [7] are also feature-based.

Research efforts have also been dedicated to other techniques apart from the distance- and feature-based approaches. For example, the ultra-fast shapelets applied a support vector machine and random forest to generate random shapelets from the input [45]. The symbolic representation for multivariate time series (SMTS) adopted random forest to divide the time series data into leaf nodes for local-pattern extraction [2]. Tuncel et al. [42] applied autoregressive forests (mv-ARF) to mine representative shapelets from multivariate time series data. Karlsson et al. [19] used generalized random shapelet forests (gRSF) to extract significant features. WEASEL + MUSE utilized a bag-of-patterns model with various sliding window sizes for multivariate feature extraction [37].

Deep learning algorithms aim at unfolding the internal representational hierarchy of the data, which helps to capture the intrinsic connections among representations [23]. These algorithms are roughly classified into two models: single-network-based and dual-network-based. A single-network-based model uses one (usually hybridized) network to handle feature and relation extraction. These models focus on mining the basic representation hierarchy of data and the significant connections within the hierarchy, and include ConvTimeNet [20], InceptionTime [12], and OmniScale 1-dimensional convolutional neural network (OS-CNN) [41]. A dual-network-based model is composed of a local-feature extraction network and a relation extraction network in parallel. The first network, usually convolutional structure-based, concentrates on local features. In contrast, the second network focuses on relationships among the features extracted. Examples of the second network include the Transformer-based model [16] and ALSTM-FCN [17]. Compared with feature extraction, relation extraction aims at capturing the hidden connections among the previously extracted features. In other words, a relation extraction network is able to compensate for the loss of the representations ignored by its corresponding feature extraction network. Hence, it is vital to design an effective relation extraction network for different applications [11,16]. Currently, attention and long short-term memory (LSTM) networks are widely used for relation extraction in time series classification. This is because the attention mechanism can relate different positions of a sequence to derive the relationships at certain positions [43] and LSTM is able to explore long- and short-period dependencies in the data, both of which help to enhance relation extraction [17,18].

Currently, hybridizing attention and LSTM networks for relation extraction has attracted increasingly more research attention in time series classification. Cascading and embedding models are the two main ways to combine them. A cascading model stacks attention and LSTM networks one after another to realize some specific functions. Neither attention or LSTM networks require significant changes in their structures, and the attention LSTM (AttLSTM) model is an example of this [17,18]. The cascading models usually suffer from two drawbacks. First, almost all attention networks are based on fully connected networks, which are insensitive to the intricate features hidden in the data. Second, the useful representations previously extracted are easily lost as they are processed through subsequent networks. An embedding model integrates attention and LSTM networks in a compact manner. Fewer features are lost during data transmission with fewer layers of neural networks cascaded. Thus, more useful features and relationships are available to be utilized by the model. This type of model is sensitive to regular and periodic data and hence able to concentrate on the local and periodical variations of data, for example, an LSTM with trend attention gate (LSTMTAG) [26]. An embedding model may not be aware of the global variations of non-periodical data if not designed properly, especially when handling long univariate and multivariate datasets. Theoretically speaking, embedding LSTM networks into an attention structure helps it to attain significantly more temporal features for calculation, improving its sensitivity to the global variations of non-periodical data when compared with fully connected networks. Unfortunately, this type of structure has not been considered in the time series data mining community.

To take advantage of the dual-network-based model and the hybridization of attention and LSTM networks, we propose a robust temporal feature network (RTFN) for feature extraction in the area of time series classification. The RTFN consists of a temporal feature network (TFN) as its local-feature extraction network and an LSTM-based attention network as its relation extraction network. Our main contributions are summarized below.

- The temporal feature network is a CNN-based residual structure, responsible for extracting sufficient local features. Multi-head CNN layers are used to diversify multi-scale features, and self-attention is adopted to relate different positions of the previously extracted features. In addition, we use the leaky rectified linear unit (LeakReLU) as the activation function to reduce the loss of features during their transmission.

- The LSTM-based attention network contains two identical layers. In each layer, LSTM networks are used instead of fully connected networks to obtain the query, key, and value matrices for their corresponding attention structure. Unlike the existing structures that combine attention and LSTM networks, the LSTM-based attention network can focus on the global variations of non-periodical data, which helps to mine useful relationships among the features already learned.
- The RTFN is embedded into a supervised structure and tested on 85 UCR2018 datasets and 30 UEA2018 datasets. The RTFN-based algorithm outperforms a number of existing supervised algorithms on 39 UCR2018 and 15 UEA2018 datasets. In addition, we embed the RTFN into a simple unsupervised clustering as an encoder. Our structure wins 9 out of 36 UCR2018 datasets when compared with 13 unsupervised algorithms.

The rest of the paper is organized as follows. Section 2 reviews the state-of-the-art deep learning algorithms for time series classification and various combinations of attention and LSTM networks. Section 3 introduces the overview of the RTFN, its key components, the RTFN-based supervised structure, and RTFN-based unsupervised clustering. Experimental results are provided in Section 4, and Section 5 concludes the paper.

2. Related work

This section first reviews the deep learning algorithms for time series classification and then discusses the existing means to hybridize attention and LSTM networks.

2.1. Deep learning algorithms

Since the introduction of the fully convolutional network (FCN) [44], increasingly more algorithms have been proposed to address time series classification problems [11]. In general, these algorithms are either single-network-based or dual-network-based. Single-network-based algorithms focus on significant features of data. For example, a 34-layer CNN was constructed to handle the ECG classification problem [34]. Serrá et al. developed a universal encoder based on CNN and convolutional attention to mine the temporal representations from input data [38]. In [20], an off-the-shelf deep CNN (ConvTimeNet) with four convolutional blocks was proposed as a transferable network to quickly adapt to the requirements of datasets. Fawaz et al. used a fast gradient sign method to fool a ResNet model, called adversarial attacks for time series classification, where a set of synthetic samples was generated [10]. InceptionTime [12] and OS-CNN [41] are often regarded as two representative single-network-based models, achieving decent performance on many univariate time series datasets. InceptionTime uses an inception structure to explore multi-scale representations from data, and OS-CNN adopts a 1-dimensional CNN to mine local features and the relationships among the data. Conversely, dual-network-based models have not yet received much research attention as an emerging trend for time series classification. A few LSTM-FCN-based models were designed to cope with univariate and multivariate time series classification problems [17,18], where FCN and LSTM were used for feature and relation extraction, respectively. In [16], Huang et al. proposed a residual attention net consisting of a ResNet-based feature network and a transformer-based relation network, obtaining promising performance on UCR datasets. The dual-network-based algorithms usually achieved better classification performance than single-network-based algorithms [16–18].

2.2. Hybridization of attention and LSTM

Hybridizing attention and LSTM models is an emerging solution to temporal and spatial relation extraction. The existing works mainly include cascading and embedding models. The former simply stacks the attention and LSTM structures together. For example, an attention-LSTM model was adopted to cope with univariate and multivariate time series classification on UCR and UEA datasets, respectively [17,18]. An attention-LSTM model was integrated into convolution–deconvolution word embedding to merge context-specific and task-specific information [40]. In [48], an attention-based time-incremental CNN cascaded attention and LSTM networks for temporal and spatial information fusion of ECG signals. Additionally, the cascading models have been applied to scholarly venue recommendations [31], semantic relation extractions [14], software defined networks [5], and so on. Alternatively, the embedding models focus on the compact integration of attention and LSTM networks. For example, a TAG-embedded LSTM model was devised to explore the local variations of quasi-periodic time series data [26]. Wang et al. [50] proposed a novel online attentional recurrent neural network model for video tracking, where inter- and intra-attention models were embedded into a bi-directional LSTM to distinguish different background scenarios. In [6], Chen et al. proposed an identity-aware single shot multi-boxes detector for object detection, using an attention-embedded LSTM structure to locate positions of interesting objects.

2.3. Analysis and motivation

The dual-network-based models realize feature and relation extraction by using two separate networks in parallel. This type of model usually performs better than a single-network-based model in supervised classification and unsupervised clustering, according to several references [16–18] and our observations in Section 4.4. Nevertheless, designing a

dual-network-based model is quite challenging because its structure should meet dataset requirements, especially its relation extraction network. The hybridization of attention and LSTM offers a promising means to discover the relationships among the representations obtained from data. However, the existing cascading and embedding models cannot adequately handle the global variations of non-periodical time series data. This motivated us to design a dual-network-based algorithm for time series classification and was the reason why we embedded LSTM into an attention structure for relation extraction.

3. RTFN

This section first provides an overview of the structure of the RTFN and then describes the convolutional neural block and LSTM-based attention layer components. Finally, the RTFN-based supervised structure and unsupervised clustering are introduced.

3.1. Overview

The structure of the RTFN primarily consists of a TFN and LSTM-based attention network (LSTMaN), as shown in Fig. 1. In the TFN, a convolutional neural block, namely 'Conv1D', is seen as the basic building block responsible for capturing the local features from the input. Two multi-head convolutional neural layers, each consisting of four Conv1D blocks, are used to discover higher-level multi-scale features from the previously extracted lower-level features. A self-attention layer [43] is placed between the two multi-head layers to relate the positions of the local features obtained by the first multi-head layer, which enriches the input features of the second multi-head layer. Detailed observations can be found in Section 4.3. LSTMaN is composed of two LSTM-based attention layers, aiming to determine the intrinsic relationships among the features learned from the input. This helps to compensate for the loss of the representations ignored by the TFN. The TFN and LSTMaN are used as the local-feature and relation extraction networks, respectively. Combining them provides the RTFN with sufficient features and relationships. We present the RTFN embedded into a supervised structure in Section 3.4 and into an unsupervised clustering in Section 3.5.

3.2. Convolutional Neural Block (Conv1D)

A Conv1D block consists of a 1-dimensional CNN module, batch normalization module, and *LeakReLU* activation function [35], defined as:

$$O_{Conv1D} = f_{LeakReLU}(f_{BN}(f_{conv}(x))) \tag{1}$$

where, O_{Conv1D} and x are the output and input of the Conv1D block, respectively. $f_{LeakReLU}$, f_{BN}, and f_{conv} denote the *LeakReLU* activation, batch normalization, and CNN functions, respectively. The CNN module is used to explore the local features from the input, defined as:

$$f_{conv}(x) = W_{cnn} \otimes x + b_{cnn} \tag{2}$$

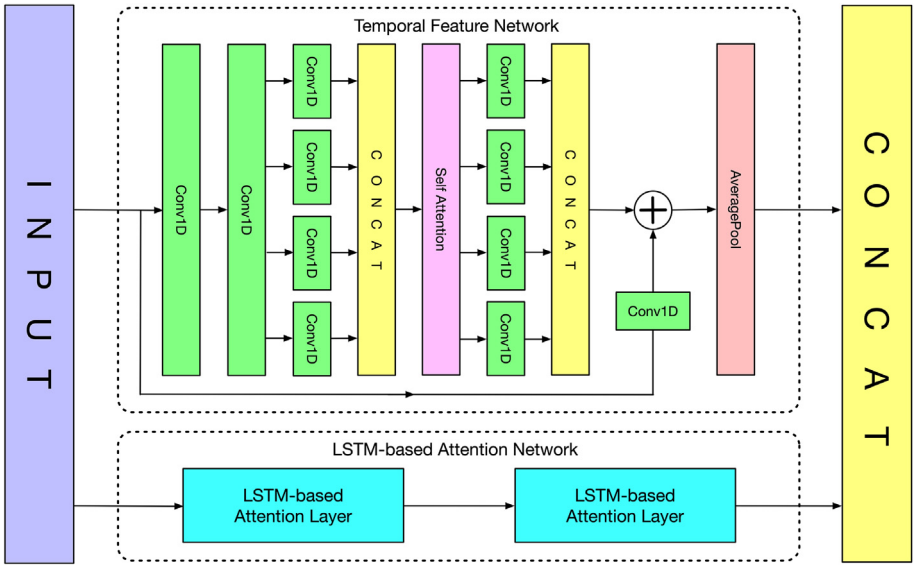

Fig. 1. Structure of RTFN.

where, W_{cnn} and b_{cnn} are the weight and bias matrices of CNN, respectively. $\otimes$ is the convolutional computation operation.

Let $x_{bn} = \{a_1, a_2, \ldots, a_N\}$ be the input of the batch normalization module, where a_i and N are the i-th instance and the batch size, respectively. Let $\mu = \frac{1}{N}\sum_{i=1}^{N} a_i$ and $\delta = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(a_i - \mu)^2}$ denote the mean and standard deviation of x_{bn}, respectively. $f_{BN}(x_{bn})$ is defined in Eq. (3).

$$f_{BN}(a_1, a_2, \ldots, a_N) = \left(\gamma\frac{a_1 - \mu}{\delta + \epsilon} + \beta, \gamma\frac{a_2 - \mu}{\delta + \epsilon} + \beta, \ldots, \gamma\frac{a_N - \mu}{\delta + \epsilon} + \beta\right) \tag{3}$$

where, $\gamma \in \mathbb{R}^+$ and $\beta \in \mathbb{R}$ are the parameters to be learned during training and $\epsilon > 0$ is an arbitrarily small number.

