RTFN: A Robust Temporal Feature Network for Time Series Classification

Zhiwen Xiao, Xin Xu, Huanlai Xing, Shouxi Luo, Penglin Dai, Dawei Zhan

Southwest Jiaotong University, Chengdu, China
China University of Mining and Technology, Xuzhou, China
Northwest A&F University, Yangling, China

Abstract

Time series data usually contains local and global patterns. Most of the existing feature networks pay more attention to local features rather than the relationships among them. The latter is, however, also important yet more difficult to explore. To obtain sufficient representations by a feature network is still challenging. 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 an LSTM-based attention network (LSTMaN). TFN is a residual structure with multiple convolutional layers. It functions as a local-feature extraction network to mine sufficient local features from data. LSTMaN is composed of two identical layers, where attention and long short-term memory (LSTM) networks are hybridized. This network acts as a relation extraction network to discover the intrinsic relationships among the extracted features at different positions in sequential data. In experiments, we embed RTFN into a supervised structure as a feature extractor and into an unsupervised structure as an encoder, respectively. The results show that the RTFN-based structures achieve excellent supervised and unsupervised performance on a large number of UCR2018 and UEA2018 datasets.

Keywords: attention mechanism, convolutional neural network, data mining,
1. Introduction

Time series data has been used in various domains, such as weather forecasting \[67\], traffic analysis \[34, 58\], human activity recognition \[1, 66\], clinical diagnosis \[1\], human heart record \[7\], electricity demand \[40\], etc. How to make full use of the data in real-world applications is crucial, depending on how well features are extracted. Different from other types of data, such as ImageNet for image classification \[8\], SemEval-2014 for sentiment recognition \[38\], and ICDAR2019 for natural scene text processing \[56\], a time series is a sequence of time-ordered data points recording certain processes \[2\]. 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 \[2, 11, 23, 31\].

So far, there have been two classes of approaches for addressing the challenge above, including traditional algorithms and deep learning ones \[11\]. The traditional algorithms are mainly distance-based. They mine features and regularizations from data, by revealing the significant differences and connections within the data. There are two streams of widely used algorithms, namely nearest neighbor (NN) methods based on a distance function, especially the dynamic time warping (DTW) distance, and linear logistic methods. Thanks to the use of DTW, the NN methods compensate for possible confounding offset by allowing some realignment of the series to show differences in the arrangement of data \[2, 28\]. The hierarchical vote collective of transformation-based ensembles (HIVE-COTE) \[29\] and the local cascade ensemble (LCE) \[9\] are two representative algorithms in the literature. Meanwhile, the linear logistic methods rely on logistic models to build internal logical structures of data and explore basic regularizations from them, e.g. the fisher kernel learning (KFL) \[32\], the hidden unit logistic model (HULM) \[36\] and the hidden state conditional random field
On the other hand, the deep learning algorithms aim at unfolding the internal representation hierarchy of data, which helps to capture the intrinsic connections among representations. These algorithms can be roughly classified into two categories, namely single-network-based and dual-network-based models. To be specific, a single-network-based model simply uses one (usually hybridized) network to handle both feature and relation extraction. These models pay attention to mining the basic representation hierarchy of data and significant connections within the hierarchy, e.g., ConvTimeNet, InceptionTime, and OmniScale 1-dimensional convolutional neural network (OS-CNN). 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 while the second one focuses on relationships among the features extracted, e.g., Transformer-based model, ALSTM-FCN, etc. Compared with feature extraction, relation extraction aims at capturing those hidden connections among the features extracted before. 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 of vital importance to design an effective relation extraction network for different applications. In particular, attention and long short-term memory (LSTM) networks are widely used for relation extraction in the context of time series classification. This is because attention mechanism can relate different positions of a sequence to derive the relationships at certain positions while LSTM is able to explore long- and short-period dependencies in data, both of which help to enhance the relation extraction.

Nowadays, hybridizing attention and LSTM networks for relation extraction has attracted increasingly more research efforts in time series classification. There are mainly two ways to combine them, namely cascading and embedding models. A cascading model simply stacks attention and LSTM networks one after another to realize some specific functions. But, neither attention networks
nor LSTM networks require significant changes in their structures, e.g. the attention LSTM (AttLSTM) models [19, 20]. However, the cascading models usually suffer from two drawbacks. Firstly, almost all attention networks are based on fully connected networks, which are not sensitive to the intricate features hidden in data. Secondly, the useful representations extracted before are easily lost as they go through subsequent networks. An embedding model, on the other hand, integrates attention and LSTM networks in a compact manner. With less layers of neural networks cascaded, less features are lost during data transmission. Thus, more useful features and relationships are made use of by the model. Such a model is sensitive to regular and periodic data and hence able to concentrate on the local and periodical variations of data, e.g. LSTM with trend attention gate (LSTMTAG) [30]. If not designed properly, an embedding model may not be aware of the global variations of those non-periodical data, especially when handling long univariate and multivariate datasets. Theoretically speaking, compared with fully connected networks, embedding LSTM networks into an attention structure helps to provide it with significantly more temporal features for calculation, improving its sensitivity to the global variations of non-periodical data. Unfortunately, such a structure has not been raised in the time series data mining community.

To take the advantages of the dual-network-based model and the hybridization of attention and LSTM networks, this paper proposes a robust temporal feature network (RTFN) for feature extraction in the area of time series classification. RTFN consists of a temporal feature network, 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 while self-attention is adopted to relate different positions of the features extracted before. To reduce the loss of features during their transmission, the leaky rectified linear unit is used as the activation function.
- The LSTM-based attention network contains two identical layers. In each layer, instead of fully connected networks, LSTM networks are used to obtain the query, key and value matrices for their corresponding attention structure. Different from the existing structures that combine attention and LSTM networks, the LSTM-based attention network can pay attention to the global variations of non-periodical data, which helps to mine useful relationships among the features already learned.

- We embed RTFN into a supervised structure, and test it on 85 UCR2018 datasets and 30 UEA2018 datasets, where the RTFN-based algorithm outperforms a number of state-of-the-art supervised algorithms on 39 UCR2018 and 14 UEA2018 datasets, respectively. Moreover, we also use RTFN in a simple unsupervised clustering as an encoder and ours wins 9 out of 36 UCR2018 datasets, 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 RTFN, its key components, the RTFN-based supervised structure and the RTFN-based unsupervised clustering. Experimental results and analysis are given in Section 4. 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) [59], 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 fea-
tures of data. For example, a 34-layer convolutional neural network was constructed to handle the ECG classification problem [43]. Serrà et al. developed a universal encoder based on CNN and convolutional attention to mine the temporal representations from input data [18]. In [22], an off-the-shelf deep CNN (ConvTimeNet) with four convolutional blocks was proposed as a transferable network to adapt quickly for the requirements of datasets. Fawaz et al. used a fast gradient sign method to fool a ResNet model, called an adversarial attacks for time series classification, where a set of synthetic samples was generated [10]. Besides, InceptionTime [12] and OS-CNN [53] 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 while OS-CNN adopts 1-dimensional CNN to mine local features and the relationships among them. On the other hand, as an emerging trend for time series classification, the dual-network-based models have not received much research attention yet. A few LSTM-FCN-based models were designed to cope with univariate and multivariate time series classification problems [19, 20], where FCN and LSTM were used for feature and relation extraction, respectively. In [18], Huang et al. proposed a residual attention net (RAN) consisting of a ResNet-based feature network and a transformer-based relation network, obtaining promising performance on UCR datasets. Meanwhile, the dual-network-based algorithms usually achieved better classification performance than those single-network-based ones [18, 19, 20].

