RSAST: Sampling Shapelets for Time Series Classification

Nicolas Rojas Varela, Michael Franklin Mbouopda, Engelbert Mephu Nguifo

RSAST: Sampling Shapelets for Time Series Classification

Nicolas Rojas Varela
LIMOS
University Clermont Auvergne
Clermont–Ferrand, France
nirojasva@unal.edu.co

Michael Franklin Mbouopda
Huawei Technologies
Ireland Reasearch Centre
Dublin, Ireland
mf.mbouopda@gmail.com

Engelbert Mephu Nguifo
LIMOS
University Clermont Auvergne
Clermont–Ferrand, France
engelbert.mephu_nguifo@uca.fr

***Abstract*—Shapelet-based techniques are widely utilized in time series classification due to their combination of interpretability and accuracy. However, these methods tend to be less scalable than other state-of-the-art approaches. Because of this, we propose RSAST as a shapeled-based technique that utilizes a stratified technique and certain statistical criteria to randomly select the shapelets. We conducted experiments on 128 datasets from the UCR archive, showcasing the comparative accuracy of RSAST against its baseline methods SAST and STC, as well as other state-of-the-art techniques. Notably, the method we propose, RSAST, preserve the accuracy and interpretability of the baseline methods while reducing the computation time. For example, with Earthquakes, one of the largest datasets from the UCR archive, RSAST took 6 minutes and 36 seconds to complete the training process, whereas SAST took 2 hours and 37 minutes.**

***Index Terms*—Time series Classification, Shapelet, Scalability, Interpretability, ANOVA, ACF, PACF**

I. Introduction

Time series data is a type of data that is measured over time. This type of data has applications in various tasks such as time series anomaly detection, time series forecasting and time series classification. In particular, this work is focus on the time series classification task which contemplated various methodologies such as STC, SAST, HIVE-COTE, ROCKET, RDST among others. Notably, the Scalable and Accurate Subsequence Transform (SAST) [1] stands out as one of the most precise and interpretable techniques. So, our work aims to study SAST approach and propose an improvement of the method.

In the state-of-the-art there are many algorithms for time series classification which uses different features for the classification. Depending of the features the methods can be categorized into eight groups [2]:

- Distance-based methods: In this category, entire time series are compared using distance measures in combination with nearest neighbor (NN) classifiers. Examples of such approaches include the Elastic Ensemble (EE) [3], which involves an ensemble of 11 1-NN classifiers weighted with a variety of elastic distance measures. Another method is Proximity Forest (PF) [4], which utilizes the same 11 distance functions as EE but with improved accuracy and scalability..
- Feature-based methods: These methods involve extracting features from the time series and using them in a classifier. Various feature extraction techniques can be employed, including statistical measures, frequency domain analysis, and time-domain characteristics. For example, Catch22 [5], one of these methods, utilizes 22 hctsa features identified as the most discriminatory in a set. These features encompass a broad spectrum, including basic statistics of time series values, linear correlations, and entropy. Similarly, TSFresh [6] was introduced as a collection of approximately 800 features extracted from time series.
- Interval-based methods: In these approaches, it is selected one or more intervals within a time series rather than using the entire sequence. Multiple intervals can be chosen, and summary measures from these intervals are then utilized as features for classification. One example of this approach is TSF [7], which extracts three summary statistics (mean, variance, and slope) from randomly selected intervals. Similarly, RISE [8] generates random intervals but employs spectral features instead.
- Shapelet-based methods: Algorithms in this category aim to identify short patterns known as shapelets, which define a class. These shapelets can appear anywhere in the series and are not dependent on the phase. A class is distinguished by the presence or absence of one or more shapelets within the entire series. The baseline from this methods in the literature it is STC [9] since is a pipeline which searches the training data for shapelets, transforms the time series and builds a classifier with the tranformed data. Later RDST [10] uses randomly selecting shapelets from the train data to transform the dataset and train a classifier introducing the concept of dilated shapelets.
- the frequency of subseries repetition is more crucial than their mere presence or absence. These methods involve creating frequency counts of recurring patterns and developing classifiers based on the resulting histograms. For instance, BOSS [11] utilizes sliding windows that are transformed into words using SFA. This process generates a feature vector by counting the occurrences of each word across all windows.
- Convolution-based methods: These methods consist in

the application of convolution and pooling operations to time series, transforming the original space into a new dataset suitable for classification. The most representative method in this category is the Random Convolutional Kernel Transform (ROCKET) [12]. ROCKET generates a large number of randomly parameterized convolutional kernels, typically ranging from thousands to tens of thousands, and uses them to transform the data. Also, there are several variants of ROCKET, such as MiniROCKET [13], which significantly accelerates ROCKET with no substantial difference in accuracy.

- Deep learning-based methods: This category utilizes neural network structures for time series classification. Examples of methods in this category include the Residual Network (ResNet) [14] and InceptionTime [15], each employing distinct architectures for classification.
- Hybrid approaches: These approaches involve combining algorithms from two or more of the previously mentioned approaches into a single classifier. These methods aim to leverage the strengths of different techniques for improved performance. For example, HIVE-COTE [16] is a heterogeneous ensemble containing five modules, each from a different representation. Another example is TS-CHIEF, which is a homogeneous ensemble where hybrid features are embedded in tree nodes.

Despite the high accuracy of state-of-the-art time series classification algorithms, they often suffer from computationally intensive calculations or lack interpretability of the results. Because of this, our work will focus on shapelet approaches, which offer interpretability and accuracy.

The Scalable and Accurate Subsequence Transform (SAST) [1] method was proposed as a method based on shapelets and builded upon STC [9]. SAST reduces the search space for shapelets by selecting only a few instances per class from the dataset, resulting in a complexity of $O(nm^3) + O(\text{classifier})$.

So, in order to increase the scalability of SAST, we propose a novel approach called Random SAST (RSAST). RSAST avoids the use of a searching algorithm by employing a stratified sampling technique to select candidate shapelets. By doing so, RSAST significantly reduces the computational burden while maintaining competitive accuracy and interpretability.

Mainly, RSAST utilizes fundamental statistical concepts such as ANOVA, Autocorrelation Function (ACF), and Partial Autocorrelation Function (PACF). So, the ANOVA test enables a more accurate selection of starting points for the shapelets, while ACF and PACF assist in identifying the most relevant lengths for these subsequences.

The Analysis of Variance (ANOVA) assesses variability within and between classes using a statistical test known as the F-test, allowing for a comparison of mean similarities. Depending on the number of classes being analyzed, various ANOVA variations can be employed [17]. Notably, RSAST predominantly utilizes one-factor ANOVA, as many time series classification tasks typically involve considering just one factor or class for classification. Also for RSAST it is assumed the independence, normality, and homogeneity of variances of the values.

Although there are alternative methods for ANOVA to compare the similarity of groups, such as the Kruskal-Wallis test, Mood's median test, or even information gain, [9] demonstrated that there are no significant differences in accuracy when choosing between these alternatives. Consequently, RSAST only considers ANOVA to compare values across time series.

Thus, the contributions of our work can be summarized as follows:

- Utilizing a stratified sampling strategy for subsequences selection, as in SAST, but taking into account certain statistical criteria such as ANOVA, ACF and PACF.
- Providing an open-source implementation of RSAST, based on the SAST model. This implementation significantly reduces training time compared to SAST. For example, when applied to the Earthquakes dataset, which is one of the largest datasets from the UCR archive, SAST took 2 hours and 37 minutes for model training with an accuracy of 0.68, whereas RSAST completed the training process in only 6 minutes and 36 seconds with 0.72 of accuracy.

The subsequent sections of our work will delve into the background and related work to the topic (Section II). Also, we will provide a detailed explanation of the method (Section III), and showcase the experimental results obtained using RSAST (Section IV). By presenting these sections, we aim to demonstrate the effectiveness and advantages of RSAST in time series classification tasks.