The batch normalization module eliminates the internal covariate shift and thus ensures a faster training process. In addition, it regularizes the proposed model and enhances its local-feature extraction ability in supervised classification and unsupervised clustering. *LeakReLU* processes the positive and negative numbers and reduces the loss of features during the data transmission process, unlike the rectified linear unit (*ReLU*) that only considers positive numbers. Detailed observations are shown in Section 4.3. The *LeakReLU* activation is defined as:

$$f_{LeakReLU}(x_{actv}) = \begin{cases} \alpha x_{actv}, & x_{actv} < 0 \\ x_{actv}, & x_{actv} \geqslant 0 \end{cases} \tag{4}$$

where, x_{actv} is the input of the *LeakReLU* unit and α is a coefficient for negative numbers. Following the widely recognized YOLOv3 [35], we set $\alpha = 0.1$ in this paper.

3.3. LSTM-based attention layer

As aforementioned, LSTMaN is proposed for relation extraction and includes two LSTM-based attention layers. The two layers have the same structure, as shown in Fig. 2, where 'MatMul' is a matrix multiplication operation. The first layer is used to extract basic relationships from the input, while the second layer is responsible for mining the intricate connections among them. The second layer helps to extract more complex regularizations hidden in the data than the first layer by extending the details of the previously obtained relationships.

An LSTM-based attention layer incorporates LSTM networks into an attention structure, unlike the existing models that hybridize attention and LSTM networks (see Section 2.2). In this network, a temporal query and a set of key-value pairs are mapped to an output. Query, Key, and Value are matrices obtained from the feature extraction by the LSTM networks. The output is defined as a weighted sum of the values with sufficient representations. In this study, each value is obtained by a compatibility function that mines the hidden relationships between a query and its corresponding key that already carries basic features. The query, key, and value matrices output by the three LSTM networks, I_q, I_k, I_v, are defined in Eqs. ()()()(5)–(7), respectively.

$$I_q = f_{LSTM-Q}(x) \tag{5}$$

$$I_k = f_{LSTM-K}(x) \tag{6}$$

$$I_v = f_{LSTM-V}(x) \tag{7}$$

where, x is the input of the layer and f_{LSTM-Q}, f_{LSTM-K} and f_{LSTM-V} are the LSTM functions for obtaining the query, key, and value matrices, respectively.

Note that each of the three LSTM networks involves the same computational procedure at time step t as a traditional LSTM network. The following describes the computation operations in an LSTM network. Let x_t and h_t be the input vector and the hidden state vector of the LSTM network at t, respectively. Let g_t^u, g_t^f, g_t^o, and g_t^c are the activation vectors of the input, forget, output, and cell state gates at t, respectively. Denote the elementwise multiplication by $\odot$. Let σ and *tanh* denote the logistic sigmoid and hyperbolic tangent functions, respectively. Let W_{ux}, W_{fx}, W_{ox}, and W_{cx} be the weight matrices of x_t at gates g_t^u, g_t^f, g_t^o, and g_t^c, respectively. Let W_{uh}, W_{fh}, W_{oh}, and W_{ch} denote the weight matrices of h_t at gates g_t^u, g_t^f, g_t^o, and g_t^c,

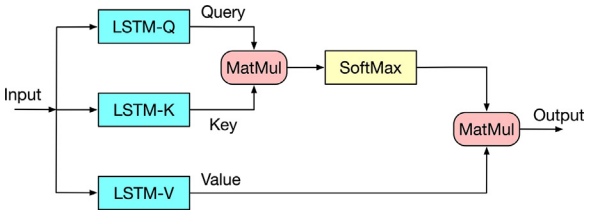

Fig. 2. Structure of the LSTM-based attention layer.

respectively. Let b_u, b_f, b_o, and b_c be the bias matrices of h_t at gates g_t^u, g_t^f, g_t^o, and g_t^c, respectively. g_t^u, g_t^f, g_t^o, g_t^c, and h_t are defined in Eqs. (8)–(12), respectively.

$$g_t^u = \sigma(W_{ux}x_t + W_{uh}h_{t-1} + b_u) \tag{8}$$

$$g_t^f = \sigma(W_{fx}x_t + W_{fh}h_{t-1} + b_f) \tag{9}$$

$$g_t^o = \sigma(W_{ox}x_t + W_{oh}h_{t-1} + b_o) \tag{10}$$

$$g_t^c = g_t^f \odot g_{t-1}^c + g_t^u \odot tanh(W_{cx}x_t + W_{ch}h_{t-1} + b_c) \tag{11}$$

$$h_t = g_t^o \odot tanh(g_t^c) \tag{12}$$

After x goes through the LSTM networks, I_q, I_k, and I_v carry sufficient long- and short-term features. They are then fed into the attention structure, and its output matrix, O_{Att}, is defined in Eq. (13).

$$O_{Att} = f_{SoftMax}\left(I_q \cdot I_k^T\right) \cdot I_v \tag{13}$$

where, $f_{SoftMax}$ is a commonly used function to compute the possibilities of a certain matrix, and I_k^T is the transpose of I_k.

Note that we concatenate the local features obtained by TFN and the global relationships obtained by LSTMaN in RTFN. Let O_{TFN} and O_{LSTMaN} denote the output matrices of TFN and LSTMaN, respectively. The output of RTFN, O_{RTFN}, is defined in Eq. (14).

$$O_{RTFN} = f_{concat}([O_{TFN}, O_{LSTMaN}]) \tag{14}$$

where, f_{concat} is the CONCAT function.

3.4. RTFN-based supervised structure

The RTFN-based supervised structure is shown in Fig. 3, where a dropout layer and fully-connected layer are cascaded to the output of the RTFN. Specifically, we introduce the dropout layer to avoid overfitting during the training process. The fully-connected layer operates as the classifier. The dropout and fully-connected layers are used because the features extracted by the RTFN are sufficient. Thus, a complicated classifier network is not necessary. Similar to other commonly used supervised algorithms [11,16,18], we use the cross-entropy function to compute the average difference between the ground truth labels and their corresponding prediction results, $\mathcal{L}_{super}$, written as:

$$\mathcal{L}_{super} = -\frac{1}{n}\sum_{i=1}^{n}\left(\widehat{Y}_i^{train} log(p_i)\right) \tag{15}$$

where, n is the number of samples, and $\widehat{Y}_i^{train}$ and p_i, $i = 1, 2, \ldots, n$, are the ground truth label of the i-th sample and its corresponding prediction output, respectively.

3.5. RTFN-based Unsupervised Clustering

The RTFN-based unsupervised clustering is based on a widely adopted unsupervised auto-encoder structure, as shown in Fig. 4. It is primarily composed of an RTFN-based encoder, a decoder, and a K-means algorithm. The RTFN is responsible for

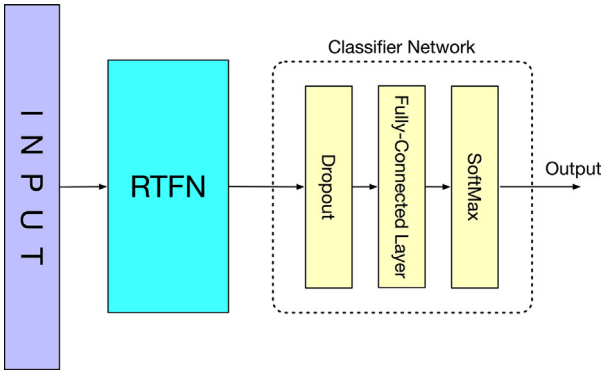

Fig. 3. RTFN-based supervised structure.

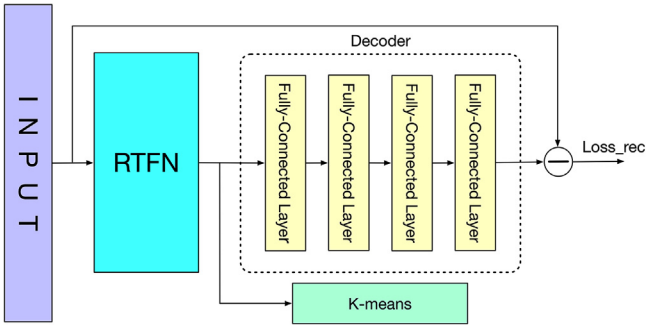

Fig. 4. RTFN-based unsupervised clustering.

obtaining as many useful representations from the input as possible. The decoder is made up of four fully-connected layers, helping to reconstruct the features captured by the RTFN. The K-means algorithm acts as the unsupervised classifier.

Different from taking the k-means loss into account [15,27,30], the RTFN-based unsupervised clustering only depends on the reconstruction loss (i.e., the mean square error), $\mathcal{L}_{rec}$, as defined in Eq. (16).

$$\mathcal{L}_{rec} = \frac{1}{n}\sum_{i=1}^{n}\left(\breve{X}_i^{train} - X_i^{rec}\right)^2 \tag{15}$$

where $\breve{X}_i^{train}$ and X_i^{rec}, $i = 1, 2, \ldots, n$, are the input and the decoder output of the i-th sample, respectively.

No matter the supervised or unsupervised structure, our goal is to minimize its loss function, $\mathcal{L}$, by finding the optimal parameters, θ^*, where $\mathcal{L}(\theta^*)$ is infinitely close to 0. This paper uses the gradient-descent method to approximate θ^* for our proposed structure. Let θ_t and l_{rate} denote the parameters and the learning rate at the t-th training epoch, respectively. θ_t is updated by Eq. (17).

$$\theta_t = \theta_{t-1} - l_{rate}\nabla_{\theta_{t-1}}\mathcal{L}(\theta_{t-1}) \tag{16}$$

where, $\nabla_{\theta_{t-1}}$ represents the gradient at the $(t-1)$-th training epoch.

4. Experiments and Analysis

This section first introduces the experimental setup and performance metrics and then focuses on the ablation study. Finally, the RTFN-based supervised structure and unsupervised clustering are evaluated.

4.1. Experimental setup

Extensive experiments were conducted in supervised classification and unsupervised clustering. This section first introduces the standard datasets, followed by the implementation details.

Supervised Classification Datasets. A univariate time series refers to a series of time-ordered data points associated with a time-dependent variable. This type of sequence contains local and global patterns of data. Local patterns show significant changes in the data, and global patterns reflect the overall trend of the data. A multivariate time series consists of multiple concurrent univariate time series, each associated with a time-dependent variable. A multivariate time series contains local and global pattern information, similar to a univariate time series. A multivariate time series also contains relationship information between variables because each variable has some dependency on other variables. Time series data is labeled in the supervised area. A supervised algorithm learns the characteristics of the input data and maps the data to the labels. For a univariate time series, a supervised algorithm needs to mine the local and global patterns from the data, such as in the EE [24], COTE [11], LPS [3], LCE [9], ConvTime [20], and ResNet-Transformer [16] models. For a multivariate time series, a supervised algorithm focuses on the local and global patterns of each variable as well as the relationships between variables, such as in the SMTS [2], mv-ARF [42], WEASEL + MUSE [37], TapNet [49], and FCN-MLSTM [18] models.

We evaluated the performance of the RTFN-based supervised structure using a number of univariate and multivariate time series datasets. We used UCR2018[4] for the univariate time series, which is one of the authoritative data archives and contains 128 datasets with different lengths in a variety of application areas. We selected 85 standard datasets from the UCR2018 archive, consisting of 65 'short-medium' and 20 'long' time series datasets. In this study, a 'long' dataset is a dataset with a length of over 500. The details of these datasets are shown in Table 1. As for the multivariate time series, UEA2018[5] is a com-

[4] http://www.timeseriesclassification.com.
[5] http://www.timeseriesclassification.com/dataset.php.

Table 1
Details of 85 univariate time series datasets. Those marked with 'YES' are also used for unsupervised clustering experiments.

Dataset	TrainSize	TestSize	Classes	SeriesLength	Type	Unsupervised
Adiac	390	391	37	176	Image	
ArrowHead	36	175	3	251	Image	YES
Beef	30	30	5	470	Spectro	YES
BeetleFly	20	20	2	512	Image	YES
BirdChicken	20	20	2	512	Image	YES
Car	60	60	4	577	Sensor	YES
CBF	30	900	3	128	Simulated	
ChlorineConcentration	467	3840	3	166	Sensor	YES
CinCECGTorso	40	1380	4	1639	Sensor	
Coffee	28	28	2	286	Spectro	YES
CricketX	390	390	12	300	Motion	
CricketY	390	390	12	300	Motion	
CricketZ	390	390	12	300	Motion	
DiatomSizeReduction	16	306	4	345	Image	YES
DistalPhalanxOutlineAgeGroup	400	139	3	80	Image	YES
DistalPhalanxOutlineCorrect	600	276	2	80	Image	YES
Earthquakes	322	139	2	512	Sensor	
ECG200	100	100	2	96	ECG	YES
ECG5000	500	4500	5	140	ECG	
ECGFiveDays	23	861	2	136	ECG	YES
FaceFour	24	88	4	350	Image	
FacesUCR	200	2050	14	131	Image	
FordA	3601	1320	2	500	Sensor	
FordB	3636	810	2	500	Sensor	
GunPoint	50	150	2	150	Motion	YES
Ham	109	105	2	431	Spectro	YES
HandOutlines	1000	370	2	2709	Image	
Haptics	155	308	5	1092	Motion	
Herring	64	64	2	512	Image	YES
InlineSkate	100	550	7	1882	Motion	
InsectWingbeatSound	220	1980	11	256	Sensor	
ItalyPowerDemand	67	1029	2	24	Sensor	
Lightning2	60	61	2	637	Sensor	YES
Lightning7	70	73	7	319	Sensor	
Mallat	55	2345	8	1024	Simulated	
Meat	60	60	3	448	Spectro	YES
MedicalImages	381	760	10	99	Image	
MiddlePhalanxOutlineAgeGroup	400	154	3	80	Image	YES
MiddlePhalanxOutlineCorrect	600	291	2	80	Image	YES
MiddlePhalanxTW	399	154	6	80	Image	YES
MoteStrain	20	1252	2	84	Sensor	YES
OliveOil	30	30	4	570	Spectro	
OSULeaf	200	242	6	427	Image	YES
Plane	105	105	7	144	Sensor	YES
ProximalPhalanxOutlineAgeGroup	400	205	3	80	Image	YES
ProximalPhalanxOutlineCorrect	600	291	2	80	Image	
ProximalPhalanxTW	400	205	6	80	Image	YES
ShapeletSim	20	180	2	500	Simulated	
ShapesAll	600	600	60	512	Image	
SonyAIBORobotSurface1	20	601	2	70	Sensor	YES
SonyAIBORobotSurface2	27	953	2	65	Sensor	YES
Strawberry	613	370	2	235	Spectro	
SwedishLeaf	500	625	15	128	Image	YES
Symbols	25	995	6	398	Image	YES
SyntheticControl	300	300	6	60	Simulated	
ToeSegmentation1	40	228	2	277	Motion	YES
ToeSegmentation2	36	130	2	343	Motion	YES
Trace	100	100	4	275	Sensor	
TwoPatterns	1000	4000	4	128	Simulated	YES
TwoLeadECG	23	1139	2	82	ECG	YES
UWaveGestureLibraryAll	896	3582	8	945	Motion	
UWaveGestureLibraryX	896	3582	8	315	Motion	
UWaveGestureLibraryY	896	3582	8	315	Motion	
UWaveGestureLibraryZ	896	3582	8	315	Motion	
Wafer	1000	6164	2	152	Sensor	YES
Wine	57	54	2	234	Spectro	YES
WordSynonyms	267	638	25	270	Image	YES
ACSF1	100	100	10	1460	Device	

