applied sciences

MDPI

Article

Time Series Classification with Shapelet and Canonical Features

Hai-Yang Liu, Zhen-Zhuo Gao, Zhi-Hai Wang * and Yun-Hao Deng

School of Computer and Information Technology, Beijing Jiaotong University, Beijing 100044, China
* Correspondence: zhhwang@bjtu.edu.cn

Abstract: Shapelet-based time series classification methods are widely adopted models for time series classification tasks. However, the high computational cost greatly limits the practicability of the Shapelet-based methods. What is more, traditional Shapelet can only describe the overall shape characteristics of subsequences under the Euclidean distance metric, so it is vulnerable to noise. Other than Shapelet, there are other types of discriminative information contained in the subsequences. To deal with the aforementioned problems, an accurate and efficient time series classification algorithm, named Shapelet with Canonical Time Series Features, is proposed in this paper. The proposed algorithm is based on the following three key strategies: (1) randomly selecting Shapelet and limiting the scope of Shapelet to improve efficiency; (2) embedding multiple canonical time series features in Shapelet to improve the adaptability of the algorithm to different classification problems and make up for the accuracy loss caused by the random selection of Shapelet; and (3) building a random forest classifier based on the new feature representations to ensure the generalization ability of the algorithm. Experimental results on 112 UCR time series datasets show that the proposed algorithm is more accurate than the STC algorithm which is based on Shapelet exact search and the Shapelet transform technique, as well as many other types of state-of-the-art time series classification algorithms. Moreover, extensive experimental comparisons verify the significant advantages of the proposed algorithm in terms of efficiency.

Keywords: time series; Shapelet; canonical time series features; classification

Citation: Liu, H.-Y.; Gao, Z.-Z.; Wang, Z.-H.; Deng, Y.-H. Time Series Classification with Shapelet and Canonical Features. *Appl. Sci.* **2022**, *12*, 8685. https://doi.org/10.3390/app12178685

Academic Editors: Shengzong Zhou and Jingsha He

Received: 7 June 2022
Accepted: 22 August 2022
Published: 30 August 2022

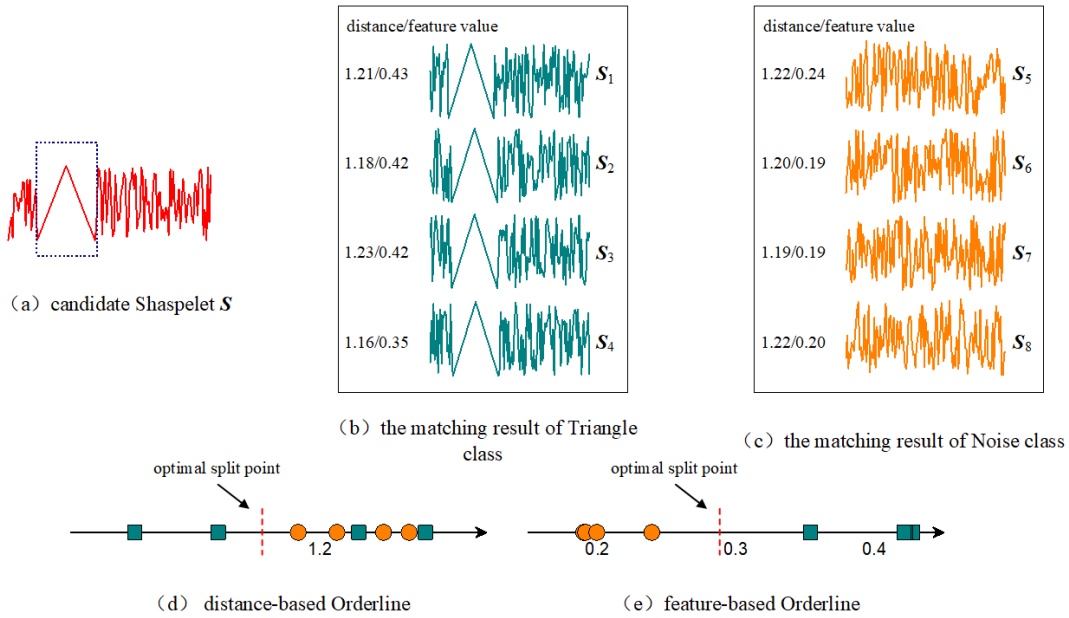
1. Introduction

Time series classification is an important research area in data mining and has received more and more extensive attention in recent years [1,2]. The solutions of many practical applications are supported by time series classification technology, such as road condition prediction [3], disease diagnosis [4], remote sensing data analysis [5] and so on, which have greatly promoted the rapid development of time series classification research. However, the increasing scale of data [6,7] and the constant introduction of complex classification tasks [7] make it still extremely challenging to achieve accurate and efficient time series classification.

One of the key problems in time series classification research is how to define and find patterns that distinguish time series from different categories, which affects and even determines the performance of time series classification algorithms. Ye et al. [8] first proposed the concept of Shapelet, defining it as a discriminating subsequence in a time series. Shapelet distinguishes different categories by the local shape of the time series, has strong predictive ability and interpretability, and has received widespread attention in the field of time series classification. However, Shapelet's brute force search algorithm requires iterating through all subsequences in the dataset with a time complexity of up to $O(n^2 \cdot l^4)$ (n is the number of time series in the dataset, l is the average length of the time series) [8]. The expensive computational cost strongly limits the usefulness of the time series classification algorithms based on Shapelet [9,10]. To address the issue, researchers improved the search efficiency of Shapelet by reducing the search space of Shapelet [9–12] or trading space for time [13] (see Section 2.1 for details). However, these methods usually

are designed with the goal of bringing only a certain efficiency gain without significantly reducing accuracy.

In addition to the efficiency problem, we realize that there is another flaw in the traditional Shapelet algorithm—that is, it can only describe the overall shape characteristics of the subsequence under the Euclidean distance metric. So, it is extremely susceptible to noise, and it is difficult to mine other types of features embedded in the subsequence. Consider the example in Figure 1 taken from the "ShapeletSim" dataset [7]. Figure 1a is a candidate Shapelet S chosen from the "Triangle" class, and Figure 1b,c present the subsequences most similar to candidate Shapelet S in each of the two categories found using sliding window technique. In fact, the characteristic that distinguishes the two categories of the "ShapeletSim" dataset is the artificially embedded triangular-shaped segment in the "Triangle" class (the area within the box in Figure 1a), and other data points in the two categories are all random noise. As can be seen from Figure 1, the candidate Shapelet S intuitively meets our expectations for Shapelet, as it has the triangular-shaped feature that the "Noise" class does not have. However, due to the influence of noise, it is impossible to correctly classify these two categories by sorting the Euclidean distances between the eight subsequences and the candidate Shapelet S (as shown in Figure 1d).



(a) candidate Shaspelet S

(b) the matching result of Triangle class

(c) the matching result of Noise class

(d) distance-based Orderline

(e) feature-based Orderline

Figure 1. An example of mining discriminative information of subsequences via embedding canonical time series features.

In order to deal with the above problems, this paper proposes a new time series classification algorithm—Random Shapelet Forest Embedded with Canonical Time Series Features (RSFCF). RSFCF is a random, tree-based integrated classification algorithm designed to achieve high accuracy and efficient time series classification. RSFCF's time complexity is reduced by several orders of magnitude relative to the Shapelet brute force search by randomly selecting Shapelets. In addition, inspired by interval-based time series classification methods [3,14], we believe that for most real-world datasets, the local offset of the time series on the timeline is generally within a limited range. As a result, RSFCF limits the scope of Shapelet, which further improves the efficiency of Shapelet matching while retaining Shapelet's location information to a large extent.

Lubba et al. proposed 22 typical time series features ("Catch22" for short; see Section 3.3 for details), including statistical features, spectral features and other types of features [15]. In order to improve the applicability of the algorithm and compensate for the accuracy loss caused by the random selection of Shapelet, this paper combines the Shapelet transformation technology [16] with multiple typical time series features,

and proposes a random Shapelet transformation method that embeds typical time series features, and the final random forest classification model is constructed based on the new feature representation of the data. As in the example shown in Figure 1, the "triangle" causes the power spectrum of the subsequence to have a stronger response in the lower frequency band. Based on the eigenvalue of the subsequence on the feature SP_Summaries_welch_rect_area_5_1 (the sum of the energies of the five lowest frequencies in the Fourier power spectrum, which is one of the features in the Catch22), the two categories can be correctly distinguished (as shown in Figure 1e). In this case, although the candidate Shapelet S is not precise (i.e., contains a lot of random noise in addition to the "triangle"), we are still able to tap into the discriminatory information contained therein and rely on it for accurate classification.

In order to fully verify the performance of the RSFCF algorithm proposed in this paper, we have conducted extensive experimental comparison and analysis with a number of current advanced time series classification algorithms on a large number of UCR time series datasets [7]. Experimental results on 112 datasets show that: (1) embedding typical time series features can effectively improve the accuracy of random Shapelet forests; (2) RSFCF surpasses the STC algorithm based on Shapelet precision search and Shapelet transformation techniques [17] in terms of accuracy (a recent work by Bagnall et al. [18] showed that STC is the most accurate time series classification algorithm based on Shapelet), and is an order of magnitude faster than it is in training; (3) besides the STC algorithm, RSFCF surpasses many other types of advanced time series classification algorithms in terms of accuracy, including residual neural networks (ResNet) [19], Proximity Forest [5], and Canonical Interval Forests (CIF) [3].

