Applied Soft Computing 128 (2022) 109494

Contents lists available at ScienceDirect

Applied Soft Computing

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

Time series classification based on temporal features

Cun Ji[a,*], Mingsen Du[a], Yupeng Hu[b], Shijun Liu[b,*], Li Pan[b], Xiangwei Zheng[a]

[a] School of Information Science and Engineering, Shandong Normal University, Jinan, China
[b] School of Software, Shandong University, Jinan, China

ARTICLE INFO

Article history:
Received 6 May 2022
Received in revised form 23 July 2022
Accepted 6 August 2022
Available online 13 August 2022

Keywords:
Time series classification
Segmentation
Temporal feature
Feature importance measures
Fully convolutional network

ABSTRACT

Along with the widespread application of Internet of things technology, time series classification have been becoming a research hotspot in the field of data mining for massive sensing devices generate time series all the time. However, how to accurately classify time series based on intuitively interpretable features is still a huge challenge. For this, we proposed a new Time Series Classification method based on Temporal Features (TSC-TF). TSC-TF firstly generates some temporal feature candidates through time series segmentation. And then, TSC-TF selects temporal feature according the importance measures with the help of a random forest. Finally, TSC-TF trains a fully convolutional network to obtain high accuracy. Experiments on various datasets from the UCR time series classification archive demonstrate the superiority of our method. Besides, we have released the codes and parameters to facilitate the community research.

1. Introduction

In the era of Internet of Things, digital sensors usually collect data periodically in widespread domains including Industry 4.0, smart city, medicine, and so on [1]. The collected data is usually sequences of observational values successively at uniform time interval, aka time series [2]. As a matter of course, discovering the valuable knowledge hidden in time series has been attracting significant interest in the data mining community recently.

In especial, time series classification (TSC), which aims to assigned time series to one predefined classes [3], arises in many real-word fields [4], including: electrocardiogram classification, emotion recognition, motor fault detection, gesture recognition, speaker identification, etc.

Time series data has the unique characteristics of large amount, random noise, high dimension and updated continuously [5]. These characteristics bring great challenges to TSC. Therefore, various TSC methods are proposed by researchers [6–8].

In many field domains, understanding reasons behind the classification decision is very important for ensuring the corresponding TSC method can be safely used on the field. Among TSC methods, feature based methods focus on exploring representative features from time series [9]. So, these methods are easier to be accepted by domain experts. For example, the method in Fig. 1 classified ECG time series as hypocalcemia or normal according to whether there is specific feature in the ECG time series. As shown in Fig. 1, this judgment process has the corresponding medical explanation: "The ECG of patients with hypocalcemia showed low or inverted T wave".

In recent years, a large number of feature based TSC methods have been proposed. Despite its exciting prospect, feature based TSC methods still face the following research challenges: (1) **Intuitive interpretability**. Most feature based TSC methods adopted statistical characteristics as features. For example, some published investigations [10–12] adopted 'CO_f1ecac' (i.e. the first $1/e$ crossing of autocorrelation function) as one representative feature to classify time series. However, domain experts may not understand the meaning of this feature. Also, they cannot directly obtain this feature, because 'CO_f1ecac' is obtained through a series of calculations. (2) **High accuracy**. Up to now, the selected features are usually combined with traditional classification methods, such as K nearest neighbor (kNN), support vector machine (SVM), Bayesian networks, random forest, and etc. Compared with traditional methods, deep learning methods usually have higher accuracy. Combining deep learning classification methods with the interpretable features to get high accuracy is another fundamental challenge.

To address the aforementioned challenges, in this paper, we proposed a novel TSC method based on Temporal Features (TSC-TF). Firstly, TSC-TF generates some temporal feature candidates with the help of one time series segmentation method [13]. Next, a random forest is used to obtain importance measures of the temporal feature candidates. Following by, TSC-SF selects the temporal features according to the importance measures. Finally, aiming to obtain high accuracy, a classifier based on Fully Convolutional Network (FCN) [14] is trained to classify the

* Corresponding authors.
E-mail addresses: jicun@sdnu.edu.cn (C. Ji), lsj@sdu.edu.cn (S. Liu).

https://doi.org/10.1016/j.asoc.2022.109494
1568-4946/© 2022 Elsevier B.V. All rights reserved.

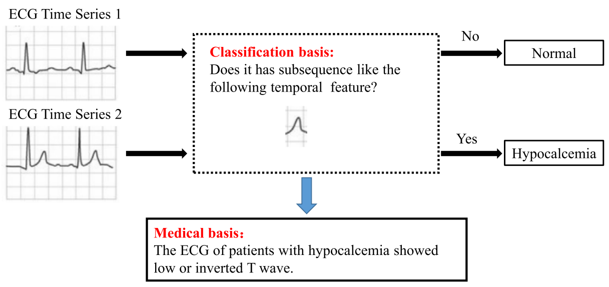

ECG Time Series 1
ECG Time Series 2
Classification basis:
Does it has subsequence like the following temporal feature?
No
Yes
Normal
Hypocalcemia
Medical basis:
The ECG of patients with hypocalcemia showed low or inverted T wave.

Fig. 1. Hypocalcemia judgment process.

transformed representation of time series based on the selected temporal features.

The main contributions of this study can be summarized as follows:

- Firstly, a temporal feature candidate generation method is introduced. This generation method selected some intuitive time series subsequences as temporal feature candidates.
- Secondly, a candidate importance measure evaluation method is put forward. This method evaluates all candidates at once. Only candidates with high importance measures will be selected as temporal features.
- Thirdly, we proposed TSC-TF, which aims to classify time series with high accuracy.
- We make experiments to show the effects of our method on various datasets from the UCR TSC archive [15]. The experimental results demonstrate that the proposed method can classify time series accuracy with the help of intuitive temporal features.

The rest of paper is structured as follow: Section 2 briefly describes some related works. Section 3 introduces our proposed method. Section 4 reports and discusses the experimental results, and our conclusions are provided in Section 5.

2. Related work

2.1. Feature based TSC methods

In recent years, a large number of feature based TSC methods have been proposed. These methods focus on mining the most representative features from time series [9].

Some feature based TSC methods [16,17] adopted the statistical features in time domain, such as: mean, maximum, minimum, standard deviation, mean absolute deviation, skewness, and kurtosis, etc. Some methods [18–20] adopted the structural features of time series, such as: trend, seasonality, periodicity, serial correlation, self-similarity, etc. Also, some researcher classified time series based on the features in frequency domain [21–23]. These methods first map time series from time domain to frequency domain through Fourier transforms or wavelet transforms, and extract the representative features from the frequency domain [24]. In addition to frequency domain, researcher extracted features through converting time series into area graph, Gramian fields, recurrence plots, Markov transition fields, and so on [25].

Of course, these features can also be used in combined. Fulcher and Jones [10] computed thousands of time series features and automatic selected a set of important features from them. Following by, Fulcher and Jones [11] introduced *hctsa*, which can automatically select the useful features from over 7,700 time series features. On the basis of this work, Lubba et al. [12] introduce the 22 canonical features (catch22) which are reduced from 4791 time series features.

In recent years, distances between time series are adopted as features for TSC [6]. As Abanda et al. [6] described, the distance features can be divided into global distance features and local distance features. Global distance features mean that the distance between different series are used as features. For example, Kate [26] used the DTW distances as features within SVMs. Local distance features mean distance to some local patterns of the series are used as features. Shapelet [27,28] based methods are the representative of local distance features.

Recently, some researcher extract features through convolutional kernels. For example, Dempster et al. [29,30] used the proportion of positive values (PPV) and max pooling operator to get two features from each convolutional kernels. In addition to PPV, Tan et al. [31] extracted mean of positive values, mean of indices of positive values and longest stretch of positive values from each convolutional kernels.

The features mentioned above have a good performance and high stability on TSC. However, these features are not directly included in the time series. So they are not interpretable enough.

2.2. Interpretable feature based TSC methods

There are mainly two categories of interpretable feature based TSC methods: dictionary based methods and shapelet based methods.