Table 1 (*continued*)

Dataset	TrainSize	TestSize	Classes	SeriesLength	Type	Unsupervised
BME	30	150	3	128	Simulated	
Chinatown	20	345	2	24	Traffic	
Crop	7200	16800	24	46	Image	
DodgerLoopDay	78	80	7	288	Sensor	
DodgerLoopGame	20	138	2	288	Sensor	
DodgerLoopWeekend	20	138	2	288	Sensor	
GunPointAgeSpan	135	316	2	150	Motion	
GunPointMaleVersusFemale	135	316	2	150	Motion	
GunPointOldVersusYoung	136	315	2	150	Motion	
InsectEPGRegularTrain	62	249	3	601	EPG	
InsectEPGSmallTrain	17	249	3	601	EPG	
MelbournePedestrian	1194	2439	10	24	Traffic	
PowerCons	180	180	2	144	Power	
Rock	20	50	4	2844	Spectrum	
SemgHandGenderCh2	300	600	2	1500	Spectrum	
SemgHandMovementCh2	450	450	6	1500	Spectrum	
SemgHandSubjectCh2	450	450	5	1500	Spectrum	
SmoothSubspace	150	150	3	15	Simulated	
UMD	36	144	3	150	Simulated	

Table 2
Details of 30 multivariate time series datasets. Abbreviations: AS - Audio Spectra, ECG - Electrocardiogram, EEG - Electroencephalogram, HAR - Human Activity Recognition, MEG - Magnetoencephalography.

Index	Dataset	TrainSize	TestSize	NumDimensions	SeriesLength	Classes	Type
AWR	ArticularWordREcognition	275	300	9	144	25	Motion
AF	AtrialFibrillation	15	15	2	640	3	ECG
BM	BasicMotions	40	40	6	100	4	HAR
CT	CharacterTrajectories	1422	1436	3	182	20	Motion
CR	Critcket	108	72	6	1197	12	HAR
DDG	DuckDuckGeese	50	50	1345	270	5	AS
EW	EigenWorm	128	131	6	17894	4	Motion
EP	Epliyepsy	137	138	3	206	4	HAR
EC	EthanolConcentration	261	263	3	1751	4	HAR
ER	ERing	30	270	4	65	6	Other
FD	FaceDetection	5890	3524	144	62	2	EEG/MEG
FM	FingerMovements	316	100	28	50	2	EEG/MEG
HMD	HandMovementDirection	160	74	10	400	4	EEG/MEG
HW	Handwriting	150	850	3	152	26	HAR
HB	Heartbeat	204	205	61	405	2	AS
IW	InsectionWingbeat	30000	20000	200	30	10	AS
JV	JapaneseVowels	270	370	12	29	9	AS
LIB	Libras	180	180	2	45	15	HAR
LSST	LSST	2459	2466	6	36	14	Other
MI	MotorImagery	278	100	64	3000	2	EEG/MEG
NATO	NATOPS	180	180	24	51	6	HAR
PD	PenDigits	7494	3498	2	8	10	EEG/MEG
PEMS	PEMSF	267	173	963	144	7	EEG/MEG
PS	Phoneme	3315	3353	11	217	39	AS
RS	RacketSports	151	152	6	30	4	HAR
SRS1	SelfRegulationSCP1	268	293	6	896	2	EEG/MEG
SRS2	SelfRegulationSCP2	200	180	7	1152	2	EEG/MEG
SAD	SpokenArbicDigits	6599	2199	13	93	10	AS
SWJ	StandWalkJump	12	15	4	2500	3	ECG
UW	UWaveGuestureLibrary	120	320	3	315	8	HAR

monly used data archive, including 30 datasets in seven application areas, i.e., audio spectra, electrocardiogram, electroencephalogram, human activity recognition, motion, eagnetoencephalography and other. Their details are listed in Table 2.

Unsupervised Clustering Datasets. Time series data is not labeled in the unsupervised area. An unsupervised algorithm discovers the previously undetected patterns in a dataset. For a univariate time series, an unsupervised algorithm aims to learn the underlying structure or distribution in the data, such as in the DEC [46], DTCR [27], IDEC [15], and DTC [30] models.

For a multivariate time series, an unsupervised algorithm learns the underlying structure or distribution of each variable as well as the relationships between the variables, as in the USAD model [1].

Following the protocol used in [15,27,30], we verified the performance of the RTFN-based unsupervised clustering with 36 standard datasets selected from the UCR2018 archive. These datasets are marked with 'YES' in the 'Unsupervised' column in Table 1.

Implementation Details. First, we introduce the parameter settings for TFN. As mentioned in Section 3.2, each Conv1D block contains a 1-dimensional CNN module. The 1-dimensional CNN module also has 128 channels in the Conv1D block that are directly connected to the residual junction, each with a kernel size of 1. Each of the four 1-dimensional CNN modules has 32 channels in each multi-head Conv1D layer. Motivated by InceptionTime [12], we set the kernel sizes of the four CNN modules to 5, 8, 11, and 17. Following the previous works in [16,35], we set the decay value of the batch normalization module to 0.9, which helps to accelerate the training by reducing the internal covariate shift of the time series data.

Each 1-dimensional CNN module has 128 channels with a kernel size of 11 in the two Conv1D blocks next to the input of the RTFN. The following explains why 11 was chosen. As references [12,16,17,20,41,44] suggest, we chose 3, 5, 7, 9, 11, and 13 as the candidate kernel sizes for the two ConvlD blocks above. We selected eight univariate and four multivariate datasets from the UCR2018 and UEA2018 archives, respectively. The eight univariate datasets contained four 'short-medium' and four 'long' time series datasets. Table 3 shows the top-1 accuracy results for the six different kernel sizes used in the two Conv1D blocks. A larger kernel size usually resulted in higher accuracies because it has a broader receptive field and thus captures richer local features. Clearly, kernel sizes 11 and 13 result in better performance than 3, 5, 7, and 9. Kernel sizes 11 and 13 lead to similar accuracy results on each dataset. However, a larger kernel size means the corresponding convolutional operations consume more computing resources. To compromise between accuracy and complexity, we set the kernel size of the two Conv1D blocks to 11.

Secondly, we introduce the parameter settings for LSTMaN. As described in Section 3.3, there are two LSTM-based attention layers. In each layer, as references [17,18] suggest, we set the number of hidden units in each LSTM network to 128. Finally, we dynamically adjusted the learning rate during the training process. The total number of training epochs and size of each decay period are denoted by N_{tot} and N_{dec}, respectively. Let $l_{rate}(j), j = 1, 2, \ldots, J$, denote the learning rate of the $j-th$ decay period, where $J = \lceil N_{tot}/N_{dec} \rceil - 1$. Its definition is written in Eq. (18).

$$l_{rate}(j) = (1 - d_{rate}) \times l_{rate}(j-1) \tag{17}$$

where d_{rate} and J are the decay rate of l_{rate} and the total number of the decay periods, respectively. In this paper, we set $l_{rate}(0) = 0.01$ and $d_{rate} = 0.1$. Once l_{rate} is smaller than 0.0001, we fix it to 0.0001. The RMSPropOptimizer of Tensorflow[6] was used to tune the parameters of our proposed RTFN structure for supervised classification and unsupervised clustering. According to [36], we set the RMSPropOptimizer's momentum term to 0.9 to avoid falling into local minima during training. The dropout layer and L_2 regularization were used to avoid overfitting during the training process. As references [12,34,41,50] suggest, the dropout layer's ratio value was set to 0.5.

All experiments are run on a computer with Ubuntu 18.04 OS, an Nvidia GTX 1070Ti GPU with 8 GB, an Nvidia GTX 1080Ti GPU with 11 GB, and an AMD R5 1400 CPU with 16G RAM.

4.2. Performance metrics

To evaluate the performance of various algorithms in terms of supervised classification and unsupervised clustering, we adopted a number of well-known performance metrics, explained below.

Supervised Classification. Three metrics are used to rank different supervised algorithms in terms of the top-1 accuracy, including 'win'/'tie'/'lose', mean accuracy (MeanACC), and AVG_rank. To be specific, for an arbitrary algorithm, its 'win', 'tie', and 'lose' values indicate on how many datasets this algorithm performs better than, equivalent to, and worse than the others, respectively. For each algorithm, the 'best' value is the summation of its corresponding 'win' and 'tie' values, while the 'total' value is the total number of datasets tested. In addition, the AVG_rank score measures the average difference between the accuracy values of a model and the best accuracy values among all models [11,9,29,16].

Unsupervised Clustering. Note that the top-1 accuracy is not applicable to unsupervised clustering. Instead, we use a widely adopted performance indicator, the rand index (RI) [27], *RI*, as defined in Eq. (16).

$$RI = \frac{PTP + NTP}{s(s-1)/2} \tag{18}$$

where *PTP* and *NTP* are the numbers of the positive and negative time series pairs in the clustering, respectively, and s is the dataset size. Besides, we denote the average RI value of a certain algorithm by 'AVG RI'.

[6] https://tensorflow.google.cn/.

Table 3
Results of the top-1 accuracy vs. the kernel size.

Type	Dataset	Kernel size = 3	Kernel size = 5	Kernel size = 7	Kernel size = 9	Kernel size = 11	Kernel size = 13
Univariate Time Series	ECG200	0.9	0.91	0.91	0.92	0.92	**0.93**
	Lighting7	0.863014	0.876712	0.890411	**0.90411**	**0.90411**	**0.90411**
	Ham	0.761905	0.761905	0.780952	0.780952	**0.809524**	**0.809524**
	Wine	0.87037	0.87037	0.888889	0.888889	**0.907407**	**0.907407**
	SemgHandGenderCh2	0.876667	0.96	0.91	0.913333	**0.923333**	**0.923333**
	SemgHandMovementCh2	0.6	0.616667	0.66667	0.716667	**0.757778**	**0.757778**
	SemgHandSubjectCh2	0.788889	0.816667	0.833333	0.85	**0.897778**	**0.897778**
	Rock	0.84	0.84	0.84	0.86	**0.88**	**0.88**
	MeanACC	0.812606	0.831540	0.840032	0.854244	0.874991	**0.876241**
Multivariate Time Series	AF	0.467	0.467	**0.533**	**0.533**	**0.533**	**0.533**
	FD	0.629	0.631	0.649	0.656	**0.67**	**0.67**
	HMD	0.649	0.649	**0.662**	**0.662**	**0.662**	**0.662**
	HB	0.727	0.748	0.751	0.751	**0.785**	**0.785**
	MeanACC	0.618	0.624	0.649	0.651	**0.663**	**0.663**

4.3. Ablation study

As shown in Fig. 1, RTFN mainly consists of a temporal feature network for local-feature extraction, i.e., TFN, and an LSTM-based attention network for relation extraction, i.e., LSTMaN.

4.3.1. Temporal Feature Network

TFN is featured with *LeakReLU*-based activation and self-attention. To study the effectiveness of the two components, we compare three TFN variants listed below.

- TFN: the proposed TFN, where *LeakReLU* and self-attention are used.
- TFN w *ReLU*: TFN with *ReLU* instead of *LeakReLU*.
- TFN w/o *SelAtt*: TFN without the self-attention layer.

We selected 12 datasets from the UCR2018 and UEA2018 archives for the performance comparison of supervised classification, including eight univariate and four multivariate datasets. These datasets are shown in Table 3. We selected four univariate datasets from the UCR2018 archive for the performance comparison of unsupervised clustering.