The main contributions in this paper are summarized as follows:

(1) Considering the characteristics of a real dataset, a method that can effectively improve the matching efficiency of Shapelet without a significant loss of accuracy is verified to limit the scope of Shapelet;

(2) A novel method of embedding typical time series features in Shapelet is proposed, and experimental results show that this method can effectively compensate for the loss of accuracy caused by the random selection of Shapelet;

(3) Based on the above method, an accurate and efficient time series classification algorithm is proposed—Random Shapelet Forest embedded with Canonical Time Series features (referred to as RSFCF).

The rest of this article is organized as follows. Section 2 introduces the relevant work and recent progress made in time series classification, Section 3 introduces the relevant definitions and background knowledge, Section 4 describes the RSFCF algorithm proposed in this paper in detail, Section 5 verifies the performance of the RSFCF algorithm through extensive experimental comparison and analysis, and a final summary and possible future work are given in Section 6.

2. Related Work

This section reviews relevant research work on Shapelet and briefly introduces other types of time series classification method in light of the latest research advances.

2.1. Classification Method Based on Shapelet

The classification method of the time series classification algorithm based on Shapelet distinguishes between different categories based on whether the discriminating subsequence (i.e., Shapelet [8]) appears in the time series, regardless of where it appears. The original method of Ye et al. [8] used information gain as an evaluation criterion to search for optimal Shapelet by enumerating all subsequences. Since the high computational cost of the method severely hampers Shapelet's practicality, much of the research on Shapelet has focused on how to accelerate the discovery of Shapelet. For example, Mueen et al. [13] employed an intelligent caching technique that traded space for time, reducing the time complexity of an exact search for Shapelet by an order of magnitude. The fast Shapelet

method [9] uses SAX technique [20] to discretize subsequences and search for "approximate" optimal Shapelet in low-dimensional spaces using random projection techniques, reducing the temporal complexity of searching for Shapelet to $O(n \cdot l^2)$. When Karlsson et al. [10] constructed each node of a decision tree, only randomly selected k subsequences (k are much less than the number of all subsequences). In [11], Shapelet pre-screening was performed based on variance located at key points in the time series at each time point. Ref. [12] applied local Fisher discriminant analysis (LFDA) to find key dimensions in time series to reduce the Shapelet search space. Another method of improving efficiency is called LearningShapelet [1], which learns Shapelet by optimizing strategies rather than directly using subsequences in the dataset as candidates.

There are two main classification strategies for the above methods. The first way is to fuse the Shapelet search process with the building process of the decision tree [8–10,13], and the second is to transform the dataset using the multiple Shapelets found (i.e., using the Shapelet transformation technique [16], see Section 3.2 for details) and then classifying them using traditional classifiers such as SVM [12]. Although a series of Shapelet-based classification methods have been proposed one after another, a recent assessment by Bagall et al. [18] suggests that in terms of accuracy, the STC classification algorithm [17] based on Shapelet precision search and Shapelet transformation techniques (the converted data are used to train the rotating forest classifier [18]) represents the most advanced level of this type of method. In the experimental analysis in Section 5, we will show that the accuracy and efficiency of the RSFCF algorithm proposed in this paper on the UCR time series dataset exceed the STC algorithm.

2.2. Other Types of Classification Methods

Classification methods based on intervals assume that local features depend on where they appear and are generally more efficient. Typical methods include time series forest (TSF) [14], random interval spectrum integration (RISE) [21], and typical interval forest (CIF) [3]. TSF randomly selects a set of intervals in the time series to perform transformations over three time domains (mean, variance, and slope, respectively), and trains an integrated classifier based on a decision tree with new feature representations [14]. Unlike TSF, RISE performs four frequency domain transitions for each set of randomly selected intervals, including autocorrelation functions, partial autocorrelation functions, autoregressive models, and power spectra [21]. CIF adds the 22 features in Catch22 [15] to TSF, significantly exceeding TSF and RISE in accuracy [3].

Classification methods based on dictionaries convert time series into a bag of patterns, distinguishing between different categories by the relative frequency with which patterns appear. Representative algorithms include Pattern Package (BoP) [22], SAX-VSM [23], BOSS [24], and WINCL [25]. Among them, BoP and SAX-VSM use symbolic aggregation approximation (SAX) [20] techniques to convert subsequences into words, building feature vectors based on word frequency [23]. BOSS is the most commonly used dictionary-based classification method [21], which constructs words using symbolic Fourier approximation (SFA) [26] techniques and constructs an integrated classifier [24] based on nearest neighbor and specially tailored distance metrics. WEASEL uses a "supervised" symbolic Fourier approximation technique to screen words with chi-square tests and ultimately train a logistic regression classifier [25]. In terms of accuracy, WEASEL represents an advanced level of lexicography-based classification methods [3,18]. However, researchers generally point out that the spatial complexity of WEASEL is extremely high, mainly due to the large characteristic space [5,27].

Classification methods based on distance usually use elastic distance measurements (i.e., distance measurement methods that can cope with phenomena such as local shifts or distortions of time series to some extent [28–30]) to quantify the distance between time series and classify them according to the distance between test instances and training instances. In order to improve the accuracy, Lines et al. [30] proposed an elastic integration algorithm consisting of 11 nearest neighbor classification algorithms based on different

elastic distance measurements. The training and classification complexity of this method are high, at $O(n^2 \cdot l^2)$ and $O(n \cdot l^2)$, respectively. Lucas et al. [5] proposed the Proximity Forest algorithm to improve the accuracy and efficiency of the elastic integration algorithm. A neighboring forest is an integration of multiple neighboring trees, where the data on each node are split according to the distance from a randomly selected time series in each class, and the distance measurement and its required parameters are also randomly selected.

The time series classification methods introduced earlier, including STC [17], BOSS [24], elastic integration [31], neighboring forests [5], etc., are all integrated methods, but they are all homogeneous integrations, or integrations based on a single representation of time series. The meta-ensemble method [21,32] is an integration method based on several homogeneous integration methods, that is, integrated integration. HIVE-COTE [21] is the most accurate and representative meta-integration method available, and is an improved version of FLATCOTE [31] that integrates elastic integration [31], STC [17], BOSS [24], time series forest [14], and random interval spectrum integration [21]. HIVE-COTE achieves the highest accuracy on UCR time series datasets [7], but is extremely complex [5,21], making it difficult to apply to large-scale datasets.

The application of deep learning method to the task of time series classification has gained attention in recent years [6,19]. Wang et al. [19] and Fawaz et al. [6] demonstrated the powerful performance of fully convolutional neural networks (FCN) and residual neural networks (ResNet) in time series classification tasks.

3. Definition and Background

This section firstly provides a definition of the basic concepts, and then introduces the key techniques and theoretical foundations of this work: Shapelet transformation techniques and typical time series characteristics.

3.1. Definitions

Definition 1. *Time series. A time series T of length l is an ordered sequence of l observations of a variable, which can be expressed as* $T = <t_1, t_2, \ldots, t_l>$, *where* $t_i \in \mathbb{R}$.

Compared with the entire time series, here we pay more attention to the local fragments of the time series, which is a time series subsequence.

Definition 2. *Time series subsequence. A subsequence* $T_{i,m} = < t_i, t_{i+1}, \ldots, t_{i+m-1} >$ *of time series T refers to a fragment consisting of consecutive m values from index i to index* $i + m - 1$ *in* $\boldsymbol{T}$.

In this article, any subsequence can be seen as a Shapelet. According to a Shapelet's discriminating ability, there will be expressions such as "optimal Shapelet" or "optimal *k* Shapelets".

Definition 3. *Distance metric. The distances between subsequences can be used to reflect their similarity. For two subsequences* $\boldsymbol{S}^1$ *and* $\boldsymbol{S}^2$ *with the same length m, here we use the normalized Euclidean distance metric shown in Equation (1).*

$$dist(\boldsymbol{S}^1, \boldsymbol{S}^2) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(S_i^1 - S_i^2)^2} \quad (1)$$

When calculating the distance between a subsequence $\boldsymbol{S}$ with length m and a time series $\boldsymbol{T}$ with length l ($l > m$), the subsequence $\boldsymbol{S}$ needs to slide over the time series T to find the best matching subsequence $\boldsymbol{S}'$, and then take the best matching distance (i.e., the

distance between the subsequence $\boldsymbol{S}$ and $\boldsymbol{S'}$) as the distance between the subsequence $\boldsymbol{S}$ and the time series $\boldsymbol{T}$ (as shown in Equation (2)). Figure 2 describes the above process.

$$subDist(\boldsymbol{S}, \boldsymbol{T}) = \min_{1 \le i \le l-m+1} dist(\boldsymbol{S}, \boldsymbol{T}_{i,m}), \tag{2}$$

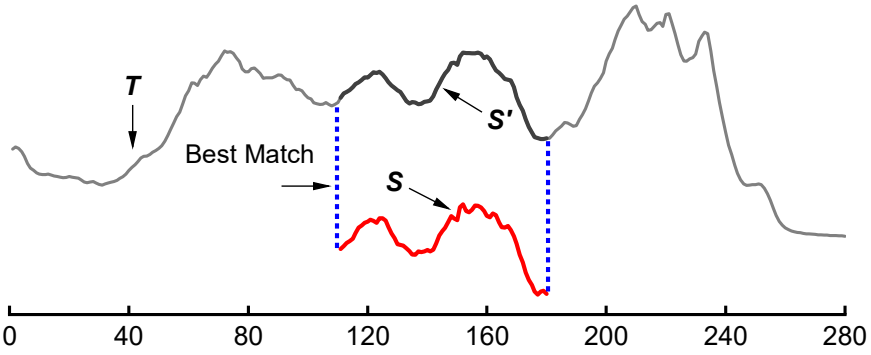


Figure 2. Illustration of subsequence matching.

