Research Article

Time Series Classification by Shapelet Dictionary Learning with SVM-Based Ensemble Classifier

Jitao Zhang,¹ Weiming Shen,¹ Liang Gao,¹ Xinyu Li,¹ and Long Wen²

¹State Key Laboratory of Digital Manufacturing Equipment and Technology, School of Mechanical Science and Engineering, Huazhong University of Science and Technology, Wuhan 430074, China
²School of Mechanical Engineering and Electronic Information, China University of Geosciences, Wuhan 430074, China

Correspondence should be addressed to Weiming Shen; wshen@ieee.org

Received 30 January 2021; Revised 4 March 2021; Accepted 12 March 2021; Published 22 March 2021

Academic Editor: Mario Versaci

Copyright © 2021 Jitao Zhang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Timeseriesclassificationisabasicandimportantapproachfortimeseriesdatamining.Nowadays,moreresearcherspayattention
totheseshapesimilaritymethodincludingShapelet-basedalgorithmsbecausethecanextractdiscriminativesubsequencesfromtimeseries.However,mostShapelet-basedalgorithmsdiscoverShapeletsbysearchingcandidatessubsequencesintrainingdatasets,whichbringstwodrawbacks:highcomputationalburdenandpoorgeneralizationability.Toovercomethesedrawbacks,this
paperproposesanovelalgorithmnamedShapeletDictionaryLearningwithSVM-basedEnsembleClassifier(SDL-SEC).
SDL-SECmodifiestheshapeletalgorithmfromtwoaspects:Shapeletdiscoverymethodandclassifier.Firstly,aShapelet
DictionaryLearning(SDL)isproposedasanovelShapeletdiscoverymethodtogenerat SHAPELETSinsteadofsearchingthem.In
thisway,SDLownstheadvantagesoflowercomputationalcostandhighergeneralizationability.Then,anSVM-basedEnsemble
Classifier(SEC)isdevelopedasanovelensembleclassifierandassembledtothetSDLalgorithm.DifferentfromtheclassicSVMthat
needspreciseparametersliningandappropriatefeaturesselection,SECcanavoidoverfittingcausedbyalargenumberoffeatures
andparameters.Comparedwiththebaselineson45datasets,theproposedSDL-SECalgorithmachievesacompetitiveclas-
sificationaccuracywithlowercomputationalcost.

1. Introduction

Timeseriesclassification(TSC)isathoreticalabstractionof
manyengineeringproblems,suchasfaultdiagnosis,speech
recognition,andelectroencephalogram(EEG)identification.
It has become an active research field in recent years [1].
Toaddressthechallengeoftscproblems,severalmethodshave
beenproposedintheliterature.Anintuitiveway	omodeltimeseriesbycomparingthedifferencesintime
domainiscalledtimedomain similaritymethod. Timedomain similaritymethodsprocesstimeseriesashigh-
dimensional points. A basic framework of timedomain similaritybasedTSCmethodisutilizingdifferentdistance
measurements to quantify similarity between time series and
then combining 1-NN algorithm as the classifier. Several L₁
or L₂ norm distance measurements have been widely
researchedin[2]. Tosolvetimeaxisdistortionproblemintime
series, elastic distance measures are employed in TSC.
Dynamic timewarping(DTW)comestobea popularelastic
distance measurement. Several variants of DTW are com-
pared in [2]. At the same time, some timeriesis
discretizationtechniquesareproposed,suchasymbolic
aggregateapproximation(SAX)[3],toreduceseries
dimension and improve computational efficiency. Fuzzy
similarity(FS)[4]isadaptedintothecharacterizingdefects
problem, which is capable of processing time series signals
affected by uncertainty and inaccuracy. In literatures, tim
domain similarity methods are proved intuitive and efficient
but not suitable for complex dynamic systems.

The modelsimilaritymethodisanotherwell-knownway
todealwithTSCproblems.Thesemethodsrepresenttimeseriesbyusingmultiplestatisticalmodels,suchasAuto
Regressive Moving Average (ARMA) [5], Hidden Markov
Model (HMM) [6], and Gaussian Mixture Models (GMM)
[7]. By comparing the parameters of the models, different class time series can be distinguished. In recent years, Deep Learning has become a common approach in machine learning and artificial intelligence fields, more and more neural network-based models have been introduced into TSC [8, 9]. Although model similarity methods have a high accuracy, the models used have a high level of abstraction, so these methods are not interpretable.

Recently, more researchers are interested in the shape similarity method. It is human instinct to distinguish objects by their shapes. Such methods suggest imitating human intuition and use shapes to distinguish different classes of time series. Shape similarity methods discover local shape features, while other methods discover global statistic features. Therefore, these methods can obtain high accuracy with better interpretability. The base model of this article, the Shapelet algorithm, is a kind of shape similarity methods. Shapelet is a special concept which means one discriminative subseries in the time series [10]. More than high accuracy, Shapelet can also provide visualization results, which can point out further research directions for domain experts [11].

Most of the existing Shapelet-based algorithms attend to discover Shapelets by searching candidates in the training dataset. Such algorithms have two drawbacks. First, the search requires extensive computation. As an illustration, the complexity of the original Shapelet algorithm is $O(n^m n^2)$, where $m$ is the length of time series and $n$ is the number of instances in the dataset. Second, the searched Shapelet lacks generalization. Each Shapelet must be a segment in the existed instance, while the most discriminative Shapelet may never appear in the historical data. Thus, this paper proposes a novel algorithm named Shapelet Dictionary Learning with SVM-based Ensemble Classifier (SDL-SEC), which contributes following two points:

(i) **Shapelet Dictionary Learning.** Inspired by Dictionary Learning (DL), the proposed algorithm generates subseries (Shapelets) by optimizing an object function. Different from the searching method, SDL generates Shapelets directly.

(ii) **SVM-Based Ensemble Classifier.** According to our experience, different time series are sensitive to different features and different parameters. Classic SVM takes a lot of time to tune parameters and select feature subsets. To address this problem, we train a set of SVM models with randomly selected features and parameters and get final results through majority voting.

The rest of this paper is organized as follows: Section 2 reviews the research literature related to Shapelet algorithms and Dictionary Learning; Section 3 describes the implementation of the proposed algorithm SDL-SEC; Section 4 presents the results with the baselines in 45 datasets; Section 5 concludes the paper and discusses future research directions.