II. Background and related works

A. Definitions

In this section, we provide an overview of the key concepts used to describe RSAST methodology.

1) **Time Series Classification:** It is a task in machine learning that involves predicting the class or category of a time series dataset.
2) **Time Series:** A time series (TS) is a sequence of data points collected over time, where each data point is associated with a specific timestamp. TS can be used within a instance pair $\{\mathbf{x}, y\}$, where $\mathbf{x}$ is the TS with m observations $(x_1, x_2, ..., x_m)$, and y is a discrete class variable with $|c|$ possible values. The list of n cases with associated class labels can be represented as $\mathbf{T} = (\mathbf{X}, \mathbf{y}) = ((\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_n, y_n))$.
3) **Classifier:** refers to a function or mapping that operates on the input space and assigns a probability distribution over the possible class variable values.
4) **Shapelets:** shapelets are subsequences, denoted as $S = \{s_1, ..., s_l\}$ with length l, that can effectively differentiate between different classes or categories in a time series dataset [18].
5) **Autocorrelation Function (ACF):** is a tool used to measure the similarity between a time series and a delayed version of itself [19].

6) **Partial Autocorrelation Function (PACF):** similar to ACF, is a measure of the correlation between a time series and a lagged version of itself, after removing the effects of all the shorter lags [20].
7) **Analysis of variance (ANOVA):** is a statistical method used to analyze the difference between the means of two or more groups. It is used to determine if there is a significant difference between the means of two or more groups based on the variation in the data [17].

B. Shapelet Approaches

As previously mentioned, time series classification (TSC) methods can be categorized into different categories, each with methods varying in their levels of accuracy, complexity, and interpretability [2].

Shapelets methods focus on finding subseries that allows distinguishing the different classes of time series. The process involves sliding these subseries across the time series and calculating the distance between the subseries and the segments of the time series.

The initial shapelet approach in the literature emerged with the work by [18]. This procedure introduced embedded shapelets within a decision tree classifier, showcasing the potential of employing shape-based features to enhance classification performance for time series data.

Following this, subsequent studies have been dedicated to two primary aspects: refining the accuracy of the original shapelet algorithm and addressing the inherent computational complexity associated with shapelet methods.

One improvement of accuracy is proposed with Shapelet Tranform [9], STC takes a multi-step approach to enhance accuracy in time series classification. Firstly, it conducts an exhaustive search for shapelets within the training data. Then, it transforms the time series into distance-based vectors, enabling the selection of the most distinguishable shapelets among classes using the information gain criterion. Then, with the selected shapelets, the algorithm constructs a classifier using the transformed series data.

Recently, some changes were introduced for STC [21] like a randomised search of shapelets with a search time as parameter, a binary shapelets evaluation against all other classes, and the use of Rotation Forest instead of HESCA classifier [2].

The Scalable and Accurate Subsequence Transform (SAST) aims to improve the complexity of STC by reducing the number of patterns to be assessed [1], and as a consequence, making the model faster to train. To accomplish this, SAST reduces the number of shapelet candidates by selecting randomly k instances per class. Furthermore, it does not select the top best shapelets beforehand, as in STC, instead all shapelets are used to transform the data.

Moreover, in Fast Shapelets (FS) [22], it was introduced an enhanced version of the decision tree shapelet approach proposed in the original algorithm [18]. This approach focuses on accelerating shapelet discovery. Instead of performing an exhaustive search at every node, FS utilizes symbolic aggregate approximation (SAX) to discretize and approximate shapelets. SAX is a technique that transforms time series into strings, reducing their dimensionality using piecewise aggregate approximation (PAA) and discretizing them into bins based on equal probability areas of the normal distribution.

Similarly in Learned Shapelets (LS) [23], it is introduced the use of a heuristic gradient descent shapelet search approach instead of the traditional enumeration strategy. Unlike FS and STC, LS is not constrained to identifying shapelets solely from subseries within the training data.

Another shapelet approach is the Generalised Random Shapelet Forest or RSF [24]. This method consists of a bagging-based tree ensemble that aims to improve computational efficiency and predictive accuracy. At each node of a different tree, r univariate shapelets are selected from the training data. Each shapelet has a random length within some predefined upper and lower bounds. Multiple trees are ensembled, so new instances are predicted by a majority vote on the trees' predictions.

Also, the Multiple Representation Sequence Learner (MrSEQL) [25], is another ensemble classifier which is based on SEQL classifier [26]. MrSEQL looks for the existence or non-existence of a shapelet within the data. However, instead of employing a distance-based technique to quantify this presence or absence, MrSEQL converts the subseries into symbolic words. Subsequently, these words are generated via two symbolic representations—SAX for the time domain and SFA for the frequency domain. A collection of distinctive words is chosen using Sequence Learner (SEQL). The training procedure culminates in a logistic regression model, yielding a set of important subsequences determined by the model's weights.

Lastly, a recent advancement in phase-independent approaches has been introduced with the Random Dilated Shapelet Transform or RDST [10]. This algorithm integrates various convolution techniques. RDST employs dilation with shapelets, which means that some spaces are defined between time points. Hence, a shapelet with dilation d is compared to time points d steps apart when computing distances between shapelet and time series. The method also uses two features in addition to distance, it uses the position of the minimum distance, and it measures the frequency of occurrences of the shapelet, based on a distance threshold. Consequently, the transformed data comprises 3k features (for k shapelets) subsequently employed in a classifier.

Although highly accurate RDST has deficiency in interpretability since the shapelets found could exhibits dilation, which is not natural for domain expert. An example of the lack of interpretability can be seen training the model with Coffee dataset from the UCR archive. Here, the first shapelet, according to a feature importance analysis, has a dilation of 13, the second most important shapelet has a dilation of 12 and the third shapelet has a dilation of 15 (see Fig. 1). As a consequence, when these shapelets are plotted on random instances from each class of the training set (see Fig. 2), it

is observed that they do not seem to match perfectly with the time series, as it is the case of other methods like STC, SAST and RSAST. In consequence, it is not clear the distinct pattern which enables the differentiation of classes of the RDST approach.

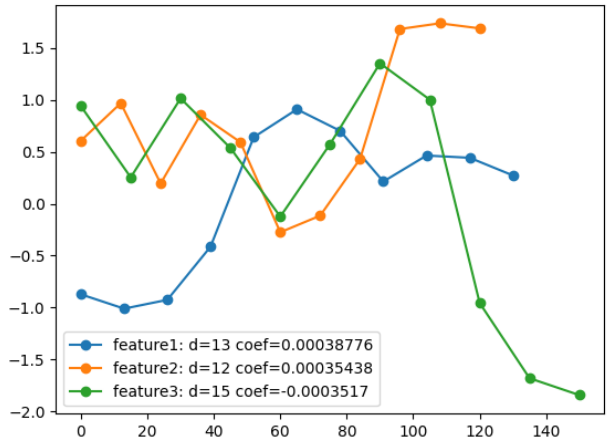


Fig. 1: Shapelets for RDST using Coffee dataset.

Besides, in RDST, the features are categorized into 3 types: the presence or absence of the shapelet (min), the position (arg min) and the frequency (SO). So, for the interpretation of the method, it is necessary to take into account the feature type as part of the analysis of the results. For instance, for Coffee dataset (see Fig. 2), the 3 most important shapelet are argmin type, indicating that their importance is determined by the location of the shapelets. For more details of RDST method it is suggested to read [10].

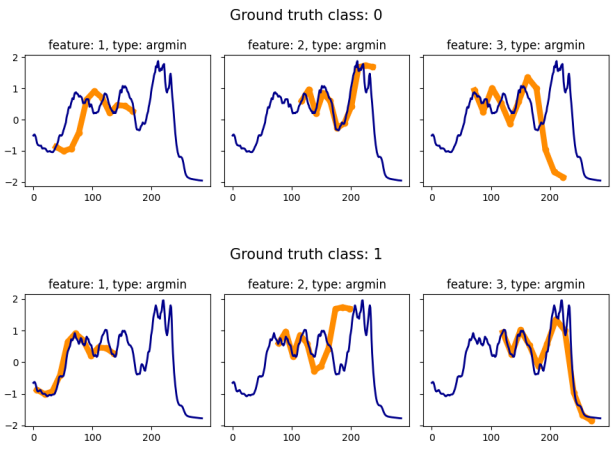


Fig. 2: Top 3 shapelets, on TS, learned by RDST on Coffee dataset.