It should be noted that most of classification methods based on Shapelet generally use Z-standard Euclidean disdance measurements in order to ensure that the amplitude offset of the subsequence is invariant—that is, Z-score is calculated according to Formula (3) before using Formula (1) to obtain the normalized European distance (μ is the mean of subsequence $\boldsymbol{S}$ and σ is the standard deviation of $\boldsymbol{S}$).

$$S_i^{norm} = \frac{S_i - \mu}{\sigma}, i = 1, 2, \cdots, m, \tag{3}$$

For computational efficiency reasons, the RSFCF algorithm uses a more efficient normalized Euclidean distance metric when searching for the best match $\boldsymbol{S'}$, and only uses the Z standardized Euclidean distance metric when calculating the best match distance.

Definition 4. *Time series classification. Given a training set* $\boldsymbol{D} = \{(T_1, y_1), (T_2, y_2), \cdots, (T_n, y_n)\}$ *containing n instances, each instance consists of a time series and its corresponding class label. The time series classification task aims to learn a classification model by training set* $\boldsymbol{D}$ *and uses the model to predict the categories of unlabeled time series.*

3.2. Shapelet Transforamtion Technology

In order to overcome the shortcomings of the original Shapelet-based classification algorithm that can only build a decision tree classification model, Hills et al. [16] proposed the Shapelet transformation technology, which separates the two stages of Shapelet discovery and the classifier training, and the converted data can be directly used to train various classification models, which greatly improves the flexibility of the application of Shapelet. Algorithm 1 describes the specific process of the Shapelet transformation algorithm.

The algorithm first scans the dataset to find the optimal k Shapelets (line 1, the specific process can be found in references [8,16]). The time series in the dataset are then converted into feature vectors in the Shapelet space. Specifically, for each time series $\boldsymbol{T}_i$ in the dataset, its distances from k Shapelet are calculated according to Equation (2). The vector $\boldsymbol{F}_i$ is then formed by the adding the k distances and $\boldsymbol{T}_i$'s class label y_i together, which is added as an instance to the transformed dataset $\boldsymbol{D'}$ (lines 3–10).

Algorithm 1. shapeletTransform(*D*,*min*,*max*,*k*)

Input: Dataset $\boldsymbol{D}$, Shapelet minimum length *min*, Shapelet maximum length *max*, number of Shapelets *k*
Output: Transformed dataset $\boldsymbol{D'}$
1. $\boldsymbol{Shapelets}$←*findBestKShapelets*($\boldsymbol{D}$,*min*,*max*,*k*);
2. $\boldsymbol{D'}$←{$\boldsymbol{F}_1$, $\boldsymbol{F}_2$,···,$\boldsymbol{F}_n$};
3. for $\boldsymbol{T}_i$ in $\boldsymbol{D}$
4. $\quad\boldsymbol{F}_i$←*null*;
5. $\quad$for Shapelet $\boldsymbol{S}_j$ in $\boldsymbol{Shapelets}$
6. $\quad\quad$*dist*, $\boldsymbol{S'}$←*subDist*($\boldsymbol{S}_j$,$\boldsymbol{T}_i$);
7. $\quad\quad\boldsymbol{F}_i$.*add*(*dist*);
8. end for
9. $\boldsymbol{F}_i$.*add*(y_i);
10. end for
11. return $\boldsymbol{D'}$;

As described in Section 3.1, when calculating the distance between Shapelet $\boldsymbol{S}_j$ and the time series $\boldsymbol{T}_i$, we obtained the subsequence $\boldsymbol{S'}$ (line 6) that is most similar to $\boldsymbol{S}_j$ in the $\boldsymbol{T}_i$ under the Euclidean distance measure. However, $\boldsymbol{S'}$ is not shown to be explicitly utilized in Algorithm 1. In Section 4, we will describe how to fully exploit the discriminative information contained in it by embedding typical time series features. It is also important to note that the Shapelets found in the training set are used when transforming the test set.

3.3. Typical Time Series Characteristics (Catch22)

Catch22 (22 canonical time series characteristics) is a feature set of 22 typical time series features which was designed to assist time series analysis, particularly time series classification, through a concise, diverse, and informative set of descriptive features.

The vast majority of time series in UCR time series datasets have been normalized by Z-score [7,15] (standardized time series whose mean value equals to 0, variance equals to 1). Catch22 was originally proposed to perform feature transformation on the entire time series, thus deliberately excluding features sensitive to mean and variance. However, the RSFCF algorithm proposed in this paper is based on the local characteristics of the time series, which performs feature transformation on the subsequence instead of the entire time series. Generally, the mean value and variance of subsequences contain a wealth of discriminating information [14]. For example, mean value can distinguish between subsequences of similar shapes but different amplitudes; variance reflects the degree of dispersion of subsequences. Therefore, in addition to the Catch22 feature set, we also introduce two features: mean and variance. Moreover, since the slope feature reflects the trend of subsequences well and was successfully applied in [3,14], we also introduce it into the algorithm. In total, the RSFCF algorithm proposed in this paper uses a total of 25 features, namely mean, variance, slope, and 22 Catch22 features. In the following, we refer to these 25 characteristics collectively as typical time series features.

4. Algorithm

This section describes in detail the random Shapelet forest algorithm (RSFCF). Firstly, a novel data transformation method is introduced, which fully excavates the discriminating information in Shapelet by embedding multiple time series features in Shapelet. Secondly, the construction and classification process of RSFCF model is described. The time complexity analysis of RSFCF algorithm is given at last.

4.1. Random Shapelet Transformation Embedded with Typical Time Series Features

To improve efficiency while reducing accuracy loss, a new random Shapelet transformation method embedded with typical time series features is proposed on the basis of the traditional Shapelet transformation technique introduced in Section 3.2.

The transformation method (Algorithm 2) first randomly selects k Shapelets from all possible subsequences of the training set according to the specified minimum and maximum length of Shapelet (line 1), and records the starting position of each Shapelet (e.g., ***Locations***[i] = 10 indicates that the starting position of the ith Shapelet is at index 10). To improve efficiency, for 25 time series typical features, we randomly select a features to perform subsequent transformations of the dataset instead of using all of them (line 2).

The transformation process that follows has two key differences from the traditional Shapelet transformation method. First, the restrictedSubDist method (line 7) limits the scope of Shapelet matching, i.e., allows Shapelet to have the maximum offset of each *shift* size on the left and right (as shown in Figure 3, Shapelet $\boldsymbol{S}$ can only look for the best match with other time series between the two dotted lines). In this way, the algorithm is still able to overcome the time warping problem that is prevalent in time series to a large extent, and the computational complexity of Shapelet matching is reduced from $O(l \cdot m)$ to $O(shift \cdot m)$ ($shift << l$).

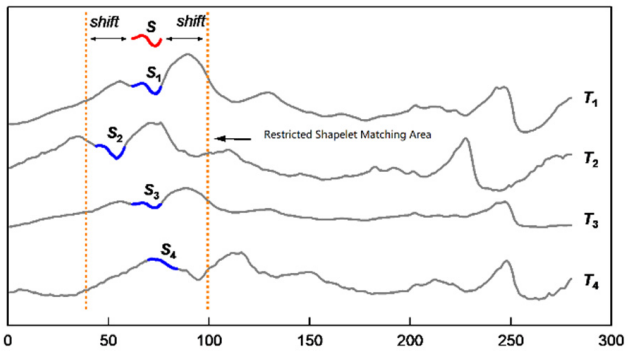


Figure 3. Illustration of Shapelet matching under restriction.

Second, when transforming time series $\boldsymbol{T}_i$ with Shapelet $\boldsymbol{S}_j$, the traditional method finds the subsequence $\boldsymbol{S'}$ most similar to $\boldsymbol{S}_j$ in $\boldsymbol{T}_i$, and only takes the normalized Euclidean distance of $\boldsymbol{S'}$ and $\boldsymbol{S}_j$ as a feature value of the transformed instance. On this basis, the proposed method calculates a different feature values of $\boldsymbol{S'}$, and merges the value of a value into the transformed instance (lines 9–12), so that the transformed data representation contains richer information.