## 2. Related Work

### 2.1. Shapelet Algorithm

The original Shapelet algorithm was constituted in [10] for time-series classification problem. In brief, there are three steps in the training stage of the original edition, as exhibited in Figure 1. First, a sliding window is used to extract time series segments, which are candidate Shapelets, and further figure out the minimum distances between candidate Shapelets and total time series in the dataset. Then, a candidate Shapelet orderline is built by arranging the minimum distances from small to large; thus, we can figure out the Information Gain (IG) and Optimal Split Point (OSP) of each orderline. High IG represents a good discrimination. The third step is to choose k-best Shapelets with higher IG. At the inference stage, a decision tree is built by k-best Shapelets’ OSPs, and untagged time series are classified by this decision tree.

Followed the principle of Shapelet, many modified Shapelet-based algorithms have been proposed in recent years. These approaches have two main directions: speed-up the running time and improve the accuracy.

The original Shapelet algorithm as described above is a recursive search method, calls for a high time complexity. Hence, some speed-up techniques are indispensable for improving Shapelet algorithm efficiency. Some efficient pruning strategies are suggested to avoid searching for unproductive Shapelets [11]. The representation methods map time series from the source space to a reduced-dimensional space, which reduce the searching time significantly [12]. Moreover, parallelizing the Shapelet algorithm makes it possible for GPU operations [13]. However, the speed-up techniques do not involve the substantial improvement of Shapelet discovery and therefore limit the accuracy of this family of methods.

On the other hand, some Shapelet-based algorithms focus on improving accuracy. Grabocka et al. [11] suggest getting optimal Shapelet by optimizing a logistic loss objective function, which can further improve the classification accuracy. Instead of IG, Kruskall–Wallis and Mood’s median are employed as quality measurement for Shapelet selection [14]. Meanwhile, Hills et al. [14] suggest a feature transformation technique, which unbind the Shapelet algorithm from the decision tree classifier. Generally, this family of methods obtains high accuracy, but the computation cost is still high.

In summary, most Shapelet-based algorithms do not achieve a good balance between efficiency and performance. A better Shapelet discovery mechanism is needed to improve the accuracy and reduce the runtime.

### 2.2. Dictionary Learning

Dictionary Learning is a widely used machine learning algorithm. Its main idea is assuming that signals can be represented by a linear combination of dictionaries. Strictly, the mathematical form of Dictionary Learning can be organized as

$$
\text{arg min}_{a, Z} \frac{1}{2} ||T - aZ||_2^2 + \lambda ||a||_1,
$$

subject to $||z_k||_2 \leq c, k = 1, \ldots, K$,

where $T = [T_1, T_2, \ldots, T_n] \in \mathbb{R}^{m \times n}$ is the input time series signal, $Z = [Z_1, Z_2, \ldots, Z_K] \in \mathbb{R}^{K \times m}$ is a dictionary,
\( \alpha \in \mathbb{R}^{n \times K} \) is a sparse coefficient, and we call \( \{Z_k\} \in Z \) an atom of dictionary.

Problem (1) is nonconvex when optimizing \( \alpha \) and \( Z \) together. However, we can draw this problem to a two-step method: coefficient updating step and dictionary updating step. If \( Z \) is fixed, \( \alpha \) updating subproblem is convex. Orthogonal Matching Pursuit (OMP) [15], Lasso [16], or Alternating Direction Method of Multipliers (ADMM) [17] are common methods to update \( \alpha \). If \( \alpha \) is fixed, \( Z \) updating subproblem is a constrained least squares problem. Many convex optimization methods can solve this problem such as Fast Iterative Shrinkage-Thresholding Algorithm (FISTA) [18], gradient descent [19], and K-SVD [20]. Perform the above two steps alternately until convergence.

There are some similarities between Shapelet and Dictionary. Shapelet represents the time series local shape features; thus, the linear combination of Shapelets can represent time series partially or totally. This behaviour is similar to the main idea of Dictionary Learning. The linear combination of Dictionaries is considered to have the high representation ability for raw data. From this sight, those two kinds of methods can share the same underlying technique.

To this end, Dictionary Learning technique is introduced to Shapelet discovery, which obtains a set of Shapelets by solving several convex optimization problems. This will greatly reduce the amount of computation in the training phase. Different from search-based methods, the optimized Shapelet is a novel segment which does not appear in the existing data. This will improve the generalization ability of Shapelets.

### 3. Shapelet Dictionary Learning with SVM-Based Ensemble Classifier

In this section, we introduce the structure of Shapelet Dictionary Learning with SVM-based Ensemble Classifier. SDL is a generative Shapelet discovery algorithm which combines DL and Shapelet. We train SDL in a supervised way and get subdictionary for each class. Then, the transformation technique uses subdictionaries to map time series from time domain to feature domain. Last, we present an SVM-based Ensemble Classifier to improve accuracy further.

#### 3.1. Build SDL Atom

The atoms of the original DL present global features, while Shapelets present local features. So, we must reconstruct the SDL atoms before making the overall model. From the introduction in Section 2.2, the original DL atom dimension is equal to the input signal dimension. More specifically, consider the following situation, an input signal \( T \) with \( m \) dimension, and the atom in DL with \( p \) dimension. In the original DL based algorithms, \( p \) is set equal to \( m \) inevitably. Nevertheless, in the SDL algorithm, an atom is a representation of the Shapelet; thus, the dimension \( p \) needs small than \( m \). To address this problem, we introduce the Shift-Invariant Dictionary Learning (SIDL) technique [21] into our work, which use a sliding window to build the SDL atom and relax the constraint of \( p \) to \( p \leq m \).

Figure 2 explains how to generate the SDL atom. The SDL atom is denoted as \( d' \in \mathbb{R}^m \), and the Shapelet is denoted...
as \(d \in \mathbb{R}^p\), where \(p \leq m\). Shapelet may match anywhere in the time series, thus the sliding window which represents a Shapelet, which is sliding along the SDL atom to match the optimal location. In an SDL atom, only the Shapelet part has practical significance, and the rest part is assigned 0. In Figure 2, a Shapelet is matched at location \(q\), the values in red blocks need to be optimized, while the blue blocks are assigned 0. The matched location \(q\) is unique for each time series.

\[
Q(d, q) = \text{a function describing the mapping relationship between the Shapelet and the SDL atom. Given } q \text{ and } d, \text{ the SDL atom can be generated as follows } [21]:
\]

\[
d'_{i} = Q(d, q)_i = \begin{cases} 
    d_{i-q}, & \text{if } q + 1 \leq i \leq q + p, \\
    0, & \text{otherwise.}
\end{cases} \tag{2}
\]