2.2.1. Dictionary based methods

Dictionary based methods classify time series according to the frequency of some similar subsequences [32]. These methods usually transformed time series into representative words through the Symbolic Aggregate approXimation (SAX) [33] or the Symbolic Fourier Approximation (SFA) [34] for counting the frequency of similar subsequences [4,35].

Similar to the “bag of words” approach widely used in the field of information retrieval, Lin et al. [36] presented the “bag of patterns” method. This method used each SAX word as a pattern in the time series and get a similarity measure through comparing the frequencies of these SAX patterns. On this basis, Senin and Malinchik [37] applied vector space model to form the word distributions by using *tf* $*$ *idf* weighting and Cosine similarity. To improve the noise tolerance, Schäfer proposed the Bag of SFA Symbols (BOSS) method which replaced SAX with SFA to obtain the word pattern. Following by, Schäfer combined BOSS model with vector space classifier [38]. To achieve both fast and accurate, Schäfer and Leser [39] proposed Word ExtrAction for Time SEries cLassification (WEASEL). WEASEL used windows of varying lengths when converting time series into words. And the order of windows is also be considered in WEASEL. Middlehurst et al. [32] proposed a more scalable version of the BOSS classifier through randomly selecting the BOSS ensemble parameters. Le Nguyen et al. [40] combined multiple symbolic representations with linear classification models for interpreting the classification decision.

Dictionary based TSC methods can interpret the classification decision through the distributions of symbolic words. However, this explanation is not intuitive enough because the symbolic word distributions are difficult to judge directly.

2.2.2. Shapelet based methods

Shapelets are identifiable subsequences of the original time series [27]. As the local patterns, shapelets classified time series into different classes through localized similarity [41]. For shapelet based methods are interpretable, they have been paid close attention by many researchers around the world recently.

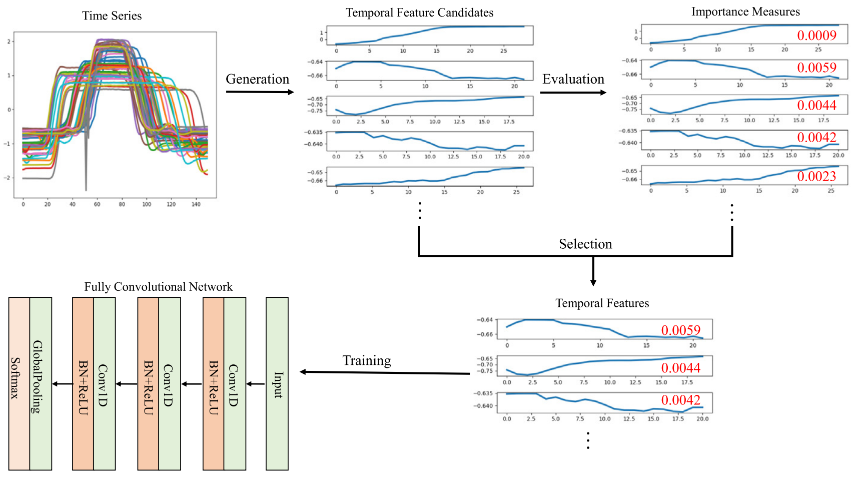


Fig. 2. The training processes of TSC-TF.

Some researcher focused on speed up the shapelet discovery process. To accelerated the shapelet selection process, some researchers pruned some similar shapelet candidates with SAX [42], special points [3,43], bloom filters [44], or clustering [45]. Random rules are also used to accelerated the shapelet selection process [46]. Some researcher speed up this process through converting the shapelet searching process into a regularized learning problem [47,48].

Some researcher paid attention to the classifiers based on shapelets. Ye and Keogh [27,28] combined shapelets with a decision tree. Lines et al. separated the processes of classification and shapelet selection [49], so various classifiers can be adopted in their method. Ji et al. [50] combined shapelets with an XGBoost classifier to achieve higher accuracy. Ma et al. [51] combined triple types of shapelets, and constructed triple shapelet networks for classification.

Through a lot of shapelet based methods are proposed, most methods evaluate candidates one by one. And few researchers applied shapelet features to deep learning classifier.

3. Our method

For intuitive interpretability and high accuracy, TSC-TF are proposed in this paper. Firstly, TSC-TF selected temporal features from some subsequences. In this way, the selected temporal features are intuitive for each of them is a part of the original time series. Secondly, to speed up the feature selection process, all temporal features are obtained at once through a random forest. Finally, TSC-TF combined the selected temporal features with a FCN classifier to achieve high accuracy.

As shown in Fig. 2 Algorithm 1, there are four main training processes in the proposed method, TSC-TF:

- **Temporal feature candidate generation**, which generates candidates with the help of time series segmentation (Line 1 - Line 5 in Algorithm 1).
- **Candidate importance measure evaluation**, which obtain the candidates' importance measures through a random forest (Line 6 in Algorithm 1).
- **Temporal feature selection**, which selects the final temporal features based on importance measures (Line 7 in Algorithm 1).
- **FCN training**, which trains a FCN classifier to obtain high accuracy (Line 8 in Algorithm 1).

In the following subsection, we will describe these four main processes in detail, respectively.

Algorithm 1 TSC-TF

```
Input: training set: $D$, feature number: $n_f$, epochs: $epoch$
Output: FCN classifier: $C$;
1: $candidates = \emptyset$
2: for each time series $T$ in $D$ do
3:     $TF$=generation($T$)    ▷ Process 1: Temporal feature candidate generation
4:     $candidates = candidates \cup TF$
5: end for
6: $IM$=evaluation($candidates$)    ▷ Process 2: Candidate importance measure evaluation
7: $features$=selection($candidates$,$IM$,$n_f$)    ▷ Process 3: Temporal feature selection
8: $C$=training($D$,$features$,$epoch$)    ▷ Process 4: FCN training
9: return $C$
```

3.1. Temporal feature candidate generation

Firstly, our method generates temporal feature candidates through a time series segmentation method which was introduced by Chung et al. [13].

Algorithm 2 Temporal feature candidate generation

```
Input: time series: $T$, segment number: $ns$
Output: temporal feature candidates: $TF$;
1: $TF = \emptyset$
2: $ns_{temp}$ = 1
3: $Set$ = $\{T\}$
4: while $ns_{temp} < ns$ do
5:     $S$ = $getMaxError(Set)$
6:     $PIP$ = $getPIP(S)$
7:     $S_L, S_R$ = $splitting(S, PIP)$
8:     if $S_L$ meets length condition then
9:         $TF = TF \cup \{S_L\}$
10:    end if
11:    if $S_R$ meets length condition then
12:        $TF = TF \cup \{S_R\}$
13:    end if
14:    $Set$ = $Set \cup \{S_L, S_R\} - \{S\}$
15: end while
16: return $TF$
```

For each time series in training dataset, the temporal feature candidate generation process is shown in Fig. 3 and Algorithm 2. As Fig. 3 and Algorithm 2 shows, TSC-TF generates temporal feature candidates through loops (Line 4 - Line 15 in Algorithm 2). In each cycle, the algorithm has the following operations:

(1) Firstly, the subsequence with maximum sum fitting error are selected (Line 5 in Algorithm 2). The sum fitting error for one

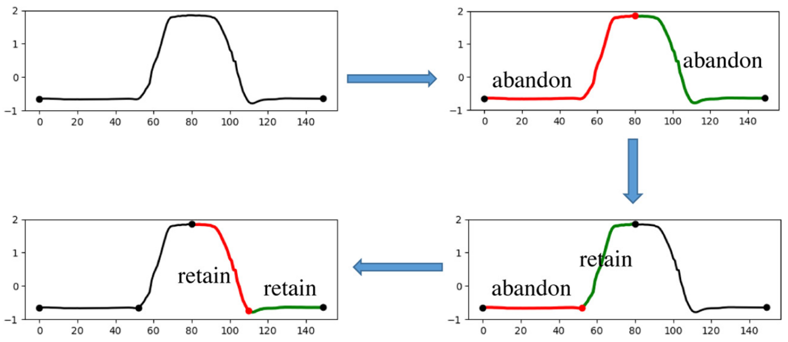


Fig. 3. The temporal feature candidate generation process. In each subfigure, the red dot is the new perceptually important point. The left new segment is mark in red, the right new segment is mark in blue.