Algorithm 2. ***randShapeletTransform-CF*(*D, min, max, k, a, shift*)**

Input: Dataset $\boldsymbol{D}$, Shapelet minimum length *min*, Shapelet maximum length *max*, number of Shapelets k, number of typical time series features a, Shapelet maximum offset *shift*
Output: Randomly selected ***Shapelets*** and their ***locations***, randomly selected feature index ***Atts***, transformed dataset $\boldsymbol{D'}$

1. ***Shapelets***, ***Locations***←*randSampShapelets*(***D***,*min*,*max*,*k*);
2. ***Atts***←*randSampAttIndices*({1,2,· · · ,25},*a*);
3. $\boldsymbol{D'}$←{$\boldsymbol{F}_1$, $\boldsymbol{F}_2$,· · · ,$\boldsymbol{F}_n$};
4. for $\boldsymbol{T}_i$ in $\boldsymbol{D}$
5. $\quad\boldsymbol{F}_i$←*null*;
6. $\quad$for Shapelet $\boldsymbol{S}_j$ in ***Shapelets***
7. $\quad\quad$*dist*, $\boldsymbol{S'}$←*restrictedSubDist*($\boldsymbol{S}_j$,$\boldsymbol{T}_i$,*shift*,***Locations***[*j*]);
8. $\quad\quad\boldsymbol{F}_i$.*add*(*dist*);
9. $\quad\quad$for *c*←1 to *a*
10. $\quad\quad\quad$*val*←*computeFeature*($\boldsymbol{S'}$,***Atts***[*c*]);
11. $\quad\quad\quad\boldsymbol{F}_i$.*add*(*val*);
12. $\quad\quad$end for
13. $\quad$end for
14. $\quad\boldsymbol{F}_i$.*add*(y_i);
15. end for
16. return ***Shapelets***, ***Locations***, ***Atts***, $\boldsymbol{D'}$;

4.2. Ensemble Classification Model Building

Algorithm 3 describes the process of building an RSFCF ensemble classification model. Algorithm 2 is first invoked to transform the training dataset (line 3), and then the transformed dataset is used to train the decision tree classifier (line 4). The Shapelets and features used in the dataset transformation are saved together in the corresponding decision tree (lines 5–7) for converting the test dataset when classifying. The above process will be repeated r times to build a forest containing r trees.

The construction of the time series tree follows the recursive strategy of the standard decision tree from top to bottom, which takes the information gain as the criterion, divides the instances of the current node with the best splitting threshold of the best splitting attribute at each node, constructing two subnodes on the left and right, and recursively carries out the process until all instances of the node belong to the same category. Regarding the calculation of the optimal split threshold for numeric attributes, the time series tree adopts a more efficient method, it divides the interval composed of the minimum and maximum values of the attribute into κ equal parts (here we set κ to a fixed value of 20), and the boundary between each cell is tested one by one as the candidate split threshold, and the threshold for obtaining the maximum information gain is the optimal split threshold for the attribute. In addition, for the split threshold to achieve the same information gain, the time series tree uses the method of maximizing the decision boundary to break the draw (to calculate a reasonable boundary value, the various attributes of the dataset need to be Z-score standardized before training the time series tree to avoid the algorithm's preference for attributes of a larger magnitude).

$$Margin_j^{\tau} = \min_{i=1,2,\cdots,n} |Att_i^j - \tau|, \tag{4}$$

Equation (4) shows how the decision boundary is calculated for the split threshold τ of the jth attribute, where Att_i^j is the value of the ith instance on the jth attribute.

Algorithm 3. ***buildRSFCF*(*D*,*min*,*max*,*k*,*a*,*shift*,*r*)**

Input: Dataset ***D***, Shapelet minimum length *min*, Shapelet maximum length *max*, number of Shapelets *k*, number of typical time series features a, maximum offset *shift*, number of decision trees *r*
Output: RSFCF classification model
1. ***RSFCFModel***←{***Tree***$_1$,***Tree***$_2$,$\cdots$,***Tree***$_r$};
2. for *i*←1 to *r*
3. ***Shapelets***, ***Locations***, ***Atts***, ***D'***←*randShapelet Transform-CF*(***D***,*min*,*max*,*k*,*a*,*shift*);
4. ***Tree***$_i$←*buildTimeSeriesTree*(***D'***);
5. ***Tree***$_i$.*add*(***Shapelets***);
6. ***Tree***$_i$.*add*(***Locations***);
7. ***Tree***$_i$.*add*(***Atts***);
8. end for
9. return ***RSFCFModel***;

Algorithm 4 describes the classification process of the RSFCF ensemble classification model. When classifying, RSFCF aggregates the classification results of all time series trees and gives the final classification by majority vote. It should be noted that before classifying using a time series tree, it is first necessary to transform the time series to be classified using the Shapelets stored in the tree and the attribute indices according to the method described in lines 6 to 13 of Algorithm 2.

Algorithm 4. ***classification*(*T*,*RSFCFModel*,*shift*)**

Inputs: Time series to be classified $\boldsymbol{T}$, Classification model ***RSFCFModel***, Shapelet maximum offset *shift*
Output: Category y of time series $\boldsymbol{T}$
1. $\boldsymbol{Y} \leftarrow null$;
2. for $\boldsymbol{Tree}_i$ in ***RSFCFModel***
3. $\quad \boldsymbol{T'} \leftarrow instanceTransform(\boldsymbol{T}, \boldsymbol{Tree}_i.\boldsymbol{Shapelets}, \boldsymbol{Tree}_i.\boldsymbol{Locations}, \boldsymbol{Tree}_i.\boldsymbol{Atts}, shift)$;
4. $\quad y_i \leftarrow \boldsymbol{Tree}_i.classifyInstance(\boldsymbol{T'})$;
5. $\quad \boldsymbol{Y}.add(y_i)$;
6. end for
7. $y \leftarrow majorityVote(\boldsymbol{Y})$;
8. return y;

4.3. Time Complexity Analysis

Training time complexity: RSFCF is an ensemble classification model composed of r time series trees. Building a time series tree first requires dataset transformation and then training on the transformed dataset. The time complexity of the latter is $O(n \cdot \log(n) \cdot k \cdot a)$ [14], where n is the number of training set instances, $\log(n)$ is the average depth of the tree, and the converted dataset has $O(k \cdot a)$ attributes (k is the number of randomly selected Shapelets, a is the number of features used when transfroming the dataset). Randomly selected Shapelets vary in length and the computational complexity of each feature varies, making it difficult to analyze the precise time complexity of the dataset transformation process, but still making reasonable estimates possible. The transformation of time series $\boldsymbol{T}_i$ over Shapelet $\boldsymbol{S}_j$ consists of two processes: (1) finding the subsequence $\boldsymbol{S'}$ most similar to $\boldsymbol{S}_j$ on the Euclidean distance measure in $\boldsymbol{T}_i$; (2) calculating the value of $\boldsymbol{S'}$ on a randomly selected features. Process 1 requires 2·*shift* normalized Euclidean distance calculations with complexity of $O(m)$ (m is the length of Shapelet $\boldsymbol{S}_j$), so the time complexity of the process is $O(shift \cdot m)$. On the basis of the Catch22 feature set, we introduce three features of linear computational complexity (mean, variance and slope), and randomly select a features of 25 features to calculate the feature values, since the average computational complexity of the 22 Catch22 features is approximately linear ($O(m^{1.16})$) [15], so the computational complexity of process 2 is also approximately linear in the mean sense, and the total time complexity of the two processes can be approximated as $O(shift \cdot m)$. Since n time series need to be transformed with k Shapelets, the time complexity of the dataset transformation is $O(n \cdot k \cdot shift \cdot \overline{m})$, where $\overline{m}$ is the average length of the Shapelets. Overall, the time complexity of training an RSFCF classification model is:

$$O(r \cdot (\underbrace{n \cdot k \cdot shift \cdot \overline{m}}_{\text{data transformation}} + \underbrace{n \cdot \log(n) \cdot k \cdot a}_{\text{time series tree training}}))$$

Classification time complexity: The time complexity of transforming test dataset is $O(k \cdot shift \cdot \overline{m})$, after which a traversal of the average $\log(n)$ nodes completes the classification of each tree. Therefore, the time complexity of the classification process is:

$$O(r \cdot (\underbrace{k \cdot shift \cdot \overline{m}}_{\text{testset transformation}} + \underbrace{\log(n)}_{\text{classification}}))$$

5. Experimental Analysis

This section first analyzes the parameter settings of the proposed algorithm RSFCF, then compares it with several of the most advanced time series classification algorithms to evaluate the accuracy and efficiency of RSFCF, and finally verifies the effectiveness of the RSFCF design strategy through experiments, finding that embedding typical time series features in Shapelet can effectively improve the classification accuracy. To improve

the reproducibility of the work, we provide the Java source code of the algorithm (https://github.com/gaozhenzhuo/RSFCF, accessed on 6 June 2022).