3.2. The SDL Model. To build the complete SDL model, further more constraints must be made. The complete SDL model is

\[
\begin{align*}
\underset{a_k, d}{\text{arg min}} & \frac{1}{2} \sum_{i=1}^{n} \left\| T_i - \sum_{k=1}^{K} a_{ik} Q(d_k, q_i) \right\|_2^2 + \lambda \sum_{i=1}^{n} \| a_i \|_1 \\
\text{s.t.} & 0 \leq q_{ik} \leq m - p, \quad i = 1, \ldots, n; \quad k = 1, \ldots, K \\
& a_{ik} \geq 0, \quad i = 1, \ldots, n; \quad k = 1, \ldots, K,
\end{align*} \tag{3}
\]

where \(T_i\) is a time series from the input dataset; the sparse coefficient \(\{a_{ik}\}\), the corresponding shifts \(\{q_{ik}\}\), and the Shapelet dictionary \(D = [d_1, d_2, \ldots, d_K] \in \mathbb{R}^{K \times p}\) are outputs of the model which need to be optimized; the length of the Shapelet \(p\), the number of Shapelets \(K\), the weight \(\lambda\), and the scale factor \(c\) are hyperparameters which need to be tuned. The scale constraint of \(d_k\) avoids producing trivial solutions. The constraint of \(q\) controls the sliding window does not exceed the boundary. The nonnegative constraint of sparse coefficients is added to avoid the Shapelet inverted.

Equation (3) is an unsupervised learning model, while time series classification is a supervised problem. To deal with this, we learn sub-Shapelet dictionaries for each class independently.

3.3. Optimization of the SDL. Equation (3) is a nonconvex problem. Commonly, a two-step optimization strategy is suitable for this problem. Sparse coefficients and Shapelet dictionary are updated alternately. The subobject of each step is convex.

3.3.1. Update Sparse Coefficients. In this step, sparse coefficient is updated. Fix the \(d\), and equation (3) turns to

\[
\begin{align*}
\underset{a_k}{\text{arg min}} & \frac{1}{2} \left\| T_i - \sum_{k=1}^{K} a_{ik} Q(d_k, q_i) \right\|_2^2 + \lambda \sum_{i=1}^{n} \| a_i \|_1 \\
0 & \leq q_{ik} \leq m - p, \quad i = 1, \ldots, n; \quad k = 1, \ldots, K \\
\text{s.t.} & a_{ik} \geq 0, \quad i = 1, \ldots, n; \quad k = 1, \ldots, K,
\end{align*} \tag{4}
\]

where \(\{a_{i,j}\}_{ij}\) and \(\{q_{ik}\}_{ik}\) are fixed.

We use an enumeration method to find the best \(q_k\), which calculates the object value with all possible locations, and the solution is located which minimizes equation (5):

\[
q^*_k = \underset{q_k}{\text{arg max}} Q(d_k, q_k)^T \tilde{T}_i, \tag{6}
\]

where \(\tilde{T} = T_i - \sum_{j \neq k} a_{ij} Q(d_j, q_j)\).

Because of the nonnegative constraint of \(a\), equation (5) is different from that of [21]. With the optimal \(q^*_k\), the update solution of \(a_k\) is

\[
a^*_k = \begin{cases} 
    \frac{Q(d_k, q^*_k)^T \tilde{T} - \lambda}{\| d_k \|^2}, & \text{if } Q(d_k, q^*_k)^T \tilde{T} > \lambda, \\
    0, & \text{otherwise.}
\end{cases} \tag{7}
\]

3.3.2. Update Dictionary. In this step, we fix \(a\) and the \(q\) and update the Shapelet dictionary \(d\). Further, we fix \(\{d_j\}_{j \neq k}\) and optimize \(d_k\) independently. Thus, the problem (3) turns to

\[
\begin{align*}
\underset{d_k}{\text{arg min}} & \frac{1}{2} \sum_{i=1}^{n} \left\| T_i - a_{ik} Q(d_k, q_i) \right\|_2^2 + \sum_{j \neq k} \alpha_{ij} Q(d_j, q_{ij}) \right\|_2^2 \\
\text{s.t.} & \| d_k \|^2 \leq c.
\end{align*} \tag{8}
\]

Equation (8) is a least square with quadratic constraints, and it can be optimized via the Lagrange Multiplier method. The optimal \(d_k\) is
Supervised Shapelet Dictionary Learning Method for Classification Problem. In this subsection, we introduce the supervised SDL to obtain subdictionary of each class and use subdictionaries to transform time series from the time domain to the feature domain.

We train SDL model for L classes independently and obtain L subdictionaries. The complete Shapelet dictionary \( S = [D_1, D_2, \ldots, D_h, D_l, \ldots, D_L] \), where \( D_i \) means \( i \)th subdictionary. See Algorithm 1 for pseudo-code of complete supervised SDL.

In the original Shapelet algorithm, the decision tree is embedded in the training process. In other words, the original Shapelet algorithm cannot use classifiers other than the decision tree. In order to unbind Shapelet discovery and decision tree, we use Shapelet transformation technique to generate features which are suitable for any classifiers. Suppose \( H = L \times K \) means the number of Shapelets in \( S \). Transformation technique calculates the minimum distance \( v_{ij} \) between \( T_i \) and \( d_p \) and formed the feature vectors \( V = [v_{i1}, v_{i2}, \ldots, v_{iH}] \). \( V \) is actually a local reconstruction error. We do not feed sparse coefficients \( \alpha \) to classifier because they lack discriminative ability. By implementing the transformation technique in both training set and testing set, we get \( V_{\text{train}} \) and \( V_{\text{test}} \).

The choice of classifier is discretionary, and we use an SVM-based Ensemble Classifier which is described below.

3.5. SVM-Based Ensemble Classifier. SVM [22, 23] is a widely used classifier with high performance and low computational cost. We choose SVM as a base classifier. However, a single classifier is easily affected by uncontrollable factors such as noise, resulting in unstable performance. The Ensemble Learning method combines the results of weak base classifiers to form a strong classifier and improves the overall robustness of the algorithm [24]. Many ensemble SVM algorithms have been proved to be more powerful than single SVM algorithms [25, 26].

Specifically, two problems need to be considered in the construction of ensemble methods. The first problem is the selection of features. We use redundant dictionaries, which result in a large number of Shapelets, and the sample after the transformation operation is a high-dimensional vector. Generally, researchers useICA or other dimension reduction methods to reduce the features. However, different samples may be sensitive to the features of different dimensions. Dimension reduction will lead to the loss of information. The second problem is the selection of SVM parameters. There are many superparameters that need to be tuned in the SVM algorithm. Finding the optimal parameters is a complex problem. At the same time, like the first problem, different samples may be sensitive to different parameters.