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 attention and LSTM structures together. For instance, an attention-LSTM model was adopted to cope with univariate and multivariate time series classification on UCR and UEA datasets, respectively [19, 20]. An attention-LSTM model was integrated into
the convolution–deconvolution word embedding to merge context-specific and
task-specific information [51]. In [64], an attention-based time-incremental CNN
cascaded attention and LSTM networks for temporal and spatial information
fusion of ECG signals. Besides, the cascading models have been applied to video
segmentation [57], semantic relation extraction [14], visual question answer [68],
urban flow prediction [46], and so on. On the other hand, the embedding models
focus on the compact integration of attention and LSTM networks. For instance,
a TAG-embedded LSTM model was devised to explore the local variations of
quasi-periodic time series data [30]. Wang et al. [70] proposed a novel on-
line attentional recurrent neural network (ARNN) model for video track, where
inter- and intra-attention models was embedded into a bi-directional LSTM to
distinguish different background scenarios. In [6], Chen et al. put forward an
identity-aware single shot multiboxes detector for object detection, where an
attention-embedded LSTM structure was used to locate positions of interesting objects. In [52], an attention-based LSTM was introduced to capture the
representation hierarchy of data in power consumption forecasting.

2.3. Analysis and Motivation

The dual-network-based models realize feature and relation extraction by
two separate networks in parallel. Such a model usually performs better than a
single-network-based model in supervised classification and unsupervised clus-
tering, according to references [18, 19, 20] and our observations in Section 4.4.
Nevertheless, designing a dual-network-based model is quite challenging since
its structure should be constructed properly according to the requirements of
datasets, especially its relation extraction network. The hybridization of atten-
tion and LSTM offers a promising means to discover the relationships among the
representations obtained from data. But, the existing cascading and embedding
models cannot well handle the global variations of non-periodical time series
data. The above is the motivation why we design a dual-network-based algo-
thesis for time series classification and why we embed LSTM into an attention
structure for relation extraction.
3. RTFN

This section first overviews the structure of RTFN, and then describes two important components, namely a convolutional neural block and an LSTM-based attention layer. In the end, the RTFN-based supervised structure and unsupervised clustering are introduced.

![Figure 1: Structure of RTFN.](image)

3.1. Overview

The structure of RTFN is shown in Fig. 1. It primarily consists of a temporal feature network (TFN) and an LSTM-based attention network (LSTMaN). In TFN, a convolutional neural block, namely ‘Conv1D’, is seen as a 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 lower-level features extracted before. Besides, we place a self-attention layer \[55\] between the two multi-head layers to relate the positions of the local features obtained by the first multi-head layer, which helps to enrich 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 at digging out the intrinsic relationships among the features learned from the input, which helps to compensate for the loss of the representations ignored by TFN. TFN and LSTMaN are used as the local-feature and relation extraction networks, respectively. Combining them together provides RTFN with sufficient features and relationships. In this paper, RTFN is embedded into a supervised structure in Section 3.4 and an unsupervised clustering in Section 3.5, respectively.

3.2. Convolutional Neural Block (Conv1D)

A Conv1D block consists of a 1-dimensional CNN module, a batch normalization module and a leaky rectified linear unit (LeakReLU) activation function [45]. To be specific, the CNN module is used to explore the local features from the input [26]. The batch normalization module eliminates the internal covariate shift and is thus able to ensure a faster training process. This module also helps to regularize the proposed RTFN and enhance its local-feature extraction ability in supervised classification and unsupervised clustering. Meanwhile, different from the rectified linear unit (ReLU) that only considers positive numbers, LeakReLU takes care of both positive and negative numbers, reducing the loss of features during the data transmission process. Detailed observations are shown in Section 4.3. The LeakReLU activation is defined in Eq. (1).

\[
f_{\text{LeakReLU}}(x) = \begin{cases} 
\alpha x, & x < 0 \\
x, & x \geq 0 
\end{cases}
\]  

(1)

where, \(x\) is the input of the LeakReLU unit and \(\alpha\) is a coefficient for negative numbers. Following the widely recognized YOLOv3 [45], we set \(\alpha = 0.1\) in this paper.

3.3. LSTM-based attention Layer

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

![Figure 2: Structure of the LSTM-based attention layer.](image)

Different from the existing models that hybridize attention and LSTM networks (see Section 2.2), an LSTM-based attention layer incorporates LSTM networks into an attention structure. In this network, a temporal query and a set of key-value pairs are mapped to an output, where the 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 paper, each value is obtained by a compatibility function that helps to mine the hidden relationships between a query and its corresponding key that already carry basic features. This helps to strengthen the robustness of each LSTM-embedded attention layer, where detailed observations can be seen in Section 4.3. The query, key and value matrices output by the three LSTM networks, $I_q$, $I_k$, $I_v$, are defined in Eqs. (2), (3) and (4), respectively.

\[
I_q = f_{LSTM-Q}(x) \tag{2}
\]

\[
I_k = f_{LSTM-K}(x) \tag{3}
\]

\[
I_v = f_{LSTM-V}(x) \tag{4}
\]

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.
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. (5).

\[ O_{Att} = f_{\text{SoftMax}}(I_q \cdot I_k^T) \cdot I_v \] (5)

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

3.4. RTFN-based Supervised Structure

The RTFN-based supervised structure is shown in Fig. 3, where a dropout layer and a fully-connected layer are cascaded to the output of RTFN. To be specific, we introduce the dropout layer to avoid overfitting during the training process. The fully-connected layer functions as the classifier. The reason we simply use the dropout and fully-connected layers is that the features extracted by RTFN are sufficiently good and thus a complicated classifier network is not necessary. Like other commonly used supervised algorithms [2, 11, 18, 20], we use the cross entropy function to compute the average difference between the ground truth labels and their corresponding prediction results, $L_{\text{super}}$, as written in Eq. (6).

\[ L_{\text{super}} = -\frac{1}{n} \sum_{i=1}^{n} (\hat{Y}_i^{\text{train}} \log(p_i)) \] (6)
where, \( n \) is the number of samples, and \( \hat{Y}^{train}_{i} \) and \( p_{i} \), \( i = 1, 2, ..., n \), 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 auto-encoder unsupervised structure, as shown in Fig. 4. To be specific, it is primarily composed of a RTFN-based encoder, a decoder and a K-means algorithm [17]. RTFN is responsible for 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 RTFN. Besides, the K-means algorithm acts as the unsupervised classifier.