5.1. Parameter Settings

RSFCF has a total of six parameters to set (see Table 1). Since the length range of the pattern in the learning task cannot be known in advance, we simply set the minimum and maximum lengths of Shapelet to 3 and l, respectively (l is the time series length in the learning task). The number of Shapelets k and the Shapelet maximum offset *shift* randomly selected per tree can be used to flexibly control the efficiency of the algorithm, and larger parameter values can theoretically obtain higher accuracy with a decline in efficiency. Parameter k is set to $\sqrt{l}$ to seek a compromise between accuracy and efficiency. We conducted specialized experiments to verify the effect of the Shapelet maximum offset *shift*, the number of features a used in the data transformation and the number of trees r in the RSFCF classification model on RSFCF performance to guide parameter value setting.

Table 1. Parameter setting of RSFCF.

Parameter	Description	Value
min	Shapelet minimum length	3
max	Shapelet maximum length	l
k	The number of randomly selected Shapelets for each tree	$\sqrt{l}$
a	The number of features used in the data transformation	8
shift	The maximum offset of Shapelet	$l/10$
r	The number of trees in the RSFCF classification model	500

Effect of the number of trees r on the performance of RSFCF (a is set to 8, *shift* is set to $l/10$). For a reasonable assessment and to fully account for experimental efficiency, we selected all 60 datasets with a total number of instances less than 2000 and time series length less than 600 in the 112 datasets shown in Table 2 (i.e., the datasets with names bolded in Table 2, which we will abbreviate as Small60 datasets later). Since RSFCF is essentially a random algorithm, we repeated the experiment 10 times on each dataset of Small60.

Table 2. Accuracy of RSFCF on 112 UCR time series datasets (%).

Dataset	Accuracy	Dataset	Accuracy	Dataset	Accuracy	Dataset	Accuracy	Dataset	Accuracy
ACSF1	86.00	**ECG200**	86.00	**Herring**	67.19	PhaOC	82.52	**Strwbe**	96.76
Adiac	77.75	ECG5000	94.53	House20	97.48	Phoneme	38.19	**SwdLeaf**	95.20
ArrHead	81.71	**ECG5D**	100.00	InlSka	47.45	PigAirP	40.38	**Symbols**	98.69
Beef	76.67	ElecDev	74.57	InEPGRT	100.00	PigArtP	97.60	**SynCtl**	99.33
BeetFly	85.00	EOGHS	65.47	InEPGST	100.00	PigCVP	60.58	**ToeSeg1**	96.93
BirdChi	90.00	EOGVS	56.63	InWSnd	65.35	**Plane**	100.00	**ToeSeg2**	89.23
BME	100.00	EthLevel	57.60	**ItPwDem**	96.02	**PowCons**	99.44	**Trace**	100.00
Car	90.00	FaceA	76.51	LKitApp	80.00	**PrPhOAG**	82.93	**TwLECG**	99.82
CBF	100.00	**FaceF**	100.00	Light2	77.05	**PrPhOC**	87.63	TwPatt	99.80
Chtown	97.38	FacesU	92.63	**Light7**	75.34	**PrPhTW**	80.49	**UMD**	97.22
ChConc	70.89	**50Words**	78.68	Mallat	98.59	RefDev	58.13	UWaAll	97.54
CinCECG	93.70	**Fish**	97.14	**Meat**	93.33	Rock	86.00	UWaX	82.72
Coffee	100.00	FordA	92.27	**MdImg**	74.87	ScrType	52.00	UWaY	76.33
Comput	72.80	FordB	77.90	**MdPhOAG**	62.99	SeHGCh2	94.50	UWaZ	77.44
CriketX	76.67	FreeRT	100.00	**MdPhOC**	83.16	SeHMCh2	88.89	Wafer	99.74
CriketY	81.54	FreeST	99.93	**MdPhTW**	58.44	SeHSCh2	93.56	**Wine**	70.37
CriketZ	83.08	**GunP**	99.33	MixSRT	95.75	**ShpSim**	98.33	**WordSyn**	70.53
Crop	77.00	**GunPAS**	99.05	MixSST	94.14	**ShpAll**	85.67	Worms	75.32
DiaSRed	85.29	**GunPMF**	99.68	**MtStr**	93.37	SKitApp	81.87	WormsTC	79.22
DiPhOAG	74.82	**GunPOY**	100.00	NoECGT1	93.18	**SmthSub**	98.67	Yoga	89.47
DiPhOC	78.26	**Ham**	76.19	NoECGT2	94.71	**SonyRS1**	83.19		
DiPhTW	69.06	HandOut	92.70	**OliOil**	83.33	**SonyRS2**	85.73		
Earthqu	74.82	Haptics	53.57	**OSULeaf**	75.62	StarCur	98.13		

As shown in Figure 4, the accuracy of RSFCF increases with the size of ensemble. Although the average accuracy tends to stabilize when the number of trees exceeds 100, it can be seen from the box diagram in Figure 4 that the accuracy rate of RSFCF in 10 experiments at r = 500 is more stable than at r = 100. The efficiency comparison analysis in the next section shows that RSFCF can still maintain a significant advantage in efficiency when r is set to 500, and when there is a higher requirement for efficiency, setting r to 100 can achieve a five-fold efficiency improvement without causing a significant reduction in classification accuracy.

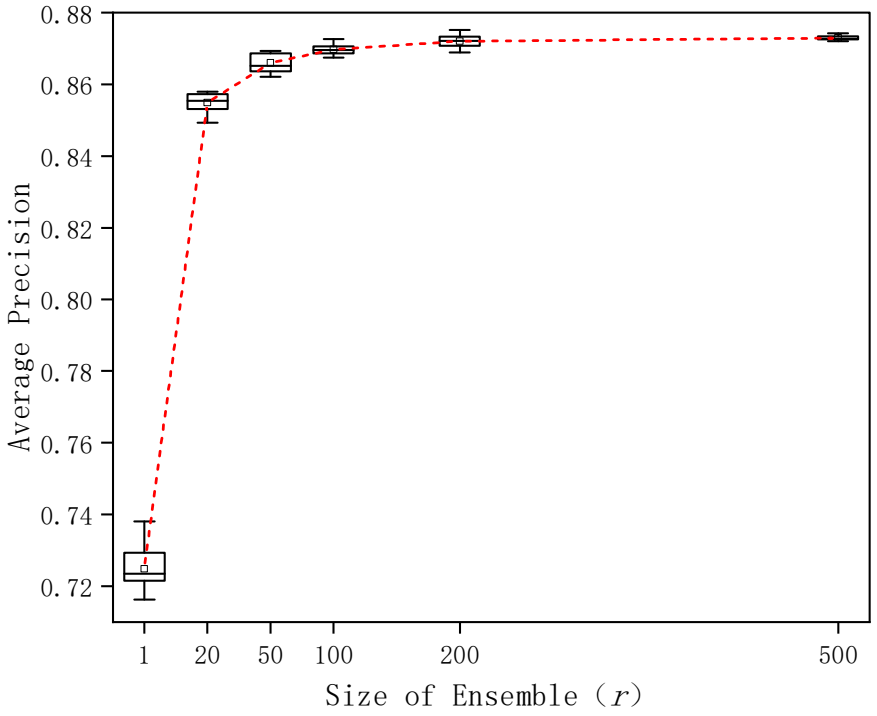


Figure 4. Average accuracy and variance of RSFCF over Small60 datasets under different ensemble sizes.

Effect of the number of features used in data transformation a on the performance of RSFCF (r set to 500, *shift* is set to $l/10$). Figure 5 shows how the average accuracy of RSFCF on the Small60 dataset compares to the average training time at different settings of parameter a. As the value of parameter a increases, the average accuracy generally shows an upward trend, and the average training time also increases (nearly linear). Since the design goal of RSFCF is to achieve accurate and efficient time series classification, we set a to 8, which loses only a small amount of accuracy but achieves nearly twice the efficiency improvement compared to using all features (i.e., setting a to 25).

Effect of the maximum offset of Shapelet *shift* on the performance of RSFCF (r set to 500, a is set to 8). Parameter *shift* is set to restrict the scope of the Shapelet matching process. To find the optimal setting of *shift*, different values from l·5% to l·50% are tested with Small60 datasets. As shown in Figure 6, as the *shift* value increases, the training time becomes significantly longer. This is the result of expanding the searching space of Shapelet matching. The classification accuracy increases from the beginning, reaches a stable state when the value of shift reaches 20% of the time series length, and even encounters a small drop at the end. This shows that although time warping exists commonly in time series datasets, the vast majority of shifts occur only within small areas, and it may be counterproductive to expand the matching space of the Shapelet. Again, to reach a compromise between efficiency and accuracy, we choose $l/10$ as the optimal value of *shift*. To sum up, subsequent experiments take the default parameter settings given in Table 1.