C. SOTA Approaches

There are some state-of-the-art methods, that focus on combining different types of features in order to achieve highest accuracy.

One of these main techniques that achieve the highest accuracy is a convolutional approach, called ROCKET, which stands for Random Convolutional Kernel Transform [12]. This method generates an extensive assortment of convolutional kernels with randomized parameters. These kernels are then utilized to transform the data through convolution and pooling operations like max pooling (identifying the maximum value) and the proportion of positive values (PPV). The results gives two features which are concatenated for all kernels. Thus, for k kernels, the transformed data has $2k$ features. These feature vectors are harnessed to train a Ridge Classifier, incorporating cross-validation to determine the L2-regularization parameter α. In scenarios involving larger datasets, a logistic regression classifier is proposed as an alternative.

Another accurate method is the Hierarchical Vote Collective of Transformation Ensembles (HIVE-COTE) [16]. This ensemble consists of five distinct modules, each capturing different features. These modules encompass the EE from distance-based representations, TSF from interval-based methods, BOSS from dictionary-based approaches, STC from shapelet-based methods, and a spectral method called RISE. Cross-validation Accuracy Weighted Probabilistic Ensemble (CAWPE) [27] is used to ensemble the results of the five modules.

HIVE-COTE demonstrated enhanced scalability in the HIVE-COTE v1.0 (HC1) [28], incorporating several advancements. HC1 introduced four modules, with cBOSS [29] replacing BOSS. The updated randomised version of STC [21] with a Rotation Forest classifier and a default one-hour shapelet search. Usability improvements were made to TSF and RISE. Notably, this version omitted the computationally intensive EE algorithm, resulting in improved efficiency without sacrificing accuracy.

Similarly HC1 was improved through HIVE-COTE v2.0 (HC2) [30]. In HC2, only STC is retained from the prior version, while the remaining modules undergo transformation. The substitution of modules is prominent. TDE supplants cBOSS, serving as the chosen dictionary classifier. Instead of TSF and RISE, the DrCIF method is introduced to handle the interval and frequency representations. Additionally, HC2 introduces Arsenal method, an ensemble of ROCKET classifiers.

Another notably accurate hybrid approach is The Time Series Combination of Heterogeneous and Integrated Embedding Forest (TS-CHIEF) [31]. This is a homogeneous ensemble of methods which means that hybrid features are embedded in tree nodes rather than separated in modules. In TS-CHIEF, each tree node evaluates a certain number of splitting criteria. Notably, for dictionary-based splits, the criteria are rooted in the BOSS method, while distance-based splits rely on EE, and interval-based splits draw from RISE. TS-CHIEF is distinct in its design to avoid the resource-intensive processing associated with HIVE-COTE.

III. RSAST Method

Although SAST demonstrated to be more efficient than STC there exist still a high complexity in the method attributed

to its search space for subsequences (see Sect. III-B). Thus, we propose an improvement on SAST called Random SAST (RSAST). Through RSAST we propose randomly selecting the shapelets, taking into account certain statistical criteria. This is done by randomly selecting the starting point (t_0) of the subsequence from (k) randomly chosen time series (T_i) for each class. Moreover, the length of the shapelet is defined using the autocorrelation function (ACF) and partial autocorrelation function (PACF) of the time series (T_i) instead of using a pre-defined length list (l).

A. Main Ideas RSAST

The method starts with the precomputation of a list of weights which facilitates the selection of initial points for subsequences. This makes that when a starting point (t_0) is randomly chosen from a time series the selection prioritize the more relevant points according to this list of weights.

To compute the aforementioned list of weights, a one-way ANOVA test is executed for each time value, denoted as $T_i[j]$, with j varying from 0 to m, and m the length of the time series (refer to Algorithm 1). So the test detects distinctive starting points that exhibit statistically significant differences between classes, thereby prevents the selection of points that are inherently similar.

Algorithm 1 CalculateWeights

Require: Dataset: **D**
$classes \leftarrow uniqClasses(D)$
$weights \leftarrow []$
$n, m \leftarrow size(D)$
for $i \leftarrow 1$ to m **do**
  $statsPerClass \leftarrow \emptyset$
  for c in $classes$ **do**
    $statsPerClass \leftarrow statsPerClass \cup \{(T_c[i], c)\}$
  end for
  $pValue \leftarrow ANOVA(statsPerClass)$
  $weights[i] \leftarrow 1 - pValue$
end for
return weights

Then, analogous to the initial SAST algorithm, k instances per class are drawn (without replacement) from the training dataset. So, the autocorrelation function (ACF) and the Partial Autocorrelation Function (PACF) are applied to each of these selected instances.

The computation of PACF and ACF for each instance yields to a set of highly correlated lagged time points. These points are subsequently employed as potential lengths for the shapelets, utilizing the significant values obtained from each test. This approach leads to the exclusion of non-correlated points within the time series, thereby eliminating the need to exhaustively explore the complete range of potential lengths within a time series (ranging from 3 to m).

Finally, with the shapelet length determined, and utilizing the precalculated list of weights, a certain number of admissible starting points are sampled (without replacement) from each instance.

Thus, with these instances of reference, the possible lengths for the subsequences, and some chosen starting points, it is obtained a collection of relevant shapelet candidates (see Algorithm 2).

Additionally, given the selection of a specific amount of starting points (p), there is a upper limit, or maximum of shapelets to obtain, since the total number depends on the difference between the length of the time series and the length of the possible shapelets. Thus, this difference must be less than the required number of starting points ($p > m - l + 1$) in order to select all the required p values. Otherwise, if the difference is greater ($p < m - l + 1$), the set of possible starting points, and therefore the total number of shapelets, will be truncated.

Algorithm 2 generateRandomShapeletCandidates

Require: Dataset: **D**, Length list: L, Random Points: p, Weights: W
$S \leftarrow \emptyset$
$n, m \leftarrow size(D)$
for $i \leftarrow 1$ to n **do**
  $rPoints \leftarrow []$
  for l in L **do**
    $rPoints \leftarrow randomlySelectPoints(T_i, W, p)$
    for j in $rPoints$ **do**
      $S \leftarrow S \cup selectSubsequence(T_i, j, l)$
    end for
  end for
end for
return S

Once the subsequences have been generated, the followed stages of the algorithm involves computing distances between the shapelets and the time series. Thus, after the transformation is completed, a classifier is trained using the newly transformed dataset.

After the training of the model a post hoc method can be applied to interpret the results [32]. This allows to obtain the most important features and therefore the most important shapelets of the model. The analysis of importance identify how much a feature is correlated to the target variable, so, in a linear model, the absolute value of the weight will distinguish the importance of a feature with respect to another less important [33].

For a general understanding of the method, refer to Algorithm 3, which presents a comprehensive overview of the RSAST procedure. Additionally, Figure 3 provides an illustrative summary of the RSAST method.

B. Search Space of Shapelet Approaches

The time taken by shapelet approaches, which make the extraction of subsequences from the training dataset, is primarily due to the exploration and evaluation of these subsequences.