subsequence $S = \{t_b, t_{b+1}, \dots, t_i, \dots, t_{e-1}, t_e\}$ can be calculated as Eqs. (1) and (2). In Eqs. (1) and (2), b and e are the begin and end index of the subsequence, t_b and t_e are the corresponding values. i and t_i are the index and value of the point which is calculated.

$$err_s = \sum_{i=b}^{e} err_i \tag{1}$$

$$err_i = \left| \frac{t_i - t_b}{i - b} * (e - b) + t_b - t_i \right| \tag{2}$$

(2) Secondly, this algorithm get the Perceptually Important Point (PIP) of the subsequence obtain by the first operation (Line 6 in Algorithm 2). PIP is the point which has the maximum fitting error in S, and it can be get through Eq. (3). In Fig. 3, the new PIP in every cycle is marked in red dots.

$$PIP = \underset{b \le i \le e}{argmax}\{err_i\} \tag{3}$$

(3) Next, the selected subsequence are split into two subsequence S_L and S_R based on PIP (Line 7 in Algorithm 2). S_L in the subsequence before PIP, which is marked in red in Fig. 3. S_R in the subsequence after PIP, which is marked in green in Fig. 3.

(4) Following by, S_R and S_L are determined to be temporal feature candidates or not (Line 8 - Line 13 in Algorithm 2). If S_L meets the length conditions, it will be retained as one candidates, otherwise it will be abandoned. S_R is judged by the same rules.

(5) Finally, S is removed from the subsequence set. Meanwhile, S_L and S_R is added to the set. (Line 14 in Algorithm 2)

The algorithm repeats these operations until the give segment number is met.

An example of the temporal feature candidate generation process is given in Fig. 4. As shown in the left part of Fig. 4, Algorithm 2 initializes the subsequence set *Set* with the time series (Line 7 in Algorithm 2). The middle and right parts of Fig. 4 show the two iterative processes respectively. The middle and right parts of the graph have three subfigures respectively:

- The subfigures at the top show the selected subsequence (Line 5 in Algorithm 2) and the corresponding PIP (Line 6 in Algorithm 2).
- The subfigures in the middle show the changes in the subsequence set *Set*. In these subfigures, the left new segment is marked in red, and the right new segment is marked in blue. The new segments are added to the subsequence set *Set*, and the selected subsequences are removed (Line 14 in Algorithm 2).
- The subfigures at the bottom show the changes in temporal feature candidates. In Iteration 1, the new segments do not meet the length conditions, so they are not selected as candidates. In Iteration 2, the new segments meet the length conditions and are selected as candidates.

3.2. Candidate importance measure evaluation

Previous methods [27,49,52] usually evaluated temporal feature candidates one by one. Unlike them, TSC-TF evaluated all the candidates at once.

As Fig. 5 shows, three are mainly two steps for evaluating the candidates measures:

Step 1: time series transformation.

In this step, we transform the original time series into a new representation. In the new representation, each temporal feature candidate is viewed as a feature, and the corresponding value is the distance between a temporal feature candidate S and the original time series T. The distance between S and T is the minimum possible distance from T to S, and it can be calculated as Eq. (4).

$$D(T, S) = \min_{1 \le i \le \|T\| - \|S\| + 1} d(S^i, S) \tag{4}$$

In Eq. (4), $\|T\|$ is the length of T and $\|S\|$ is the length of S. $S^i = \{t_i, t_{i+1}, \dots, t_{i+\|S\|-1}\}$ is one subsequence of T, which start index is i, and the length of S^i is $\|S\|$. The distance between subsequence $S^i = \{t_i, t_{i+1}, \dots, t_{i+\|S\|-1}\}$ and candidates $S = \{s_1, s_2, \dots, s_{\|S\|}\}$ can be calculated as Eq. (5)

$$d(S^i, S) = \sqrt{\frac{1}{\|S\|} \sum_{j=1}^{\|S\|} (t_{i+j} - s_j)^2} \tag{5}$$

Suppose there are m temporal feature candidates, one original time series T will be converted into the following forms as Eq. (6). In Eq. (6), $S_1, \dots, S_m$ are the temporal feature candidates.

$$T_D = \{D(T, S_1), \dots, D(T, S_m)\} \tag{6}$$

Step 2: importance measure evaluation.

Using the transformed training data as input, this step trains a random forest classifier. Then, the importance measures of each temporal features are counted. Through this way, the importance measures of all candidates are evaluated at once.

The importance measure IM_j for one temporal feature candidate S_j can be obtained through five major operations:

- Calculated the Gini impurity [53] for each tree node as Eq. (7). In Eq. (7), $\|C\|$ is the class number, p_c is the class frequency for class c in the tree node.

$$GI = 1 - \sum_{c=1}^{\|C\|} p_c^2 \tag{7}$$

- Calculated importance measure IM_m for tree node m. As Fig. 6 shows, IM_m is calculated as Eq. (8). As shown in

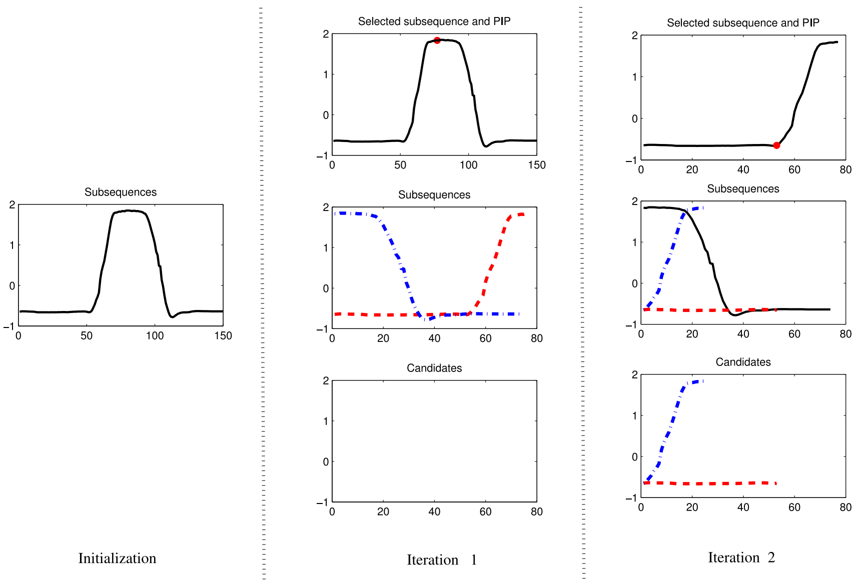
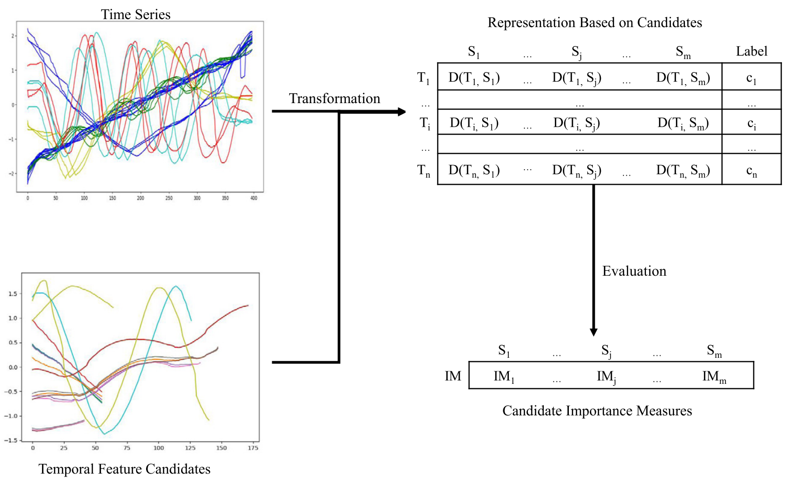


Fig. 4. An example of the temporal feature candidate generation process.