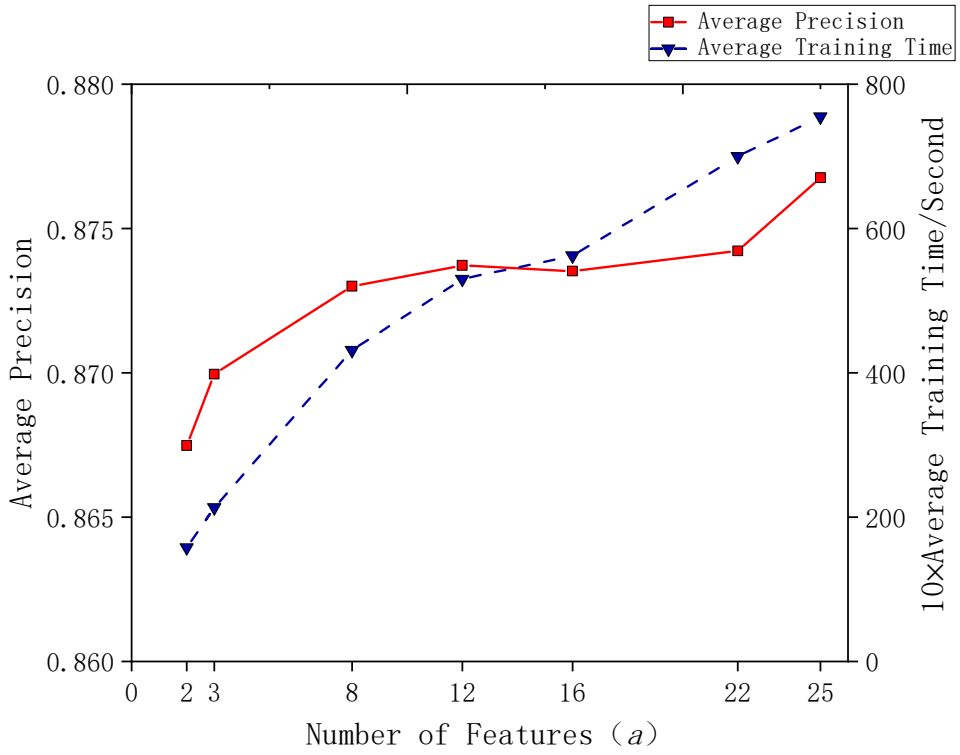


Figure 5. Average accuracy and training time of RSFCF over Small60 datasets under different settings of parameter *a*.

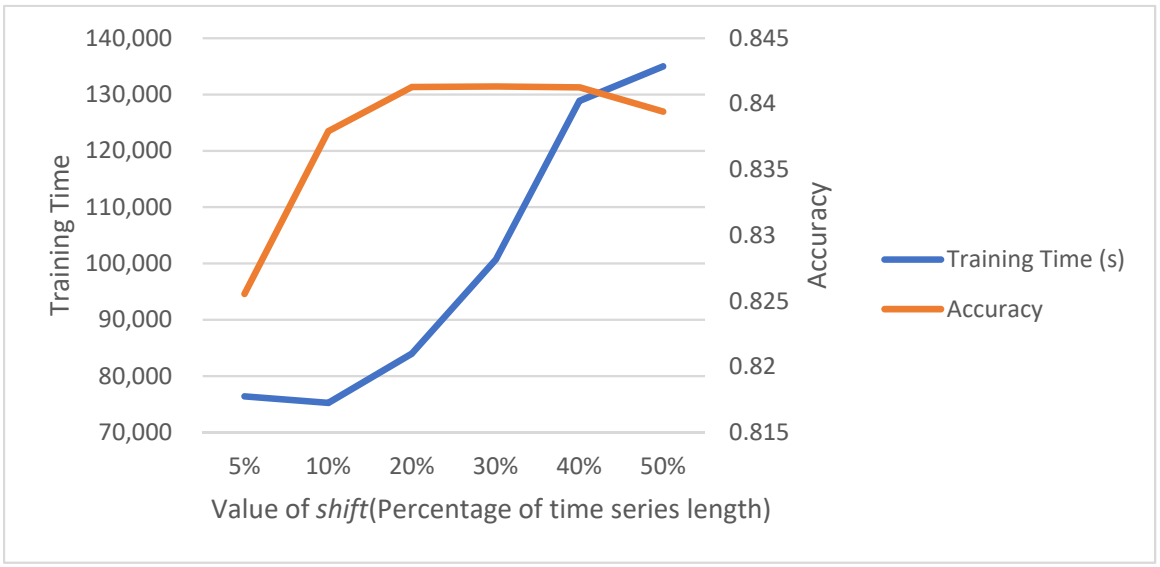


Figure 6. Average accuracy and training time of RSFCF over Small60 datasets under different settings of parameter *shift*.

5.2. Ablation Experiment

RSFCF randomly selects 8 out a total of 25 features, namely mean, variance, slope, and 22 Catch22 features. The ablation experiment is designed to test whether eight features are good enough for describing a time series by comparing it with using all Catch22 features and using mean, variance, slope features only. The experiment is conducted with all 112 time series datasets under three different settings. With each dataset, the algorithms with Catch22 features and with mean, variance, slope features are run only once, while the

original RSFCF is run 10 times to avoid contingency, as the eight features are randomly selected. The average accuracies are finally calculated as shown in Table 3.

Table 3. Ablation experiment result.

	RSFCF (8 Features)	Catch22 (22 Features)	Meanstdslope (3 Features)
Accuracy	84.52	84.50	82.52

The average accuracy of RSFCF is slightly higher than Catch22, showing that the algorithm benefits from adding the three features. What is more, it should be noted that the feature size of RSFCF is much smaller than Catch22. Using only mean, variance and slope features acquires the lowest average accuracy among the three. This means these three features are not enough to describe a time series when dealing with classification tasks.

5.3. Accuracy Comparison

The UCR time series dataset [7] is the standard dataset in the area of time series classification study [30], containing a total of 128 datasets. Bagnall et al. [18] recently conducted a new evaluation of time series classification algorithms using 112 of them (15 of the 16 excluded datasets are not typically used to evaluate algorithm performance due to inconsistent time series lengths or defect values [7,18], and the other is a "Fungi" dataset with only one training instance per category), and gave the accuracy of time series classification algorithms that represent the most advanced level of the current state of affairs.

In order to fully verify the performance of the RSFCF algorithm proposed in this paper, we used the same 112 datasets and compared them with the seven algorithms that performed the best and are most relevant: gRSF [10], STC [17], CIF [3], WEASEL [25], Proximity Forest [5], HIVE-COTE [21], ResNet [19] (the above algorithms are abbreviated as: gRSF, STC, CIF, WS, PF, HCT, RN). These algorithms belong to the six different types of time series classification methods introduced in Section 2, representing the advanced level of the corresponding types. The experiment follows the original training and test set split of the dataset [7], and the results of the comparison algorithms are taken from the results proposed by Bagnall et al. [18]. Table 2 shows the accuracy rate of algorithm RSFCF on 112 datasets.

We use the critical difference plots described in [32] to compare the accuracy of multiple classifiers across multiple datasets. We set the significance level α to 0.05, and then use the Friedman test to determine whether the hypothesis "there is no difference in the average ranking of accuracy between multiple classifiers on multiple datasets" is true, and if the hypothesis is not true, we then use the Nemenyi test to group the classifiers (the classifiers connected to the same line in Figure 7 form a group), and there is a significant difference in the average accuracy ranking of any two classifiers in different groups.

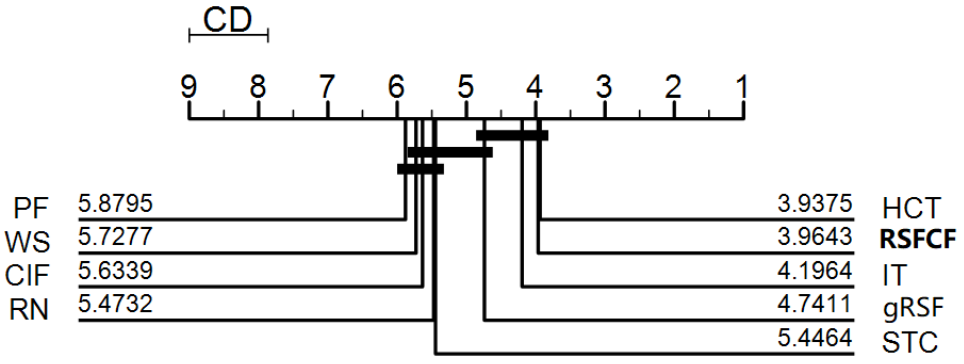


Figure 7. Average ranks and critical differences on accuracy of RSFCF and 7 comparison algorithms over 112 UCR datasets.

The critical difference plot shown in Figure 7 shows that the average ranking of RSFCF's accuracy rates outperforms gRSF, STC, CIF, and WEASEL, and significantly exceeds Proximity Forest. It is worth mentioning that according to the recent evaluation of Bagalll et al. [18], the above four algorithms represent the current most advanced level of time series classification methods based on Shapelet, interval, dictionary, and distance. Compared with deep learning methods, RSFCF surpasses the powerful baseline algorithm ResNet. The meta-ensemble method HIVE-COTE obtained the highest average accuracy ranking (2.9063), but there was no significant difference from the algorithm RSFCF in this paper, and the efficiency comparison in the next section verified that the efficiency of RSFCF was much higher than that of HIVE-COTE.