Different from taking the k-means loss into account [15, 31, 33], the RTFN-based unsupervised clustering simply depends on the reconstruction loss (i.e. the mean square error), \( \mathcal{L}_{rec} \), as defined in Eq. (7).

\[
\mathcal{L}_{rec} = \frac{1}{n} \sum_{i=1}^{n} (\hat{X}^{train}_{i} - X^{rec}_{i})^2
\]  

where \( \hat{X}^{train}_{i} \) and \( X^{rec}_{i} \), \( i = 1, 2, ..., n \), are the input and the decoder output of the \( i \)-th sample, respectively.
4. Experiments and Analysis

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

4.1. Experimental Setup

Extensive experiments in supervised classification and unsupervised clustering have been conducted. This section introduces the standard datasets first and the implementation details later.

Supervised Classification Datasets. We evaluate the performance of the RTFN-based supervised structure by a number of univariate and multivariate time series datasets. For the univariate time series, UCR2018 [7] is one of the authoritative data archives, which contains 128 datasets with different lengths in a variety of application areas. We select 85 standard datasets from the UCR2018 archive, consisting of 65 'short-medium' and 20 'long' time series datasets. In this paper, a 'long' dataset is a dataset with a length of over 500. The details of these datasets are shown in Table 1. For the multivariate time series, UEA2018 [1] is a commonly used data archive, including 30 datasets in seven application areas, namely audio spectra, electrocardiogram, electroencephalogram, human activity recognition, motion, eagnetoencephalography and other. Their details are listed in Table 2.

Unsupervised Clustering Datasets. Following the protocol used in [15, 31, 33], we verify the performance of the RTFN-based unsupervised clustering by 36 standard datasets selected from the UCR2018 archive. They are marked with 'YES' in column 'Unsupervised' in Table 1.

Implementation Details. Firstly, we introduce the parameter settings for TFN. As mentioned in Section 3.2, each Conv1D block contains a 1-dimensional CNN module. In the two Conv1D blocks next to the input, each 1-dimensional CNN module has 128 channels, each with a kernel size of 11. In the Conv1D block directly connected to the residual junction, the 1-dimensional CNN module also has 128 channels, each with a kernel size of 1. In each multi-head
| Classes | Features | Duration | Description | Type | Supervised |
|---------|----------|----------|-------------|------|-------------|
| 1194 | 1000 | 896 | YES | YES | YES |
| 2439 | 370 | 23 | YES | YES | YES |
| 6164 | 810 | 17 | YES | YES | YES |
| 1000 | 1320 | 2 | YES | YES | YES |
| 896 | 2524 | 8 | YES | YES | YES |
| 896 | 2524 | 8 | YES | YES | YES |
| 896 | 2524 | 8 | YES | YES | YES |
| 1000 | 1320 | 2 | YES | YES | YES |
| 1000 | 1320 | 2 | YES | YES | YES |
| 1000 | 1320 | 2 | YES | YES | YES |

Table 1: Details of the 85 univariate time series datasets. Those marked with 'YES' are also used for unsupervised clustering experiments.
Table 2: Details of the 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       | BAR          |
| CT    | CharacterTrajectories     | 1422      | 1436     | 3             | 182          | 20      | Motion       |
| CR    | Cricket                  | 108       | 72       | 6             | 1197         | 12      | BAR          |
| DDG   | DuckDuckGeeses           | 50        | 50       | 1345          | 270          | 5       | AS           |
| EW    | EigenWorm                | 128       | 131      | 6             | 17894        | 4       | Motion       |
| EP    | Epilepsy                 | 137       | 138      | 3             | 206          | 4       | BAR          |
| EC    | EthanolConcentration     | 284       | 283      | 3             | 1751         | 4       | BAR          |
| ER    | Elling                   | 30        | 270      | 4             | 65           | 6       | Other        |
| FD    | FaceDetection            | 5890      | 3524     | 144           | 62           | 2       | EGG/MEG      |
| FM    | FingerMovements          | 336       | 100      | 28            | 50           | 2       | EGG/MEG      |
| RMD   | HandMovementDirection    | 160       | 74       | 10            | 400          | 4       | EGG/MEG      |
| HW    | Handwriting              | 150       | 850      | 3             | 152          | 26      | BAR          |
| HH    | Heartbeat                | 204       | 205      | 61            | 405          | 2       | AS           |
| EW    | InsectWingbeat           | 30000     | 28000    | 200           | 30           | 10      | AS           |
| JV    | JapaneseVowels           | 270       | 370      | 12            | 20           | 9       | AS           |
| LIB   | Libras                   | 180       | 180      | 2             | 45           | 15      | BAR          |
| LSST  | LSST                     | 2459      | 2466     | 6             | 36           | 14      | Other        |
| MI    | MotorImagery             | 278       | 100      | 64            | 1000         | 2       | EGG/MEG      |
| NATO  | NATOIPS                  | 180       | 180      | 24            | 51           | 6       | BAR          |
| PD    | ProDigits                | 7094      | 3498     | 2             | 8            | 10      | EGG/MEG      |
| PEMS  | PEMS                     | 287       | 173      | 963           | 144          | 7       | EGG/MEG      |
| PS    | Phoneme                  | 3315      | 3353     | 11            | 217          | 39      | AS           |
| RS    | RacketSports             | 154       | 152      | 6             | 30           | 4       | BAR          |
| SRS1  | SelfRegulationSCP1       | 268       | 293      | 6             | 896          | 2       | EGG/MEG      |
| SRS2  | SelfRegulationSCP2       | 200       | 180      | 7             | 1152         | 2       | EGG/MEG      |
| SAD   | SpokenArabicDigits       | 6699      | 6599     | 13            | 93           | 10      | AS           |
| SWJ   | StandWalkJump            | 11        | 15       | 4             | 2500         | 3       | ECG          |
| UW    | UWaveGestureLibrary      | 120       | 320      | 3             | 315          | 8       | BAR          |

Conv1D layers, each of the four 1-dimensional CNN modules has 32 channels. Besides, these modules are with kernel sizes of 5, 8, 11 and 17, respectively.

Secondly, we introduce the parameter settings for LSTMaN. As described in Section 3.3, there are two LSTM-based attention layers. In each layer, the number of hidden units in each LSTM network is set to 128.

Last but not least, we dynamically adjust the learning rate during the training process. Let the total number of training epochs and the size of each decay period denoted by $N_{tot}$ and $N_{dec}$, respectively. Let $l_{rate}(j), j = 1, 2, ..., J$, 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. (8).

$$l_{rate}(j) = (1 - d_{rate}) \times l_{rate}(j - 1)$$ (8)
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 is used to tune the parameters of our proposed RTFN structures for supervised classification and unsupervised clustering.

All experiments are run on a computer with Ubuntu 18.04 OS, a Nvidia GTX 1070Ti GPU with 8GB, a Nvidia GTX 1080Ti GPU with 11GB 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 adopt 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 [2, 9, 11, 18].

**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) [44], \( RI \), as defined in Eq. (9).

\[
RI = \frac{PTP + NTP}{s(s-1)/2} \tag{9}
\]

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’.