Time Series
Transformation
Representation Based on Candidates
Label
Evaluation
Candidate Importance Measures
Temporal Feature Candidates

	S_1	...	S_j	...	S_m	Label
T_1	$D(T_1, S_1)$	...	$D(T_1, S_j)$	...	$D(T_1, S_m)$	c_1
...			...			...
T_i	$D(T_i, S_1)$	...	$D(T_i, S_j)$		$D(T_i, S_m)$	c_i
...			...			...
T_n	$D(T_n, S_1)$	...	$D(T_n, S_j)$	...	$D(T_n, S_m)$	c_n

	S_1	...	S_j	...	S_m
IM	IM_1	...	IM_j	...	IM_m

Fig. 5. Steps for evaluating the candidate importance measures.

Fig. 6, l and r are the left and right tree nodes after splitting. In Eq. (8), GI_m, GI_l, GI_r are the Gini impurity for the corresponding tree node.

$$IM_m = GI_m - GI_l - GI_r \tag{8}$$

- Sum the importance measure IM_{ij} for candidates S_j in one decision tree $tree_i$ as Eq. (9). In Eq. (9), M is the tree nodes which use S_j as decision conditions in decision tree $tree_i$.

$$IM_{ij} = \sum_{m \in M} IM_m \tag{9}$$

- Sum the importance measure IM_j for candidates S_j in the random forest as Eq. (10). In Eq. (10), N_{tree} is the number of decision trees in the random forests.

$$IM_j = \sum_{i=1}^{N_{tree}} IM_{ij} \tag{10}$$

- Normalized the importance measure as Eq. (11). In Eq. (11), N_c is the number of temporal feature candidates.

$$IM_j = \frac{IM_j}{\sum_{i=1}^{N_c} IM_i} \tag{11}$$

3.3. Temporal feature selection

This process aims to obtain the final temporal features and to represent the original time series with the final temporal features. This process adopted the following three steps to achieve these two goals:

(1) **Threshold determination**. For some temporal feature candidates may have same importance measures, this step convert the given number N_f of temporal features to importance measure threshold ε. Through ranking the candidates' importance measures from large to small, and this step uses the N_f largest value as the threshold ε.

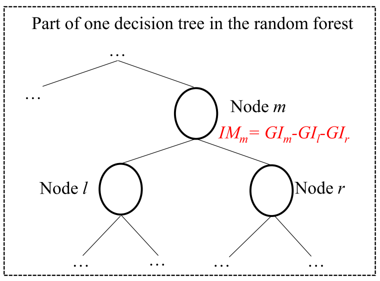

Part of one decision tree in the random forest
Node m
$IM_m = GI_m - GI_l - GI_r$
Node l
Node r

Fig. 6. Importance measure for tree node m.

(2) **Final temporal feature selection**. As shown in Fig. 7(a), the candidates with a importance measure of not less than ε are selected as final temporal features. Note than the final temporal feature number may be more than the given number N_f when some candidates' importance measures are equals to ε.

(3) **Representation dimension reduction**. After obtaining the temporal features, the dimensions of time series representation (which are get by step 1 in Section 3.2) will be reduced. As shown in Fig. 7(b), only the dimensions related to selected temporal features are retained.

3.4. FCN training

Recently, classifiers based on FCN achieves premium performance for time series [54], images [14], natural language processing [55], etc. In order to achieve high accuracy, a classifier based on FCN are training for the reduced representation.

The architecture of the trained classifier is shown in Fig. 8. As shown in Fig. 8, the classifier uses reduced representation as input. After that, the data is processed by three hidden blocks. As Fig. 8 shows, each hidden block are composed of a convolutional layer, a batch normalization layer and a ReLU activation layer. As described in Fig. 8, each convolutional layer is 1-D kernel. The kernel sizes for these three 1-D kernels are 8, 5 and 3, respectively. And the numbers of filters in these convolutional layer are 128, 256, and 128. Finally, a global average pooling layer are used reduces the number of weights, and a softmax layer are used to produced the final class label.

4. Experiments

4.1. Experimental setup

4.1.1. Datasets

The UCR TSC archive [15] are the common benchmark datasets for TSC experiments. Considering the performance of our computers, the initial 43 datasets of them are used in our experiments. We list the detailed information of the selected datasets in Table 1.

4.1.2. Baseline methods

The following five methods are selected as baseline methods:

- **1NN-DTW**, which is usually used as the general baseline in TSC [4,7].
- **catch22** [12] is selected as the representative of feature based methods that are mentioned in Section 2.1.
- **DTW-F** [26], which uses the distances between global time series as features.
- **FS** [42] is selected as the representative of interpretable feature based TSC methods that are mentioned in Section 2.2. This method uses identifiable subsequences as features.

Table 1
Experiments datasets.

Dataset	Train size	Test size	Number of classes	Length
Adiac	390	391	37	176
Beef	30	30	5	470
CBF	30	900	3	128
ChlorineConcentration	467	3840	3	166
CinCECGTorso	40	1380	4	1639
Coffee	28	28	2	286
CricketX	390	390	12	300
CricketY	390	390	12	300
CricketZ	390	390	12	300
DiatomSizeReduction	16	306	4	345
ECG200	100	100	2	96
ECGFiveDays	23	861	2	136
FaceAll	560	1690	14	131
FaceFour	24	88	4	350
FacesUCR	200	2050	14	131
FiftyWords	450	455	50	270
Fish	175	175	7	463
GunPoint	50	150	2	150
Haptics	155	308	5	1092
InlineSkate	100	550	7	1882
ItalyPowerDemand	67	1029	2	24
Lightning2	60	61	2	637
Lightning7	70	73	7	319
Mallat	55	2345	8	1024
MedicalImages	381	760	10	99
MoteStrain	20	1252	2	84
OliveOil	30	30	4	570
OSULeaf	200	242	6	427
SonyAIBORobotSurface1	20	601	2	70
SonyAIBORobotSurface2	27	953	2	65
StarLightCurves	1000	8236	3	1024
Symbols	25	995	6	398
SwedishLeaf	500	625	15	128
SyntheticControl	300	300	6	60
Trace	100	100	4	275
TwoLeadECG	23	1139	2	82
TwoPatterns	1000	4000	4	128
UWaveGestureLibraryX	896	3582	8	315
UWaveGestureLibraryY	896	3582	8	315
UWaveGestureLibraryZ	896	3582	8	315
Wafer	1000	6164	2	152
WordSynonyms	267	638	25	270
Yoga	300	3000	2	426

4.1.3. Reproducibility

For reproducibility, we released out codes and parameters on Github.[1] The results can be independently replicated.

4.2. Performance comparison with baselines

4.2.1. Comparison of all methods

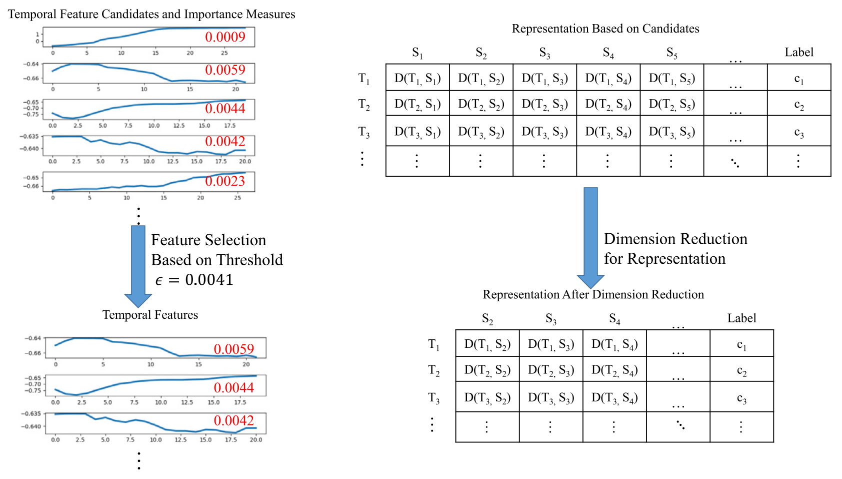
We compare the classification accuracy of TSC-TF and the baseline methods. Table 2 lists the classification accuracy rate of the methods. In Table 2, the accuracy listed for catch22 is taken from [25]. And the accuracy listed for 1NN-DTW, DTW-F and FS is taken from [4].