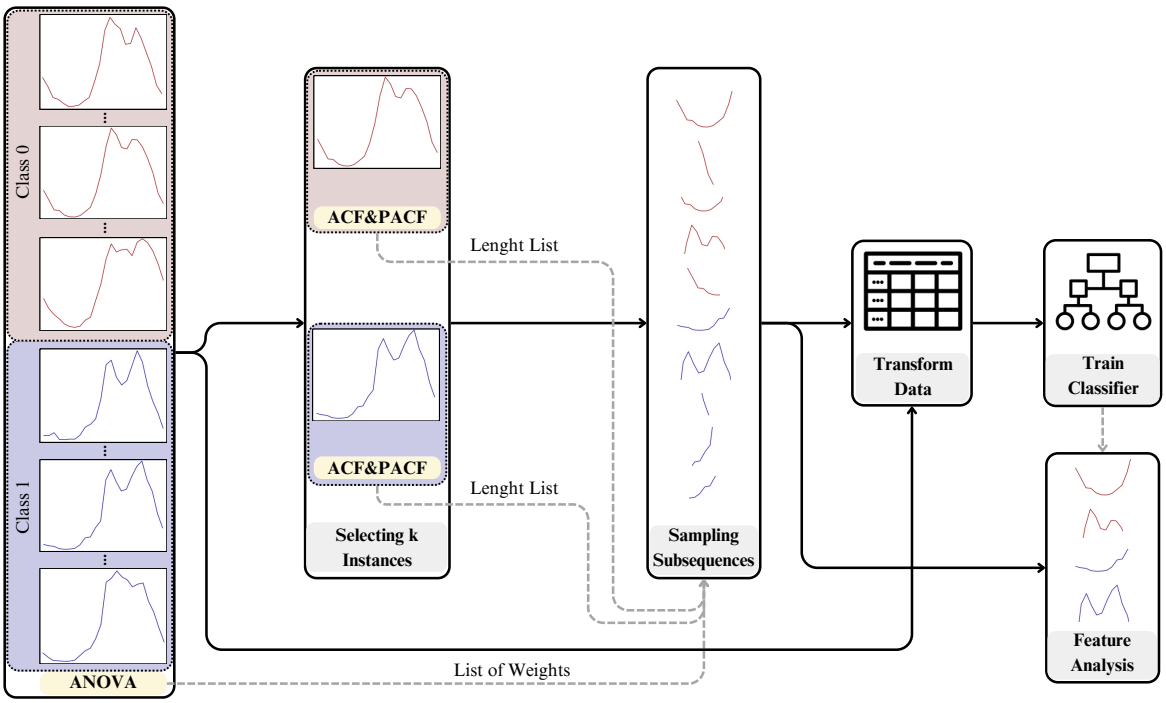


Fig. 3: Overview of RSAST method.

Algorithm 3 Random ScalableAndAccurateSubsequenceTransform

Require: Dataset: **D**, Instances per Class: k, classifier to use: C, Number of Starting Point: p

$n, m \leftarrow size(D)$
$W \leftarrow CalculateWeights(D)$
$D_c \leftarrow randomlySelectInstancesPerClass(D, k)$
$L \leftarrow generateLenghtListACF(D_c)$
$L \leftarrow L \cup generateLenghtListPACF(D_c)$
$S \leftarrow generateRandomShapeletCandidates(D_c, L, p, W)$
$D_f \leftarrow \emptyset$
for $i \leftarrow 1$ to n **do**
  $x_i \leftarrow []$
  for $j \leftarrow 1$ to $|S|$ **do**
    $x_i[j] \leftarrow dist(T_i, S_j)$
  end for
  $D_f \leftarrow D_f \cup \{(x_i, c_i)\}$
end for
$clf \leftarrow trainClassifier(C, D_f)$ ▷ train the classifier on the transformed dataset
return (clf, S)

As a result, reducing the number of shapelets can substantially decrease the overall training time of the method.

RSAST, in particular, does not explore the entire space of possible subsequences but instead performs a sampling of them, distinguishing it from its baseline STC and SAST. For instance, compared to SAST, RSAST does not utilize all potential starting points for the subsequences along the time series due to a parameter that controls the extraction of these random points. Thus, only when this parameter is set equal to or higher than the length of the time series does RSAST consider the same number of subsequences as SAST. Similarly, since RSAST consider a reduced amount of subsequences within the randomly selected instances of the training dataset, these can be adjusted to utilize all instances and all possible subsequence lengths, enabling the method to explore the same number of subsequences as STC.

Moreover, other shapelet approaches such as FS and RDST also utilize the training dataset to select candidate shapelets. For instance, FS employs multiple SAX words for the subsequences losing information in the new representation, while RDST generates the shapelets randomly considering some parameters like the number of shapelets to generate, the possible lengths, the proportion of shapelets that use normalization and the threshold for the shapelet frequency. Lastly, LS does not search for subsequences in the training dataset; instead, it utilizes a heuristic to learn near-optimal shapelets by exploring shapelet interactions.

C. Time Complexity

The complexity analysis of RSAST starts by the analysis of the computation of the list of weights. Since the method applies an ANOVA test for each point along the n time series of length m. This is accomplished with a complexity of $O(n|c|m)$ with $|c|$ the number of classes. Also, similar to SAST, RSAST selects k reference time series with $|c|$ classes, in $O(|c|)$ steps, resulting in a total of $O(k|c|)$ reference time series within the dataset. So, given a predetermined set of p starting points and $|L|$ possible lengths for the

subsequences, $O(k|c|p|L|)$ shapelet candidates are randomly selected. Therefore, in the worst scenario when $|L|$ reach the maximum possible values m, the shapelet selection process has a complexity of $O(k|c|pm)$.

Additionally, for the transformation step the method requires $O(k|c|pm)$ distance computations over the $O(n)$ time series with $O(m)$ arithmetic operations, culminating in a time complexity of $O(k|c|pnm^2)$.

Moreover, ACF and PACF tests are done in $O(k|c|mlog(m))$ and $O(k|c|m^3)$ steps, respectively, when using ACF with Fast Fourier Transform and PACF with Ordinary Least Square regression.

Therefore, the total time complexity of RSAST is $O(n|c|m) + O(c) + O(k|c|mlog(m)) + O(k|c|m^3) + O(k|c|pm)+O(k|c|pnm^2)+O(classifier)$. Hence the overall complexity of RSAST is $O(nm^2) + O(classifier)$.

IV. Experiments

RSAST was implemented in the Python programming language. The implementation was developed under SAST framework, following the scikit-learn design principles (URL: Repository RSAST). The repository contains the outcomes for the experimentation of the method.

A. Accuracy

In the following section, the results are presented in terms of accuracy, with a comparative analysis of RSAST against the original SAST algorithm and STC. Additionally, a comparison is provided between RSAST and other shapelet-based techniques like Fast Shapelet (FS) [22], Learned Shapelet (LS) [23] and Random Dilated Shapelet Transform (RDST) [10]. Finally, a comparison is made between the proposed algorithm and some ensemble methods comprising state-of-the-art techniques like HIVE-COTE, TS-CHIEF, and ROCKET.

During the experiments the supervised classifier employed within RSAST is the Ridge classifier, likewise to ROCKET. So it is utilized the Ridge Classifier with Leave-One-Out (LOO) cross-validation for scenarios where the transformed dataset contains more features than instances. In cases where the opposite holds true, a Logistic Regression classifier is utilized. All the parameters for the classifiers are left at their default values and are not fine-tuned.

In addition, we employed the Wilcoxon significance test with a significance level of 0.05 to make the comparison of accuracy among the methods. So in order to visualize this, a critical difference diagrams were constructed following the methodology proposed in [34]. These plots show the average ranks across all datasets and visualize them on a line, grouping classifiers into cliques where there exists no statistically significant difference in rank.

The experiment was run on an AMD EPYC 7763 64-Core Processor with 256 CPUs. The experiment encompassed the utilization of 128 datasets obtained from the UEA & UCR repository [35], employing the default training and test set provided by the repository.

Furthermore, for specific comparative analyses involving other methods, RSAST was run with 10 resampling iterations, covering the same 128 datasets as before. This approach aimed to comprehensively evaluate RSAST's performance across different scenarios and datasets.

