



# Scalable and Accurate Subsequence Transform

Michael F Mbouopda •, Engelbert Mephu

## ► To cite this version:

Michael F Mbouopda •, Engelbert Mephu. Scalable and Accurate Subsequence Transform. Pattern Recognition, 2023, 147, pp.1-18. 10.1016/j.patcog.2023.110121 . hal-03087686v3

**HAL Id: hal-03087686**

**<https://uca.hal.science/hal-03087686v3>**

Submitted on 21 Nov 2023

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Distributed under a Creative Commons Attribution - NonCommercial - NoDerivatives 4.0 International License

# Scalable and Accurate Subsequence Transform for Time Series Classification

Michael Franklin MBOUOPDA\*\*, Engelbert MEPHU NGUIFO\*

University Clermont Auvergne, Clermont Auvergne INP, CNRS, ENSMSE, LIMOS, F-63000, Clermont-Ferrand, France

---

## ARTICLE INFO

Keywords:

Time series

Classification

Shapelet

Scalability

Interpretability

---

## ABSTRACT

Time series classification using phase-independent subsequences called shapelets is one of the best approaches in the state of the art. This approach is especially characterized by its interpretable property and its fast prediction time. However, given a dataset of  $n$  time series of length at most  $m$ , learning shapelets requires a computation time of  $O(n^2 m^4)$  which is too high for practical datasets. In this paper, we exploit the fact that shapelets are shared by the members of the same class to propose the SAST (Scalable and Accurate Subsequence Transform) algorithm which has a time complexity of  $O(nm^3)$ . SAST is accurate, interpretable and does not learn redundant shapelets. The experiments we conducted on the UCR archive datasets showed that SAST is more accurate than the state of the art Shapelet Transform algorithm while being significantly more scalable.

---

## 1. Introduction

The world is surrounded today by data that change through time. Such data are used to model the dynamic of one or multiple variables and are called time series. They are exploited in many domains such as physics, chemistry, finance, medicine, and industry. Many tasks can be performed on time series data and one of them is the classification task. Time series classification (TSC) is a task that consists of learning a function able to classify objects represented as time series. This task has been successfully performed in astronomy in order to classify galaxies and stars (Moss, 2018), in smart appliances in order to identify faults (Gupta, Gupta, Biswas and Dutta, 2020), in medicine for rapid pathogen identification (Papagiannopoulou, Parchen and Waegeman, 2019), in satellite image analysis (Sanchez, Serrurier and Ortner, 2019) and in many other applications.


There are many methods for the classification of time series. Regarding the feature used for classification, these methods are grouped into seven categories: whole series, interval, shapelet, dictionary, spectral, hybrid, and deep learning. In this paper, we put our focus on shapelet based methods. This class of algorithms is very interesting because they are accurate, robust to noise and interpretable (Ye and Keogh, 2009). In particular, the shapelet transform

---

\* This work is funded by the French Ministry of Higher Education, Research and Innovation. Thanks to the UEA & UCR Time Series Classification Repository which provides the datasets used for our experiments. Many thanks to the reviewers for their time and constructive feedback. Great thanks to SK Imran Hossain and Jean-Jacques Givry for proofreading this paper.

\*Corresponding author

\*\*Principal corresponding author

 michael.mbouopda@uca.fr (M.F. MBOUOPDA); engelbert.mephu\_nguifo@uca.fr (E. MEPHU NGUIFO)

 <https://frank11.github.io/> (M.F. MBOUOPDA); <https://perso.isima.fr/~enmephun/> (E. MEPHU NGUIFO)

ORCID(s): 0000-0003-1916-1464 (M.F. MBOUOPDA); 0000-0001-9119-678X (E. MEPHU NGUIFO)

algorithm is proved to be among the most effective when tested on the UEA & UCR archive (Bagnall, Lines, Bostrom, Large and Keogh, 2017). They have also proved to be effective in clustering (Siyu Fotso, Mephu Nguifo and Vaslin, 2020), showing how useful shapelets are. The interpretability of a shapelet method is obtained by visualizing the subsequences that trigger the class label of a given instance. Since the introduction of TSC using shapelets, one of the major limitations of the developed algorithms is their time complexity. In fact, the state of the art time complexity of shapelet based methods is  $n^2m^4$  where  $n$  is the number of time series in the dataset and  $m$  is the length of the longest time series. This high time complexity is due to the large number of shapelet candidates that need to be evaluated in order to find the top best shapelets.