In Table 2, the highest accuracy rates for each dataset are given in bold. The last two rows of Table 2 are used to count the average accuracy and the best number of each method.

In order to compare the accuracy of these methods more intuitively, we draw the result of Table 2 in a critical difference diagram as shown in Fig. 9.

As Fig. 9 and Table 2 show, the proposed method has the smallest average rank, highest average accuracy among these methods. And the proposed method works best on most of the dataset. These demonstrate that out method is better than the other methods in terms of accuracy.

[1] Our code: https://github.com/sdujicun/TSC_TF


Temporal Feature Candidates and Importance Measures
0.0009
0.0059
0.0044
0.0042
0.0023
Feature Selection Based on Threshold $\epsilon = 0.0041$
Temporal Features
0.0059
0.0044
0.0042
Representation Based on Candidates
S_1 S_2 S_3 S_4 S_5 … Label
T_1 $D(T_1, S_1)$ $D(T_1, S_2)$ $D(T_1, S_3)$ $D(T_1, S_4)$ $D(T_1, S_5)$ … c_1
T_2 $D(T_2, S_1)$ $D(T_2, S_2)$ $D(T_2, S_3)$ $D(T_2, S_4)$ $D(T_2, S_5)$ … c_2
T_3 $D(T_3, S_1)$ $D(T_3, S_2)$ $D(T_3, S_3)$ $D(T_3, S_4)$ $D(T_3, S_5)$ … c_3
Dimension Reduction for Representation
Representation After Dimension Reduction
S_2 S_3 S_4 … Label
T_1 $D(T_1, S_2)$ $D(T_1, S_3)$ $D(T_1, S_4)$ … c_1
T_2 $D(T_2, S_2)$ $D(T_2, S_3)$ $D(T_2, S_4)$ … c_2
T_3 $D(T_3, S_2)$ $D(T_3, S_3)$ $D(T_3, S_4)$ … c_3

(a) An example for temporal feature selection

(b) Dimension reduction for the corresponding representations

Fig. 7. An example for temporal feature selection and dimension reduction.

Table 2
Classification accuracy with different methods.

Dataset	1NN-DTW	DTW-F [26]	catch22[12]	FS [42]	TSC-TF
Adiac	0.604	0.611	0.685	0.593	**0.783**
Beef	0.633	0.667	0.473	0.567	**0.767**
CBF	0.997	0.991	0.954	0.940	**1.000**
ChlorineConcentration	0.648	0.634	0.598	0.546	**0.703**
CinCECGTorso	0.651	0.715	0.803	**0.859**	0.708
Coffee	**1.000**	0.964	0.980	0.929	**1.000**
CricketX	0.754	**0.764**	0.609	0.485	0.723
CricketY	0.744	**0.779**	0.591	0.531	0.730
CricketZ	**0.754**	0.728	0.628	0.464	0.700
DiatomSizeReduction	**0.967**	0.935	0.925	0.866	0.875
ECG200	0.770	0.810	0.789	0.810	**0.920**
ECGFiveDays	0.768	0.974	0.816	**0.998**	**0.998**
FaceAll	0.808	0.757	**0.811**	0.626	0.753
FaceFour	0.830	**0.989**	0.680	0.909	0.966
FacesUCR	**0.905**	0.710	0.709	0.706	**0.905**
FiftyWords	0.690	**0.747**	0.598	0.481	0.627
Fish	0.823	**0.949**	0.773	0.783	0.933
GunPoint	0.907	0.980	0.943	0.947	**0.991**
Haptics	0.377	0.435	0.386	0.393	**0.526**
InlineSkate	0.384	0.373	**0.472**	0.189	0.430
ItalyPowerDemand	0.950	**0.960**	0.878	0.917	0.948
Lighting2	**0.869**	0.787	0.745	0.705	0.863
Lighting7	0.726	0.575	0.646	0.644	**0.776**
Mallat	0.934	0.948	0.906	0.976	**0.987**
MedicalImages	0.737	0.724	0.757	0.624	**0.763**
MoteStrain	0.835	**0.909**	0.849	0.777	0.866
OliveOil	0.833	0.900	0.746	0.733	**0.933**
OSULeaf	0.591	0.810	0.724	0.678	**0.847**
SonyAIBORobotSurface1	0.725	0.740	0.883	0.686	**0.955**
SonyAIBORobotSurface2	0.831	0.856	**0.902**	0.790	0.878
StarLightCurves	0.907	0.962	**0.970**	0.918	0.969
SwedishLeaf	0.792	0.901	0.881	0.768	**0.931**
Symbols	**0.950**	0.949	0.948	0.934	0.942
SyntheticControl	**0.993**	0.990	0.967	0.910	0.977
Trace	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
TwoLeadECG	0.905	**0.985**	0.854	0.924	0.962
TwoPatterns	**1.000**	**1.000**	0.849	0.908	0.929
UWaveGestureLibraryX	0.728	**0.802**	0.769	0.695	0.714
UWaveGestureLibraryY	0.634	0.703	**0.704**	0.596	0.604
UWaveGestureLibraryZ	0.658	**0.732**	0.706	0.638	0.671
Wafer	0.980	**0.997**	**0.997**	**0.997**	0.991
WordsSynonyms	0.649	**0.674**	0.544	0.431	0.566
Yoga	0.837	**0.868**	0.804	0.695	0.861
Average	0.793	0.821	0.773	0.734	**0.837**
Best Number	9	15	7	4	**18**

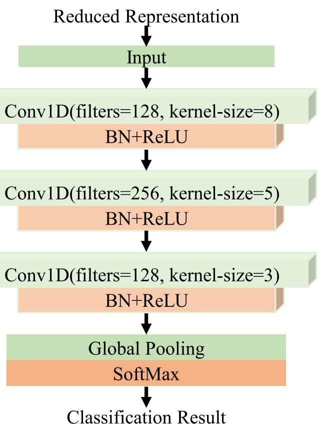

Reduced Representation
Input
Conv1D(filters=128, kernel-size=8)
BN+ReLU
Conv1D(filters=256, kernel-size=5)
BN+ReLU
Conv1D(filters=128, kernel-size=3)
BN+ReLU
Global Pooling
SoftMax
Classification Result

Fig. 8. The architecture of FCN.

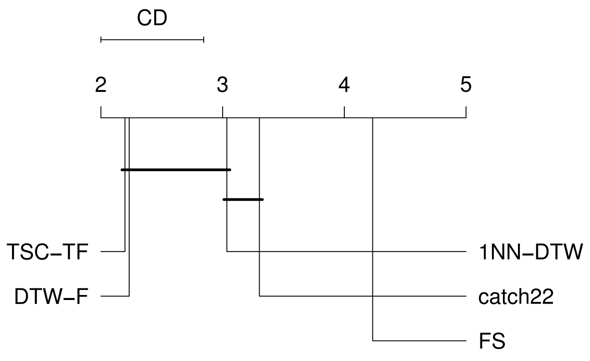


Fig. 9. Critical difference diagram for the proposed method and baseline methods.

Table 3
Accuracy comparison results between TSC-TF and each baseline methods.

	TSC-TF better	Equal	TSC-TF worse
1NN-DTW Vs. TSF-TF	**26**	3	14
catch22 Vs. TSF-TF	**31**	1	11
FS Vs. TSF-TF	**39**	2	2
DTW-F Vs. TSF-TF	**21**	1	**21**

4.2.2. Comparison between TSC-TF and each baseline methods

In this group of experiments, we compare TSC-TF with each baseline method directly. The comparison results are shown in Table 3.