After evaluating the performance of RSAST by varying the hyperparameters of the method (k instances per class and p starting point) we set this values to $k = 10$ and $p = 10$. Increasing the number of instances did not yield a significant improvement in accuracy but exponentially increased the method's execution time. Also with this number of instances reduces the variability in accuracy that could lead to chose just one instance per class.

The outcomes for SAST were taken from the SAST original paper (72 datasets). While the results for the remaining methods were taken directly from the UEA & UCR webpage (85 Bake Off datasets) for the default train/test results as well as the resampled results.

1) Comparison RSAST, SAST and STC: We evaluated and compared RSAST in terms of accuracy against the original SAST algorithm altogether STC and 1-nearest neighbour with dynamic time warping (1NN-DTW) as a benchmark method. For the results of SAST there are 72 results from the original paper using the default train/test split from the UCR archive, which corresponds to 56 datasets from the 85 Bake Off [36]. Similarly, the results from STC correspond to the 85 Bake Off datasets from the UEA & UCR webpage with the default train/test results, leading to a comparable dataset of 56 datasets for further analysis.

Fig.4, provides a visualization of the relative ranks of each method. Thus, it is observed that RSAST outperforms the accuracy of the TSC benchmark 1NN-DTW. Also, as it is showed by the clique in the critical difference diagram, RSAST achieves a similar level of accuracy to SAST and STC. So, RSAST attains comparable accuracy to its baseline counterparts while being faster to train.

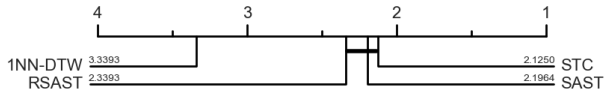
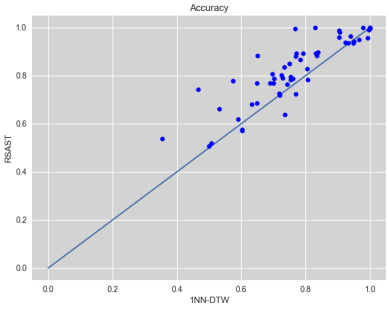


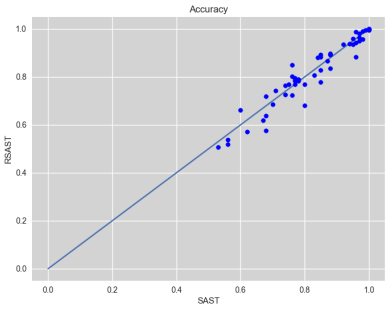
Fig. 4: Critical difference diagram between RSAST, SAST, STC and the benchmark 1NN-DTW

Furthermore, when it is compared one and one the accuracy of RSAST against the other methods, for the different datasets, we found that RSAST wins, against the benchmark 1NN-DTW, in 41 times, losses in 10 and draws in 5 times. Similarly, against SAST, RSAST wins 24 times, losses 21 times and draws 11 times. Finally, comparing with STC, RSAST wins 20 times, losses 30 times and draws 6 times (see Fig 5).

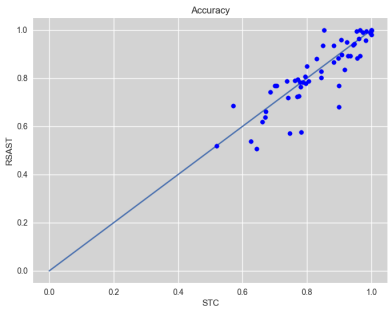
2) Comparison RSAST and Shapelet Methods: For this section, we compared shapelet methods based on the availability of results in the UCR Repository [35], or using well-



(a) RSAST(41 wins) vs 1NN-DTW(10 wins), 5 draws



(b) RSAST (24 wins) vs SAST (21 wins), 11 draws



(c) RSAST (20 wins) vs STC (30 wins), 6 draws

Fig. 5: Pairwise comparison of RSAST against SAST, STC and 1NN-DTW

established implementations, that allow testing the model locally.

Hence, the comparison among shapelet methods includes Fast Shapelet (FS), Learned Shapelet (LS), Random Dilated Shapelet Transform (RDST), and the benchmark 1NN-DTW. In the case of RSAST, this is evaluated using 128 datasets from the UCR archive with 10 resamples. For the rest of methods the available results corresponds to the 85 Bake Off datasets. Consequently, the results are computed for the 85 datasets of the evaluation of RSAST in the Bake Off.

So, Fig 6 illustrates that RSAST outperforms FS and the benchmark method. Moreover, RSAST demonstrates a similar level of accuracy to LS. However, when comparing RSAST and RDST, the latter method performs better in most of the datasets.

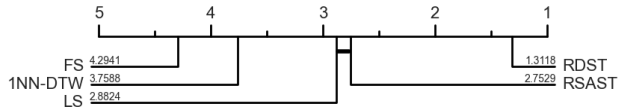


Fig. 6: Critical difference diagram between shapelet approaches and the benchmark 1NN-DTW.

When comparing the different accuracy of RSAST against the other shapelet approaches (see Fig 7), it is observed that RSAST outperforms FS in 73 cases, loses in 7 cases, and has 5 draws. Similarly, against LS, RSAST wins 40 times, loses 34 times, and has 11 draws. Finally, when compared with RDST, RSAST wins 6 times, loses 71 times, and has 8 draws.

Some datasets exhibit substantial variations in results among shapelet approaches. For instance, in ElectricDevices and Lightning7, RSAST surpasses FS, with scores of 0.82 compared to 0.29, and 0.75 compared to 0.15, respectively. Additionally, RSAST outperforms LS significantly in the OliveOil dataset, achieving a score of 0.87 compared to 0.17. However, RSAST shows notably lower performance in the ShapeletSim dataset, compared to the rest of shapelet methods recording an accuracy of 0.48 compared to 1.0, 0.93, and 0.99 for FS, LS, and RDST, respectively.

3) Comparison RSAST and Other Categories: For this comparison we employed the most representative methods, previously introduced, from each category along with RSAST.

Therefore, specifically, we used Time Series Forest (TSF) for interval-based methods, Elastic Ensemble (EE) for distance-based methods, and Bag of Symbolic Fourier Approximation Symbols (BOSS) for dictionary-based methods. Similar to the shapelet methods comparison, we used the 128 results for RSAST with 10 resampling, but this time narrowed to 83 results of BOSS from the UCR webpage. As a result, the comparison against RSAST it is computed for a total of 83 datasets.

The mean rank (see Fig 8) for each of these methods shows that BOSS is the most accurate, followed by RSAST. However, RSAST is not significantly worse; in fact, it is comparable in terms of accuracy with all the methods. Lastly, the distance-based method (EE) and the interval-based method (TSF) present similar accuracy.

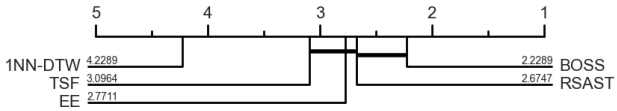
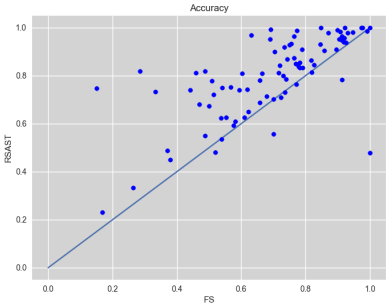


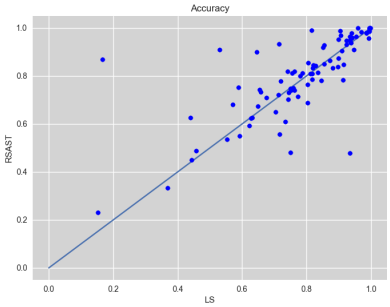
Fig. 8: Critical difference RSAST with other categories

In one-on-one accuracy comparisons among these methods (see Fig 9), RSAST outperforms TSF 42 times, loses 32 times, and results in 9 draws. Against EE, RSAST wins 37 times, loses 35 times, and draws 11 times. Finally, when compared with BOSS, RSAST wins 29 times, loses 40 times, and draws 14 times. Particularly, in the dataset named ShapeletSim, BOSS outperforms all the methods by a significant margin.