In order to address these two problems, we designed two strategies. First, instead of using all features, each base SVM uses randomly selected features for training. As long as the number of base SVMs is enough, each feature will be used, and no information will be lost without dimension reduction. Second, the parameters of each base SVM are generated randomly, which avoids the overfitting caused by using a set of specific parameters. As shown in Figure 3, the steps of the SVM-based Ensemble Classifier are as follows:

1. Construct \( B \) training subsets and testing subsets. Each training subset randomly selects \( E \) features from the training feature set \( V_{\text{train}} \), and the same \( E \) features are selected from the testing feature set \( V_{\text{test}} \) to construct the corresponding testing subset. A feature is a column in the feature set. So, we get a series of subsets, \( V_{\text{train}} = [v_{b1}^{\text{train}}, v_{b2}^{\text{train}}, \ldots, v_{bE}^{\text{train}}, b = 1, 2, \ldots, B] \), where \( v_{bE}^{\text{train}} \) is a random column of \( V_{\text{train}} \), and \( v_{bf}^{\text{train}} \neq v_{bj}^{\text{train}} \), \( v_{bf} \neq v_{bf} \). In the same way, we get testing subsets \( V_{\text{test}} = [v_{v1}^{\text{test}}, v_{v2}^{\text{test}}, \ldots, v_{vE}^{\text{test}}, v_{vE}^{\text{test}}, b = 1, 2, \ldots, B] \).
2. For each training subset \( V_{b}^{\text{train}} \), train an SVM model: \( SVM_b(k_{tb}, d_{tb}, g_{tb}, p_{tb}) \), where kernel type \( k_{tb} \), degree in kernel function \( d_{tb} \), gamma in kernel function \( g_{tb} \), and coef0 in kernel function \( p_{tb} \) are all chosen randomly. Then, test \( V_{b}^{\text{test}} \) by using model \( SVM_b \) and return test result \( \hat{Y}_{b}^{\text{test}} \).
3. Deploy majority voting to get the final test result \( \hat{Y}_{\text{test}} \).

4. Experimental Results and Analyses

4.1. Dataset and Baseline Description. The 45 datasets from UCR Time Series Classification Repository [27] are used to verify the proposed algorithm. Table 1 summarises the details of 45 datasets.

SDL-SEC is a modified version of the classical Shapelet algorithm; thus, we chose two typical Shapelet-based algorithms for comparison. More than that, a Deep Learning algorithm is also compared. The following three algorithms are chosen as baselines:

- **Scalable Shapelet Discovery (SD) [11]** uses an online clustering and pruning technique to avoid repeatedly measuring the classification accuracy of similar subsequences. SD takes low computational cost and handles GB-scale data in several minutes.
- **Learning Time-Series Shapelets (LTS) [28]** learns top-K Shapelets through optimizing a cost function which consists of classification accuracy and regularization terms. LTS reaches a strong performance among many Shapelet-based algorithms.
- **Fully Convolutional Network (FCN) [29]** is a Deep Learning structure method. Since the Deep Learning
method plays an important role in many domains, and FCN is a strong baseline as authors claimed in their papers.

We reproduced the results of LTS and FCN with the open-source code (http://www.timeseriesclassification.com/code.php, https://github.com/cauchyturing/UCR_Time-Series_Classification_Deep_Learning_Baseline). SD is well tested, and the hardware performance is close to ours, so we reuse the results in [11] directly. Also, we compare single SVM classifier with SVM-based Ensemble Classifier. Single SVM version is denoted as SDL-S, and ensemble version is denoted as SDL-SEC. Classification Accuracy is used as a quantitative performance comparison measure, which is the ratio of truly classified instances to total test instances. The runtime is used as a quantitative efficiency comparison measure, which includes training time and testing time.

Our hardware environment is CPU: i7-9750H, RAM: 16 GB. We use Matlab 2019b to implement the SDL-SEC algorithm. SD and LTS are implemented by JAVA, and FCN is implemented by Python with TensorFlow. To be fair, we use CPU to compute only and use the default setting for all baselines.

4.2. Hyperparameter Search. There are four hyperparameters to be tuned in the SDL-SEC model, $p$, $K$, $c$, and $\lambda$. The classification accuracy has different sensitivities to these

\[
\text{Input: time series data set } T, \text{ parameters } p, K, \lambda, c, \text{ stopping threshold } \varepsilon; \\
\text{Output: complete Shapelet dictionary } S. \\
(1) \text{For } l = 1, 2, \ldots, L \text{ do} \\
(2) \text{Initialize sub-dictionaries } D_l, \text{ sub-coefficient } A_l, \text{ sub-location } q_l \\
(3) \text{Repeat} \\
(4) \text{For } k = 1, 2, \ldots, K \text{ do} \\
(5) \text{update } q_{lk} \text{ with (4)} \\
(6) \text{update } a_{lk} \text{ with (5)} \\
(7) \text{End For} \\
(8) \text{For } k = 1, 2, \ldots, K \text{ do} \\
(9) \text{update } d_{lk} \text{ with (7)} \\
(10) \text{End For} \\
(11) \text{Until convergence} \\
(12) \text{End For} \\
(13) S = [D_1, D_2, \ldots D_L] \\
(14) \text{Return } S
\]

Algorithm 1: Shapelet dictionary learning.

Figure 3: Flowchart of the proposed SVM-based Ensemble Classifier, with the following steps: (1) constructing training subsets and testing subsets; (2) training SVM models on training subsets and inferencing on testing subsets; (3) majority voting.
hyperparameters. Figure 4 reveals the accuracy variation trend of the Gun Point dataset in four situations. In Figure 4(a), the value of $p$ is varying in the range of [0.1, 1] with the step size of 0.01, while the rest hyperparameters are fixed. In the same way, the remaining hyperparameters are fixed when the value of $K$ is varying in the range of [1, 20] with the step size of 1 in Figure 4(b); the value of $c$ is varying in the range of [200, 400] with the step size of 10 in Figure 4(c); the value of $\lambda$ is varying in the range of [0.1, 10] with the step size of 0.1 in Figure 4(d). Obviously, the classification accuracy fluctuates dramatically when the values of $p$ and $K$ are varying. And, the classification accuracy fluctuates straightly when the values $c$ and $\lambda$ are varying. Thus, $p$ and $K$ need to be fine-tuned.