A human brain is able to recognize a lot of variations of an object after seeing a single variant. For instance, we are able to recognize any model of car after seeing one of them, we can recognize many species of dog if we have ever seen a dog. This ability is called *core object recognition* (DiCarlo, Zoccolan and Rust, 2012). Inspired by this amazing behavior of our brain, we claim that a shapelet model should be able to recognize any variant of a shapelet if it knows one or a few number of its variants. Simply defined, a shapelet is a pattern that is shared by the time series that belong to the same class. Therefore, any single instance of a class should contain all the shapelet candidates or at least a variant of each shapelet candidate for that class. Guided by this observation, we propose the Scalable and Accurate Subsequence Transform (SAST) algorithm, a TSC algorithm that is accurate, scalable and whose predictions are interpretable.

Existing shapelet based methods use the whole dataset to generate shapelet candidates, then use information gain to select the top best shapelets before doing the classification using a supervised classifier. We claim that it is not necessary to generate the shapelet candidates from the whole dataset, only one or few instances per class is enough. We also claim that pruning shapelet candidates without taking into account the classifier can lead to inaccurate classification. We propose the SAST model to support our claims ; it uses only a single instance per class in order to generate shapelet candidates. Furthermore, shapelet candidates are not assessed beforehand of classification. The supervised classifier automatically identifies the top best shapelets during its training phase. The key points of our contribution are the following:

- We introduce the *core shapelet recognition* task which aims to recognize any variant of a shapelet from one or few variants of that shapelet. We claim that TSC by shapelets is a core shapelet recognition task and therefore the size of the shapelet space is considerably reduced without losing crucial information.
- We propose the SAST method, which successfully performs the core shapelet recognition task in order to accurately classify time series. SAST is also more scalable than the state of the art shapelet methods. In particular, SAST took 1 second to classify the Chinatown dataset with an accuracy of 96%, while the state of the art shapelet

based algorithm STC took 51 seconds and achieved an accuracy of 97% on the same computer. Furthermore our proposed method can successfully classify some datasets on which STC fails.

- We give an open source implementation of our model that is compatible with the well known scikit-learn machine learning library. This allows anyone to easily use our model on its own data, and to reproduce our experiments. The code is available here <https://github.com/frank11/sast>

The rest of this paper is organized as follows: we start by setting the background and present related works in Section 2. In Section 3 we describe our proposed method SAST, which is inspired by the core object recognition capability of human brain. In Section 4, we assess SAST on various datasets and compare it to state of the art shapelet and non-shapelet based methods. Section 5 summarizes this work and presents future direction.

## 2. Background and related works

In this section, we start by presenting the background notions required to follow this work and then we continue by presenting the related works.

### 2.1. Background

**Definition 1** (Time series). *A sequence of  $n$  real values recorded in time. In this definition,  $n$  is the length of the time series*

$$T = (t_1, t_2, \dots, t_n), \forall_{1 \leq i \leq n} t_i \in \mathbb{R} \quad (1)$$

A time series can be decomposed into subsequences of different lengths.

**Definition 2** (Subsequence or pattern). *Given a time series  $T$  of length  $n$ , a subsequence (also called pattern)  $S$  of length  $l$  is a sequence of  $l$  consecutive values of  $T$  starting at time step  $j$ .*

$$S = (s_1, s_2, \dots, s_l) = (t_j, t_{j+1}, \dots, t_{j+l-1}) \quad (2)$$

Two subsequences are said *similar* if the distance between them is less than a predefined threshold. Euclidean distance and Dynamic Time Warping (DTW) are the most used similarity distance in TSC (Bagnall et al., 2017). For shapelet based methods, Euclidean distance is preferred in the literature (Ye and Keogh, 2009; Bagnall et al., 2017) and we also use it in this work. However, any distance can be used to compute the similarity between subsequences.

A time series *is similar* to a pattern if there is a subsequence of that time series that is similar to the pattern.

**Definition 3** (Similarity). *The similarity between a time series  $T$  of length  $m$  and a subsequence  $S$  of length  $l$  is defined as follows:*

$$\text{dist}(T, S) = \min_{R \in W_l} (\sum_{i=1}^l (r_i - s_i)^2), \text{ with } W_l \text{ being the set of all subsequences of length } l \text{ in } T \quad (3)$$

The patterns  $S$  and  $R$  are put on the same scale before the distance computation using a normalization technique. Beware that we are not using the Euclidean distance, but the square of the Euclidean distance. As a matter of fact, the square root can be seen as a change of the similarity scale and therefore does not change the similarity.