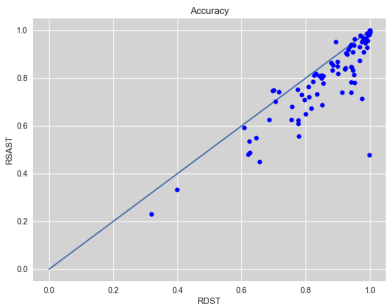
4) Comparison RSAST and SOTA methods: Similarly, as with shapelet methods, RSAST is compared with other state-of-the-art algorithms that are based on many type of features, sometimes including shapelets. For instance, some ensemble methods are included for the comparison such as: HIVE-COTE v1.0 which uses interval, dictionary, shapelet and spectral base features; TS-CHIEF which uses distance, dictionary and spectral base features; and ROCKET which extracts different



(a) RSAST (73 wins) vs FS (7 wins), 5 draws

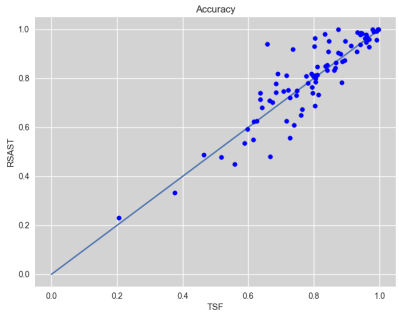


(b) RSAST (40 wins) vs LS (34 wins), 11 draws

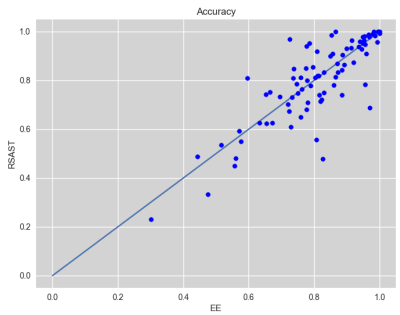


(c) RSAST (6 wins) vs RDST (71 wins), 8 draws

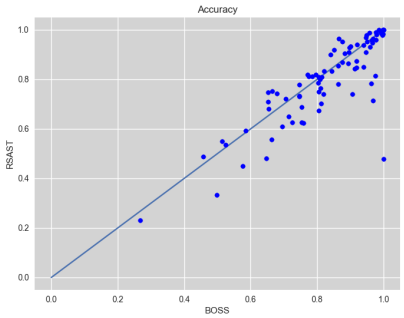
Fig. 7: Pairwise comparison of shapelet approaches



(a) RSAST (42 wins) vs TSF (32 wins), 9 draws



(b) RSAST (37 wins) vs EE (35 wins), 11 draws



(c) RSAST (29 wins) vs BOSS (40 wins), 14 draws

Fig. 9: Pairwise comparison with other categories

features through the convolution operation with random kernels.

The results indicate that RSAST appears to be less accurate than ROCKET, HIVE-COTE and TS-CHIEF (see Fig 10). However it is important to notice that RSAST is a single feature approach whereas in the ensemble methods there are many features used to make the classification. Moreover, the majority of these ensemble methods function as black box models, making it impossible to discern how the classification of time series is performed. In contrast, RSAST employs shapelets for this task, providing clarity in the classification process.

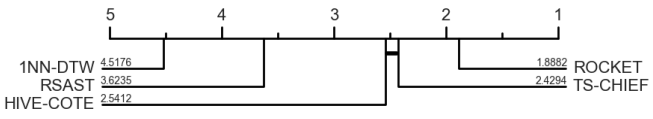


Fig. 10: Critical difference diagram SOTA algorithms

When comparing RSAST against each of the other methods for each dataset individually (see Fig 11), it is observed that RSAST wins 18 times against HIVE-COTE, loses 58 times, and draws 9 times. Similarly, RSAST wins 15 times against TS-CHIEF, loses 60 times, and draws 10 times. Lastly, RSAST wins 6 times against ROCKET, loses 68 times, and draws 11 times.

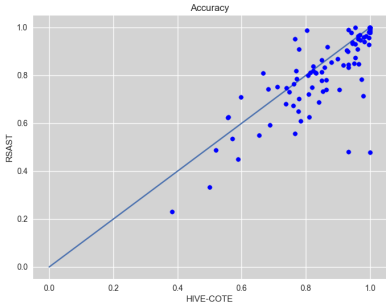
B. Scalability

The scalability assessment of RSAST followed a methodology similar to that of SAST, so we compared the training time of the models with respect the time series length and the number of time series within the dataset. So, for the length method in RSAST, we used the ACF&PACF criteria, varying the number of instances across 1, 10, while the number of random points ranged between 10 and 30. Additionally, we evaluated the most scalable methods, to our knowledge, like the original SAST algorithm, ROCKET, and RDST.

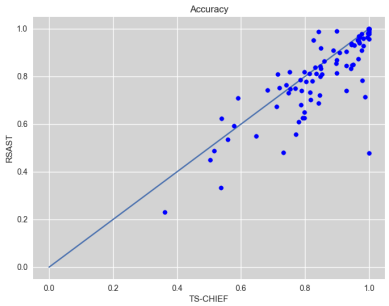
The experiments were performed using RSAST integrated with logistic regression for the section focused on increasing the training set size (Section IV-B1). However, in relation to the time series length, the experiments were conducted in conjunction with a ridge regression classifier (Section IV-B2).

Both experiments (time series length and training set size) were carried out locally utilizing an Intel Core i5-4300U processor. Also, in order to better show the results we re-scale the training time in a logarithmic scale.

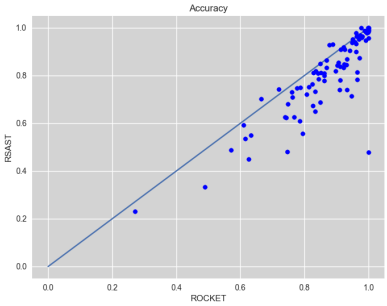
1) Training set size: To assess the scalability of the method concerning the training set size, a random oversampling of the Chinatown dataset was performed. Instances were selected with replacement from the original dataset within the range of



(a) RSAST (18 wins) vs HIVE-COTE (58 wins), 9 draws



(b) RSAST (15 wins) vs TS-CHIEF (60 wins), 10 draws



(c) RSAST (6 wins) vs ROCKET (68 wins), 11 draws

Fig. 11: Pairwise comparison with SOTA methods

2^4, 2^5, 2^6, 2^7, 2^8, 2^9 and 2^{10}. This procedure yielded a new dataset with an augmented number of time series. A logistic regression classifier was employed for all the methods.

The results are presented in Figure 12, showcasing the performance of the various tested approaches. Notably, RSAST demonstrates good performance compared to the other approaches. Specifically, as it is shown in Table I, RSAST with 1 instance and 10 random points exhibits the best time performance, taking only 0.3s, for 1024 instances. Additionally, RSAST with 10 instances, and 10 random points took 3.84s for the same number of time series.

It is noteworthy that ROCKET is the method that experiences the most significant increase in total training time as the number of time series (TS) rises, going from 0.97s to 7.79s for 16 and 1024 instances respectively.

The SAST approach, using a logistic regression classifier, took 3.1s for 1024 instances mirroring the tendency observed with RDST as the number of instances grows.

Lastly, RDST achieved a training time of 5.97 seconds for 1024 instances, showing similarity to RSAST with 10 instances and 10 random points.