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.

Table 3: The top-1 accuracy results of different supervised algorithms on 12 selected datasets.

| Dataset Series | Type | TFN | TFN w ReLU | TFN w/o SelAtt | TFN+1LSTM | TFN+2LSTM | TFN+3LSTM | TFN+AttLSTM | TFN+LSTM+TAG |
|----------------|------|-----|------------|----------------|-----------|-----------|-----------|-------------|---------------|
| Univariate Time Series | 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 |
| | Lighting7 | 0.5373 | 0.5119 | 0.4966 | 0.5729 | 0.6230 | 0.6230 | 0.5770 | 0.5770 |
| | AVG RI | 0.598 | 0.573 | 0.5502 | 0.6222 | 0.7085 | 0.7085 | 0.6931 | 0.6931 |

Table 4: The RI results of different unsupervised algorithms on 4 selected datasets.

| Dataset Series | Type | TFN | TFN w ReLU | TFN w/o SelAtt | AVG RI |
|----------------|------|-----|------------|----------------|-------|
| Beef | 0.6267 | 0.5941 | 0.5402 | 0.7034 |
| Car | 0.6418 | 0.6354 | 0.626 | 0.6708 |
| ECG200 | 0.6533 | 0.6315 | 0.6018 | 0.7018 |
| Lighting7 | 0.5373 | 0.5119 | 0.4966 | 0.5770 |
| AVG RI | 0.598 | 0.573 | 0.5502 | 0.6222 |

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 the performance of a number of 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.

For the performance comparison of supervised classification, 12 datasets are selected from the UCR 2018 and UEA 2018 archives, including 8 univariate
and 4 multivariate datasets. Besides, these univariate datasets contain 4 'short-
medium' and 4 'long' time series datasets. As for the performance comparison
of unsupervised clustering, we select 4 univariate datasets from the UCR 2018 archive.

The top-1 accuracy and RI results obtained by different supervised and un-
supervised algorithms are shown in Table 3 and Table 4, 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. Different
from ReLU that focuses on positive numbers only, LeakReLU makes use of
both 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. This is why LeakReLU improves both the supervised
and unsupervised performance of TFN, compared with ReLU. Then, we com-
pare TFN and TFN w/o SelAtt in term of the supervised classification and
unsupervised clustering. Apparently, TFN overweighs TFN w/o SelAtt on all
the datasets. The reason behind this, is the SelAtt layer can relate different
positions of time series data, enriching the extracted features. So, embedding
the SelAtt layer in TFN helps to enhance its supervised and unsupervised per-
formance. Therefore, LeakReLU and self-attention are necessary for TFN.

4.3.2. The LSTM-based Attention Network

In RTFN, LSTMaN consists of two LSTM-based attention layers. To inves-
tigate the effectiveness of LSTMaN, we compare a number of 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 [20].
- LSTMTAG: an embedding attention-LSTM model, where a trend atten-
tion gate is embedded into an LSTM structure [30].

To make a fair comparison, TFN is used in each RTFN structure as the local-feature extraction network. In other words, no matter for supervised classification or unsupervised clustering, the corresponding RTFN structures are exactly the same, except for their relation-extraction components.

Firstly, we study the impact of the number of LSTM-based attention layers on the performance of RTFN. Between TFN+2LSTMaL and TFN+1LSTMaL, one can observe from Tables 3 and 4 that the former always performs better than the latter. In the 2LSTMaL structure, the second layer unfolds the details of the relationships among the features captured by the first layer and hence can discover those complicated representations ignored before. This is why 2LSTMaL mines more intricate relationships hidden in the data than 1LSTMaL. If comparing TFN+2LSTMaL and TFN+3LSTMaL, one can see that the two achieve equivalent performance in almost all cases except dataset ‘Car’, as illustrated in Tables 3 and 4. The following explains why. In the 3LSTMaL structure, the third layer is supposed to further extend the details of the relationships among those features extracted by the first and second layers. However, all intrinsic details have been explicitly unveiled in the second layer. In this case, the third layer only acts as an information transmission layer. This layer not only may lead to loss of features during their transmission, but also consumes additional computing resources, especially to complicated datasets. Actually, 2LSTMaL aims at striking a balance between accuracy and model complexity.

To further support this, we show the model complexity comparison of different supervised algorithms on 4 long time series datasets in Table 5. It is easily seen that TFN+2LSTMaL has a lower model complexity than TFN+3LSTMaL, e.g. their CPU time on dataset ‘SemgHandGendeCh2’ is 32.421894s and 35.440211s, respectively.

Secondly, we investigate the effectiveness of the proposed LSTMaN by comparing it with two well recognized models based on attention and LSTM. It can been seen from Tables 3 and 4 that TFN+AttLSTM and TFN+LSTMTAG are beaten by TFN+2LSTMaL on each dataset for supervised classification or
Table 5: Computational complexity comparison of TFN+1LSTMaL, TFN+2LSTMaL and TFN+3LSTMaL in terms of 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        | SemgHandGenderCh2        | 2.56755        | 30.277891   | 1.506916            | 1.603434            |
| TFN+2LSTMaL        | SemgHandGenderCh2        | 2.995131       | 32.421894   | 2.127678            | 2.429863            |
| TFN+3LSTMaL        | SemgHandGenderCh2        | 3.262915       | 35.440211   | 2.893432            | 3.129045            |
| TFN+1LSTMaL        | SemgHandMovementCh2      | 3.105615       | 21.771511   | 1.305190            | 1.537634            |
| TFN+2LSTMaL        | SemgHandMovementCh2      | 3.626167       | 24.561916   | 1.903635            | 2.028537            |
| TFN+3LSTMaL        | SemgHandMovementCh2      | 4.034311       | 26.680317   | 2.465214            | 2.634242            |
| TFN+1LSTMaL        | SemgHandSubjectCh2       | 3.123979       | 20.747758   | 1.393914            | 1.529973            |
| TFN+2LSTMaL        | SemgHandSubjectCh2       | 3.426036       | 24.651912   | 1.964818            | 2.098713            |
| TFN+3LSTMaL        | SemgHandSubjectCh2       | 3.839362       | 26.720378   | 1.514096            | 2.795453            |
| TFN+1LSTMaL        | Rock                     | 4.737211       | 6.989938    | 1.237770            | 1.328797            |
| TFN+2LSTMaL        | Rock                     | 5.042245       | 8.687216    | 1.369962            | 1.576917            |
| TFN+3LSTMaL        | Rock                     | 5.453398       | 10.772195   | 1.606344            | 2.103435            |

unsupervised clustering. The following explains the reasons. On the one hand, AttLSTM lacks of 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 concentrate on the local variations of periodical data, due to the LSTM structure with TAG embedded. It is, however, not sensitive to the global variations of nonperiodical data, which is not beneficial to discover complex connections hidden in the data, especially when facing long univariate datasets and multivariate ones. For instance, for TFN+LSTMTAG, its top-1 accuracy values on datasets 'Rock' and 'AF' are only 0.68 and 0.2, respectively. On the other hand, compared with TFN, TFN+2LSTMaL always obtains a higher accuracy value on each dataset, no matter in the aspect of supervised classification or unsupervised clustering. This clearly unveils that LSTMaN plays a non-trivial role in the performance improvement. This is because the two LSTMaL are capable of extracting those intricate representations that may be ignored by TFN. In other words, LSTMaN and TFN well complement each other in RTFN.