Among all the patterns contained in a time series dataset, some can be used as discriminative features in order to find the class label of an unseen time series. These patterns are called *shapelets*. Shapelets have been introduced as primitives for TSC by Ye and Keogh (2009). The authors proposed a shapelet based decision tree in which each node is a subsequence and the time

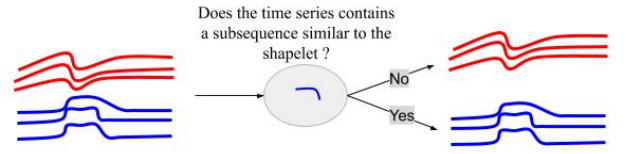


Figure 1: An illustration of a node in a shapelet decision tree for a binary TSC

series arriving at a node are split into two groups such that one group contains data that are similar to the subsequence at that node, and the other group is the set of data that are not similar to the subsequence. Figure 1 illustrates a node in the proposed decision tree. The blue time series contain the subsequence in the node (i.e they are similar to the subsequence at the node), so they follow the branch labeled with *yes*. The red time series does not contain the subsequence in the node (i.e they are not similar to the subsequence at the node), therefore they follow the branch labeled with *no*.

Training a shapelet based decision tree consists of learning the best subsequence (i.e shapelet) to use at each node, and for each shapelet the best similarity threshold. This training is done in a top-down approach as in a classical decision tree using the information gain (IG) at each node to find the best split (Ye and Keogh, 2009).

**Definition 4** (Shapelet). *Given a dataset  $D = \{(T_1, c_1), (T_2, c_2), \dots, (T_n, c_n)\}$  of time series with their class labels  $c_i$  taken from a finite set of classes  $C$ , a shapelet  $S^*$  is a subsequence that maximizes the information gain*

$$S^* = \operatorname{argmax}_{S \in W} IG(D, S), \text{ with } W \text{ being the set of all subsequences in } D. \quad (4)$$

**Definition 5** (Information gain (IG)). *Let  $D$  be a time series dataset and  $S$  a shapelet. Let  $D_L = \{T \in D \mid \text{dist}(T, S) \leq$*

$\epsilon\}$  and  $D_R = \{T \in D \mid \text{dist}(T, S) > \epsilon\}$ , then

$$IG(D, S) = \max_{\epsilon \in SP \subset \mathbb{R}} \left( H(D) - \frac{|D_L|}{|D|} H(D_L) - \frac{|D_R|}{|D|} H(D_R) \right), \text{ with } H(D) = - \sum_{c \in C} p_c \log p_c \quad (5)$$

$H(\cdot)$  is the entropy,  $p_c$  is the probability of having the class  $c$  in the dataset  $D$ ,  $C$  is the set of classes in  $D$ , and  $SP$  is the set of possible split points.

For a time series of length  $m$ , there are  $m-l+1$  subsequences of length  $l$  and  $\frac{m(m+1)}{2}$  subsequences in total for a single time series. Since any subsequence in the dataset is a shapelet candidate, and there are about  $O(nm^2)$  subsequences, evaluating them is very time consuming. Because of that, the time complexity of the shapelet decision tree algorithm is  $n^2m^4$  (Ye and Keogh, 2009). Hence, the method does not scale well with the number of time series and moreover with the length of the time series. The goal of this work is to improve the scalability of shapelet-based classification without sacrificing accuracy or interpretability.

## 2.2. Time series features for classification

Depending on the features used for classification, TSC methods have been grouped into six categories (Hills, Lines, Baranauskas, Mapp and Bagnall, 2014):

- *whole series*: these methods use the whole series as feature. The main idea is to define a similarity distance (Euclidean distance, Dynamic Time Warping and its variants, Time Warped Edit Distance, etc) between time series and then use the nearest neighbor algorithm for classification. The Fast Ensemble of Elastic Distance or FastEE (Tan, Petitjean and Webb, 2020) is an ensemble method that combines different similarity distances.
- *interval*: the discriminative features are localized at some specific intervals on the time series, and are computed using some aggregate functions (mean, median, max, etc). A supervised classifier is then trained on the computed features. Time Series Forest or TSF (Deng, Runger, Tuv and Vladimir, 2013) is to our knowledge the most popular algorithm in this category. TSF has inspired the development of more accurate and efficient methods, namely the Canonical Interval Forest or CIF (Middlehurst, Large and Bagnall, 2020) and the Supervised Time Series Forest or STSF (Cabello, Naghizade, Qi and Kulik, 2020). CIF extends the aggregate functions used in TSF with the catch22 (Lubba, Sethi, Knaute, Schultz, Fulcher and Jones, 2019) features and random sampling. STSF uses a supervised approach to examine only a set of sub-series in the original time series.
- *shapelet*: shapelet-based methods use phase independent patterns (subsequences) as features. These subsequences are shared by members of the same class and are called shapelets. An instance is classified regarding the shapelets it contains. The first proposed method in this category is the Shapelet Decision Tree (Ye and