Based on the sensitivity analysis above, the hyperparameters tuning strategy are formulated. The grid search approach is used in hyperparameter tuning. According to experiments, redundant Shapelets can capture more unusual local shape features and improve the classification accuracy. Thus, we search the number of Shapelet $K$ in the range of $[10, 100]$ with the step size of $10$. The Shapelet length $p$ is searched in the range of $[0.05, 0.9] \times m$ with the step size of $0.05 \times m$. The value of $K \times p$ determines the interpretability of Shapelet. When $K \times p > m$, it means Shapelets are redundant and duplicate, such setting may cause higher accuracy but worse interpretability. We fix the coefficient $\lambda$ to 0.01 and the coefficient $c$ to 100.

| No | Dataset          | Classes | Length | Training Instances | Testing Instances |
|----|-----------------|---------|--------|--------------------|-------------------|
| 1  | 50 words        | 50      | 270    | 450                | 455               |
| 2  | Adiac           | 37      | 176    | 390                | 391               |
| 3  | Beef            | 5       | 470    | 30                 | 30                |
| 4  | CBF             | 3       | 128    | 30                 | 900               |
| 5  | Chlorine.       | 3       | 166    | 467                | 3840              |
| 6  | CinC ECG        | 4       | 1639   | 40                 | 1380              |
| 7  | Coffee          | 2       | 286    | 28                 | 28                |
| 8  | Cricket X       | 12      | 300    | 390                | 390               |
| 9  | Cricket Y       | 12      | 300    | 390                | 390               |
| 10 | Cricket Z       | 12      | 300    | 390                | 390               |
| 11 | Diatom          | 4       | 345    | 16                 | 306               |
| 12 | ECG200          | 2       | 96     | 100                | 100               |
| 13 | ECGFive         | 2       | 136    | 23                 | 861               |
| 14 | FaceAll         | 14      | 131    | 560                | 1690              |
| 15 | FaceFour        | 4       | 350    | 24                 | 88                |
| 16 | FacesUCR        | 14      | 131    | 200                | 2050              |
| 17 | Fish            | 7       | 463    | 175                | 175               |
| 18 | Gun point       | 2       | 150    | 50                 | 150               |
| 19 | Haptics         | 5       | 1092   | 155                | 308               |
| 20 | InlineSkate     | 7       | 1882   | 100                | 550               |
| 21 | ItalyPower      | 2       | 24     | 67                 | 1029              |
| 22 | Lighting2       | 2       | 637    | 60                 | 61                |
| 23 | Lighting7       | 7       | 319    | 70                 | 73                |
| 24 | MALLAT          | 8       | 1024   | 55                 | 2345              |
| 25 | MedicalImages   | 10      | 99     | 381                | 760               |
| 26 | MoteStrain      | 2       | 84     | 20                 | 1252              |
| 27 | Non.FatalECG.1  | 42      | 750    | 1800               | 1965              |
| 28 | Non.FatalECG.2  | 42      | 750    | 1800               | 1965              |
| 29 | OliveOil        | 4       | 570    | 30                 | 30                |
| 30 | OSULeaf         | 6       | 427    | 200                | 242               |
| 31 | Sony.I          | 2       | 65     | 27                 | 953               |
| 32 | Sony.II         | 2       | 70     | 20                 | 601               |
| 33 | StarLight       | 3       | 1024   | 1000               | 8236              |
| 34 | SwedishLeaf     | 15      | 128    | 500                | 625               |
| 35 | Symbols         | 6       | 398    | 25                 | 995               |
| 36 | Synthetic       | 6       | 60     | 300                | 300               |
| 37 | Trace           | 4       | 275    | 100                | 100               |
| 38 | Two patterns    | 4       | 128    | 1000               | 4000              |
| 39 | TwoLeadECG      | 2       | 82     | 23                 | 1139              |
| 40 | uWave.X         | 8       | 315    | 896                | 3582              |
| 41 | uWave.Y         | 8       | 315    | 896                | 3582              |
| 42 | uWave.Z         | 8       | 315    | 896                | 3582              |
| 43 | Wafer           | 2       | 152    | 1000               | 6174              |
| 44 | WordsS          | 25      | 270    | 267                | 638               |
| 45 | Yoga            | 2       | 426    | 300                | 3000              |
4.3. Results and Analyses. This section describes and analyses the experimental results from two aspects: classification accuracy and runtime. At the same time, the limitation of proposed algorithm is also discussed.

4.3.1. Classification Accuracy Comparison. We use two statistical indicators to compare the classification accuracies of all algorithms, total wins and average rank. The best algorithm has the most total wins and the highest average rank. The algorithm with better generalization ability can obtain higher accuracy in more data sets, so it is also an important index to evaluate the classification accuracy.

Table 2 summarises the classification accuracy statistical indicators of experiments. It should be noted that the accuracy of SDL-S is slightly different from those presented in [30], due to the updated parameters searching strategies. As shown in Table 2, the proposed algorithm has considerable classification effects. SDL-SEC achieves the best average rank of 1.91 and wins 15 times in 45 datasets. This result is better than that of SDL-S, showing that the SVM-based Ensemble Classifier is effective. FCN obtains the most total wins, 28 times, but the average rank is 1.96, lower than SDL-SEC. LTS also has good performance, but both indicators are worse than SDL-SEC. The accuracy of SD is obviously lower than other algorithms.

Figure 5 is a 1-vs-1 comparison of SDL-SEC algorithm with other algorithms. In Figure 5, the points above the red line represent the higher accuracy of the baseline, and the points below the red line represent the higher accuracy of SDL-SEC algorithm. In comparison with SD and LTS algorithms, the points are more distributed in the area below the red line. In comparison with FCN, the points are roughly evenly distributed on both sides of the red line. It can be seen intuitively that the overall accuracy of SDL-SEC is close to FCN and higher than SD and LTS.

By analyzing Table 2 and Figure 5, we can draw a conclusion that SDL-SEC has a high classification accuracy with good generalization ability, which can adapt to most time series classification problems.

4.3.2. Runtime Comparison. The runtime of each algorithm is analysed below. Table 3 shows the runtimes of the comparison algorithms. The “n/a” in Table 3 indicates that the time or memory required by the algorithm exceeds the limitation of our hardware, and Table 2 uses the accuracy provided by the author. The results of LTS FCN and SDL-SEC are rounded. The runtime of SD algorithm is the shortest. As it was claimed, SD is an efficient time series classification algorithm. Although SDL-SEC runs longer than SD, it is generally acceptable. LTS and FCN run 2-4 orders of magnitude longer than SDL-SEC. Although FCN can be accelerated by the GPU, it is still a high computational expensive algorithm.
To sum up, SDL-SEC has achieved a good balance between accuracy and runtime, greatly reducing the operating time while also having a high accuracy.