A comparison between the classification accuracy of 1NN-DTW and TSC-TF is shown in Fig. 10. From Fig. 10 and Table 3, it is clear that TSC-TF produced better results with 26 datasets, 1NN-DTW performed better with 14 datasets, and the two techniques tied with the other 3 datasets. These results suggest that the classification results of TSC-TF are better than 1NN-DTW.

Fig. 11 shows the classification accuracy comparison result between catch22 and TSC-TF. As Fig. 11 and Table 3 show, TSC-TF produced better results with 31 datasets, catch22 performed better with 11 datasets, and the two techniques tied with the other 1 datasets. These results suggest that the classification results of TSC-TF are better than catch22.

The classification accuracy comparison between FS and TSC-TF is shown in Fig. 12. From Fig. 12 and Table 3, it is clear that TSC-TF produced better results with 39 datasets, FS performed better with 2 datasets, and the two techniques tied with the other 2 datasets. These results suggest that the classification results of TSC-TF are better than FS.

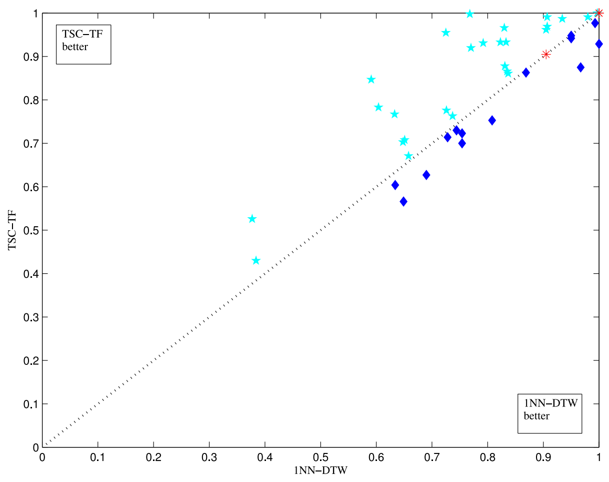


Fig. 10. Accuracy comparison between TSC-TF and 1NN-DTW.

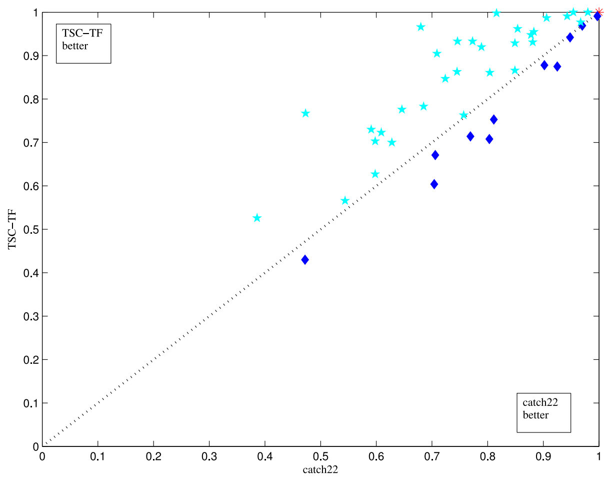


Fig. 11. Accuracy comparison between TSC-TF and catch22.

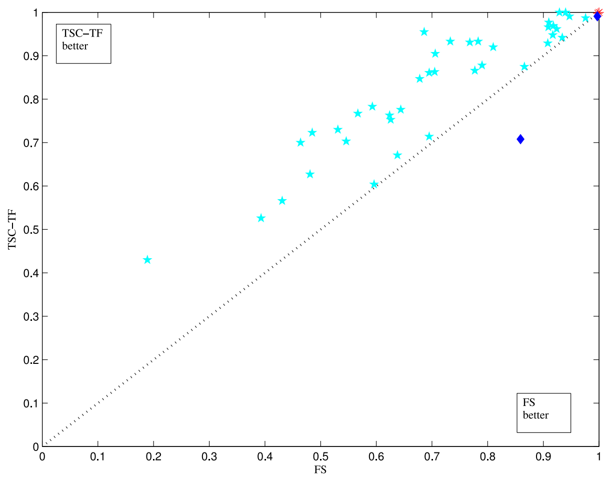


Fig. 12. Accuracy comparison between TSC-TF and FS.

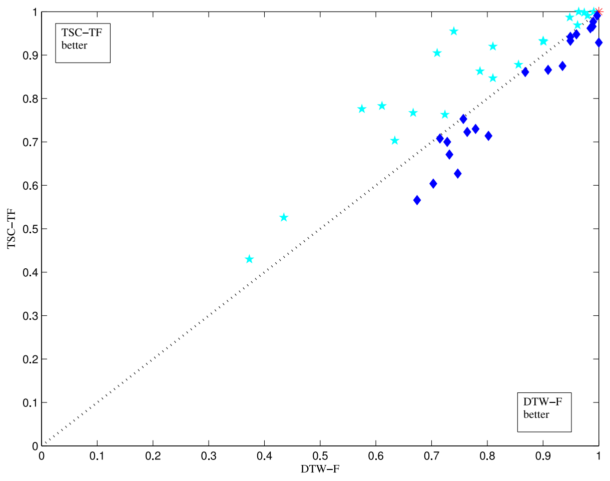


Fig. 13. Accuracy comparison between TSC-TF and DTW-F.

Fig. 13 shows the classification accuracy comparison result between DTW-F and TSC-TF. As Fig. 13 and Table 3 show, TSC-TF produced better results with 21 datasets, DTW-F performed better with 21 datasets, and the two techniques tied with the other 1 datasets. These results suggest that the classification results of TSC-TF are no worse than DTW-F.

4.3. Sensitivity analysis

4.3.1. Feature number analysis

We set the feature number to 64, 128, 256, 512, 1024, and 2048 to analyze the effect of feature number on training time and accuracy. In this set of experiments, the epoch of FCN is set to 2000. Fig. 14 shows the training times and accuracy of 'MedicalImages', 'Mallat', and 'TwoPatterns' with different feature numbers.

From Fig. 14, we can get the following rules: With the increase of feature numbers, the accuracy first increases, then maintains a high level, and finally decreases. As Fig. 14(a) shows, 'MedicalImages' achieves high accuracy when feature number ranges from 256 to 1024. As Fig. 14(b) shows, 'Mallat' achieves high accuracy when the range feature number ranges from 64 to 1024. As Fig. 14(c) shows, 'TwoPatterns' achieves high accuracy when feature number ranges from 128 .

As Fig. 14 shows, with the exponential growth of feature numbers, the FCN training time also increases exponentially. Note that, the X axis and the Y axis on the right are exponential coordinates in Fig. 14.

Considering the accuracy and FCN training time, we suggest setting the feature number to 256 or 512. Also the feature number can be increased appropriately for large datasets, and it can be appropriately reduced for small datasets.

4.3.2. Epoch analysis for FCN

To analyze the effect of epoch, we got the loss, accuracy and time with epoch ranges from 1 to 2000. Note that, the feature number is set to 256 in this group of experiments.

Fig. 15 shows the train loss and the train accuracy of 'MedicalImages', 'Mallat', and 'TwoPatterns' with different epochs. From Fig. 15, we can get that: (1) In the initial stage, with the increase of epoch, the train loss becomes smaller and the train accuracy becomes higher; (2) When epoch is greater than 500, the train loss and accuracy remains basically unchanged.

Fig. 16 shows the time of 'MedicalImages', 'Mallat', and 'TwoPatterns' with different epochs. In Fig. 16, t_e represents for the time of one single epoch, and t_s represents for the sum time from the begin to this epoch. Eq. (12) shows how to get t_s.

$$t_s^i = \sum_{k=1}^{i} t_e^k \tag{12}$$

As Fig. 16 shows, the fluctuations of one single epoch times (t_e) are within one second. In other words, there are only small fluctuations in the time of one single epoch. As a result, the overall training time t_s is basically proportional to epoch. Therefore, t_s are basically straight lines in Fig. 16

Considering the accuracy and FCN training time, it is recommended that the epoch can be adjusted in the range of 500 to 1000.