Keogh, 2009). Later, Fast Shapelet or FS (Rakthanmanon and Keogh, 2013) has been proposed to improve the time complexity of the Shapelet Decision Tree while keeping almost the same accuracy. Unlike Shapelet Decision Tree, Shapelet Transform Classifier or STC (Hills et al., 2014) separates the shapelet learning phase from the classification phase, allowing the usage of any supervised classifier and a significant accuracy improvement. ELIS++ (Zhang, Wang, Fang, Wang and Wang, 2021) is a recent shapelet method that uses data augmentation and logistic regression to optimize shapelets. ELIS++ is an improvement of a former method named LS or Learning time series Shapelets (Grabocka, Schilling, Wistuba and Schmidt-Thieme, 2014).

- *dictionary*: these methods are used when the classes have the same discriminative patterns, however at different frequencies for each class. Generally, the time series are converted to sequences of symbols before extracting patterns also called words. The Bag of Symbolic-Fourier Approximation Symbols or simply BOSS (Schäfer, 2015) and its variants the Contract BOSS (Middlehurst, Vickers and Bagnall, 2019) and the Spatial BOSS (Large, Bagnall, Malinowski and Tavenard, 2019) are some state of the art algorithms in this category. Another state of the art is the Word ExtrAction for time SEries cLassification or WEASEL (Schäfer and Leser, 2017) algorithm.
- *spectral*: these methods work in the frequency domain and are able to extract features that are very hard to find in the time domain. Some popular techniques used here are power spectrum and auto correlation function (ACF) (Bagnall and Janacek, 2014; Flynn, Large and Bagnall, 2019).
- *hybrid*: these are methods that work on different features at the same time in order to take advantage of each type of feature. The most famous model in this category is HIVE-COTE (Lines, Taylor and Bagnall, 2018) which is composed of 5 components, each focusing on one type of features (whole series, interval, shapelet, dictionary or spectral) and is an ensemble of classifiers. TS-CHIEF (Shifaz, Pelletier, Petitjean and Webb, 2020) is a tree-based hybrid method that uses all the previous features except shapelet features. A recent method is ROCKET (Dempster, Petitjean and Webb, 2020) which uses randomly generated convolutional kernels in order to extract different types of features.

Recently, deep learning has shown its effectiveness in many tasks, including TSC (Fawaz, Forestier, Weber, Idoumghar and Muller, 2019). Deep learning methods can learn different type of features and can be very accurate if there are enough data for training and computing power. The main limitation of these methods is that they are *black box* methods and therefore are difficult to explain. In fact, explanation by methods such saliency maps and class activation maps is generally unreliable and inaccurate for time series predictions (Ismail, Gunady, Corrada Bravo and Feizi, 2020). Some state of the art deep learning models for TSC are Resnet (Wang, Yan and Oates, 2017) and InceptionTime (Fawaz, Lucas, Forestier, Pelletier, Schmidt, Weber, Webb, Idoumghar, Muller and Petitjean, 2020).

### 2.3. Related works

Many works have been proposed in order to improve the scalability of shapelet-based TSC, but at the cost of accuracy and possibly interpretability. Fast Shapelet or FS (Rakthanmanon and Keogh, 2013) reduces the time complexity to  $O(nm^2)$  by using a symbolic representation of time series. Symbolic representations have the benefit of reducing the time series length and hence the number of shapelet candidates. Other authors proposed to reduce the space of shapelet candidates by randomly picking a fraction of subsequences to be evaluated (Renard, Rifqi, Erray and Detyniecki, 2015; Wistuba, Grabocka and Schmidt-Thieme, 2015). Although these methods can find competitive shapelets, they are less accurate in general.

The state of the art algorithm for shapelet based classification of time series is the Shapelet Transform Classifier (STC) algorithm proposed by Hills et al. (2014) and speed up later by Bostrom and Bagnall (2015). This algorithm designs shapelet-based classification as a three step process. The first step is the selection of the top  $k$  shapelets, this is equivalent to learning a bag of shapelets to represent the time series. The second step is the shapelet transformation where each time series in the dataset is replaced by a vector of its distances to each of the selected shapelets. The final step consists of training a supervised classifier on the transformed dataset. Although the time complexity of this algorithm is still  $n^2m^4$ , it is the shapelet based algorithm that obtains the best classification accuracy on the UEA & UCR archive (Bagnall et al., 2017).