The top-1 accuracy and RI results obtained with different supervised and unsupervised algorithms are shown in Tables 4 and 5, respectively. It is easily observed that TFN outperforms TFN w *ReLU* on each dataset for supervised classification or unsupervised clustering. For example, the top-1 accuracy values of TFN and TFN w *ReLU* are 0.833333 and 0.611111, respectively. Unlike *ReLU* that focuses on positive numbers only, *LeakReLU* makes use of positive and negative numbers, helping to avoid the loss of the extracted features during their transformation. Thus, *LeakReLU* can mine more local features from the input, and the supervised and unsupervised performance of TFN is improved when compared with *ReLU*. We then compared the TFN and TFN w/o *SelAtt* in terms of supervised classification and unsupervised clustering. The TFN outperformed the TFN w/o *SelAtt* on all the datasets because the *SelAtt* layer can relate different positions of time series data, enriching the extracted features. It is clear that embedding the *SelAtt* layer in the TFN helps to enhance its supervised and unsupervised performance. Therefore, *LeakReLU* and self-attention are necessary for the TFN.

4.3.2. The LSTM-based attention network

In RTFN, LSTMaN consists of two LSTM-based attention layers. To investigate the effectiveness of LSTMaN, we compare five RTFN structures with the following relation-extraction components.

- 1LSTMaL: one LSTM-based attention layer.
- 2LSTMaL: two LSTM-based attention layers, i.e. the proposed LSTMaN.
- 3LSTMaL: three LSTM-based attention layers.
- AttLSTM: a cascading attention-LSTM model, where attention and LSTM layers simply pile up [18].
- LSTMTAG: an embedding attention-LSTM model, where a trend attention gate is embedded into an LSTM structure [26].

To make a fair comparison, the TFN is used in each RTFN structure as the local-feature extraction network. In other words, the corresponding RTFN structures are exactly the same for supervised classification or unsupervised clustering, except for their relation-extraction components.

Table 4
The top-1 accuracy results of different supervised algorithms on 12 selected datasets.

Type	Dataset	SeriesLength	TFN	TFN w *ReLU*	TFN w/o *SelAtt*	TFN + 1LSTMaL	TFN + 2LSTMaL	TFN + 3LSTMaL	TFN + AttLSTM	TFN + LSTMTAG
Univariate Time Series	ECG200	96	0.84	0.82	0.8	0.9	**0.92**	**0.92**	0.91	0.89
	Lighting7	319	0.821918	0.808219	0.808219	0.849315	**0.90411**	**0.90411**	0.849315	0.849315
	Ham	431	0.714286	0.619048	0.619048	0.761905	**0.809524**	**0.809524**	0.780952	0.761905
	Wine	234	0.833333	0.611111	0.555556	0.888889	**0.907407**	**0.907407**	0.888889	0.87037
	SemgHandGenderCh2	1500	0.796667	0.783333	0.651667	0.866667	**0.923333**	**0.923333**	0.91	0.84
	SemgHandMovementCh2	1500	0.595555	0.56	0.513333	0.611111	**0.757778**	**0.757778**	0.56	0.56
	SemgHandSubjectCh2	1500	0.793333	0.788889	0.74	0.8	**0.897778**	**0.897778**	0.873333	0.74
	Rock	2844	0.7	0.64	0.62	0.82	**0.88**	**0.88**	0.86	0.68
	MeanACC		0.761887	0.703825	0.663478	0.812236	**0.874991**	**0.874991**	0.829061	0.773949
Multivariate Time Series	AF	640	0.4	0.333	0.267	0.467	**0.533**	**0.533**	0.467	0.2
	FD	62	0.555	0.545	0.519	0.614	**0.67**	**0.67**	0.631	0.555
	HMD	400	0.544	0.481	0.5	0.649	**0.662**	**0.662**	0.649	0.649
	HB	405	0.547	0.535	0.515	0.761	**0.785**	**0.785**	0.727	0.547
	MeanACC		0.512	0.474	0.450	0.623	**0.663**	**0.663**	0.619	0.488

Table 5
The RI results of different unsupervised algorithms on four selected datasets.

Dataset	TFN	TFN w *ReLU*	TFN w/o *SelAtt*	TFN + 1LSTMaL	TFN + 2LSTMaL	TFN + 3LSTMaL	TFN + AttLSTM	TFN + LSTMTAG
Beef	0.6267	0.5945	0.5402	0.7034	**0.7655**	**0.7655**	0.7057	0.7034
Car	0.6418	0.6354	0.626	0.6708	**0.7169**	0.7028	0.6898	0.6418
ECG200	0.6533	0.6315	0.6018	0.7018	**0.7285**	**0.7285**	0.7018	0.7018
Lighting2	0.5373	0.5119	0.4966	0.5729	**0.6230**	**0.6230**	0.5770	0.5770
AVG RI	0.6148	0.5933	0.5662	0.6622	**0.7085**	0.7050	0.6686	0.6560

First, we studied the impact of the number of LSTM-based attention layers on the performance of the RTFN. Between TFN + 2LSTMaL and TFN + 1LSTMaL, the former always performed better than the latter. In the 2LSTMaL structure, the second layer revealed the details of the relationships among the features captured by the first layer and hence could discover the previously ignored complicated representations. Thus, 2LSTMaL mines more intricate relationships hidden in the data than 1LSTMaL. When comparing TFN + 2LSTMaL and TFN + 3LSTMaL, it is clear that the two achieve equivalent performance in almost all cases except for the 'Car' dataset, as illustrated in Tables 4 and 5. This is because the third layer is supposed to further extend the details of the relationships among those features extracted by the first and second layers in the 3LSTMaL structure. However, all intrinsic details are explicitly unveiled in the second layer. In this case, the third layer only acts as an information transmission layer. This layer may lead to loss of features during their transmission and consume additional computing resources, especially on complicated datasets. 2LSTMaL aims at striking a balance between accuracy and model complexity. To further support this, the model complexity comparison of different supervised algorithms on four 'long' time series datasets is shown in Table 6. It is clear that TFN + 2LSTMaL has a lower model complexity than TFN + 3LSTMaL, with CPU times of 32.421894 and 35.440211 s, respectively, on the dataset 'SemgHandGendeCh2'.

Second, we investigated the effectiveness of the proposed LSTMaN by comparing it with two well-recognized models based on attention and LSTM. TFN + 2LSTMaL beats TFN + AttLSTM and TFN + LSTMTAG on each dataset for supervised classification or unsupervised clustering. This is because AttLSTM lacks in-depth attention to the internal connections among the already extracted representations during their transmission, and thus insufficient features are mined from data. Meanwhile, LSTMTAG is able to focus on the local variations of periodical data due to the LSTM structure with TAG embedded. However, it is not sensitive to the global variations of non-periodical data, making it difficult to discover complex connections hidden in the data, especially for long univariate and multivariate datasets. For example, the top-1 accuracy values for TFN + LSTMTAG on the datasets 'Rock' and 'AF' are only 0.68 and 0.2, respectively. TFN + 2LSTMaL always obtains a higher accuracy value on each dataset for supervised classification and unsupervised clustering when compared with the TFN. This clearly demonstrates that LSTMaN plays an important role in performance improvement, because it extracts the intricate representations that may be ignored by the TFN. In other words, the LSTMaN and TFN complement each other in the RTFN.

4.4. Evaluation of the RTFN-based Supervised Structure

To evaluate the performance of the RTFN-based supervised structure, we compared it with a number of existing supervised algorithms against 'win'/'lose'/'tie', MeanACC, and AVG_rank on 85 univariate and 30 multivariate datasets, as seen in Table 1and 2.

4.4.1. Performance comparison on univariate time series

Table 7 shows the top-1 accuracy results obtained with different supervised algorithms on the 85 selected datasets in the UCR2018 archive. The existing SOTA represented the best algorithm on each dataset [16], including ConvTimeNet [20], EE

Table 6
Computational complexity comparison of TFN + 1LSTMaL, TFN + 2LSTMaL and TFN + 3LSTMaL in terms of the supervised classification. Abbreviations: M – Measured in Millions, s – Measured in Seconds.

Algorithm	Dataset	Parameters (M)	CPU only (s)	With GPU 1080Ti (s)	With GPU 1070Ti (s)
TFN + 1LSTMaL	SemgHandGendeCh2	2.546755	30.277891	1.506916	1.603434
TFN + 2LSTMaL		2.851651	32.421894	2.127678	2.410963
TFN + 3LSTMaL		3.262915	35.440211	2.893432	3.129045
TFN + 1LSTMaL	SemgHandMovementCh2	3.315015	21.771511	1.30519	1.537834
TFN + 2LSTMaL		3.620167	24.561916	1.865625	2.028537
TFN + 3LSTMaL		4.031431	26.660317	2.345234	2.834242
TFN + 1LSTMaL	SemgHandSubjectCh2	3.12295	20.742758	1.303414	1.529351
TFN + 2LSTMaL		3.428038	24.695192	1.864818	2.040713
TFN + 3LSTMaL		3.839302	26.723078	1.934696	2.783453
TFN + 1LSTMaL	Rock	4.747221	6.989958	1.293770	1.352977
TFN + 2LSTMaL		5.042245	8.887336	1.360002	1.370617
TFN + 3LSTMaL		5.453509	10.771745	1.636234	2.103425

Table 7
Results of different supervised algorithms on 85 selected datasets.