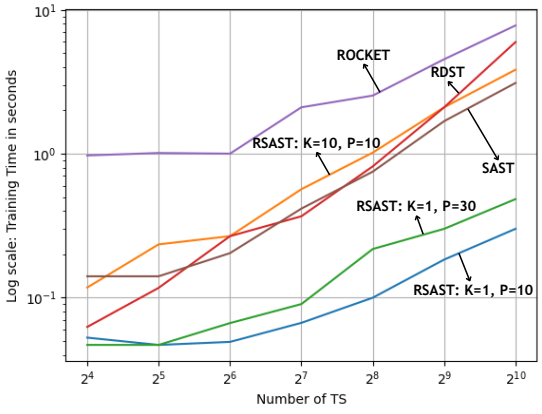


Fig. 12: training time versus training set size

2) Time series length: To evaluate the scalability of the method in terms of the time series length, the HouseTwenty dataset is utilized, with time series truncated at lengths of 2^5, 2^6, 2^7, 2^8, 2^9, and 2^{10}. A Ridge Classifier was employed for all the methods (RSAST, SAST, RDST and ROCKET). Figure 13, provides a visual representation of the escalating computation time exhibited by the various methods as the time series length grows.

Method	# Series 16	1024
RSAST: n_random_points=10 nb_inst_per_class=1	0.05s	0.3s
RSAST: n_random_points=10 nb_inst_per_class=10	0.12s	3.84s
RSAST: n_random_points=30 nb_inst_per_class=1	0.05s	0.48s
RDST: n_shapelets=10_000	0.06s	5.97s
ROCKET: num_kernels=10_000	0.97s	7.79s
SAST: min_shapelet_length=3 max_shapelet_length=m	0.14s	3.1s

TABLE I: Summary training time versus training set size

Notably, as it is shown in Table II, the SAST approach exhibits the highest training computation time, going from 0.87s with instances of length 32 to 18 hours, 45 minutes, and 48.02 seconds with instances of length 1024. On the other hand, RSAST, with 10 instances and 10 random points, took 38 minutes, and 1.96 second for time series of length 1024.

Furthermore, RDST and ROCKET emerges as the most scalable method, followed by RSAST with 1 instance and 10 random points, taking a training times of 12.45s, 7.96s and 3 minutes 15.76s, respectively for time series of length 1024.

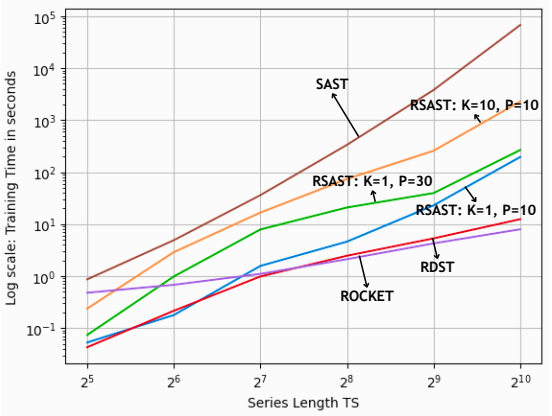


Fig. 13: Training time versus time series length

Method	Length 32	Length 1024
RSAST: n_random_points=10 nb_inst_per_class=1	0.05s	3m 15.76s
RSAST: n_random_points=10 nb_inst_per_class=10	0.24s	38m 1.96s
RSAST: n_random_points=30 nb_inst_per_class=1	0.07s	4m 26.37s
RDST: n_shapelets=10_000	0.04s	12.45s
ROCKET: num_kernels=10_000	0.48s	7.96s
SAST: min_shapelet_length=3 max_shapelet_length=m	0.87s	18h 45m 48.02s

TABLE II: Summary training time versus time series length

C. Interpretability

Similar to the primitive approach [18], and the rest of shapelet approaches within the literature, the predictions made by RSAST can be interpreted through the identification and visualization of the selected shapelets.

Thus, for our method, this is done by the application of feature importance analysis, as in SAST. In this process, each feature is linked to a shapelet candidate extracted from a time series. The shapelet candidates that correspond to the most significant features are regarded as the top-performing shapelets.

Additionally, it is important to note that the class label associated with any given shapelet candidate is derived from the class of the time series from which it was originally extracted. So, the class label of a new time series can be obtained by using the class labels of the shapelet candidates to which it is the most similar.

Thus, RSAST predictions uses the significance of each feature, determined by the absolute value of its corresponding learned weight. So, as depicted in a example of the Coffee dataset from the UCR archive (see Figure 14), the extracted shapelets spanning short subsequences, from 13 points (feature 3), and larger shapelets with a length of 78 points (feature 1).

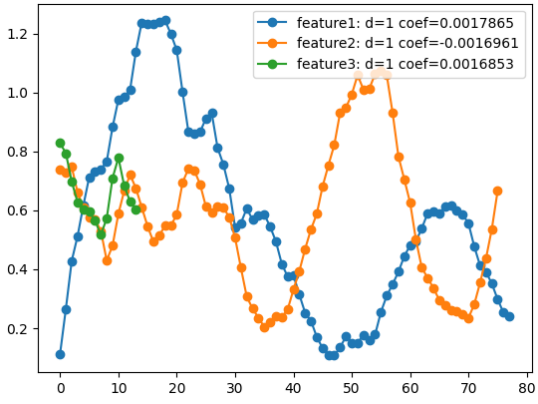


Fig. 14: Shapelets for RSAST using Coffee dataset

Moreover, Figure 15 illustrates the top 3 best shapelets extracted by RSAST, plotted on the reference time series, for the Coffee dataset. The top rows of the figures correspond to the reference time series selected from class 0, while the second rows represent the reference time series selected from class 1. A perfect match between a shapelet candidate and a reference time series implies that the shapelet has been extracted from that particular reference time series. Thus, in this case, the feature 1 is learned from class 0, while the feature 2 and 3 are learned from class 1. This visualization offers insights into the discriminatory patterns learned by RSAST for distinguishing between the two classes of the dataset.

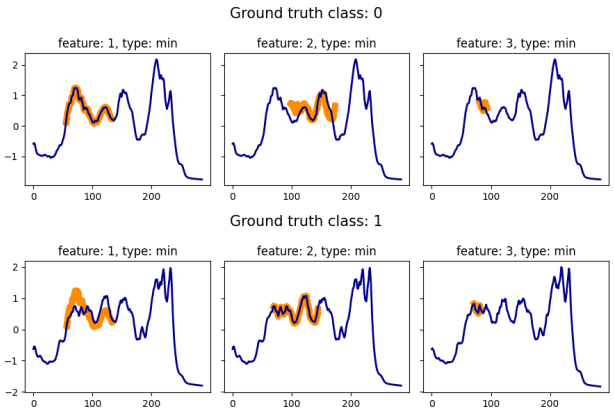


Fig. 15: Top 3 shapelets, on TS, learned by RSAST on Coffee dataset.

V. Conclusion

In summary, shapelet-based methods in time series classification aim to identify distinctive patterns within time series so as assign a class, to a new time series, from a range of possible class values.

The Scalable and Accurate Subsequence Transform (SAST) is a state-of-the-art technique that makes use of these representative subsequences to train a classifier. However, despite its accuracy, SAST faces challenges in terms of scalability, particularly when dealing with longer time series.

Thus, in order to enhance the computational efficiency of the algorithm we proposed Random SAST (RSAST) a method that generates shapelets randomly, guided by certain statistical criteria, reducing the search space of shapelets.

RSAST demonstrates competitive accuracy when compared to other shapelet-based techniques, as well as against various state-of-the-art approaches. Moreover, its interpretability is a noteworthy feature, as it relies on the extraction of continuous subsequences from the training time series. This characteristic allows for a clearer understanding of the model's decision-making process.