Although shapelets are defined as subsequences selected from the considered time series, other approaches proposed to learn shapelets through an optimization process. The first of these approaches is Learning Shapelet or LS (Grabocka et al., 2014) which starts with randomly initialized shapelets and uses logistic regression to adjust them iteratively with the goal of maximizing the classification accuracy. This allows the learning of shapelets that are not present in the training time series. However, the shapes of the learned shapelets are unpredictable and can be difficult to interpret. The time required by this method to be accurate depends on the initialization step and the optimization algorithm used. Unlike LS which initializes shapelets randomly, ELIS++ (Zhang et al., 2021), an improvement of ELIS (Fang, Wang and Wang, 2018) initializes shapelets by the subsequences contained in the training time series. Then Bayesian optimization is used to set the hyperparameters automatically. Furthermore, ELIS++ uses data augmentation in order to converge on small datasets. ELIS++ is more accurate and converges faster than LS. ELIS++ is tight to logistic regression and optimizing the hyperparameters is challenging. In this work our focus is on the classical shapelet algorithm, in which the shapelets are not learned but are selected from the training time series. This guarantees interpretability and there is almost no hyperparameter to tune. Furthermore, the STC capabilities are unlimited since it can be coupled with any supervised classifier.

When tested on the UCR archive (Anthony Bagnall and Keogh, 2018), STC is the most accurate TSC method that uses a *single type* of features. However, many methods have combined different types of features and ensemble



techniques in order to achieve the highest accuracy. The current most accurate TSC methods in the state of the art are HIVE-COTE (Lines et al., 2018), TS-CHIEF (Shifaz et al., 2020) and ROCKET (Dempster et al., 2020). The time complexity of HIVE-COTE is bounded by the complexity of its shapelet component, that is  $n^2m^4$ . Therefore decreasing the computation time of the shapelet component will make HIVE-COTE more efficient in terms of running time. TS-CHIEF is a forest of trees in which a time series follows a branch if it is closer to the reference time series (there is one reference time series for each class) associated with that branch than to the reference time series of other branches. TS-CHIEF does not use shapelet features because of the time complexity needed for their computation. ROCKET is the most scalable state of the art method, it is based on random convolutional kernels that are used to transform the dataset. ROCKET results are not easily interpretable because the kernels used are short, independent and not sampled from the dataset.

SAST, our proposal described in the next section is architecturally similar to ROCKET, however, the way both models work is different in two folds: firstly, the subsequences used in SAST (similar to convolutional kernels in ROCKET) are those of some reference time series randomly chosen from the dataset, and hence they are dependent and have variable lengths. Secondly, SAST does not use the convolution operator as similarity measure, but the Euclidean distance. These properties of SAST are the reasons of its scalability and interpretability. SAST is a scalable and accurate alternative to STC. It can be used to reduce the computation time of the shapelet module in HIVE-COTE, and can be integrated to TS-CHIEF at the cost of little computation time overhead. Adding shapelet features in TS-CHIEF would increase its classification performance.

### 3. SAST: Scalable and Accurate Subsequence Transform

In TSC, a shapelet is ideally a pattern that is shared by every instance of the same class, and that instances of other classes do not have, they are called discriminative patterns or subsequences. The number of patterns in a dataset of  $n$  time series of length  $m$  is  $O(nm^2)$ , and state of the art shapelet algorithms evaluate each of them by computing their information gain for a set of similarity thresholds before keeping the patterns and their corresponding similarity thresholds that give the highest information gain. Reducing the number of patterns to be assessed will make shapelet models faster to train. In this section we propose a way to reduce the number of shapelet candidates. Then we show that there is no need to select the top best shapelets beforehand. Finally, we present a novel method for shapelet based TSC.

#### 3.1. Reducing the number of shapelet candidates

Human brain effortlessly performs *core object recognition*, the ability to recognize objects despite substantial appearance variations (DiCarlo et al., 2012). This gives human the capability to recognize a vast number of objects

that have the same name just by seeing a few of them. Heeger (2002-2014) used Figure 2 in his lecture notes on Perception to illustrate the notion of invariance in recognition. This figure shows different ducks. Some are in water while others are not, some ducks are photographs and others are drawings. Furthermore, the ducks have different sizes, colors, etc. Despite all these variabilities, a human brain that has already seen a duck is able to recognize that each object on this figure is a duck. This ability of human brain is what computer vision is aiming at. Although, this illustration is about image recognition, the idea is transferable to TSC based on shapelet by building algorithms that are able to recognize any variant of a shapelet by comparing it to one or a few numbers of its variants.