Dataset	Existing SOTA	USRL-FordA [13]	Inception-Time [12]	Combined (1NN) [13]	OS-CNN [41]	Best:fcn-lstm [17]	Vanilla:RN- Transformer [16,43]	ResNet- Transformer [16]	ResNet- Transformer2 [16]	ResNet- Transformer3 [16]	Ours
Adiac	0.857	0.76	0.841432	0.645	0.838875	**0.869565**	0.84399	0.849105	0.849105	0.849105	0.792839
ArrowHead	0.88	0.817	0.845714	0.817	0.84	**0.925714**	0.891429	0.891429	0.891429	0.891429	0.851429
Beef	**0.9**	0.667	0.7	0.6	0.83333	**0.9**	0.866667	0.866667	0.866667	0.866667	**0.9**
BeetleFly	0.95	0.8	0.8	0.8	0.8	**1**	**1**	0.95	0.95	**1**	**1**
BirdChicken	0.95	0.9	0.95	0.75	0.9	0.95	**1**	0.9	**1**	0.7	**1**
Car	0.933	0.85	0.883333	0.8	0.933333	**0.966667**	0.95	0.883333	0.866667	0.3	0.883333
CBF	**1**	0.988	0.998889	0.978	0.998889	0.996667	**1**	0.997778	**1**	**1**	**1**
ChlorineConcentration	0.872	0.688	0.876563	0.588	0.84974	0.816146	0.849479	0.863281	0.409375	0.861719	**0.894271**
CinCECGTorso	**0.9949**	0.638	0.853623	0.693	0.830435	0.904348	0.871739	0.656522	0.89058	0.31087	0.810145
Coffee	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
CricketX	0.821	0.682	**0.853846**	0.741	0.846154	0.792308	0.838462	0.8	0.810256	0.8	0.771795
CricketY	0.8256	0.667	0.851282	0.664	**0.869231**	0.802564	0.838462	0.805128	0.825641	0.808766	0.789744
CricketZ	0.8154	0.656	**0.861538**	0.723	**0.861538**	0.807692	0.820513	0.805128	0.128205	0.1	0.787179
DiatomSizeReduction	0.967	0.974	0.934641	0.967	0.980392	0.970588	0.993464	**0.996732**	0.379085	**0.996732**	0.980392
DistalPhalanxOutlineAgeGroup	**0.835**	0.727	0.733813	0.669	0.755396	0.791367	0.81295	0.776978	0.467626	0.776978	0.719425
DistalPhalanxOutlineCorrect	0.82	0.764	0.782608	0.683	0.771739	0.791367	**0.822464**	**0.822464**	**0.822464**	0.793478	0.771739
Earthquakes	0.801	0.748	0.741007	0.64	0.683453	**0.81295**	0.755396	0.755396	0.76259	0.755396	0.776978
ECG200	0.92	0.83	0.93	0.85	0.91	0.91	0.94	**0.95**	0.94	0.93	0.92
ECG5000	0.9482	0.94	0.940889	0.925	0.940222	**0.948222**	0.941556	0.943556	0.944222	0.940444	0.944444
ECGFiveDays	**1**	**1**	**1**	0.999	**1**	0.987224	**1**	**1**	**1**	**1**	**1**
FaceFour	**1**	0.83	0.954545	0.864	0.943182	0.943182	0.954545	0.965909	0.977273	0.215909	0.924045
FacesUCR	0.958	0.835	**0.97122**	0.86	0.96439	0.941463	0.957561	0.947805	0.926829	0.95122	0.95122
FordA	0.9727	0.927	0.961364	0.863	0.958333	**0.976515**	0.948485	0.946212	0.517424	0.940909	0.939394
FordB	**0.9173**	0.798	0.861782	0.748	0.813580	0.792593	0.838272	0.830864	0.838272	0.823457	0.823547
GunPoint	**1**	0.987	**1**	0.833	**1**	**1**	**1**	**1**	**1**	**1**	**1**
Ham	0.781	0.533	0.714286	0.533	0.714286	**0.809524**	0.761905	0.780952	0.619048	0.514286	**0.809524**
HandOutlines	0.9487	0.919	0.954054	0.832	**0.956757**	0.954054	0.937838	0.948649	0.835135	0.945946	0.894595
Haptics	0.551	0.474	0.548701	0.354	0.512987	0.558442	0.564935	0.545455	**0.600649**	0.194805	**0.600649**
Herring	0.703	0.578	0.671875	0.563	0.609375	**0.75**	0.703125	0.734375	0.65625	0.703125	**0.75**
InsectWingbeatSound	0.6525	0.599	0.638889	0.506	0.637374	**0.668687**	0.522222	0.642424	0.53859	0.536364	0.651515
ItalyPowerDemand	0.97	0.929	0.965015	0.942	0.947522	0.963071	0.965015	0.969874	0.962099	**0.971817**	0.964043
Lightning2	**0.8853**	0.787	0.770492	0.885	0.819672	0.819672	0.852459	0.852459	0.754098	0.868852	0.836066
Lightning7	0.863	0.74	0.835616	0.795	0.808219	0.863014	0.821918	0.849315	0.383562	0.835616	**0.90411**
Mallat	0.98	0.916	0.955224	**0.994**	0.963753	0.98081	0.977399	0.975267	0.934328	0.979104	0.938593
Meat	**1**	0.867	0.933333	0.9	0.983333	0.883333	**1**	**1**	**1**	**1**	**1**
MedicalImages	0.792	0.725	0.794737	0.693	0.768421	**0.798684**	0.780263	0.765789	0.759211	0.789474	0.793421
MiddlePhalanxOutlineAgeGroup	**0.8144**	0.623	0.551948	0.506	0.538961	0.668831	0.655844	0.662338	0.623377	0.662338	0.662388
MiddlePhalanxOutlineCorrect	0.8076	0.839	0.817869	0.722	0.807560	0.841924	**0.848797**	**0.848797**	**0.848797**	0.835052	0.744755
MiddlePhalanxTW	0.612	0.555	0.512987	0.513	0.564935	0.603896	0.564935	0.577922	0.551948	**0.623377**	**0.62377**
MoteStrain	**0.95**	0.823	0.886581	0.853	0.939297	0.938498	0.940895	0.916933	0.9377	0.679712	0.875399
OliveOil	0.9333	0.9	0.833333	0.833	0.833333	0.766667	**0.966667**	0.9	0.933333	0.9	**0.966667**
Plane	**1**	0.981	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
ProximalPhalanxOutlineAgeGroup	0.8832	0.839	0.84878	0.805	0.843902	0.887805	0.887805	**0.892683**	0.882927	**0.892683**	0.878049
ProximalPhalanxOutlineCorrect	0.918	0.869	**0.931271**	0.801	0.900344	**0.931271**	**0.931271**	**0.931271**	0.683849	0.924399	0.910653
ProximalPhalanxTW	0.815	0.785	0.77561	0.717	0.775610	**0.843902**	0.819512	0.814634	0.819512	0.819512	0.834146
ShapeletSim	**1**	0.517	0.955556	0.772	0.827778	**1**	**1**	0.91111	0.888889	0.9777778	**1**
ShapesAll	0.9183	0.837	0.928333	0.823	0.923333	0.905	0.923333	0.876667	0.921667	**0.933333**	0.876667
SonyAIBORobotSurface1	0.985	0.84	0.8685552	0.825	0.978369	0.980525	**0.988353**	0.978369	0.708819	0.985025	0.881864
SonyAIBORobotSurface2	0.962	0.832	0.946485	0.885	0.961175	0.972718	0.976915	0.974816	**0.98426**	0.976915	0.854145
Strawberry	0.976	0.946	0.983784	0.903	0.981081	**0.986486**	**0.986486**	**0.986486**	**0.986486**	**0.986486**	**0.986486**
SwedishLeaf	0.9664	0.925	0.9774	0.891	0.9696	**0.9792**	**0.9792**	0.9728	0.9696	0.9664	0.9376
Symbols	0.9668	0.945	0.980905	0.933	0.976884	**0.98794**	0.9799	0.970854	0.976884	0.252261	0.892462
SyntheticControl	**1**	0.977	0.996667	0.977	**1**	0.993333	**1**	0.996667	**1**	**1**	**1**
ToeSegmentation1	0.9737	0.899	0.964912	0.851	0.956140	**0.991228**	0.969298	0.969298	0.97807	**0.991228**	0.982456
ToeSegmentation2	0.9615	0.9	0.938462	0.9	0.938462	0.930769	0.976923	0.953846	0.953846	**0.976923**	0.938462
Trace	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
TwoPatterns	**1**	0.992	**1**	0.998	**1**	0.99675	**1**	**1**	**1**	**1**	**1**
TwoLeadECG	**1**	0.993	0.99561	0.988	0.999122	**1**	**1**	**1**	**1**	**1**	**1**
UWaveGestureLibraryAll	**0.9685**	0.865	0.951982	0.838	0.941653	0.961195	0.856784	0.933277	0.939978	0.879118	**0.9685**
UWaveGestureLibraryX	0.8308	0.784	0.824958	0.762	0.817700	**0.843663**	0.780849	0.814629	0.810999	0.808766	0.815187
UWaveGestureLibraryY	0.7585	0.697	**0.767169**	0.666	0.749860	0.765215	0.664992	0.71636	0.671413	0.67895	0.752094
UWaveGestureLibraryZ	0.7725	0.729	0.764098	0.679	0.757956	**0.795924**	0.756002	0.761027	0.760469	0.762144	0.757677
Wafer	**1**	0.995	0.99854	0.987	0.998864	0.998378	0.99854	0.998215	0.99854	0.999027	**1**
Wine	0.889	0.685	0.611111	0.5	0.555556	0.833333	0.851852	0.87037	0.87037	**0.907407**	**0.907407**

Table 7 (*continued*)

Dataset	Existing SOTA	USRL-FordA [13]	Inception-Time [12]	Combined (1NN) [13]	OS-CNN [41]	Best:fcn-lstm [17]	Vanilla:RN- Transformer [16,43]	ResNet- Transformer [16]	ResNet- Transformer2 [16]	ResNet- Transformer3 [16]	Ours
WordSynonyms	**0.779**	0.641	0.733542	0.633	0.747649	0.680251	0.661442	0.65047	0.636364	0.678683	0.659875
ACSF1	–	0.73	0.92	0.85	0.92	0.9	**0.96**	0.91	0.93	0.17	0.9
BME	–	0.96	0.99333	0.947	**1**	0.993333	**1**	**1**	**1**	**1**	0.986667
Chinatown	–	0.962	0.985423	0.936	0.982609	0.982609	**0.985507**	**0.985507**	**0.985507**	**0.985507**	**0.985507**
Crop	–	0.727	0.772202	0.695	0.770179	0.74494	0.743869	0.742738	0.746012	0.740476	**0.774702**
DodgerLoopDay	–	–	0.15	–	0.5625	0.6375	0.5357	0.55	0.4625	0.5	**0.675**
DodgerLoopGame	–	–	0.855072	–	**0.920290**	0.898551	0.876812	0.891304	0.550725	0.905794	0.905797
DodgerLoopWeekend	–	–	0.971014	–	0.978261	0.978261	0.963768	0.978261	0.949275	0.963768	**0.985507**
GunPointAgeSpan	–	0.987	0.987342	0.991	**1**	0.996835	0.996835	0.996835	**1**	0.848101	0.990506
GunPointMaleVersusFemale	–	**1**	0.993671	0.994	**1**	**1**	**1**	**1**	0.996835	0.996835	**1**
GunPointOldVersusYoung	–	**1**	0.965079	**1**	**1**	0.993651	**1**	**1**	**1**	**1**	**1**
InsectEPGRegularTrain	–	**1**	**1**	**1**	**1**	0.995984	**1**	**1**	**1**	**1**	**1**
InsectEPGSmallTrain	–	**1**	0.943775	**1**	0.473896	0.935743	0.955823	0.927711	0.971888	0.477912	**1**
MelbournePedestrian	–	0.947	0.913899	0.914	0.908163	0.913061	0.912245	0.911837	0.904898	0.901633	0.95736
PowerCons	–	0.933	0.944444	0.894	**1**	0.994444	0.933333	0.9444444	0.927778	0.927778	**1**
Rock	–	0.54	0.8	0.5	0.56	**0.92**	0.78	**0.92**	0.82	0.76	0.88
SemgHandGenderCh2	–	0.84	0.816667	0.863	0.876667	0.91	0.866667	0.916667	0.848333	0.651667	**0.923333**
SemgHandMovementCh2	–	0.516	0.482222	0.709	0.577778	0.56	0.513333	0.504444	0.391111	0.468889	**0.757778**
SemgHandSubjectCh2	–	0.591	0.824444	0.72	0.713333	0.873333	0.746667	0.74	0.666667	0.788889	**0.897778**
SmoothSubspace	–	0.94	0.993333	0.833	**1**	0.98	**1**	**1**	0.99333	**1**	**1**
UMD	–	0.986	0.986111	0.958	0.993056	0.986111	**1**	**1**	**1**	**1**	**1**
Total	65	82	85	82	85	85	85	85	85	85	85
Win	7	0	3	1	2	**12**	2	1	1	2	8
Tie	15	7	9	6	16	15	25	22	21	23	**31**
Lose	43	75	73	75	67	58	58	62	63	60	46
Best	22	7	12	7	18	27	27	23	22	25	**39**

Table 8
Results of eight traditional algorithms and our structure on 65 selected datasets.

Dataset	DD_{DTW} [28]	DTD_C [28]	TSF [7]	TSBF [4]	LPS [3]	BOSS [21]	EE [24]	COTE [11]	Ours
Adiac	0.701	0.701	0.731	0.77	0.77	0.765	0.665	0.79	**0.792839**
ArrowHead	0.789	0.72	0.726	0.754	0.783	0.834	0.811	0.811	**0.851429**
Beef	0.667	0.667	0.767	0.567	0.6	0.8	0.633	0.867	**0.9**
BeetleFly	0.65	0.65	0.75	0.8	0.8	0.9	0.75	0.8	**1**
BirdChicken	0.85	0.8	0.8	0.9	**1**	0.95	0.8	0.9	**1**
Car	0.8	0.733	0.767	0.783	0.85	0.833	0.833	**0.9**	0.883333
CBF	0.997	0.98	0.994	0.988	0.999	0.998	0.998	0.996	**1**
ChlorineConcentration	0.708	0.713	0.72	0.692	0.608	0.661	0.656	0.727	**0.894271**
CinCECGTorso	0.725	0.725	0.983	0.712	0.736	0.887	0.942	**0.995**	0.810145
Coffee	**1**	**1**	0.964	**1**	**1**	**1**	**1**	**1**	**1**
CricketX	0.754	0.754	0.664	0.705	0.697	0.736	0.813	**0.808**	0.771795
CricketY	0.777	0.774	0.672	0.736	0.767	0.754	0.805	**0.826**	0.789744
CricketZ	0.774	0.774	0.672	0.715	0.754	0.746	0.782	**0.815**	0.787179
DiatomSizeReduction	0.967	0.915	0.931	0.899	0.905	0.931	0.944	0.928	**0.980392**
DistalPhalanxOutlineAgeGroup	0.705	0.662	**0.748**	0.712	0.669	**0.748**	0.691	**0.748**	0.719425
DistalPhalanxOutlineCorrect	0.732	0.725	0.772	**0.783**	0.721	0.728	0.728	0.761	0.771739
Earthquakes	0.705	0.705	0.748	0.748	0.64	0.748	0.741	0.748	**0.776978**
ECG200	0.83	0.84	0.87	0.84	0.86	0.87	0.88	0.88	**0.92**
ECG5000	0.924	0.924	0.939	0.94	0.917	0.941	0.939	0.941	**0.944444**
ECGFiveDays	0.769	0.822	0.956	0.877	0.879	**1**	0.82	0.999	**1**
FaceFour	0.83	0.818	0.932	**1**	0.943	**1**	0.909	0.898	0.924045
FacesUCR	0.904	0.908	0.883	0.867	0.926	0.957	0.945	0.942	**0.95122**
FordA	0.723	0.765	0.815	0.85	0.873	0.93	0.736	**0.957**	0.939394
FordB	0.667	0.653	0.688	0.599	0.711	0.771	0.662	0.804	**0.823547**
GunPoint	0.98	0.987	0.973	0.987	0.993	**1**	0.993	**1**	**1**
Ham	0.476	0.552	0.743	0.762	0.562	0.667	0.571	0.648	**0.809524**
HandOutlines	0.868	0.865	**0.919**	0.854	0.881	0.903	0.889	**0.919**	0.894595
Haptics	0.399	0.399	0.445	0.49	0.432	0.461	0.393	0.523	**0.600649**
Herring	0.547	0.547	0.609	0.641	0.578	0.547	0.578	0.625	**0.75**
InsectWingbeatSound	0.355	0.473	0.633	0.625	0.551	0.523	0.595	**0.653**	0.651515
ItalyPowerDemand	0.95	0.951	0.96	0.883	0.923	0.909	0.962	0.961	**0.964043**
Lightning2	0.869	0.869	0.803	0.738	0.82	0.836	**0.885**	0.869	0.836066
Lightning7	0.671	0.658	0.753	0.726	0.74	0.685	0.767	0.808	**0.90411**
Mallat	0.949	0.927	0.919	**0.96**	0.908	0.938	0.94	0.954	0.938593
Meat	0.933	0.933	0.933	0.933	0.883	0.9	0.933	0.917	**1**
MedicalImages	0.737	0.745	0.755	0.705	0.746	0.718	0.742	0.758	**0.793421**
MiddlePhalanxOutlineAgeGroup	0.539	0.5	0.578	0.578	0.578	0.545	0.558	0.636	**0.662388**
MiddlePhalanxOutlineCorrect	0.732	0.742	**0.828**	0.814	0.773	0.78	0.784	0.804	0.744755
MiddlePhalanxTW	0.487	0.5	0.565	0.597	0.526	0.545	0.513	0.571	**0.62377**
MoteStrain	0.833	0.768	0.869	0.903	0.922	0.879	0.883	**0.937**	0.875399
OliveOil	0.833	0.867	0.867	0.833	0.867	0.867	0.867	0.9	**0.966667**
Plane	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
ProximalPhalanxOutlineAgeGroup	0.8	0.795	0.834	0.849	0.849	0.834	0.805	0.854	**0.878049**
ProximalPhalanxOutlineCorrect	0.794	0.794	0.849	0.828	0.873	0.849	0.808	0.869	**0.910653**
ProximalPhalanxTW	0.766	0.771	0.8	0.815	0.81	0.8	0.766	0.78	**0.834146**
ShapeletSim	0.611	0.6	**1**	0.478	0.961	**1**	0.817	0.961	**1**
ShapesAll	0.85	0.838	**0.908**	0.792	0.185	**0.908**	0.867	0.892	0.876667
SonyAIBORobotSurface1	0.742	0.71	0.632	0.787	0.795	0.632	0.704	0.845	**0.881864**
SonyAIBORobotSurface2	0.892	0.892	0.859	0.81	0.778	0.859	0.878	**0.952**	0.854145
Strawberry	0.954	0.957	0.967	0.965	0.952	0.976	0.946	0.951	**0.986486**
SwedishLeaf	0.901	0.896	0.922	0.914	0.915	0.922	0.915	**0.955**	0.9376
Symbols	0.953	0.963	**0.967**	0.915	0.946	**0.967**	0.96	0.964	0.892462
SyntheticControl	0.993	0.997	0.987	0.993	0.98	0.967	0.99	**1**	**1**
ToeSegmentation1	0.807	0.807	0.741	0.781	0.877	0.939	0.829	0.974	**0.982456**
ToeSegmentation2	0.746	0.715	0.815	0.8	0.869	0.962	0.892	0.915	**0.938462**
Trace	**1**	0.99	0.99	0.98	0.98	**1**	0.99	**1**	**1**
TwoLeadECG	0.978	0.985	0.759	0.866	0.948	0.981	0.971	0.993	**1**
TwoPatterns	**1**	**1**	0.991	0.976	0.982	0.993	**1**	**1**	**1**
UWaveGestureLibraryAll	0.935	0.938	0.957	0.926	0.966	0.939	0.968	0.964	**0.9685**
UWaveGestureLibraryX	0.779	0.775	0.804	0.831	0.829	0.762	0.805	0.822	**0.815187**
UWaveGestureLibraryY	0.716	0.698	0.727	0.736	0.761	0.685	0.726	**0.759**	0.752094
UWaveGestureLibraryZ	0.696	0.679	0.743	**0.772**	0.768	0.695	0.724	0.75	0.757677
Wafer	0.98	0.993	0.996	0.995	0.997	0.995	0.997	**1**	**1**
Wine	0.574	0.611	0.63	0.611	0.63	0.741	0.574	0.648	**0.907407**
WordSynonyms	0.73	0.73	0.647	0.688	0.701	0.638	**0.779**	0.757	0.659875
Total	65	65	65	65	65	65	65	65	65
Win	0	0	1	3	0	0	2	11	**33**
Tie	4	3	6	3	3	**10**	3	9	**10**