4.4. Evaluation of the RTFN-based Supervised Structure

To evaluate the performance of the RTFN-based supervised structure, we compare it with a number of state-of-the-art supervised algorithms against 'win' / 'lose' / 'tie', MeanACC and AVG_rank on 85 univariate and 30 multivariate
| Datasets | DistalPhalanxOutlineAgeGroup | GunPointMaleVersusFemale | SonyAIBORobotSurface1 | UWaveGestureLibraryAll | SemgHandMovementCh2 | UWaveGestureLibraryX | SemgHandGenderCh2 | DiatomSizeReduction | ProximalPhalanxTW | ItalyPowerDemand | GunPointAgeSpan | WordSynonyms | HandOutlines | TwoPatterns | Earthquakes | ECG5000 | CricketZ | OliveOil | ACSF1 | Wafer | Adiac | Table 6: Results of different supervised algorithms on 85 selected datasets. |
|----------|-----------------------------|--------------------------|-----------------------|-------------------------|---------------------|-------------------|-----------------|-------------------|----------------|----------------|----------------|-------------|--------------|------------|-------------|----------|--------|--------------|--------|----------|-------------|-----------------|
| **FNN**  | 0.9482                      | 0.944                      | 0.9402                | 0.925                    | 0.9402              | 0.9448            | 0.9418          | 0.8978            | 0.9487          | 0.919          | 0.9541        | 0.832        | 0.835        | 0.723       | 0.833       | 0.86          | 0.75   | 0.75     | 0.75        | 0.75             |
| **CNN**  | 0.9949                      | 0.9737                    | 0.899                 | 0.9649                   | 0.9649              | 0.9956            | 0.9956          | 0.9768            | 0.9936          | 0.9873         | 0.9988        | 0.9936      | 0.9936      | 0.9768      | 0.9988      | 0.9988       | 0.9768 | 0.9988  | 0.9988       | 0.9988            |
| **RNN**  | 0.9949                      | 0.8409                    | 0.807                 | 0.9802                   | 0.8367              | 0.9956            | 0.9956          | 0.9768            | 0.9936          | 0.9873         | 0.9988        | 0.9936      | 0.9936      | 0.9768      | 0.9988      | 0.9988       | 0.9768 | 0.9988  | 0.9988       | 0.9988            |
| **CRF**  | 1                           | 1                         | 1                     | 1                        | 1                    | 1                 | 1               | 1                 | 1                | 1              | 1              | 1            | 1            | 1           | 1           | 1            | 1      | 1       | 1            | 1                 |
| **SVM**  | 0.9882                      | 0.843                      | 0.836                 | 0.9802                   | 0.8367              | 0.9956            | 0.9956          | 0.9768            | 0.9936          | 0.9873         | 0.9988        | 0.9936      | 0.9936      | 0.9768      | 0.9988      | 0.9988       | 0.9768 | 0.9988  | 0.9988       | 0.9988            |
| **C4.5** | 0.9949                      | 0.9737                    | 0.899                 | 0.9649                   | 0.9649              | 0.9956            | 0.9956          | 0.9768            | 0.9936          | 0.9873         | 0.9988        | 0.9936      | 0.9936      | 0.9768      | 0.9988      | 0.9988       | 0.9768 | 0.9988  | 0.9988       | 0.9988            |
| **DAF**  | 0.9949                      | 0.8409                    | 0.807                 | 0.9802                   | 0.8367              | 0.9956            | 0.9956          | 0.9768            | 0.9936          | 0.9873         | 0.9988        | 0.9936      | 0.9936      | 0.9768      | 0.9988      | 0.9988       | 0.9768 | 0.9988  | 0.9988       | 0.9988            |
| **LR**   | 0.9949                      | 0.8409                    | 0.807                 | 0.9802                   | 0.8367              | 0.9956            | 0.9956          | 0.9768            | 0.9936          | 0.9873         | 0.9988        | 0.9936      | 0.9936      | 0.9768      | 0.9988      | 0.9988       | 0.9768 | 0.9988  | 0.9988       | 0.9988            |

21
| Dataset       | Classes | SeriesLength | Existing SOTA | ConvTimeNet | EE Survey | COTE | BOSS | Asst | RTFN | RTFN | Vanilla: ResNet-Transformer | Vanilla: ResNet-Transformer | Vanilla: ResNet-Transformer | Vanilla: ResNet-Transformer | Vanilla: ResNet-Transformer | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fcn | Best: lstm-fc...
time series cases, we show the top-1 accuracy results of different algorithms in Table 7, where all the 20 'long' time series datasets in Table 1 are tested. It is easily observed that the RTFN-based supervised structure, i.e. ours, is the best algorithm as it obtains the highest 'Best' and MeanACC values. This is because RTFN is able to mine sufficient local features and the relationships among them, thanks to the efficient cooperation of TFN and LSTMaN. Especially, LSTMaN relates different locations of the obtained representations and can thus capture their intrinsic regularizations during the data transmission process.

On the other hand, Best:lstm-fcn also achieves decent performance with respect to 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 no doubt the one with the best performance among all the transformer-based networks for comparison. The reason behind this, is 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 8 shows the top-1 accuracy results obtained by different supervised algorithms on the 30 datasets in the UEA 2018 archive. For each dataset, the existing SOTA represents the best algorithm on that dataset, including STC [39], HC [25], gRSF [21], and so on; similarly, Best:DTW, Best:DTWN, and Best:EDN are the best performance DTW-based (e.g. involving DTW_I and DTW_A in [9]), DTWN-based (e.g. involving DTW-1NN_I(n) and DTW-1NN_D(n) [9]) and ED-NN-based (e.g. involving ED-1NN and ED-1NN(Normalized) [39]) approaches on that dataset, respectively.

It is observed that the RTFN-based supervised structure, i.e. ours, performs the best among all algorithms for comparison since ours obtains the highest MeanACC and 'best' values, namely, 0.763 and 14, and the smallest AVG_rank score, namely 3.833. Meanwhile, MF and LCEM are the second and third best algorithms while Best:EDN is the worst one. The following explains why. On the one hand, ours takes advantages of TFN and LSTMaN to mine sufficient local features and the relationships among them, thanks to the efficient cooperation of TFN and LSTMaN. Especially, LSTMaN relates different locations of the obtained representations and can thus capture their intrinsic regularizations during the data transmission process.
achieve decent performance in supervised classification. On the other hand, among dimensions at different timestamps. This is why MF and LCEM also.

This allows LCEM to capture enough complex relationships associated with multiple variates as many as possible. In the meantime, LCEM and MF (i.e. MLSTM-FCN) can learn significant connections among the features.