4.3.3. Limitation of the Proposed Algorithm. In this part, we discuss the limitations of the proposed algorithm by analyzing two datasets with low classification accuracy. SDL performs worse in datasets Adiac and InlineSkate. The classification accuracies are 0.790 and 0.415, respectively. Figure 6 draws time domain curves of Adiac and InlineSkate. Lines in different colors represent different instances in the dataset. In Figures 6(a) and 6(b), signals from different labels of Adiac dataset show the similar curves, and the shape features are not discriminative. In Figures 6(c) and 6(d), signals from the same label of InlineSkate show disorganized curves, and there is no common shape feature. Shapelet-based algorithms include the proposed algorithm will lose effectiveness in the above two situations: similar curves between different labels or disorganized curves.

| No | Dataset   | SD | LTS | FCN | SDL-S | SDL-SEC |
|----|-----------|----|-----|-----|-------|---------|
| 1  | 50words   | 0.680 | 0.768 | 0.679 | 0.692 | 0.789   |
| 2  | Adiac     | 0.583 | 0.522 | 0.857 | 0.591 | 0.790   |
| 3  | Beef      | 0.507 | 0.800 | 0.750 | 0.833 | 0.867   |
| 4  | CBF       | 0.975 | 0.990 | 1.000 | 0.994 | 0.997   |
| 5  | Chlorine  | 0.553 | 0.590 | 0.843 | 0.602 | 0.616   |
| 6  | CinC ECG  | 0.773 | 0.856 | 0.813 | 0.821 | 0.926   |
| 7  | Coffee    | 0.961 | 1.000 | 1.000 | 1.000 | 1.000   |
| 8  | Cricket X | 0.672 | 0.744 | 0.815 | 0.751 | 0.774   |
| 9  | Cricket Y | 0.675 | 0.718 | 0.792 | 0.741 | 0.769   |
| 10 | Cricket Z | 0.673 | 0.721 | 0.813 | 0.782 | 0.800   |
| 11 | Diatom.   | 0.896 | 0.967 | 0.930 | 0.892 | 0.922   |
| 12 | ECG200    | 0.818 | 0.850 | 0.900 | 0.850 | 0.860   |
| 13 | ECGFive   | 0.953 | 1.000 | 0.985 | 0.999 | 1.000   |
| 14 | FaceAll   | 0.714 | 0.776 | 0.929 | 0.792 | 0.803   |
| 15 | FaceFour  | 0.820 | 0.966 | 0.932 | 0.932 | 0.955   |
| 16 | FacesUCR  | 0.847 | 0.943 | 0.948 | 0.893 | 0.929   |
| 17 | Fish      | 0.755 | 0.966 | 0.971 | 0.977 | 0.977   |
| 18 | Gun point | 0.931 | 1.000 | 1.000 | 0.987 | 1.000   |
| 19 | Haptics   | 0.356 | 0.464 | 0.551 | 0.477 | 0.477   |
| 20 | InlineSkate| 0.385 | 0.364 | 0.411 | 0.356 | 0.415   |
| 21 | ItalyPower| 0.920 | 0.963 | 0.970 | 0.952 | 0.971   |
| 22 | Lighting2 | 0.795 | 0.852 | 0.803 | 0.787 | 0.787   |
| 23 | Lighting7 | 0.652 | 0.808 | 0.863 | 0.726 | 0.795   |
| 24 | MALLAT    | 0.926 | 0.953 | 0.980 | 0.921 | 0.951   |
| 25 | MedicalImages | 0.676 | 0.679 | 0.792 | 0.664 | 0.709   |
| 26 | MoteStrain| 0.783 | 0.858 | 0.950 | 0.905 | 0.905   |
| 27 | Non.FatalECG.1 | 0.814 | 0.869 | 0.961 | 0.895 | 0.906   |
| 28 | Non.FatalECG.2 | 0.855 | 0.911 | 0.955 | 0.912 | 0.938   |
| 29 | OliveOil  | 0.790 | 0.700 | 0.833 | 0.900 | 0.900   |
| 30 | OSULeaf   | 0.566 | 0.769 | 0.988 | 0.934 | 0.955   |
| 31 | Sony.I    | 0.850 | 0.827 | 0.968 | 0.813 | 0.813   |
| 32 | Sony.II   | 0.780 | 0.890 | 0.962 | 0.953 | 0.953   |
| 33 | StarLight | 0.933 | 0.937 | 0.967 | 0.974 | 0.974   |
| 34 | SwedishLeaf| 0.849 | 0.917 | 0.966 | 0.917 | 0.925   |
| 35 | Symbols   | 0.865 | 0.930 | 0.962 | 0.951 | 0.951   |
| 36 | Synthetic.| 0.983 | 0.997 | 0.990 | 0.997 | 0.997   |
| 37 | Trace     | 0.965 | 1.000 | 1.000 | 1.000 | 1.000   |
| 38 | Two patterns | 0.981 | 0.997 | 1.000 | 0.988 | 0.988   |
| 39 | TwoLeadECG| 0.867 | 0.997 | 0.897 | 0.999 | 0.999   |
| 40 | uWave.X   | 0.761 | 0.801 | 0.754 | 0.785 | 0.793   |
| 41 | uWave.Y   | 0.671 | 0.712 | 0.725 | 0.699 | 0.703   |
| 42 | uWave.Z   | 0.676 | 0.752 | 0.729 | 0.731 | 0.741   |
| 43 | Wafer     | 0.993 | 0.996 | 0.997 | 0.997 | 0.997   |
| 44 | WordsS    | 0.625 | 0.680 | 0.580 | 0.594 | 0.672   |
| 45 | Yoga      | 0.625 | 0.831 | 0.845 | 0.839 | 0.839   |

Average rank: 4.64, 2.84, 1.96, 2.87, 1.91
Total wins: 0, 11, 28, 8, 15
Figure 5: 1-vs-1 Comparison of SDL-SEC with Baselines: (a) SDL-SEC VS SD; (b) SDL-SEC VS LTS; (c) SDL-SEC VS FCN.