Table 8 (*continued*)

Dataset	DD_{DTW} [28]	DTD_C [28]	TSF [7]	TSBF [4]	LPS [3]	BOSS [21]	EE [24]	COTE [11]	Ours
Lose	61	62	58	59	62	55	60	45	**22**
Best	4	3	7	6	3	10	5	20	**43**

[24], COTE [11], and BOSS [21]. Similarly, the Best:lstm-fcn approach had the best-performance for each dataset. For example, it involved LSTM-FCN and ALSTM-FCN in [17]. Note that the existing SOTA did not consider the last 20 of the 85 datasets.

The proposed RTFN-based supervised structure performed the best in 'tie' and the second best in 'win', guaranteeing its first position in 'best'. To be specific, our proposed model won in 8 cases and performed no worse than any other algorithm in 31 cases, which led to 39 'best' cases in the comparison. The Best:lstm-fcn and Vanilla:ResNet-Transformer achieved the second and third best performances, respectively, with respect to the 'best' score. The former was a winner of 12 datasets, indicating its outstanding performance. Conversely, USRLFordA was the worst algorithm, with only 7 'tie' scores. Table 8 compares the proposed model and eight traditional algorithms on 65 selected UCR datasets in terms of the top-1 accuracy. These traditional algorithms included DD_{DTW} [28], DTD_C [28], TSF [7], TSBF [4], LPS [3], BOSS [21], EE [24], and COTE [11]. Our structure obtained better 'win', 'tie', 'lose', and 'best' scores than the rest of the algorithms.

In addition, to emphasize the performance of the proposed structure in 'long' time series cases, we show the top-1 accuracy results of different algorithms in Table 9, where all the 20 'long' time series datasets in Table 1 were tested. It is clear that the RTFN-based supervised structure is the best algorithm as it obtains the highest 'Best' and MeanACC values. This is because the RTFN can mine sufficient local features and the relationships among them, thanks to the efficient cooperation of TFN and LSTMaN. In particular, LSTMaN relates different locations of the obtained representations and can thus capture their intrinsic regularizations during the data transmission process. Best:lstm-fcn also achieves decent performance regarding the 'best' value and mean accuracy because its LSTM helps to extract additional features from the input data to enrich the features obtained by the FCN networks. Vanilla:ResNet-Transformer is undoubtedly the one with the best performance among all the compared transformer-based networks because the embedded attention mechanism can further link different positions of time series data and thus enhance the accuracy.

4.4.2. Performance comparison on multivariate time series

Table 10 shows the top-1 accuracy results obtained with different supervised algorithms on all 30 datasets in the UEA2018 archive. The existing SOTA represented the best algorithm on each dataset, including the STC [32], HC [22], gRSF [19], and mv-ARF [42] models. Similarly, Best:DTW, Best:DTWN, and Best:EDN were the best performance DTW-based (e.g., DTW_I and DTW_A [9]), DTWN-based (e.g., DTW-1NN_I(n) and DTW-1NN_D(n) [9]), and ED-NN-based (e.g., ED-1NN and ED-1NN (Normalized) [32]).

Our proposed RTFN-based supervised structure performed the best among all compared algorithms, obtaining the highest MeanACC and 'best' values of 0.763 and 15, respectively, and the smallest AVG_rank score of 3.817. MF and LCEM were the second and third best algorithms, while Best:EDN was the worst. The following explains why. The proposed RTFN takes advantage of TFN and LSTMaN to mine sufficient features from the input data. In particular, LSTMaN can discover the relationships among the representations associated with each variate, as well as those associated with different variates. This is why the proposed RTFN achieved the best performance. Due to the efficient coordination of the LSTM and FCN networks, MF (i.e., MLSTM-FCN) can learn significant connections among the features associated with as many multiple variates as possible. In the meantime, LCEM uses an explicit boosting-bagging approach to explore the interactions among the dimensions. This allows LCEM to capture adequate complex relationships among dimensions at different timestamps, which is why MF and LCEM also achieve decent performance in supervised classification. Alternatively, Best:EDN is based on the traditional DTW-1NN approach, making it quite challenging to simultaneously focus on the useful representations in univariate data and their relationships in multivariate data. This is why deep learning approaches have attracted increasingly more research attention. The AVG_rank results obtained with different supervised algorithms on 30 multivariate datasets are shown in Fig. 5.

4.5. Evaluation of the RTFN-based unsupervised clustering

To evaluate the performance of the RTFN-based unsupervised clustering, we compared it with a number of state-of-the-art unsupervised algorithms against three performance metrics: 'best' based on the results of 'win'/'tie'/'lose', AVG RI, and AVG_rank, as discussed in Section 4.2.

Following some well-recognized research works [15,27,30], we selected 36 representative datasets from the UCR2018 archive for performance evaluation, and they are marked with 'YES' in Table 1. The RI results obtained with different unsupervised algorithms on the 36 datasets are shown in Table 11. DTCR and the proposed RTFN-based unsupervised clustering ranked best and second best among all compared algorithms, respectively. DTCR takes advantage of a seq2seq structure to explore sufficient temporal features for a K-means classifier. This algorithm uses the classifier's loss to update its model

Table 9
Results of different supervised algorithms on 20 'long' time series datasets.

Dataset	Classes	SeriesLength	Existing SOTA	USRL-FordA [13]	Inception-Time [12]	Combined (1NN) [13]	OS-CNN [41]	Best: lstm-fcn [17]	Vanilla:RN-Transformer [16,43]	ResNet-Transformer1 [16]	ResNet-Transformer2 [16]	ResNet-Transformer3 [16]	*Ours*
BeetleFly	2	512	0.95	0.8	0.8	0.8	0.8	**1**	**1**	0.95	0.95	**1**	**1**
BirdChicken	2	512	0.95	0.9	0.95	0.75	0.9	0.95	**1**	0.9	**1**	0.7	**1**
CinCECGTorso	4	1639	**0.9949**	0.638	0.853623	0.693	0.830435	0.904348	0.871739	0.656522	0.89058	0.31087	0.810145
Car	4	577	0.933	0.85	0.88333	0.8	0.933333	**0.966667**	0.95	0.883333	0.866667	0.3	0.883333
Earthquake	2	512	**0.801**	0.748	0.741007	0.64	0.683453	0.81295	0.755396	0.755396	0.76259	0.755396	0.776978
HandOutlines	2	2709	0.9487	0.919	0.954054	0.832	**0.956757**	0.954054	0.937838	0.948649	0.835135	0.945946	0.894595
Haptics	5	1092	0.551	0.474	0.548701	0.354	0.512987	0.558442	0.564935	0.545455	**0.600649**	0.194805	**0.600649**
Herring	2	512	0.703	0.578	0.671875	0.563	0.609375	**0.75**	0.703125	0.734375	0.65625	0.703125	**0.75**
Lighting2	2	637	**0.8853**	0.787	0.770492	0.885	0.819672	0.819672	0.852459	0.852459	0.754098	0.868852	0.836066
Mallat	8	1024	0.98	0.916	0.955224	0.994	0.963753	**0.98081**	0.977399	0.975267	0.934328	0.979104	0.938593
OliveOil	4	570	0.9333	0.9	0.833333	0.833	0.833333	0.766667	**0.966667**	0.9	0.933333	0.9	**0.966667**
ShapesAll	60	512	0.9183	0.837	0.928333	0.823	0.923333	0.905	0.923333	0.876667	0.921667	**0.933333**	0.876667
UWaveGestureLibraryAll	8	945	**0.9685**	0.865	0.951982	0.838	0.941653	0.961195	0.856784	0.933277	0.939978	0.879118	**0.9685**
ACSF1	10	1460	—	0.73	0.92	0.85	0.92	0.9	**0.96**	0.91	0.93	0.17	0.9
InsectEPGRegularTrain	3	601	—	**1**	**1**	**1**	**1**	0.995984	**1**	**1**	**1**	**1**	**1**
InsectEPGSmallTrain	3	601	—	**1**	0.943775	**1**	0.473896	0.935743	0.955823	0.927711	0.971888	0.477912	**1**
SemgHandGendeCh2	2	1500	—	0.84	0.816667	0.863	0.876667	0.91	0.866667	0.916667	0.848333	0.651667	**0.923333**
SemgHandMovementCh2	6	1500	—	0.516	0.482222	0.709	0.577778	0.56	0.513333	0.504444	0.391111	0.468889	**0.757778**
SemgHandSubjectCh2	5	1500	—	0.591	0.824444	0.72	0.713333	0.873333	0.746667	0.74	0.666667	0.788889	**0.897778**
Rock	4	2844	—	0.54	0.8	0.5	0.56	**0.92**	0.78	**0.92**	0.82	0.76	0.88
Best			4	2	1	2	1	5	5	2	3	3	**11**
MeanACC			—	0.77145	0.8314531	0.77235	0.7914879	0.87124325	0.85910825	0.8415111	0.8336637	0.6893953	**0.8830541**

Table 10
Results of different supervised algorithms on all UEA2018 datasets. Abbreviations: MF - MLSTM-FCN [18], WM - WEASEL + MUSE [37].