A shapelet is a pattern that is similar to time series from the same class. Any pattern that is similar to a shapelet can be considered a variant of that shapelet. Therefore, we introduce the *core shapelet recognition* task, whose goal is to recognize any variant of a shapelet by just seeing one or very few numbers of its variants. This task could be used as a preprocessing step which extracts features (i.e shapelets) for a downstream time series analysis task such as classification. We argue that TSC based on shapelets must incorporate a core shapelet recognition task and that it must be solved using fewer shapelet candidates than it has been done since the introduction of shapelets by

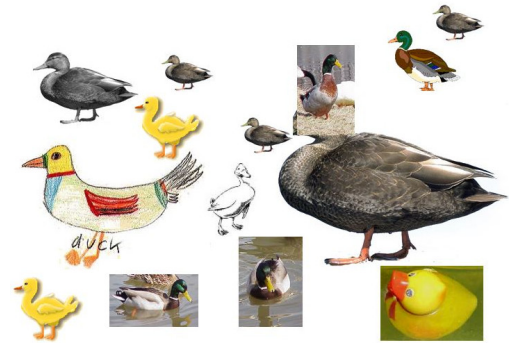


Figure 2: Illustration of invariance in recognition.

Ye and Keogh (2009). Hence, instead of generating shapelet candidates from the whole dataset, we propose to use only one or a few number of instances per class. In fact, the members of a class should contain common patterns or at least different variants of the same pattern. In this way, the learning algorithm must focus on one (or a few number) variant of each shapelet candidate to learn how to classify a time series. We acknowledge that the more variants the model learns from, the more accurate the model would be. However, this can also cause the model to overfit, especially when the training data is not representative of the testing data. There are known situations where the core shapelet recognition task cannot be solved: 1) The most obvious one is when no shapelet is present in the dataset, meaning that any shapelet approach would fail. 2) The second one is when the shapelets representing a class are not all present in each individual time series of the class, but are distributed over several instances.

In Figure 3, we have three randomly selected instances of the Chinatown dataset (containing two classes) from the UEA & UCR archive (Anthony Bagnall and Keogh, 2018). The left example is from class 1 and the second example (in the middle) is from class 2. It is easy to observe that the instance from class 1 starts by a deep valley, while the instance of class 2 does not. One reason that can be considered in order to classify the instance on the right in class 2 is that it does not start by a valley. Hence, we claim that observing only one instance per class could be enough to discover discriminative patterns and successfully perform classification of new instances. Figure 3 is of course not

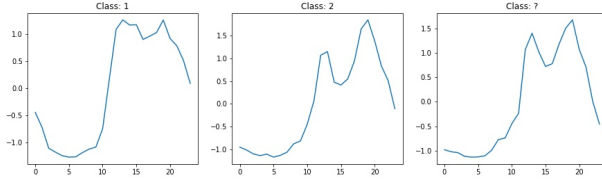


Figure 3: Three randomly selected instances from the Chinatown dataset. The instance on the right is probably from class 2 since it does not start with a valley

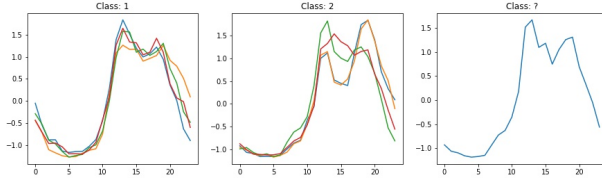


Figure 4: Some randomly selected instances from the Chinatown dataset. The instance on the right is definitely from class 2

enough to support our claim. However, the extensive experiment described in Section 4 and the obtained results give clear evidence supporting our claim.

Figure 4 shows 4 randomly selected instances for each class. Instances of the same class are superposed in order to expose global patterns. The figure emphasizes the previous observation that class 1 contains instances that start with a deep valley while class 2 are instances that are more flat at the beginning.

Based on this observation, we propose the following statement:

**Proposition 1.** *Let  $D = \{(T_1, c_1), (T_2, c_2), \dots, (T_n, c_n)\}$  be a dataset of time series. Let  $D_c$  be any subset of  $D$  that contains at most  $k$  ( $k \geq 1$ ) instances from each class. If classes in  $D$  can be distinguished using shapelets, then for any shapelet  $shp$  of  $D$ , there is a time series in  $D_c$  that is similar to  $shp$ .*

*Proof.* Let us assume that classes in  $D$  are distinguishable using shapelets and that there is a shapelet  $shp$  for the dataset  $D$  that is not similar to any time series in the set  $D_c$ . Since  $D_c$  contains at least a time series of each class in  $D$ , any shapelet for the dataset  $D$  must be similar to at least one time series in  $D_c$ . It follows from there that assuming  $shp$  to be a shapelet is wrong. Therefore the statement is true.  $\square$

Let us explain the expression “distinguishable using shapelet”, we use it to characterize datasets on which shapelet-based methods are known to be accurate. According to Bagnall et al. (2017), these datasets include electric device, ECG, sensor and simulated time series datasets. To the best of our knowledge, there is no way to know if a shapelet-based method would be suitable or not on a dataset without applying the method. This emphasizes the importance of improving the scalability of shapelet-based algorithms.