Table 3: Runtime for 45 datasets.

| No | Dataset     | SD  | LTS | FCN  | SDL-SEC |
|----|-------------|-----|-----|------|---------|
| 1  | 50 words    | 0.494| n/a | 9307 | 384     |
| 2  | Adiac       | 0.337| 68163| 5361 | 24      |
| 3  | Beef        | 0.047| 1187 | 1300 | 11      |
| 4  | CBF         | 0.131| 28  | 3114 | 15      |
| 5  | Chlorine    | 0.650| 1009 | 13934| 147     |
| 6  | CinC ECG    | 2.111| 14816| 31540| 87      |
| 7  | Coffee      | 0.044| 46  | 1026 | 1       |
| 8  | Cricket X   | 0.763| 53780| 8889 | 36      |
### Table 3: Continued.

| No | Dataset          | SD  | LTS     | FCN  | SDL-SEC |
|----|------------------|-----|---------|------|---------|
| 9  | Cricket Y        | 0.652 | 53663   | 8685 | 168     |
| 10 | Cricket Z        | 0.800 | 53699   | 8559 | 179     |
| 11 | Diatom           | 0.109 | 236     | 3446 | 3       |
| 12 | ECG200           | 0.055 | 41      | 1099 | 2       |
| 13 | ECGFive          | 0.133 | 9       | 3752 | 2       |
| 14 | FaceAll          | 1.557 | 23985   | 8173 | 113     |
| 15 | FaceFour         | 0.150 | 372     | 1569 | 2       |
| 16 | FacesUCR         | 0.677 | 8184    | 5542 | 52      |
| 17 | Fish             | 0.228 | 16309   | 5537 | 42      |
| 18 | Gun point        | 0.074 | 25      | 1250 | 1       |
| 19 | Haptics          | 2.314 | 37594   | 13097 | 229   |
| 20 | InlineSkate      | 1.451 | 141802  | 22836 | 4533 |
| 21 | ItalyPower       | 0.067 | 2       | 1598 | 2       |
| 22 | Lighting2        | 1.939 | 873     | 2793 | 109     |
| 23 | Lighting7        | 0.463 | 2920    | 2176 | 50      |
| 24 | MALLAT           | 1.743 | 28683   | 38842 | 5313 |
| 25 | MedicalImages    | 0.690 | 4060    | 4049 | 17      |
| 26 | MoteStrain       | 0.153 | 6       | 4233 | 1       |
| 27 | Non.FatalECG.1   | 8.721 | n/a     | 90922 | 2102 |
| 28 | Non.FatalECG.2   | 6.446 | n/a     | 90826 | 1276 |
| 29 | OliveOil         | 0.068 | 1043    | 1513 | 4       |
| 30 | OSULeaf          | 0.244 | 12900   | 6537 | 35      |
| 31 | Sony.I           | 0.099 | 3       | 2262 | 0.3     |
| 32 | Sony.II          | 0.105 | 3       | 3319 | 5       |
| 33 | StarLight        | 12.561 | 91024  | 160381 | 1641 |
| 34 | SwedishLeaf      | 0.456 | 19651   | 6183 | 71      |
| 35 | Symbols          | 0.356 | 1099    | 8766 | 10      |
| 36 | Synthetic        | 0.099 | 550     | 1992 | 5       |
| 37 | Trace            | 0.157 | 957     | 2375 | 3       |
| 38 | Two patterns     | 2.555 | 2829    | 16142 | 33    |
| 39 | TwoLeadECG       | 0.130 | 4       | 3870 | 2       |
| 40 | uWave.X          | 6.322 | 58636   | 37696 | 126   |
| 41 | uWave.Y          | 5.114 | 58948   | 33952 | 1280  |
| 42 | uWave.Z          | 3.154 | 58633   | 32280 | 955   |
| 43 | Wafer            | 2.236 | 948     | 23529 | 434   |
| 44 | WordsS           | 0.480 | 123991  | 7153 | 112     |
| 45 | Yoga             | 1.333 | 1985    | 23449 | 134   |

![Figure 6: Continued.](image-url)
5. Conclusion and Future Work

In this paper, we present a Shapelet-based time series classification algorithm called Shapelet Dictionary Learning with SVM-based Ensemble Classifier. First, we propose a Shapelet discovery method Shapelet Dictionary Learning, which combines Dictionary Learning and Shapelet and generates a group of Shapelets instead of searching. The generated Shapelets are totally new subsequences which contain local shape features of all the time series data but not exist in original data. This encourages the generalization ability of Shapelet, reaches a higher accuracy in time series classification, and reduces runtime simultaneously. Furthermore, we propose an SVM-based Ensemble Classifier to execute the classification operation. The proposed SVM-based Ensemble Classifier trains a series of base SVM classifiers which are fed random features and parameters. This method decreases the dependence on feature and parameter selection and thus further improves the classification accuracy. Extensive experiments are performed on 45 benchmark datasets. The results show that the proposed algorithm has high accuracy, good robustness, and high efficiency.

We also look into the future work from the limitations of the proposed algorithm. First, SDL-SEC discovers the time domain shape features. It will fail when the shape feature is not obvious. In order to address this issue, we may exploit a feature fusion method by combining frequency domain features and other statistical features. Second, the current SDL hyperparameter selection method is a grid search approach. This approach requires to design searching strategy manually. As an improvement direction, intelligent optimization algorithms can be exploited, such as evolutionary algorithms, to search hyperparameters automatically [31]. And, the third, only Euclidean Distance is considered in this work, so it is difficult to deal with real-world data with noise and uncertainty. Fuzzy Measurement [32, 33] is an excellent tool to handle this kind of problems, which we will investigate in our future work.

Data Availability

The dataset used in this paper can be found in http://www.timeseriesclassification.com.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported in part by the National Key R&D Program of China under Grant no. 2018AAA0101700 and the Natural Science Foundation of China (NSFC) under grant 51721092.

References

[1] P. Esling and C. Agon, “Time-series data mining,” ACM Computing Surveys, vol. 45, no. 1, p. 12, 2012.
[2] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. Keogh, “Querying and mining of time series data,” Proceedings of the VLDB Endowment, vol. 1, no. 2, pp. 1542–1552, 2008.
[3] S. Guo and W. Guo, “Process monitoring and fault prediction in multivariate time series using bag-of-words,” IEEE Transactions on Automation Science and Engineering, pp. 1–13, 2020.
[4] M. Versaci, S. Calcagno, M. Cacciola, F. C. Morabito, I. Palmara, and D. Pellicano, “Innovative fuzzy techniques for characterizing defects in ultrasonic nondestructive evaluation,” in Ultrasonic Nondestructive Evaluation Systems, pp. 201–232, Springer, Berlin, Germany, 2015.
[5] K. Deng, A. W. Moore, and M. C. Nechyba, “Learning to recognize time series: combining ARMA models with memory-based learning,” in Proceedings of the 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA’97), pp. 246–251, Monterey, CA, USA, June 1997.
[6] S. Zhong and J. Ghosh, “HMMs and Coupled HMMs for multi-channel EEG classification,” in Proceedings of the 2002
International Joint Conference on Neural Networks (IJCNN’02), vol. 2, pp. 1154–1159, Piscataway, NJ, USA, May 2002.