Table 8: Results of different supervised algorithms on all UEA2018 datasets. Abbreviations: MF – MLSTM-FCN [20], WM – WEASET + MUSE [17].

| Dataset | Existing | Best: | Best: | LSTM | MOB | RNN | BC | BM | CR | CT | DDG | EW | F | JV | N | Ours |
|---------|----------|------|------|------|-----|-----|----|----|----|----|-----|----|---|----|---|-----|
| NATO    | 0.900    | 0.900 | 0.900 | 0.900 | 0.883 | 0.880 | 0.867 | 0.883 | 0.867 | 0.867 | 0.883 | 0.880 | 0.900 | 0.880 | 0.880 |
| LSST    | 0.867    | 0.867 | 0.867 | 0.867 | 0.867 | 0.867 | 0.867 | 0.867 | 0.867 | 0.867 | 0.867 | 0.867 | 0.867 | 0.867 | 0.867 |
| SAD     | 0.900    | 0.900 | 0.900 | 0.900 | 0.891 | 0.888 | 0.888 | 0.888 | 0.888 | 0.888 | 0.888 | 0.888 | 0.888 | 0.888 | 0.888 |
| SWJ     | 0.900    | 0.900 | 0.900 | 0.900 | 0.891 | 0.888 | 0.888 | 0.888 | 0.888 | 0.888 | 0.888 | 0.888 | 0.888 | 0.888 | 0.888 |
| SRS1    | 0.900    | 0.900 | 0.900 | 0.900 | 0.891 | 0.888 | 0.888 | 0.888 | 0.888 | 0.888 | 0.888 | 0.888 | 0.888 | 0.888 | 0.888 |
| SRS2    | 0.900    | 0.900 | 0.900 | 0.900 | 0.891 | 0.888 | 0.888 | 0.888 | 0.888 | 0.888 | 0.888 | 0.888 | 0.888 | 0.888 | 0.888 |
| SAD     | 0.900    | 0.900 | 0.900 | 0.900 | 0.891 | 0.888 | 0.888 | 0.888 | 0.888 | 0.888 | 0.888 | 0.888 | 0.888 | 0.888 | 0.888 |
| CR      | 0.867    | 0.867 | 0.867 | 0.867 | 0.867 | 0.867 | 0.867 | 0.867 | 0.867 | 0.867 | 0.867 | 0.867 | 0.867 | 0.867 | 0.867 |
| CT      | 0.867    | 0.867 | 0.867 | 0.867 | 0.867 | 0.867 | 0.867 | 0.867 | 0.867 | 0.867 | 0.867 | 0.867 | 0.867 | 0.867 | 0.867 |
| DDG     | 0.883    | 0.883 | 0.883 | 0.883 | 0.883 | 0.883 | 0.883 | 0.883 | 0.883 | 0.883 | 0.883 | 0.883 | 0.883 | 0.883 | 0.883 |
| EW      | 0.900    | 0.900 | 0.900 | 0.900 | 0.891 | 0.888 | 0.888 | 0.888 | 0.888 | 0.888 | 0.888 | 0.888 | 0.888 | 0.888 | 0.888 |
| F       | 0.900    | 0.900 | 0.900 | 0.900 | 0.891 | 0.888 | 0.888 | 0.888 | 0.888 | 0.888 | 0.888 | 0.888 | 0.888 | 0.888 | 0.888 |
| JV      | 0.900    | 0.900 | 0.900 | 0.900 | 0.891 | 0.888 | 0.888 | 0.888 | 0.888 | 0.888 | 0.888 | 0.888 | 0.888 | 0.888 | 0.888 |
| N       | 0.900    | 0.900 | 0.900 | 0.900 | 0.891 | 0.888 | 0.888 | 0.888 | 0.888 | 0.888 | 0.888 | 0.888 | 0.888 | 0.888 | 0.888 |
| Ours    | 0.900    | 0.900 | 0.900 | 0.900 | 0.891 | 0.888 | 0.888 | 0.888 | 0.888 | 0.888 | 0.888 | 0.888 | 0.888 | 0.888 | 0.888 |

features from the input data. In particular, LSTMaN can discover the relationships among not only the representations associated with each variate, but also those associated with different variates. This is why ours achieves the best performance. Due to the efficient coordination of LSTM and FCN networks, MF (i.e. MLSTM-FCN) can learn significant connections among the features associated with multiple variates as many as possible. In the meantime, LCEM uses an explicit boosting-bagging approach to explore the interactions among the dimensions. This allows LCEM to capture enough complex relationships among dimensions at different timestamps. This is why MF and LCEM also achieve decent performance in supervised classification. On the other hand, Best:EDN is based on the traditional DTW-NN approach. This makes it quite challenging to simultaneously focus on the useful representations in univariate
data and the relationships among them in multivariate data. This is why deep learning approaches have attracted increasingly more research attention. The AVG_rank results obtained by different supervised algorithms on 30 multivariate datasets are shown in Fig. 5.

![Figure 5: Results of the AVG_ranks of different algorithms on 30 multivariate datasets.](image)

4.5. Evaluation of the RTFN-based Unsupervised Clustering

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

Following some well recognized research works [15, 31, 33, 35, 62], we select 36 representative datasets from the UCR 2018 archive for performance evaluation and they are marked with 'YES’ in Table 1. The RI results obtained by different unsupervised algorithms on the 36 datasets are shown in Table 9.

Obviously, DTCR and our RTFN-based unsupervised clustering rank the best and second best among all algorithms for comparison. DTCR takes advantage of an seq2seq structure to explore sufficient temporal features for a K-means classifier. This algorithm uses the loss of the classifier to update its model parameters, encouraging the representations extracted from data to form a cluster structure. This is why DTCR is good at mining cluster-specific representations from input data. Its complicated structure is, however, for addressing cluster-specific problems only. On the contrary, ours simply adopts an auto-encoding
structure to update the parameters of our model and utilizes a K-means algorithm to classify the features obtained by RTFN. Although it is quite simple in structure, our achieves decent performance on the 36 selected datasets, which depends on the strong feature extraction ability of RTFN.

In order to further evaluate the effectiveness of RTFN in unsupervised clustering, we compare the RTFN-based K-means algorithm with a separate K-means algorithm on the 36 datasets above and show the RI results in Fig. 6. One can see that ours outperforms the separate K-means algorithm on all but two datasets. This is because our unsupervised clustering is provided with sufficient features obtained by the proposed RTFN, especially those hiding deeply in the input data which are beyond the exploration abilities of ordinary feature extraction methods.
extraction networks. The AVG_rank results of all unsupervised algorithms for comparison are shown in Fig. 7.

![Figure 6: The RI values obtained by the K-means algorithm and ours.](image)

![Figure 7: AVG_ranks of different unsupervised algorithms.](image)

5. Conclusion

In the proposed RTFN, the temporal feature network is responsible for extracting local features while the LSTM-based attention network aims at discovering intrinsic relationships among the representations learned from data. Experimental results demonstrate that our RTFN achieves decent performance in both supervised classification and unsupervised clustering. Specifically, our RTFN-based supervised algorithm performs the best on 39 out of 85 univariate datasets in the UCR2018 archive and 14 out of 30 multivariate datasets in the UEA2018 archive, respectively, compared with the latest results from the
supervised classification community. In particular, ours wins 11 out of 20 long univariate dataset cases. Our RTFN-based unsupervised algorithm is the second best when considering all 36 datasets. Last but not least, the experimental results also indicate that RTFN has good potential to be embedded into other learning frameworks to handle time series problems of various domains in the real world.