From the previous proposition, any shapelet  $shp$  of  $D$  is always similar to a pattern in  $D_c$ . Therefore, a shapelet algorithm that generated shapelet candidates from  $D_c$  can achieve the same accuracy as if  $D$  was used. We run the shapelet transform algorithm (STC) (Hills et al., 2014) on the Chinatown dataset and plotted the top-5 shapelets that have been selected for each class in Figure 5. The shapelets on the first row clearly identify the valley at the beginning

of time series in class 1. Although they are coming from different time series, they are very similar in shape. Likewise, the shapelets on the last row identify the flat starting of instances in class 2. Generating shapelet candidates from the whole dataset makes STC learn different variants of the same patterns. The variations are in terms of starting position, length and shape.

By applying proposition 1 in the STC algorithm, we introduce STC-k, a variation of the STC algorithm that uses at most  $k$  time series from each class to generate the shapelet candidates. Hence, for a dataset with  $c$  classes, the number of shapelet candidates to be evaluated in STC-k is  $O(ckm^2)$  unlike STC in

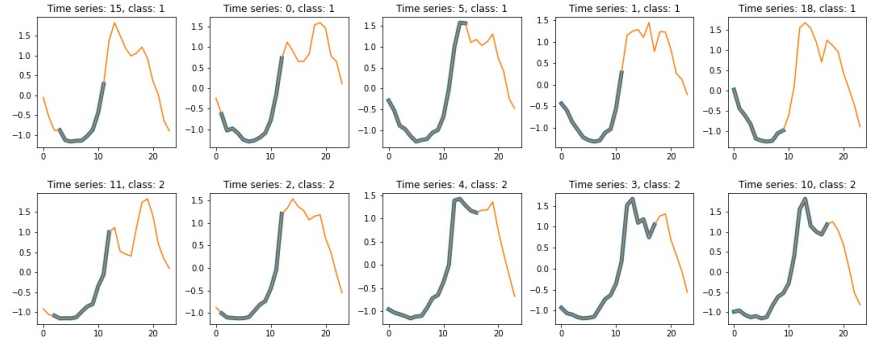


Figure 5: Top-5 shapelets extracted for each class of the Chinatown dataset by the shapelet transform algorithm.

which  $O(nm^2)$  need to be evaluated. Algorithm 1 is an outline of the STC-k algorithm. The only difference with STC is that the size of the shapelet space can be controlled by the parameter  $k$ . For a more detailed description of the algorithm, the reader should refer to the original paper of STC (Hills et al., 2014).

---

**Algorithm 1:** ShapeletTransformK
 

---

**Input:**  $D$ : dataset,  $k$ : nb of instances per class,  $C$ : classifier,  $length\_list$ : subsequence lengths,  $min\_ig$ : min IG

```

1 begin
2    $D_c \leftarrow randomlySelectInstancesPerClass(D, k)$ 
3    $S \leftarrow generateShapeletCandidates(D_c, length\_list)$ 
4    $S \leftarrow extractShapelet(S, D, min\_ig)$ 
5    $D_f \leftarrow \emptyset$ 
6   for  $i \leftarrow 1$  to  $n$  do
7      $x_i \leftarrow []$ 
8     for  $j \leftarrow 1$  to  $|S|$  do
9        $| x_i[j] \leftarrow dist(T_i, S_j)$ 
10    end
11     $D_f \leftarrow D_f \cup \{(x_i, c_i)\}$ 
12  end
13   $clf \leftarrow trainClassifier(C, D_f)$ 
14  return  $(clf, S)$ ;           // the trained classifier and the shapelet candidates
15 end
    
```

---

By default, the  $length\_list$  parameter is the set  $\{3, 4, \dots, m\}$ , where  $m$  is the length of the time series in the dataset.  $min\_ig$  is set to 0.05. In practice, two other parameters are used in STC: the maximum number of shapelets to keep

per class and the time contract. The number of shapelets to keep per class is by default set to 200. The time contract is the maximum time allocated to the algorithm to search shapelets on the given dataset. Middlehurst et al. (2020) stated that in one hour of searching per dataset, the result is not significantly worse than the full search. Following the implementation of STC in the sktime library (Löning, Bagnall, Ganesh, Kazakov, Lines and Király, 2019), our implementation of Algorithm 1 handles the time contract and the maximum number of shapelets to keep per class.