Table 4 shows the results of the two-by-two comparison of RSFCF and seven algorithms. Although HIVE-COTE ranks higher on average accuracy than RSFCF, it can be seen from Table 4 that RSFCF still surpassed HIVE-COTE on 36 datasets. Compared to the remaining six algorithms, RSFCF wins on near to or even more than half of the datasets, while the other algorithms only win on about one-third of the datasets, which highlights the accuracy advantage of RSFCF.

Table 4. Pairwise comparison of RSFCF and 6 comparison algorithms in terms of accuracy over 112 UCR datasets.

Comparison of Algorithms	Win/Draw/Loss
HIVE-COTE	36/12/**64**
STC	**65**/10/37
ResNet	**54**/13/45
CIF	**62**/19/31
WEASEL	**63**/11/38
Proximity Forest	**64**/14/34
gRSF	**60**/12/40

5.4. Efficiency Comparison

To evaluate the efficiency of the algorithm, we ran RSFCF and five comparison algorithms on the Small60 dataset using the same computer (Intel Core i7-7700 (3.60 GHz) processor, 16 GB of memory), recording the algorithm's CPU running time (ResNet was not included in this evaluation because it required a high-performance GPU to complete the training in acceptable time). Figure 8 shows the average training time (horizontal axis) and the average ranking of accuracy (vertical axis) on the Small60 dataset compared to the average ranking of accuracy (vertical axis) of the RSFCF and 4 comparison algorithms (HIVE-COTE was not included because the algorithm was not run on the full Small60 dataset after more than 5 days), and the algorithm is located in the lower left corner of the graph, indicating that its accuracy average ranking is higher and requires less training time. As can be seen from Figure 8, the efficiency of RSFCF far exceeds that of STC. RSFCF is slightly less efficient than WEASEL and CIF, but RSFCF ranks significantly higher on average for accuracy. The above comparative analysis shows that RSFCF is highly competitive in terms of both accuracy and efficiency.

5.5. Design Strategy Validation

The biggest innovation in this paper is to provide a novel idea for embedding typical time series features in randomly selected Shapelets. To verify the effectiveness of this strategy, we compared the accuracy of RSFCF and Naive Random Shapelet Forest (RSF) on 112 UCR datasets. RSF is a simplified version of RSFCF that does not embed typical time series features in Shapelet (this simplification can be achieved by setting the parameter a to 0 or commenting on lines 9–12 of Algorithm 2) (Figure 9).

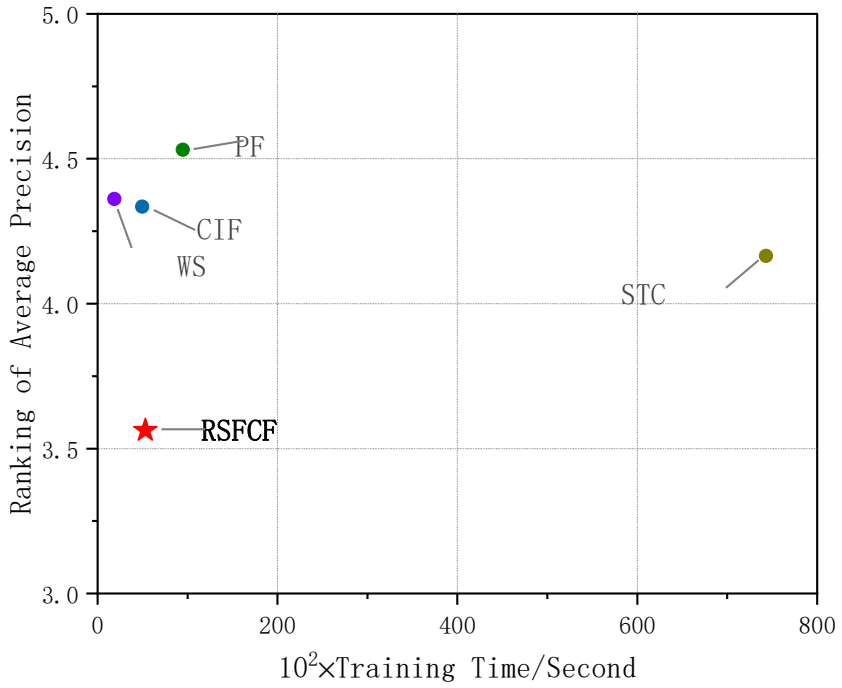


Figure 8. Comparison of RSFCF and 4 classifiers in terms of average training time and average ranks over Small60 datasets.

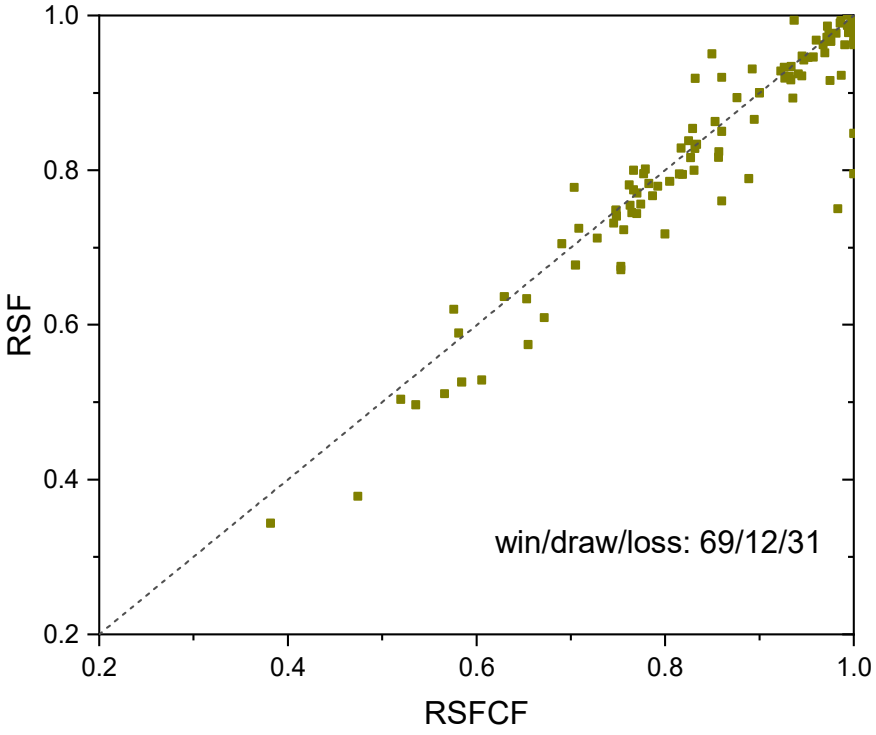


Figure 9. Comparison of random Shapelet forest classifiers with or without canonical time series features (RSFCF versus RSF) in terms of accuracy over 112 UCR datasets.

Figure 10 shows the results of the comparison of RSFCF and RSF, with RSFCF winning on 69 datasets, while RSF won on only 31 datasets. The reason why RSF is significantly weaker than RSFCF is that RSF does not embed other features, so it can only use the overall shape of Shapelet as a basis for distinguishing different categories of time series, but the randomly selected Shapelet is only a very small subset of all possible Shapelet,

so the probability of taking a discriminating "exact shape" is very low, thus affecting the classification accuracy. RSFCF has two significant advantages over RSF.

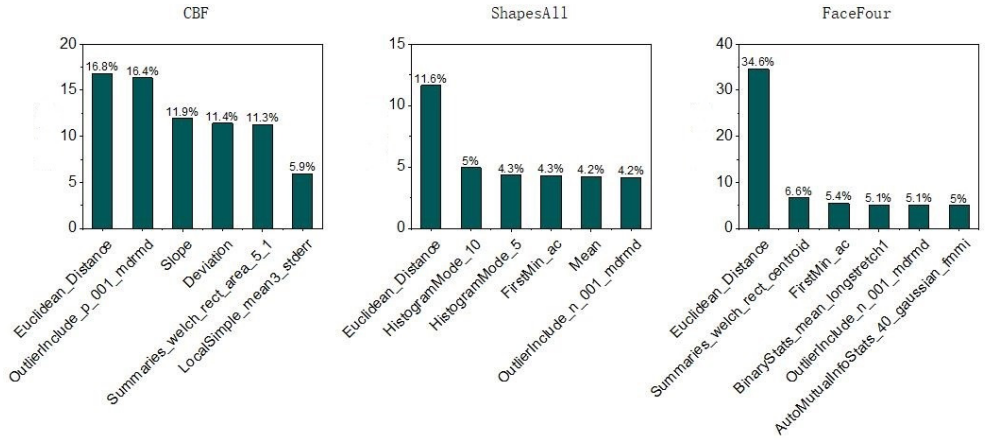


Figure 10. Feature contribution comparisons on three UCR datasets CBF, ShapesAll and FaceFour.