References

[1] A. Bagnall, H. A. Dau, J. Lines, M. Flynn, J. Large, A. Bostrom, P. Southam, and E. Keogh. The uea multivariate time series classification archive, 2018. arXiv preprint arXiv: 1811.00075, 2018.

[2] A. Bagnall., J. Lines, A. Bostrom, J. Large, and E. Keogh. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. Data Min. Knowl. Disc., 31:1–55, 2017.

[3] M. G. Baydogan and G. Runger. Time series representation and similarity based on local autopatterns. Data Min. Knowl. Disc., 30(2):476–509, 2016.

[4] Z. Che, S. Purushotham, K. Cho, D. Sontag, and Y. Liu. Recurrent neural networks for multivariate time series with missing values. Scientific Reports, 8(1):6085, 2018.

[5] J. Chen, Z. Xiao, H. Xing, P. Dai, S. Luo, and M. A. Iqbal. Stdpg: a spatio-temporal deterministic policy gradient agent for dynamic routing in sdn. in Proc. IEEE ICC 2020, pages 1–6, 2020.

[6] X. Chen, J. Yu, and Z. Wu. Temporally identity-aware ssd with attentional lstm. IEEE Trans. Cybernetics, 50(6):2674–2686, 2020.

[7] H. A. Dau, A. Bagnall, C.-C. M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatana, and E. Keogh. The ucr time series archive. arXiv preprint arXiv: 1810.07758, 2018.
[8] J. Deng, A. Berg, S. Satheesh, H. Su, A. Khosla, and L. Fei-Fei. Imagenet dataset. in Proc. ILSVRC-2012, 2012. URL http://www.image-net.org/challenges/LSVRC/2012/.

[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, and P.-A. Muller. Adversarial attack on deep neural networks for time series classification. arXiv:1903.07054, 2019.

[11] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P. A. Muller. Deep learning for time series classification: a reviewer. Data Min. Knowl. Disc., 33:917–963, 2019.

[12] H. I. Fawaz, B. Lucas, G. Forestier, C. Pelletier, D. F. Schmidt, J. Weber, G.I. Webb, L. Idoumghar, P.-A. Muller, and F. Petitjean. Inception-time: finding alexnet for time series classification. arXiv preprint arXiv:1909.04939, 2019.

[13] J.-Y. Franceschi, A. Dieuleveut, and M. Jaggi. Unsupervised scalable representation learning for multivariate time series. in Proc. NeurIPS 2019, pages 1–25, 2019.

[14] Z. Geng, G. Chen, Y. Han, G. Lu, and F. Li. Semantic relation extraction using sequential and tree-structured lstm with attention. Inform. Sciences, 509:183–192, 2020.

[15] X. Guo, L. Gao, X. Liu, and J. Yin. Improved deep embedded clustering with local structure preservation. in Proc. IJCAI 2017, pages 1753–1759, 2017.

[16] J. Han, H. Liu, M. Wang, Z. Li, and Y. Zhang. Era-lstm: an efficient reram-based architecture for long short-term memory. IEEE Trans. Parallel Distrib. Syst., 31(6):1328–1342, 2020.
[17] J. A. Hartigan and M. A. Wong. Algorithm as 136: a k-means clustering algorithm. *J. Roy. Statist. Soc. Ser. C.*, 1(28):100–108, 1979.

[18] S. H. Huang, L. Xu, and C. Jiang. Residual attention net for superior cross-domain time sequence modeling. *arXiv preprint arXiv: 2001.04077*, 2020.

[19] F. Karim, S. Majumdar, and H. Darabi. Insights into lstm fully convolutional networks for time series classification. *IEEE Access*, 7:1328–1342, 2019.

[20] F. Karim, S. Majumdar, H. Darabi, and S. Harford. Multivariate lstm-fcns for time series classification. *Neural Networks*, 116:237–245, 2019.

[21] I. Karlsson, P. Papapetrou, and H. Boström. Generalized random shapelet forests. *Data Min. Knowl. Disc.*, 30(5):1053–1083, 2016.

[22] K. Kashiparekh, J. Narwariya, P. Malhotra, L. Vig, and G. Shroff. Convtimenet: A pre-trained deep convolutional neural network for time series classification. *arXiv preprint arXiv: 1904.12546*, 2019.

[23] M. Längkvist, L. Karlsson, and A. Loutf. A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recogn. Lett.*, 42(1):11–24, 2014.

[24] J. Large, A. Bagnall, S. Malinowski, and R. Tavenard. From bop to boss and beyond: time series classification with dictionary based classifier. *arXiv preprint arXiv:1809.06751*, 2018.

[25] J. Large, J. Lines, and A. Bagnall. A probabilistic classifier ensemble weighting scheme based on cross validated accuracy estimates. *Data Min. Knowl. Disc.*, 33(6):1674–1709, 2019.

[26] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, pages 436–444, 2015.
[27] Z. Li, Y. Yang, J. Liu, X. Zhou, and H. Lu. Unsupervised feature selection using nonnegative spectral analysis. in Proc. AAAI 2012, pages 1026–1032, 2012.

[28] J. Lines and A. Bagnall. Time series classification with ensembles of elastic distance measures. Data Min. Knowl. Disc., 29:565–592, 2015.

[29] J. Lines, S. Taylor, and A. Bagnall. Time series classification with hive-cote: The hierarchical vote collective of transformation-based ensembles. ACM Trans. Knowl. Discov. D., 21(52):1–35, 2018.

[30] F. Liu, X. Zhou, J. Cao, Z. Wang, T. Wang, H. Wang, and Y. Zhang. Anomaly detection in quasi-periodic time series based on automatic data segmentation and attentional lstm-cnn. IEEE Trans. Knowl. Data En., pages 1–14, 2020.

[31] Q. Ma, J. Zheng, S. Li, and G. W. Cottrell. Learning representations for time series clustering. in Proc. NeurIPS 2019, pages 1–11, 2019.

[32] L. Maasten. Learning discriminative fisher kernels. in Proc. ICML 2011, pages 217–224, 2011.

[33] N. S. Madiraju, S. M. Sadat, D. Fisher, and H. Karimabadi. Deep temporal clustering: fully unsupervised learning of time-domain features. arXiv preprint arXiv: 1802.01059, 2018.

[34] R.H. Mousavi, M. Khansari, and R. Rahmani. A fully scalable big data framework for botnet detection based on network traffic analysis. Inform. Sciences, 512:629–640, 2020.

[35] J. Paparrizos and L. Gravano. K-shape: efficient and accurate clustering of time series. in Proc. ACM SIGMOD 2015, pages 1855–1870, 2015.