Dataset Index	Existing SOTA	Best: DTW [32]	Best: DTWN [9]	Best: EDN [32]	LCEM [9]	XGBM [22]	RFM [39]	WM [32]	CBOSS [21]	MLCN [18]	RISE [45]	TSF [7]	TapNet [49]	MUSE [37]	MF [9,18]	Ours
AWR	0.99	0.987	0.987	0.97	**0.993**	0.99	0.99	**0.993**	0.99	0.957	0.963	0.953	0.987	**0.993**	0.986	**0.993**
AF	0.267	0.267	0.267	0.267	0.467	0.4	0.333	0.267	0.267	0.333	0.267	0.2	0.333	0.4	0.2	**0.533**
BM	**1**	**1**	**1**	0.675	**1**	**1**	**1**	**1**	**1**	0.875	**1**	**1**	**1**	**1**	**1**	**1**
CT	0.986	**1**	0.969	0.964	0.979	0.983	0.985	0.99	0.986	0.917	0.986	0.931	0.997	0.986	0.993	0.993
CR	–	–	**1**	0.944	0.986	0.972	0.986	0.986	–	–	–	–	0.958	–	0.986	0.986
DDG	0.48	0.58	0.6	0.275	0.375	0.4	0.4	0.575	0.48	0.38	0.22	0.46	0.575	0.58	**0.675**	0.6
EW	0.749	0.517	0.618	0.55	0.527	0.55	**1**	0.89	0.511	0.33	0.626	0.712	0.489	–	0.809	0.685
EP	**1**	**1**	0.978	0.667	0.986	0.978	0.986	0.993	0.979	0.732	0.979	**1**	0.971	0.993	0.964	0.978
EC	**0.882**	0.361	0.323	0.293	0.372	0.422	0.433	0.316	0.304	0.373	0.445	0.487	0.323	0.476	0.274	0.38
ER	0.97	0.926	0.133	0.133	0.2	0.133	0.133	0.133	0.919	0.941	0.881	0.859	0.133	**0.974**	0.133	0.941
FD	0.656	0.529	0.529	0.519	0.614	0.629	0.614	0.545	0.513	0.555	0.64	0.508	0.556	0.631	0.555	**0.67**
FM	0.582	0.53	0.53	0.55	0.59	0.53	0.569	0.54	0.519	0.58	0.581	0.562	0.53	0.551	0.61	**0.63**
HMD	0.414	0.224	0.306	0.279	0.649	0.541	0.5	0.378	0.292	0.544	0.481	0.312	0.378	0.362	0.378	**0.662**
HW	0.478	0.601	**0.607**	0.531	0.287	0.267	0.267	0.531	0.504	0.305	0.359	0.191	0.357	0.518	0.547	0.454
HB	0.64	0.604	0.717	0.62	0.761	0.693	0.8	0.727	0.564	0.458	0.535	0.518	0.751	0.515	0.714	**0.785**
IW	–	–	0.115	0.128	0.228	0.237	0.224	–	–	–	–	–	0.208	–	0.105	**0.467**
JV	–	–	0.959	0.949	0.978	0.968	0.97	0.978	–	–	–	–	0.965	–	**0.992**	0.973
LIB	0.9	0.883	0.894	0.833	0.772	0.767	0.783	0.894	0.894	0.85	0.806	0.806	0.85	0.894	**0.922**	**0.922**
LSST	0.391	0.458	0.575	0.456	**0.652**	0.633	0.612	0.628	0.458	0.39	0.161	0.265	0.568	0.435	0.646	0.451
MI	**0.61**	0.59	0.51	0.51	0.6	0.46	0.55	0.5	0.39	0.51	0.48	0.55	0.59	–	0.53	0.6
NATO	0.889	0.883	0.883	0.85	0.916	0.9	0.911	0.883	0.85	0.9	0.8	0.839	0.939	0.906	0.961	**0.967**
PD	0.941	0.977	0.977	0.939	0.977	0.951	0.951	0.969	0.939	0.979	0.892	0.831	0.98	0.967	**0.987**	**0.987**
PEMS	0.981	0.981	0.734	0.705	0.942	0.983	0.983	–	0.73	0.745	0.982	**0.994**	0.751	–	0.653	0.936
PS	0.321	0.195	0.151	0.104	0.288	0.187	0.222	0.19	0.151	0.151	0.137	0.269	0.175	–	0.275	**0.33**
RS	0.898	0.891	0.868	0.842	**0.941**	0.928	0.921	0.914	0.854	0.856	0.895	0.823	0.868	0.933	0.882	0.862
SRS1	0.854	0.806	0.775	0.771	0.839	0.829	0.826	0.744	0.765	0.908	0.84	0.724	0.652	0.697	0.867	**0.922**
SRS2	0.533	0.539	0.539	0.533	0.55	0.483	0.478	0.522	0.533	0.506	0.483	0.494	0.55	0.528	0.522	**0.622**
SAD	–	–	0.963	0.967	0.973	0.97	0.968	0.982	–	–	–	–	0.983	–	**0.994**	0.986
SWJ	0.467	0.333	0.333	0.2	0.4	0.333	0.467	0.333	0.333	0.4	0.333	0.267	0.4	0.267	0.467	**0.667**
UW	0.897	0.903	0.903	0.881	0.897	0.894	0.9	0.903	0.869	0.859	0.775	0.684	0.894	**0.931**	0.857	0.903
Best	4	3	3	0	4	1	2	2	1	0	1	3	1	4	6	**15**
MeanACC	0.626	0.586	0.658	0.597	0.691	0.668	0.692	0.643	0.553	0.532	0.552	0.541	0.657	0.502	0.683	**0.763**
AVG_rank hline	6.817	8.517	8.633	11.800	5.900	8.417	6.917	7.833	11.117	10.583	10.450	11.433	8.083	8.933	6.750	**3.817**

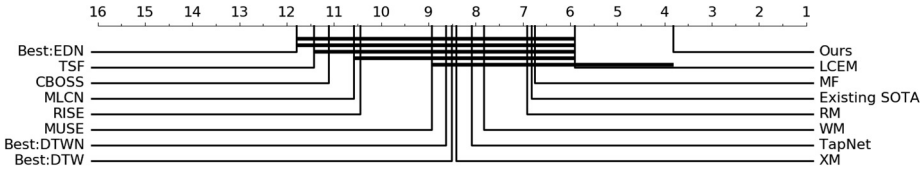

Fig. 5. Results of the AVG_ranks of different algorithms on 30 multivariate datasets.

parameters, encouraging the representations extracted from the data to form a cluster structure. This explains why DTCR is good at mining cluster-specific representations from input data. However, its complicated structure is for addressing cluster-specific problems only. On the contrary, the RTFN simply adopts an auto-encoding structure to update our model's parameters and utilizes a K-means algorithm to classify the features obtained by the RTFN. Although it is quite simple in structure, the RTFN achieves decent performance on the 36 selected datasets based on its strong feature extraction ability.

In order to further evaluate the effectiveness of the RTFN in unsupervised clustering, we compared the RTFN-based K-means algorithm with a separate K-means algorithm on the 36 datasets above, and the RI results are shown in Fig. 6. The RTFN outperformed the separate K-means algorithm on all but two datasets. This is because the unsupervised clustering was provided with sufficient features obtained by the proposed RTFN, especially those hiding deeply in the input data beyond the exploration abilities of ordinary feature extraction networks. The AVG_rank results of all compared unsupervised algorithms are shown in Fig. 7.

Table 11
The RI results of different unsupervised algorithms on 36 selected datasets.

Dataset	K-means [27]	UDFS [30]	NDFS [15]	RUFS [27]	RSFS [27]	KSC [30]	KDBA [46]	K-shape [15]	Ushapelet [27]	DTC [30]	DEC [46]	IDEC [15]	DTCR [27]	Ours
ArrowHead	0.6095	0.7254	0.7381	**0.7476**	0.7108	0.7254	0.7222	0.7254	0.6460	0.6692	0.5817	0.6210	0.6868	0.6460
Beef	0.6713	0.6759	0.7034	0.7149	0.6975	0.7057	0.6713	0.5402	0.6966	0.6345	0.5954	0.6276	**0.8046**	0.7655
BeetFly	0.4789	0.4949	0.5579	0.6053	0.6516	0.6053	0.6052	0.6053	0.7314	0.5211	0.4947	0.6053	**0.9000**	0.6632
BirdChicken	0.4947	0.4947	0.7361	0.5579	0.6632	0.7316	0.6053	0.6632	0.5579	0.4947	0.4737	0.4789	**0.8105**	0.7316
Car	0.6345	0.6757	0.6260	0.6667	0.6708	0.6898	0.6254	0.7028	0.6418	0.6695	0.6859	0.6870	**0.7501**	0.7169
ChlorineConcentration	0.5241	0.5282	0.5225	0.5330	0.5316	0.5256	0.5300	0.4111	0.5318	0.5353	0.5348	0.5350	0.5357	**0.5367**
Coffee	0.7460	0.8624	**1.0000**	0.5467	**1.0000**	**1.0000**	0.4851	**1.0000**	**1.0000**	0.4841	0.4921	0.5767	0.9286	0.8624
DiatomSizeReduction	0.9583	0.9583	0.9583	0.9333	0.9137	**1.0000**	0.9583	**1.0000**	0.7083	0.8792	0.9294	0.7347	0.9682	0.9583
DistalPhalanxOutlineAgeGroup	0.6171	0.6531	0.6239	0.6252	0.6539	0.6535	0.6750	0.6020	0.6273	0.7812	0.7785	0.7786	**0.7825**	0.7786
DistalPhalanxOutlineCorrect	0.5252	0.5362	0.5362	0.5252	0.5327	0.5235	0.5203	0.5252	0.5098	0.5010	0.5029	0.5330	0.6075	**0.6095**
ECG200	0.6315	0.6533	0.6315	0.7018	0.6916	0.6315	0.6018	0.7018	0.5758	0.6018	0.6422	0.6233	0.6648	**0.7568**
ECGFIveDays	0.4783	0.5020	0.5573	0.5020	0.5953	0.5257	0.5573	0.5020	0.5968	0.5016	0.5103	0.5114	**0.9638**	0.5964
GunPoint	0.4971	0.5029	0.5102	**0.6498**	0.4994	0.4971	0.5420	0.6278	0.6278	0.5400	0.4981	0.4974	0.6398	0.6471
Ham	0.5025	0.5219	0.5362	0.5107	0.5127	0.5362	0.5141	0.5311	0.5362	0.5648	**0.5963**	0.4956	0.5362	**0.5963**
Herring	0.4965	0.5099	0.5164	0.5238	0.5151	0.4940	0.5164	0.4965	0.5417	0.5045	0.5099	0.5099	**0.5790**	0.5322
Lighting2	0.4966	0.5119	0.5373	0.5729	0.5269	0.6263	0.5119	0.6548	0.5192	0.5770	0.5311	0.5519	0.5913	**0.6230**
Meat	0.6595	0.6483	0.6635	0.6578	0.6657	0.6723	0.6816	0.6575	0.6742	0.3220	0.6475	0.6220	**0.9763**	0.7175
MiddlePhalanxOutlineAgeGroup	0.5351	0.5269	0.5350	0.5315	0.5473	0.5364	0.5513	0.5105	0.5396	0.5757	0.7059	0.6800	**0.7982**	0.6800
MiddlePhalanxOutlineCorrect	0.5000	0.5431	0.5047	0.5114	0.5149	0.5014	0.5563	0.5114	0.5218	0.5272	0.5423	0.5423	0.5617	**0.5909**
MiddlePhalanxTW	0.0983	0.1225	0.1919	0.7920	0.8062	0.8187	0.8046	0.6213	0.7920	0.7115	0.8590	0.8626	**0.8638**	0.8062
MoteStrain	0.4947	0.5579	0.6053	0.5579	0.6168	0.6632	0.4789	0.6053	0.4789	0.5062	0.7435	0.7342	**0.7686**	0.6826
OSULeaf	0.5615	0.5372	0.5622	0.5497	0.5665	0.5714	0.5541	0.5538	0.5525	0.7329	0.7484	0.7607	**0.7739**	0.7513
Plane	0.9081	0.8949	0.8954	0.9220	0.9314	0.9603	0.9225	0.9901	**1.0000**	0.9040	0.9447	0.9447	0.9549	0.9619
ProximalPhalanxOutlineAgeGroup	0.5288	0.4997	0.5463	0.5780	0.5384	0.5305	0.5192	0.5617	0.5206	0.7430	0.4263	0.8091	0.8091	**0.8180**
ProximalPhalanxTW	0.4789	0.4947	0.6053	0.5579	0.5211	0.6053	0.5211	0.5211	0.4789	0.8380	0.8189	**0.9030**	0.9023	0.8180
SonyAIBORobotSurface1	0.7721	0.7695	0.7721	0.7787	0.7928	0.7726	0.7988	0.8084	0.7639	0.5563	0.5732	0.6900	**0.8769**	0.7812
SonyAIBORobotSurface2	0.8697	0.8745	0.8865	0.8756	0.8948	**0.9039**	0.8684	0.5617	0.8770	0.7012	0.6514	0.6572	0.8354	0.7066
SwedishLeaf	0.4987	0.4923	0.5500	0.5192	0.5038	0.4923	0.5500	0.5533	0.6154	0.8871	0.8837	0.8893	**0.9223**	0.8893
Symbols	0.8810	0.8548	0.8562	0.8525	0.9060	0.8982	**0.9774**	0.8373	0.9603	0.9053	0.8841	0.8857	0.9168	0.9053
ToeSegmentation1	0.4873	0.4921	0.5873	0.5429	0.4968	0.5000	**0.6143**	**0.6143**	0.5873	0.5077	0.4984	0.5017	0.5659	**0.6143**
ToeSegmentation2	0.5257	0.5257	0.5968	0.5968	0.5826	0.5257	0.5573	0.5257	0.5020	0.5348	0.4991	0.4991	**0.8286**	0.7825
TwoPatterns	0.8529	0.8259	0.8530	0.8385	**0.8588**	0.8585	0.8446	0.8046	0.7757	0.6251	0.6293	0.6338	0.6984	0.6298
TwoLeadECG	0.5476	0.5495	0.6328	**0.8246**	0.5635	0.5464	0.5476	**0.8246**	0.5404	0.5116	0.5007	0.5016	0.7114	0.6289
Wafer	0.4925	0.4925	0.5263	0.5263	0.4925	0.4925	0.4925	0.4925	0.4925	0.5324	0.5679	0.5597	0.7338	**0.8093**
Wine	0.4984	0.4987	0.5123	0.5021	0.5033	0.5006	0.5064	0.5001	0.5033	0.4906	0.4913	0.5157	**0.6271**	0.5471
WordSyonyms	0.8775	0.8697	0.8760	0.8861	0.8817	0.8727	0.8159	0.7844	0.8230	0.8855	0.8893	0.8947	**0.8984**	0.8973
Best	0	0	1	3	2	3	2	4	2	0	1	1	**17**	9
AVG RI	0.5957	0.6077	0.6403	0.6477	0.6542	0.6582	0.6335	0.6419	0.6402	0.6238	0.6351	0.6515	**0.7714**	0.7233
AVG_rank	10.8194	9.6944	7.2361	7.4306	6.8750	7.1667	7.9722	8.0000	8.1250	8.6111	8.7778	7.6389	**2.7778**	3.6250

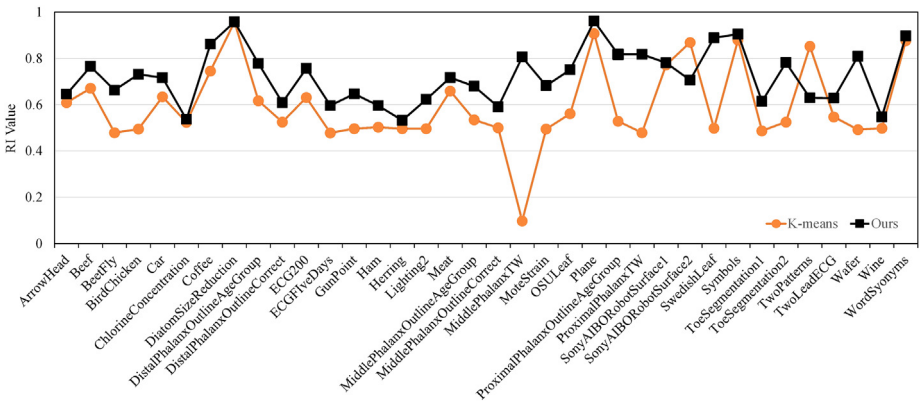

Fig. 6. The RI values obtained by the K-means algorithm and ours.