First, because the value of Shapelet on one or more features can reflect the discriminating shape information contained within it (as shown in Figure 1), RSFCF relaxes the requirements for Shapelet and thus reduces the risk of random selection of Shapelet, resulting in a loss of accuracy; second, in addition to the sequence shape, RSFCF can also capture discriminating information from multiple angles such as the numerical distribution characteristics, autocorrelation, and periodicity of the sequence, which is the most critical reason for which RSFCF surpasses RSF and STC. Based on the comparison results of RSFCF and RSF and the above analysis, we believe that embedding typical time series features in Shapelet plays a key role in improving classification accuracy.

6. Conclusions

The continuous expansion of data scale puts forward higher requirements for the efficiency of data mining algorithms. In order to perform accurate and efficient time series classification, a random Shapelet forest algorithm (RSFCF) embedded with typical time series features is proposed in this paper. RSFCF randomly selects Shapelet and limits the scope of Shapelet to improve efficiency, and embeds typical time series features in Shapelets to compensate for the loss of accuracy caused by random selection of Shapelet. The classification results on the 112 UCR time series datasets show that the accuracy of RSFCF surpasses that of multiple advanced time series classification algorithms and reaches the current leading level. The meta-integrated method HIVE-COTE is more accurate than RSFCF, but experiments have shown that its efficiency is much lower than that of RSFCF, so HIVE-COTE is difficult to apply to large-scale datasets. In summary, the RSFCF algorithm proposed in this paper takes into account both accuracy and efficiency, and has higher practicality. Future work includes studying a fusion strategy to embed RSFCF into the meta-integration method HIVE-COTE for a more precise classification of scenarios where real-time requirements are not high.

Author Contributions: Conceptualization, H.-Y.L.; methodology, Z.-H.W.; software, Z.-Z.G.; writing—original draft preparation, H.-Y.L.; writing—review and editing, Y.-H.D. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Fundamental Research Funds for the Central Universities, grant number 2019RC052, Beijing Natural Science Foundation, grant number 4214067 and National Key Research and Development Program of China, grant number 2021ZD0113002.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Grabocka, J.; Schilling, N.; Wistuba, M.; Schmidt-Thieme, L. Learning time-series shapelets. In Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (ACM 2014), Samelsonplatz, Hildesheim, Germany, 24 August 2014; pp. 392–401.
2. Yan, W.H.; Li, G.L. Research on time series classification based on shapelet. *Comput. Sci.* **2019**, *46*, 29–35.
3. Middlehurst, M.; Large, J.; Bagnall, A. The canonical interval forest (CIF) classifier for time series classification. *arXiv* **2010**, arXiv:2008.09172.
4. Lee, W.; Park, S.; Joo, W.; Moon, C.I. Diagnosis prediction via medical context attention networks using deep generative modeling. In Proceedings of the 2018 IEEE International Conference on Data Mining, Singapore, 17–20 November 2018; pp. 1104–1109.
5. Lucas, B.; Shifaz, A.; Pelletier, C.; O'Neill, L.; Zaidi, N.; Goethals, B.; Petitjean, F.; Webb, G.I. Proximity forest: An effective and scalable distance-based classifier for time series. *Data Min. Knowl. Discov.* **2019**, *33*, 607–635. [CrossRef]
6. Fawaz, H.I.; Forestier, G.; Weber, J.; Idoumghar, L.; Muller, P.A. Deep learning for time series classification: A review. *Data Min. Knowl. Discov.* **2019**, *33*, 917–963. [CrossRef]
7. Dau, H.A.; Keogh, E.; Kamgar, K.; Weber, J.; Idoumghar, L.; Muller, P.-A. The UCR Time Series Classification Archive. Available online: https://www.cs.ucr.edu/~{}eamonn/time_series_data_2018 (accessed on 6 June 2022).
8. Ye, L.; Keogh, E. Time series shapelets: A novel technique that allows accurate, interpretable and fast classification. *Data Min. Knowl. Discov.* **2011**, *22*, 149–182. [CrossRef]
9. Rakthanmanon, T.; Keogh, E. Fast shapelets: A scalable algorithm for discovering time series shapelets. In Proceedings of the 13th SIAM International Conference on Data Mining (SIAM), Austin, TX, USA, 2–4 May 2013; pp. 668–676.
10. Karlsson, I.; Papapetrou, P.; Boström, H. Generalized random shapelet forests. *Data Min. Knowl. Discov.* **2016**, *30*, 1053–1085. [CrossRef]
11. Li, G.L.; Yan, W.H.; Wu, Z.D. Discovering shapelets with key points in time series classification. *Expert Syst. Appl.* **2019**, *132*, 76–86. [CrossRef]
12. Zhang, Z.G.; Zhang, H.W.; Wen, Y.L.; Zhang, Y.; Yuan, X. Discriminative extraction of features from time series. *Neurocomputing* **2018**, *275*, 2317–2328. [CrossRef]
13. Mueen, A.; Keogh, E.; Young, N. Logical-shapelets: An expressive primitive for time series classification. In Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (ACM), San Diego, CA, USA, 11–24 August 2011; pp. 1154–1162.
14. Deng, H.; Runger, G.; Tuv, E.; Vladimir, M. A time series forest for classification and feature extraction. *Inf. Sci.* **2013**, *239*, 142–153. [CrossRef]
15. Lubba, C.H.; Sethi, S.S.; Knaute, P.; Schultz, S.R.; Fulcher, B.D.; Jones, N.S. Catch22: Canonical time-series characteristics. *Data Min. Knowl. Discov.* **2019**, *33*, 1821–1852. [CrossRef]
16. Hills, J.; Lines, J.; Baranauskas, E.; Mapp, J.; Bagnall, A. Classification of time series by shapelet transformation. *Data Min. Knowl. Discov.* **2014**, *28*, 851–881. [CrossRef]
17. Bostrom, A.; Bagnall, A. Binary shapelet transform for multiclass time series classification. In *Transactions on Large-Scale Data-and Knowledge-Centered Systems XXXII*; Springer: Berlin/Heidelberg, Germany, 2017; pp. 24–46.
18. Bagnall, A.; Lines, J.; Vickers, W. The UEA & UCR Time Series Classification Repository. Available online: http://www.timeseriesclassification.com (accessed on 6 June 2022).
19. Wang, Z.; Yan, W.; Oates, T. Time series classification from scratch with deep neural networks: A strong baseline. In Proceedings of the 2017 International Joint Conference on Neural Networks, Anchorage, AK, USA, 14–19 May 2017; pp. 1578–1585.
20. Lin, J.; Keogh, E.; Wei, L.; Lonardi, S. Experiencing SAX: A novel symbolic representation of time series. *Data Min. Knowl. Discov.* **2007**, *15*, 107–144. [CrossRef]
21. Lines, J.; Taylor, S.; Bagnall, A. Time series classification with HIVE-COTE: The hierarchical vote collective of transformation-based ensembles. *ACM Trans. Knowl. Discov. Data* **2018**, *12*, 1–35. [CrossRef]
22. Lin, J.; Khade, R.; Li, Y. Rotation-invariant similarity in time series using bag-of-patterns representation. *J. Intell. Inf. Syst.* **2012**, *39*, 287–315. [CrossRef]
23. Senin, P.; Malinchik, S. SAX-VSM: Interpretable time series classification using SAX and vector space model. In Proceedings of the 2013 IEEE 13rd International Conference on Data Mining, Dallas, TX, USA, 7–10 December 2013; pp. 1175–1180.
24. Schäfer, P. The BOSS is concerned with time series classification in the presence of noise. *Data Min. Knowl. Discov.* **2015**, *29*, 1505–1530. [CrossRef]
25. Schäfer, P.; Leser, U. Fast and accurate time series classification with WEASEL. In Proceedings of the 2017 ACM Conference on Information and Knowledge Management, Singapore, 6–10 November 2017; pp. 637–646.
26. Schäfer, P.; Högqvist, M. 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, Berlin, Germany, 26–29 March 2012; pp. 516–527.
27. Middlehurst, M.; Vickers, W.; Bagnall, A. Scalable dictionary classifiers for time series classification. In Proceedings of the International Conference on Intelligent Data Engineering and Automated Learning, Manchester, UK, 14–16 November 2019; pp. 11–19.

28. Yuan, J.D.; Lin, Q.H.; Zhang, W.; Wang, Z.H. Locally slope-based dynamic time warping for time series classification. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management (ACM 2019), Beijing, China, 3–7 November 2019; p. 17131722.
29. Bagnall, A.; Lines, J.; Bostrom, A.; Large, J.; Keogh, E. The great time series classification bake off: A review and experimental evaluation of recent algorithmic advances. *Data Min. Knowl. Discov.* **2017**, *31*, 606–660. [CrossRef] [PubMed]
30. Lines, J.; Bagnall, A. Time series classification with ensembles of elastic distance measures. *Data Min. Knowl. Discov.* **2015**, *29*, 565–592. [CrossRef]
31. Bagnall, A.; Lines, J.; Hills, J. Time-series classification with COTE: The collective of transformation-based ensembles. *IEEE Trans. Knowl. Data Eng.* **2015**, *27*, 2522–2535. [CrossRef]
32. Demšar, J. Statistical comparisons of classifiers over multiple datasets. *J. Mach. Learn. Res.* **2006**, *7*, 1–30.