5. Conclusion

Recently, feature based TSC methods have been becoming a research hotspot because they are able to explain the classification results. However, how to accurately classify time series based on intuitively interpretable features is still a huge challenge. To address this changeling, TSC-TF is proposed in this work to obtain high accuracy with intuitively temporal features. Firstly, TSC-TF generates some temporal feature candidates with the help of a time series segmentation method. And then, a random forest is used to obtain importance measures of the candidates. Following by, TSC-SF selects the temporal features according to the importance measures. Finally, a classifier based on FCN is trained to classify the transformed representation of time series based on the selected temporal features. Experimental results on various datasets from the UCR TSC archive demonstrate that the proposed method can achieve better accuracy based on intuitively temporal features. In the future, we will reduce the similar feature candidate when generating them. Also, we will combine some speedup strategies to speed up the time series transformation.

CRediT authorship contribution statement

Cun Ji: Conceptualization, Methodology, Validation, Writing – original draft, Writing – review & editing. **Mingsen Du:** Methodology, Validation, Writing – original draft, Writing – review & editing. **Yupeng Hu:** Methodology, Validation. **Shijun Liu:** Supervision, Validation. **Li Pan:** Methodology, Validation. **Xiangwei Zheng:** Supervision, Project administration.

Declaration of competing interest

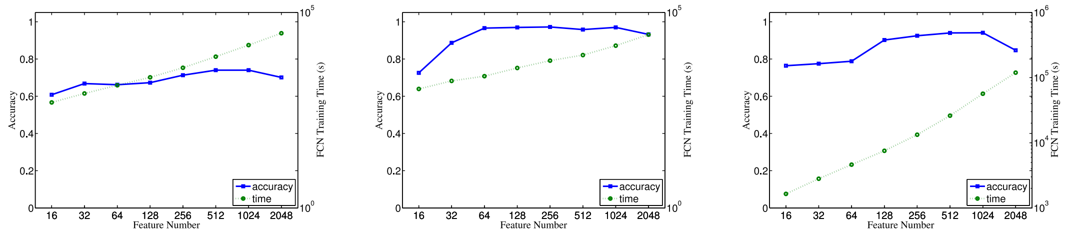
The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

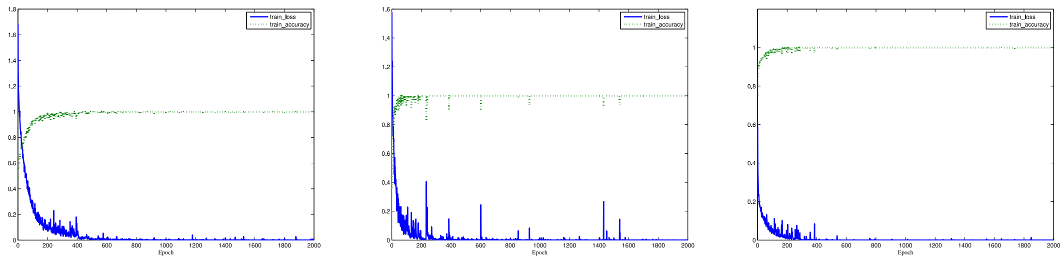
Acknowledgments

This work is partially funded by the Innovation Methods Work Special Project [grant number 2020IM020100], and the Natural Science Foundation of Shandong Province [grant number ZR2020QF112]. The authors would like to thank the anonymous reviewers and the editors for their insightful comments and suggestions, which are greatly helpful for improving the quality of this paper.



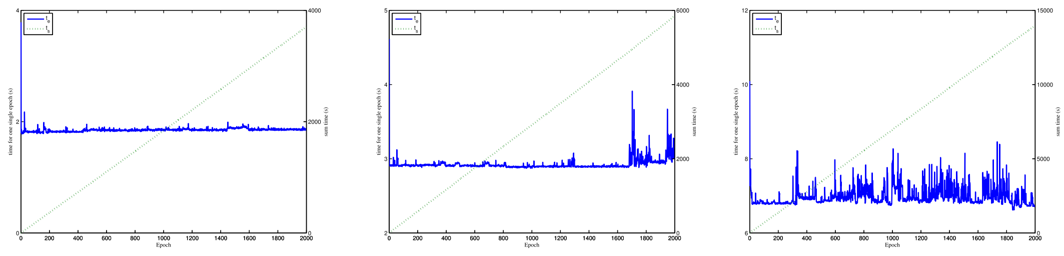
(a) MedicalImages (b) Mallat (c) TwoPatterns

Fig. 14. Training time and accuracy comparison with different feature numbers.



(a) MedicalImages (b) Mallat (c) TwoPatterns

Fig. 15. Training loss and accuracy comparison with different epochs.



(a) MedicalImages (b) Mallat (c) TwoPatterns

Fig. 16. Training time comparison with different epochs.

References

[1] H. Zhang, H. Gao, D. Jin, HTF: An effective algorithm for time series to recover missing blocks, in: International Conference on Database Systems for Advanced Applications, Springer, 2021, pp. 29–44.

[2] P. Esling, C. Agon, Time-series data mining, ACM Comput. Surv. 45 (1) (2012) 1–34.

[3] C. Ji, C. Zhao, S. Liu, C. Yang, L. Pan, L. Wu, X. Meng, A fast shapelet selection algorithm for time series classification, Comput. Netw. 148 (2019) 231–240.

[4] A. Bagnall, J. Lines, A. Bostrom, J. Large, E. Keogh, The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances, Data Min. Knowl. Discov. 31 (3) (2017) 606–660.

[5] S.J. Wilson, Data representation for time series data mining: time domain approaches, Wiley Interdiscip. Rev. Comput. Stat. 9 (1) (2017) e1392.

[6] A. Abanda, U. Mori, J.A. Lozano, A review on distance based time series classification, Data Min. Knowl. Discov. 33 (2) (2019) 378–412.

[7] H.I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, P.-A. Muller, Deep learning for time series classification: a review, Data Min. Knowl. Discov. 33 (4) (2019) 917–963.

[8] A.P. Ruiz, M. Flynn, J. Large, M. Middlehurst, A. Bagnall, The great multivariate time series classification bake off: a review and experimental evaluation of recent algorithmic advances, Data Min. Knowl. Discov. (2020) 1–49.

[9] Z. Xiao, X. Xu, H. Xing, S. Luo, P. Dai, D. Zhan, RTFN: A robust temporal feature network for time series classification, Inform. Sci. 571 (2021) 65–86.

[10] B.D. Fulcher, N.S. Jones, Highly comparative feature-based time-series classification, IEEE Trans. Knowl. Data Eng. 26 (12) (2014) 3026–3037.

[11] B.D. Fulcher, N.S. Jones, Hctsa: A computational framework for automated time-series phenotyping using massive feature extraction, Cell Syst. 5 (5) (2017) 527–531.

[12] C.H. Lubba, S.S. Sethi, P. Knaute, S.R. Schultz, B.D. Fulcher, N.S. Jones, Catch22: Canonical time-series characteristics, Data Min. Knowl. Discov. 33 (6) (2019) 1821–1852.

[13] F.-L. Chung, T.-C. Fu, V. Ng, R.W. Luk, An evolutionary approach to pattern-based time series segmentation, IEEE Trans. Evol. Comput. 8 (5) (2004) 471–489.

[14] J. Long, E. Shelhamer, T. Darrell, Fully convolutional networks for semantic segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 3431–3440.

[15] H.A. Dau, E. Keogh, K. Kamgar, C.-C.M. Yeh, Y. Zhu, S. Gharghabi, C.A. Ratanamahatana, Yanping, N. Hu, A. Bagnall, A. Mueen, G. Batista, M. Hexagon, The UCR time series classification archive, 2018, https://www.cs.ucr.edu/~eamonn/time_series_data_2018/.