[7] R. J. Povinelli, M. T. Johnson, A. C. Lindgren, and J. Jinjin Ye, “Time series classification using Gaussian mixture models of reconstructed phase spaces,” IEEE Transactions on Knowledge and Data Engineering, vol. 16, no. 6, pp. 779–783, 2004.

[8] A. Gautam and V. Singh, “CLR-based deep convolutional spiking neural network with validation based stopping for time series classification,” Applied Intelligence, vol. 50, no. 3, pp. 830–848, 2020.

[9] L.-P. Jin and J. Dong, “Ensemble deep learning for biomedical time series classification,” Computational Intelligence and Neuroscience, vol. 2016, Article ID 6212684, 13 pages, 2016.

[10] L. Ye and E. Keogh, “Time series shapelet: a new primitive for data mining,” in Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD ’09), pp. 947–955, Paris, France, June 2009.

[11] J. Grabocka, M. Wistuba, and L. Schmidt-Thieme, “Fast classification of univariate and multivariate time series through shapelet discovery,” Knowledge and Information Systems, vol. 49, no. 2, pp. 429–454, 2016.

[12] T. Rakthanmanon and E. Keogh, “Fast Shapelet: a scalable algorithm for discovering time series Shapelet,” in Proceedings of the 13th SIAM international conference on Data Mining, pp. 668–676, Austin, Texas, USA, May 2013.

[13] K. Chang, B. Deka, W. W. Hwu, and D. Roth, “Efficient pattern-based time series classification on GPU,” in Proceedings of the 2012 IEEE 12th International Conference on Data Mining, pp. 131–140, Brussels, Belgium, December 2012.

[14] J. Hills, J. Lines, E. Baranauskas, J. Mapp, and A. Bagnall, “Classification of time series by shapelet transformation,” Data Mining and Knowledge Discovery, vol. 28, no. 4, pp. 851–881, 2014.

[15] M. Li, “Example-based learning using heuristic orthogonal matching pursuit teaching mechanism with auxiliary coefficient representation for the problem of de-fencing and its affiliated applications,” Applied Intelligence, vol. 48, no. 9, pp. 2884–2893, 2018.

[16] R. Tibshirani, “Regression shrinkage and selection via the lasso: a retrospective,” Journal of the Royal Statistical Society: Series B (Statistical Methodology), vol. 73, no. 3, pp. 273–282, 2011.

[17] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, “Distributed optimization and statistical learning via the alternating direction method of Multipliers,” Found Trends Mach Learn, vol. 3, no. 1, pp. 1–122, 2011.

[18] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, “Online dictionary learning for sparse coding,” in Proceedings of the 26th Annual International Conference on Machine Learning, pp. 689–696, Association for Computing Machinery, Montreal, Canada, June 2009.

[19] A. Beck and M. Teboulle, “A Fast iterative shrinkage-thresholding algorithm for linear inverse problems,” SIAM Journal on Imaging Sciences, vol. 2, no. 1, pp. 183–202, 2009.

[20] K. Huang, Y. Wu, C. Yang, G. Peng, and W. Shen, “Structure dictionary learning-based multimode monitoring and its application to aluminum electrolysis process,” IEEE Transactions on Automation Science and Engineering, vol. 17, no. 4, pp. 1989–2003, 2020.

[21] G. Zheng, Y. Yang, and J. Carbonell, “Efficient shift-invariant dictionary learning,” in Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD’16), pp. 2095–2104, San Francisco, CA, USA, August 2016.

[22] C.-C. Chang and C.-J. Lin, “LIBSVM: A library for support vector machines,” ACM Transactions on Intelligent Systems and Technology, vol. 2, no. 3, pp. 1–27, 2011.

[23] R. Gupta, M. A. Alam, and P. Agarwal, “Modified support vector machine for detecting stress level using EEG signals,” Computational Intelligence and Neuroscience, vol. 2020, Article ID 8660841, 14 pages, 2020.

[24] A. Onan, “Two-stage topic extraction model for bibliometric data analysis based on word embeddings and clustering,” IEEE Access, vol. 7, pp. 145614–145633, 2019.

[25] A. Onan, S. Korukoglu, and H. Bulut, “Ensemble of keyword extraction methods and classifiers in text classification,” Expert Systems with Applications, vol. 57, pp. 232–247, 2016.

[26] J. Cao, G. Lv, C. Chang, and H. Li, “A feature selection based serial SVM ensemble classifier,” IEEE Access, vol. 7, pp. 144516–144523, 2019.

[27] “The UCR time series data mining archive,” 2015, http://www.cs.ucr.edu/eamonn/etimeseriesdata/.

[28] J. Grabocka, N. Schilling, M. Wistuba, and L. Schmidt-Thieme, “Learning time-series shapelet,” in Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD’14), pp. 392–401, New York, NY, USA, August 2014.

[29] Z. Wang, W. Yan, and T. Oates, “Time series classification from scratch with deep neural networks: a strong baseline,” in Proceedings of the 2017 International Joint Conference on Neural Networks (IJCNN’17), pp. 1578–1585, Anchorage, AL, USA, May 2.

[30] J. Zhang, X. Li, L. Gao, L. Wen, and G. Liu, “A shapelet dictionary learning algorithm for time series classification,” in Proceedings of the 2019 IEEE 15th International Conference on Automation Science and Engineering (CASE’20), pp. 299–304, Vancouver, BC, Canada, August 2019.

[31] A. Onan and S. Korukoglu, “A feature selection model based on genetic rank aggregation for text sentiment classification,” Journal of Information Science, vol. 43, no. 1, pp. 25–38, 2017.

[32] M. Versaci and F. C. Morabito, “Image edge detection: a new approach based on fuzzy entropy and fuzzy divergence,” International Journal of Fuzzy Systems, pp. 1–19, 2021.

[33] M. N. Postorino and M. Versaci, “A geometric fuzzy-based approach for airport clustering,” Advances in Fuzzy Systems, vol. 2014, Article ID 201243, 12 pages, 2014.