References

[1] M. F. Mbouopda and E. Mephu Nguifo, "Scalable and accurate subsequence transform for time series classification," *Pattern Recognition*, vol. 147, p. 110121, 2024. 1, 2, 3

[2] M. Middlehurst, P. Schäfer, and A. Bagnall, "Bake off redux: a review and experimental evaluation of recent time series classification algorithms," *arXiv preprint arXiv:2304.13029*, 2023. 1, 3

[3] J. Lines and A. Bagnall, "Time series classification with ensembles of elastic distance measures," *Data Min. Knowl. Discov.*, vol. 29, no. 3, p. 565–592, 2015. 1

[4] B. Lucas, A. Shifaz, C. Pelletier, L. O'Neill, N. Zaidi, B. Goethals, F. Petitjean, and G. Webb, "Proximity forest: an effective and scalable distance-based classifier for time series," *Data Mining and Knowledge Discovery*, vol. 33, no. 3, pp. 607–635, 2019. 1

[5] C. H. Lubba, S. S. Sethi, P. Knaute, S. R. Schultz, B. D. Fulcher, and N. S. Jones, "catch22: Canonical time-series characteristics: Selected through highly comparative time-series analysis," *Data Mining and Knowledge Discovery*, vol. 33, no. 6, pp. 1821–1852, 2019. 1

[6] M. Christ, N. Braun, J. Neuffer, and A. W. Kempa-Liehr, "Time series feature extraction on basis of scalable hypothesis tests (tsfresh–a python package)," *Neurocomputing*, vol. 307, pp. 72–77, 2018. 1

[7] H. Deng, G. Runger, E. Tuv, and M. Vladimir, "A time series forest for classification and feature extraction," *Information Sciences*, vol. 239, pp. 142–153, 2013. 1

[8] M. Flynn, J. Large, and T. Bagnall, "The contract random interval spectral ensemble (c-rise): the effect of contracting a classifier on accuracy," in *Hybrid Artificial Intelligent Systems: 14th International Conference*. Springer, 2019, pp. 381–392. 1

[9] J. Hills, J. Lines, E. Baranauskas, J. Mapp, and A. Bagnall, "Classification of time series by shapelet transformation," *Data Min. Knowl. Discov.*, vol. 28, no. 4, p. 851–881, 2014. 1, 2, 3

[10] A. Guillaume, C. Vrain, and W. Elloumi, "Random dilated shapelet transform: A new approach for time series shapelets," in *International Conference on Pattern Recognition and Artificial Intelligence*. Springer, 2022, pp. 653–664. 1, 3, 4, 7

[11] P. Schäfer, "The boss is concerned with time series classification in the presence of noise," *Data Mining and Knowledge Discovery*, vol. 29, pp. 1505–1530, 2015. 1

[12] A. Dempster, F. Petitjean, and G. I. Webb, "Rocket: exceptionally fast and accurate time series classification using random convolutional kernels," *Data Mining and Knowledge Discovery*, vol. 34, no. 5, pp. 1454–1495, 2020. 2, 4

[13] A. Dempster, D. F. Schmidt, and G. I. Webb, "Minirocket: A very fast (almost) deterministic transform for time series classification," in *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, 2021, pp. 248–257. 2

[14] R. Zhang, Q. Wang, and Y. Lu, "Combination of resnet and center loss based metric learning for handwritten chinese character recognition," in *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, vol. 5. IEEE, 2017, pp. 25–29. 2

[15] H. Ismail Fawaz, B. Lucas, G. Forestier, C. Pelletier, D. F. Schmidt, J. Weber, G. I. Webb, L. Idoumghar, P.-A. Muller, and F. Petitjean, "Inceptiontime: Finding alexnet for time series classification," *Data Mining and Knowledge Discovery*, vol. 34, no. 6, pp. 1936–1962, 2020. 2

[16] J. Lines, S. Taylor, and A. Bagnall, "Time series classification with hive-cote: The hierarchical vote collective of transformation-based ensembles," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 12, no. 5, pp. 1–35, 2018. 2, 4

[17] L. Sthle and S. Wold, "Analysis of variance (anova)," *Chemometrics and Intelligent Laboratory Systems*, vol. 6, no. 4, pp. 259–272, 1989. 2, 3

[18] L. Ye and E. Keogh, "Time series shapelets: A new primitive for data mining," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Association for Computing Machinery, 2009, p. 947–956. 2, 3, 11

[19] K. Park, *Fundamentals of Probability and Stochastic Processes with Applications to Communications*. Springer International Publishing, 2017. [Online]. Available: https://books.google.fr/books?id=OmRADwAAQBAJ 2

[20] R. Shumway and D. Stoffer, *Time Series Analysis and Its Applications: With R Examples*, ser. Springer texts in statistics. Springer, 2006. [Online]. Available: https://books.google.fr/books?id=J-moNKkSNrEC 3

[21] A. Bostrom and A. Bagnall, "Binary shapelet transform for multiclass time series classification," *Transactions on Large-Scale Data- and Knowledge-Centered Systems XXXII: Special Issue on Big Data Analytics and Knowledge Discovery*, pp. 24–46, 2017. 3, 4

[22] T. Rakthanmanon and E. Keogh, "Fast shapelets: A scalable algorithm for discovering time series shapelets," in *proceedings of the 2013 SIAM International Conference on Data Mining*. SIAM, 2013, pp. 668–676. 3, 7

[23] J. Grabocka, N. Schilling, M. Wistuba, and L. Schmidt-Thieme, "Learning time-series shapelets," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 392–401. 3, 7

[24] I. Karlsson, P. Papapetrou, and H. Boström, "Generalized random shapelet forests," *Data mining and knowledge discovery*, vol. 30, pp. 1053–1085, 2016. 3

[25] T. Le Nguyen, S. Gsponer, I. Ilie, M. O'reilly, and G. Ifrim, "Interpretable time series classification using linear models and multi-resolution multi-domain symbolic representations," *Data mining and knowledge discovery*, vol. 33, pp. 1183–1222, 2019. 3

[26] T. Le Nguyen, S. Gsponer, and G. Ifrim, "Time series classification by sequence learning in all-subsequence space," in *2017 IEEE 33rd international conference on data engineering (ICDE)*. IEEE, 2017, pp. 947–958. 3

[27] J. Large, J. Lines, and A. Bagnall, "A probabilistic classifier ensemble weighting scheme based on cross-validated accuracy estimates," *Data mining and knowledge discovery*, vol. 33, no. 6, pp. 1674–1709, 2019. 4

[28] A. Bagnall, M. Flynn, J. Large, J. Lines, and M. Middlehurst, "A tale of two toolkits, report the third: on the usage and performance of hive-cote v1.0," in *AALTD@PKDD/ECML*, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:215745319 4

[29] M. Middlehurst, W. Vickers, and A. Bagnall, "Scalable dictionary classifiers for time series classification," in *Intelligent Data Engineering and Automated Learning–IDEAL 2019: 20th International Conference, Manchester, UK, November 14–16, 2019, Proceedings, Part I 20*. Springer, 2019, pp. 11–19. 4

[30] M. Middlehurst, J. Large, M. Flynn, J. Lines, A. Bostrom, and A. Bagnall, "Hive-cote 2.0: a new meta ensemble for time series classification," *Machine Learning*, vol. 110, no. 11-12, pp. 3211–3243, 2021. 4

[31] A. Shifaz, C. Pelletier, F. Petitjean, and G. I. Webb, "Ts-chief: a scalable and accurate forest algorithm for time series classification," *Data Mining and Knowledge Discovery*, vol. 34, no. 3, pp. 742–775, 2020. 4

[32] M. Dash and H. Liu, "Feature selection for classification," *Intelligent data analysis*, vol. 1, no. 1-4, pp. 131–156, 1997. 5

[33] C. Molnar, *Interpretable Machine Learning*. Leanpub, 2020. [Online]. Available: https://books.google.fr/books?id=jBm3DwAAQBAJ 5

[34] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *The Journal of Machine learning research*, vol. 7, pp. 1–30, 2006. 7

[35] H. A. Dau, A. Bagnall, K. Kamgar, C.-C. M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatana, and E. Keogh, "The ucr time series archive," *CAA Journal of Automatica Sinica*, vol. 6, no. 6, pp. 1293–1305, 2019, [Online; accessed August 18, 2023]. 7

[36] A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh, "The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances," *Data mining and knowledge discovery*, vol. 31, pp. 606–660, 2017. 7