[16] M. Baghizadeh, K. Maghooli, F. Farokhi, N.J. Dabanloo, A new emotion detection algorithm using extracted features of the different time-series generated from ST intervals Poincaré map, Biomed. Signal Process. Control 59 (2020) 101902.

[17] A. Nanopoulos, R. Alcock, Y. Manolopoulos, Feature-based classification of time-series data, Int. J. Comput. Res. 10 (3) (2001) 49–61.
[18] X. Wang, K. Smith, R. Hyndman, Characteristic-based clustering for time series data, Data Min. Knowl. Discov. 13 (3) (2006) 335–364.
[19] H. Deng, G. Runger, E. Tuv, M. Vladimir, A time series forest for classification and feature extraction, Inform. Sci. 239 (2013) 142–153.
[20] S. Wu, X. Wang, M. Liang, D. Wu, PFC: A novel perceptual features-based framework for time series classification, Entropy 23 (8) (2021) 1059.
[21] I. Batal, M. Hauskrecht, A supervised time series feature extraction technique using dct and dwt, in: 2009 International Conference on Machine Learning and Applications, IEEE, 2009, pp. 735–739.
[22] P. Chaovalit, A. Gangopadhyay, G. Karabatis, Z. Chen, Discrete wavelet transform-based time series analysis and mining, ACM Comput. Surv. 43 (2) (2011) 1–37.
[23] H. Zhang, T. Ho, W. Huang, Blind feature extraction for time-series classification using haar wavelet transform, in: International Symposium on Neural Networks, Springer, 2005, pp. 605–610.
[24] M.G. Baydogan, G. Runger, E. Tuv, A bag-of-features framework to classify time series, IEEE Trans. Pattern Anal. Mach. Intell. 35 (11) (2013) 2796–2802.
[25] B. Wang, T. Jiang, X. Zhou, B. Ma, F. Zhao, Y. Wang, Time-series classification based on fusion features of sequence and visualization, Appl. Sci. 10 (12) (2020) 4124.
[26] R.J. Kate, Using dynamic time warping distances as features for improved time series classification, Data Min. Knowl. Discov. 30 (2) (2016) 283–312.
[27] L. Ye, E. Keogh, Time series shapelets: a new primitive for data mining, in: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2009, pp. 947–956.
[28] L. Ye, E. Keogh, Time series shapelets: a novel technique that allows accurate, interpretable and fast classification, Data Min. Knowl. Discov. 22 (1) (2011) 149–182.
[29] A. Dempster, F. Petitjean, G.I. Webb, ROCKET: exceptionally fast and accurate time series classification using random convolutional kernels, Data Min. Knowl. Discov. 34 (5) (2020) 1454–1495.
[30] A. Dempster, D.F. Schmidt, G.I. Webb, Minirocket: A very fast (almost) deterministic transform for time series classification, in: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, 2021, pp. 248–257.
[31] C.W. Tan, A. Dempster, C. Bergmeir, G.I. Webb, MultiRocket: Effective summary statistics for convolutional outputs in time series classification, 2021, arXiv preprint arXiv:2102.00457.
[32] M. Middlehurst, W. Vickers, A. Bagnall, Scalable dictionary classifiers for time series classification, in: International Conference on Intelligent Data Engineering and Automated Learning, Springer, 2019, pp. 11–19.
[33] J. Lin, E. Keogh, L. Wei, S. Lonardi, Experiencing SAX: a novel symbolic representation of time series, Data Min. Knowl. Discov. 15 (2) (2007) 107–144.
[34] P. Schäfer, M. Högqvist, Sfa: a symbolic fourier approximation and index for similarity search in high dimensional datasets, in: Proceedings of the 15th International Conference on Extending Database Technology, 2012, pp. 516–527.
[35] B. Lucas, A. Shifaz, C. Pelletier, L. O'Neill, N. Zaidi, B. Goethals, F. Petitjean, G.I. Webb, Proximity forest: an effective and scalable distance-based classifier for time series, Data Min. Knowl. Discov. 33 (3) (2019) 607–635.
[36] J. Lin, R. Khade, Y. Li, Rotation-invariant similarity in time series using bag-of-patterns representation, J. Intell. Inf. Syst. 39 (2) (2012) 287–315.
[37] P. Senin, S. Malinchik, Sax-vsm: Interpretable time series classification using sax and vector space model, in: 2013 IEEE 13th International Conference on Data Mining, IEEE, 2013, pp. 1175–1180.
[38] P. Schäfer, Scalable time series classification, Data Min. Knowl. Discov. 30 (5) (2016) 1273–1298.
[39] P. Schäfer, U. Leser, Fast and accurate time series classification with weasel, in: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, 2017, pp. 637–646.
[40] T. Le Nguyen, S. Gsponer, I. Ilie, M. O'Reilly, G. Ifrim, Interpretable time series classification using linear models and multi-resolution multi-domain symbolic representations, Data Min. Knowl. Discov. 33 (4) (2019) 1183–1222.
[41] D. Guijo-Rubio, V.M. Vargas, P.A. Gutiérrez, C. Hervás-Martínez, Studying the effect of different L_p norms in the context of time series ordinal classification, in: Conference of the Spanish Association for Artificial Intelligence, Springer, 2021, pp. 44–53.
[42] T. Rakthanmanon, E. Keogh, Fast shapelets: A scalable algorithm for discovering time series shapelets, in: Proceedings of the 2013 SIAM International Conference on Data Mining, SIAM, 2013, pp. 668–676.
[43] G. Li, W. Yan, Z. Wu, Discovering shapelets with key points in time series classification, Expert Syst. Appl. 132 (2019) 76–86.
[44] G. Li, B.K.K. Choi, J. Xu, S.S. Bhowmick, K.-P. Chun, G.L. Wong, Efficient shapelet discovery for time series classification, IEEE Trans. Knowl. Data Eng. (2020) 1–15.
[45] X. Zou, X. Zheng, C. Ji, Y. Zhang, An improved fast shapelet selection algorithm and its application to pervasive EEG, Pers. Ubiquitous Comput. (2021) 1–13.
[46] G. Li, W. Yan, Extracting distinctive shapelets with random selection for early classification, in: Proceedings of the 13th International Conference on Knowledge Science, Engineering and Management, vol. 12274, Springer, 2020, pp. 471–484.
[47] J. Grabocka, N. Schilling, M. Wistuba, L. Schmidt-Thieme, Learning time-series shapelets, in: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2014, pp. 392–401.
[48] Y. Hu, P. Zhan, Y. Xu, J. Zhao, Y. Li, X. Li, Temporal representation learning for time series classification, Neural Comput. Appl. 33 (8) (2021) 3169–3182.
[49] J. Lines, L.M. Davis, J. Hills, A. Bagnall, A shapelet transform for time series classification, in: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2012, pp. 289–297.
[50] C. Ji, X. Zou, Y. Hu, S. Liu, L. Lyu, X. Zheng, XG-SF: An xgboost classifier based on shapelet features for time series classification, Procedia Comput. Sci. 147 (2019) 24–28.
[51] Q. Ma, W. Zhuang, G. Cottrell, Triple-shapelet networks for time series classification, in: Proceedings of the 2019 IEEE International Conference on Data Mining, IEEE, 2019, pp. 1246–1251.
[52] C. Ji, Y. Hu, K. Wang, P. Zhan, X. Li, X. Zheng, Identifiable temporal feature selection via horizontal visibility graph towards smart medical applications, Interdiscip. Sci. Comput. Life Sci. 13 (4) (2021) 717–730.
[53] S. Nembrini, I.R. König, M.N. Wright, The revival of the gini importance? Bioinformatics 34 (21) (2018) 3711–3718.
[54] Z. Wang, W. Yan, T. Oates, Time series classification from scratch with deep neural networks: A strong baseline, in: 2017 International Joint Conference on Neural Networks, IEEE, 2017, pp. 1578–1585.
[55] Y. Xing, L. Zhong, X. Zhong, An encoder-decoder network based FCN architecture for semantic segmentation, Wirel. Commun. Mob. Comput. 2020 (2020).