[36] W. Pei, H. Dibeklioğlu, D. M. J. Tax, and L. van der Maaten. Multivariate time-series classification using the hidden-unit logistic model. IEEE Trans. Neur. Net. Lear., 29(4):920–931, 2018.
[37] F. Petitjean, A. Ketterlin, and P. Gancarski. A global averaging method for dynamic time warping, with the applications to clustering. *Pattern Recogn.*, 44(3):678–693, 2011.

[38] M. Pontiki, D. Galanis, J. Pavlopoulos, H. Papageorgiou, I. Androutsopoulos, and S. Manandhar. Semeval-2014 task 4: Aspect based sentiment analysis. *in Proc. 8th SemEval 2014*, 2014. URL http://alt.qcri.org/semeval2014/task4/.

[39] A. R. Puiz, M. Flynn, and A. Bagnall. Benchmarking multivariate time series classification algorithms. *arXiv preprint arXiv:2007.13156*, 2020.

[40] R. Pérez-Chacón, G. Asencio-Cortés, F. Martínez-Álvarez, and A. Troncoso. Big data time series forecasting based on pattern sequence similarity and its application to the electricity demand. *Inform. Sciences*, 540:160–174, 2020.

[41] M. Qian and C. Zhai. Robust unsupervised feature selection. *in Proc. IJCAI 2013*, pages 1621–1627, 2013.

[42] A. Quattoni, S. Wang, L. Morency, M. Collins, and T. Darrell. Hidden conditional random fields. *IEEE Trans. Pattern Anal.*, 29(10):1848–1852, 2007.

[43] 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.

[44] W. M. Rand. Objective criteria for the evaluation of clustering methods. *J. Am. Stat. Assoc.*, 166(336):845–850, 1971.

[45] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv: 1804.02767*, 2018.

[46] F. Rodrigues, I. Markou, and F. C. Pereira. Combining time-series and textual data for taxi demand prediction in event areas: a deep learning approach. *Inform. Fusion*, 49:120–129, 2019.
[47] P. Schäfer and U. Leser. Multivariate time series classification with weasel+muse. *arXiv preprint arXiv:1711.11343*, 2017.

[48] J. Serrà, S. Pascual, and A. Karatzoglou. Towards a universal neural network encoder for time series. *in Proc. CCIA 2018*, pages 120–129, 2018.

[49] L. Shi, L. Du, and Y. Shen. Robust spectral learning for unsupervised feature selection. *in Proc. IEEE ICMD 2014*, pages 977–982, 2014.

[50] M. Shokoohi-Yekta, B. Hu, H. Jin, J. Wang, and E. Keogh. Generalizing dtw to the multi-dimensional case requires an adaptive approach. *Data Min. Knowl. Disc.*, 31(1):1–31, 2017.

[51] K. Shuang, Z. Zhang, J. Loo, and S. Su. Convolution-deconvolution word embedding: an end-to-end multi-prototype fusion embedding method for natural language processing. *Inform. Fusion*, 53:112–122, 2020.

[52] C. Siridhipakul and P. Vateekul. Multi-step power consumption forecasting in thailand using dual-stage attentional lstm. *in Proc. IEEE ICITEE 2019*, pages 1–6, 2019.

[53] W. Tang, G. Long, L. Liu, T. Zhou, J. Jiang, and M. Blumenstein. Re-thinking 1d-cnn for time series classification: a stronger baseline. *arXiv preprint arXiv: 2002.10061*, 2020.

[54] K. S. Tuncel and M. G. Baydogan. Autoregressive forests for multivariate time series modeling. *Pattern Recogn.*, 73:202–215, 2018.

[55] A. Vaswani, N. Shazeer, J. Uszkoreit N. Parmar, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *in Proc. NeurIPS 2017*, pages 5998–6008, 2017.

[56] D. Wang, R. Zhang, M. Liao, M. Yang, B. Shi, X. Liu, Y. Zhou, D. Karatzas, S. Lu, C. V. Jawahar, and X. Bai. Icdar 2019 robust reading challenge on reading chinese text on signboard.
in Proc. ICDAR 2019 robust reading competitions, 2019. URL https://rrc.cvc.uab.es/?com=introduction.

[57] Q. Wang, C. Yuan, and Z. Lin. Learning attentional recurrent neural network for visual tracking. in Proc. ICME 2017, pages 1237–1242, 2017.

[58] W. Wang, Y. Shang, Y. He, Y. Li, and J. Liu. Botmark: Automated botnet detection with hybrid analysis of flow-based and graph-based traffic behaviors. Inform. Sciences, 511:284–296, 2020.

[59] Z. Wang, W. Yan, and T. Oates. Time series classification from scratch with deep neural networks: A strong baseline. in Proc. IEEE IJCNN 2017, pages 1578–1585, 2017.

[60] M. Wistuba, J. Grabocka, and L. Schmidt-Thieme. Ultra-fast shapelets for time series classification. arXiv preprint arXiv:1503.05018, 2015.

[61] J. Xie, R. Girshick, and A. Farhadi. Unsupervised deep embedding for clustering analysis. in Proc. ICML 2016, pages 478–487, 2016.

[62] J. Yang and J. Leskovec. Patterns of temporal variation in online media. in Proc. ACM WSDM 2011, pages 177–186, 2011.

[63] Y. Yang, H. Shen, Z. Ma, Z. Huang, and X. Zhou. L2, 1-norm regularized discriminative feature selection for unsupervised. in Proc. IJCAI 2011, pages 1589–1594, 2011.

[64] Q. Yao, R. Wang, X. Fan, J. Liu, and Y. Li. Multi-class arrhythmia detection from 12lead varied-length ecg using attention-based time-incremental convolutional neural network. Inform. Fusion, 53:174–182, 2020.

[65] J. Zakaris, A. Mueen, and E. Keogh. Clustering time series using unsupervised shapelets. in Proc. IEEE ICMD 2012, pages 785–794, 2012.

[66] H. Zhang, Z. Xiao, J. Wang, F. Li, and E. Szczerbicki. A novel iot-perceptive human activity recognition (HAR) approach using multi-head convolutional attention. IEEE Internet Things J., 7(2):1072–1080, 2020.
[67] S. Zhang, Y. Chen, W. Zhang, and R. Feng. A novel ensemble deep learning model with dynamic error correction and multi-objective ensemble pruning for time series forecasting. *Inform. Sciences*, 544:427–445, 2021.

[68] W. Zhang, J. Yu, H. Hu, and Z. Qin. Multimodal feature fusion by relational reasoning and attention for visual question answering. *Inform. Fusion*, 55:116–126, 2020.

[69] X. Zhang, Y. Gao, J. Lin, and C.-T. Lu. Tapnet: Multivariate time series classification with attentional prototypical network. *in Proc. AAAI 2020*, pages 1–8, 2020.

[70] Y. Zhu, C. Zhao, H. Guo, J. Wang, X. Zhao, and H. Lu. Attention couplenet: fully convolutional attention coupling network for object detection. *IEEE Trans. Image Process.*, 28(1):1170–1175, 2018.