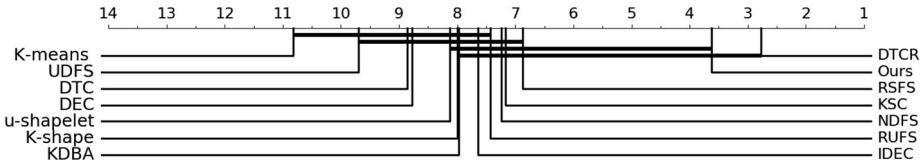

Fig. 7. AVG_ranks of different unsupervised algorithms.

5. Conclusion

In the proposed RTFN, the temporal feature network is responsible for extracting local features, while the LSTM-based attention network discovers intrinsic relationships among the representations learned from data. Experimental results demonstrated that the proposed RTFN achieved decent performance in both supervised classification and unsupervised clustering. Specifically, the proposed RTFN-based supervised algorithm performed the best on 39 out of 85 univariate datasets in the UCR2018 archive and 15 out of 30 multivariate datasets in the UEA2018 archive, compared with the latest results from the supervised classification community. In particular, the proposed RTFN won 11 out of 20 'long' univariate dataset cases. Our RTFN-based unsupervised algorithm performed second best when considering all 36 datasets. Finally, the experimental results also indicated that the RTFN had a strong potential to be embedded in other learning frameworks to handle time series problems of various domains in the real world.

CRediT authorship contribution statement

Zhiwen Xiao: Methodology, Conceptualization, Writing - original draft. **Xin Xu:** Methodology, Validation. **Huanlai Xing:** Methodology, Writing - review & editing, Supervision. **Shouxi Luo:** Software, Validation. **Penglin Dai:** Investigation, Visualization. **Dawei Zhan:** 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.

Acknowledgements

This work was supported in part by National Natural Science Foundation of China (No. 61802319, No. 62002300), China Postdoctoral Science Foundation (No. 2019M660245, No. 2019M663552, No. 2020T130547), the Fundamental Research Funds for the Central Universities, and China Scholarship Council, P. R. China.

References

[1] J. Audibert, P. Michiardi, F. Guyard, S. Marti, M.A. Zuluaga, Usad: unsupervised anomaly detection on multivariate time series, in: Proc. ACM KDD'20, 2020, pp. 23–27.
[2] M.G. Baydogan, G. Runger, Learning a symbolic representation for multivariate time series classification, Data Min. Knowl. Disc. 29 (2015) 400–422.
[3] M.G. Baydogan, G. Runger, Time series representation and similarity based on local autopatterns, Data Min. Knowl. Disc. 30 (2016) 476–509.
[4] M.G. Baydogan, G. Runger, E. Tuv, A bag-of-features framework to classify time series, IEEE Trans. Pattern Anal. 35 (11) (2013) 2796–2802.
[5] J. Chen, Z. Xiao, H. Xing, P. Dai, S. Luo, M.A. Iqbal, Stdpg: a spatio-temporal deterministic policy gradient agent for dynamic routing in sdn, in: Proc. IEEE ICC 2020, 2020, pp. 1–6.
[6] X. Chen, J. Yu, Z. Wu, Temporally identity-aware ssd with attentional lstm, IEEE Trans. Cybern. 50 (6) (2020) 2674–2686.
[7] H. Deng, G. Runger, E. Tuv, M. Vladimir, A time series forest for classification and feature extraction, Inform. Sci. 239 (2013) 142–153.
[8] M. Fahim, K. Fraz, A. Sillitti, Tsi: Time series to imaging based model for detecting anomalous energy consumption in smart buildings, Inform. Sci. 523 (2020) 1–13.
[9] K. Fauvel, É. Fromont, V. Masson, P. Faverdin, and A. Termier. Local cascade ensemble for multivariate data classification. arXiv preprint arXiv:2005.03645, 2020..
[10] H.I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, P.-A. Muller, Adversarial attack on deep neural networks for time series classification, in: Proc. IJCNN 2019, 2019, pp. 1–8.
[11] H.I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, P.-A. Muller, Deep learning for time series classification: a review, Data Min. Knowl. Disc. 33 (2019) 917–963.
[12] H.I. Fawaz, B. Lucas, G. Forestier, C. Pelletier, D.F. Schmidt, J. Weber, G.I. Webb, L. Idoumghar, P.-A. Muller, F. Petitjean, Inceptiontime: finding alexnet for time series classification, Data Min. Knowl. Disc. 34 (2020) 1936–1962.
[13] J.-Y. Franceschi, A. Dieuleveut, M. Jaggi, Unsupervised scalable representation learning for multivariate time series, in: Proc. NeurIPS 2019, 2019, pp. 1–25.
[14] Z. Geng, G. Chen, Y. Han, G. Lu, F. Li, Semantic relation extraction using sequential and tree-structured lstm with attention, Inform. Sci. 509 (2020) 183–192.
[15] X. Guo, L. Gao, X. Liu, J. Yin, Improved deep embedded clustering with local structure preservation, in: Proc. IJCAI 2017, 2017, pp. 1753–1759.
[16] S.H. Huang, L. Xu, C. Jiang. Residual attention net for superior cross-domain time sequence modeling. arXiv preprint arXiv: 2001.04077, 2020..
[17] F. Karim, S. Majumdar, H. Darabi, Insights into lstm fully convolutional networks for time series classification, IEEE Access 7 (2019) 1328–1342.
[18] F. Karim, S. Majumdar, H. Darabi, S. Harford, Multivariate lstm-fcns for time series classification, Neural Networks 116 (2019) 237–245.

[19] I. Karlsson, P. Papapetrou, H. Boström, Generalized random shapelet forests, Data Min. Knowl. Disc. 30 (2016) 1053–1085.
[20] K. Kashiparekh, J. Narwariya, P. Malhotra, L. Vig, G. Shroff, Convtimenet: A pre-trained deep convolutional neural network for time series classification, in: Proc. IJCNN 2019, 2019, pp. 1–8.
[21] J. Large, A. Bagnall, S. Malinowski, R. Tavenard. From bop to boss and beyond: time series classification with dictionary based classifier. arXiv preprint arXiv:1809.06751, 2018..
[22] J. Large, J. Lines, A. Bagnall, A probabilistic classifier ensemble weighting scheme based on cross validated accuracy estimates, Data Min. Knowl. Disc. 33 (2019) 1674–1709.
[23] Y. LeCun, Y. Bengio, G. Hinton. Deep learning. Nature, pages 436–444, 2015..
[24] J. Lines, A. Bagnall, Time series classification with ensembles of elastic distance measures, Data Min. Knowl. Disc. 29 (2015) 565–592.
[25] J. Lines, S. Taylor, A. Bagnall, Time series classification with hive-cote: The hierarchical vote collective of transformation-based ensembles, ACM Trans. Knowl. Discov. D. 12 (5) (2018) 1–35.
[26] F. Liu, X. Zhou, J. Cao, Z. Wang, T. Wang, H. Wang, Y. Zhang, Anomaly detection in quasi-periodic time series based on automatic data segmentation and attentional lstm-cnn, IEEE Trans. Knowl. Data En. (2020) 1–14.
[27] Q. Ma, J. Zheng, S. Li, G.W. Cottrell, Learning representations for time series clustering, in: Proc. NeurIPS 2019, 2019, pp. 1–11.
[28] Q. Ma, W. Zhuang, S. Li, D. Huang, G.W. Cottrell, Adversarial dynamic shapelet networks, in: Proc. AAAI 2020, 2020, pp. 5069–5076.
[29] L. Maaten, Learning discriminative fisher kernels, in: Proc. ICML 2011, 2011, pp. 217–224.
[30] N.S. Madiraju, S.M. Sadat, D. Fisher, H. Karimabadi, Deep temporal clustering: fully unsupervised learning of time-domain features, in: Proc. ICLR 2018, 2018, pp. 1–11.
[31] T. Pradhan, P. Kumar, S. Pal, Claver: An integrated framework of convolutional layer, bidirectional lstm with attention mechanism based scholarly venue recommendation, Inform. Sci. 559 (2020) 212–235.
[32] A.R. Puiz, 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. Disc. 35 (2021) 401–449.
[33] A. Quattoni, S. Wang, L. Morency, M. Collins, T. Darrell, Hidden conditional random fields, IEEE Trans. Pattern Anal. 29 (10) (2007) 1848–1852.
[34] P. Rajpurkar, A.Y. Hannun, M. Haghpanahi, C. Bourn, and A.Y. Ng. Cardiologist-level arrhythmia detection with convolutional neural networks. arXiv:1707.01836, 2017..
[35] J. Redmon, A. Farhadi. Yolov3: An incremental improvement. arXiv preprint arXiv: 1804.02767, 2018..
[36] S. Ruder. An overview of gradient descent optimization algorithms. arXiv preprint arXiv: 1609.04747v2, 2017..
[37] P. Schäfer and U. Leser. Multivariate time series classification with weasel+muse. arXiv preprint arXiv:1711.11343, 2017..
[38] J. Serrá, S. Pascual, A. Karatzoglou, Towards a universal neural network encoder for time series, in: Proc. CCIA 2018, 2018, pp. 120–129.
[39] M. Shokoohi-Yekta, B. Hu, H. Jin, J. Wang, E. Keogh, Generalizing dtw to the multi-dimensional case requires an adaptive approach, Data Min. Knowl. Disc. 31 (2017) 1–31.
[40] K. Shuang, Z. Zhang, J. Loo, S. Su, Convolution-deconvolution word embedding: an end-to-end multi-prototype fusion embedding method for natural language processing, Inform. Fusion 53 (2020) 112–122.
[41] W. Tang, G. Long, L. Liu, T. Zhou, J. Jiang, and M. Blumenstein. Rethinking 1d-cnn for time series classification: a stronger baseline. arXiv preprint arXiv: 2002.10061, 2020..
[42] K.S. Tuncel, M.G. Baydogan, Autoregressive forests for multivariate time series modeling, Pattern Recogn. 73 (2018) 202–215.
[43] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Łstrok;. Kaiser, I. Polosukhin, Attention is all you need, in: Proc. NeurIPS 2017, 2017, pp. 5998–6008.
[44] Z. Wang, W. Yan, T. Oates, Time series classification from scratch with deep neural networks: A strong baseline, in: Proc. IEEE IJCNN 2017, 2017, pp. 1578–1585.
[45] M. Wistuba, J. Grabocka, L. Schmidt-Thieme. Ultra-fast shapelets for time series classification. arXiv preprint arXiv:1503.05018, 2015..
[46] J. Xie, R. Girshick, A. Farhadi, Unsupervised deep embedding for clustering analysis, in: Proc. ICML 2016, 2016, pp. 478–487.
[47] D. Yang, X. Zhou, X. Wang, J. Huang, Mirco-earthquake source depth detection using machine learning techniques, Inform. Sci. 544 (2021) 325–342.
[48] Q. Yao, R. Wang, X. Fan, J. Liu, Y. Li, Multi-class arrhythmia detection from l2-lead varied-length ecg using attention-based time-incremental convolutional neural network, Inform. Fusion 53 (2020) 174–182.
[49] X. Zhang, Y. Gao, J. Lin, C.-T. Lu, Tapnet: Multivariate time series classification with attentional prototypical network, in: Proc. AAAI 2020, 2020, pp. 6845–6852.
[50] Y. Zhu, C. Zhao, H. Guo, J. Wang, X. Zhao, H. Lu, Attention couplenet: fully convolutional attention coupling network for object detection, IEEE Trans. Image Process. 28 (1) (2018) 1170–1175.