We use the Chinatown dataset from the UCR archive (Anthony Bagnall and Keogh, 2018) as a toy dataset to assess STC-1 (that is STC- $k$  with  $k = 1$ ). STC-1 took about 10 seconds to classify the Chinatown dataset with an accuracy of 96% (average over five runs), while the original STC algorithm took 51 seconds and gave an accuracy of 97% on the same computer. Therefore, the STC-1 algorithm is about 5 times faster and achieves almost the same accuracy as the original algorithm. The extracted shapelets are shown in Figure 6. Different variants of the same shapelet are not learned anymore. For this dataset, exactly one shapelet has been selected for each class. As we will show in Section 4, STC- $k$  is significantly less accurate than STC, even when  $k$  is equal to 75% of the number of time series in each class.

### 3.2. Identify shapelets using feature importance analysis

When TSC using shapelets was introduced, shapelets were learned while building a decision tree (Ye and Keogh, 2009). Later, shapelet transform (STC) (Hills et al., 2014) has been proposed to allow the use of any supervised classifier. The algorithm proceeds by finding the top best shapelets, then transforms the dataset using the found shapelets and finally trains a classifier

on the transformed dataset (Hills et al., 2014; Bostrom and Bagnall, 2015; Karlsson, Papapetrou and Boström,

2016). Therefore, there are three main steps: feature extraction where best shapelets are selected, dataset transformation where each time series is replaced by a vector of its distance to the selected shapelets and finally training where a classifier is trained on the transformed dataset.

We propose to remove the feature extraction step and use every shapelet candidate to transform the dataset. After training the classifier on the transformed dataset, a post hoc method for model explanation can be used to find the most important features. The importance of a feature represents how much that feature is correlated to the target variable (Dash and Liu, 1997; Molnar, 2022). The importance of a feature in a tree based algorithm determines how much it reduces the variance of the data compared to the parent node (Dash and Liu, 1997; Molnar, 2022). This corresponds exactly to the definition of a shapelet (see Definition 4). In a linear model, the absolute value of the weight of an

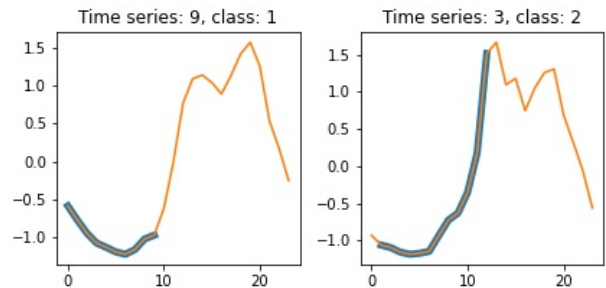


Figure 6: Shapelets extracted by STC on the Chinatown dataset using a single randomly selected instance per class to generate shapelet candidates.

important feature will be greater than the one of a less important feature (Molnar, 2022). Classifiers such as decision trees and linear models are said to be inherently interpretable since a post hoc analysis is not required to interpret their predictions. More generally, when a classifier is fitted, a post hoc explainer can be used to find the most important features (Murdoch, Singh, Kumbier, Abbasi-Asl and Yu, 2019) in order to interpret predictions. Two examples of these post hoc explainers are LIME (Ribeiro, Singh and Guestrin, 2016) and SmoothGrad (Smilkov, Thorat, Kim, Viégas and Wattenberg, 2017) for saliency maps. More methods can be found in the review of Samek, Montavon, Lapuschkin, Anders and Müller (2020). Hence, selecting shapelets beforehand of classification using information gain can be skipped, since the classifier can automatically learn the top best shapelets during its training iterations and feature analysis can be used after training to get the learned shapelets.

**Proposition 2.** *Let  $D = \{(T_1, c_1), (T_2, c_2), \dots, (T_n, c_n)\}$  be a dataset of time series, and  $S$  the set of all subsequences in  $D$ . Let  $D_f = \{(x_1, c_1), (x_2, c_2), \dots, (x_n, c_n)\}$  be a dataset such that  $x_i = [x_{i:1}, x_{i:2}, \dots, x_{i:|S|}]$ , where  $x_{i:j} = \text{dist}(T_i, S_j)$ . If the  $j^{\text{th}}$  feature is an important feature given by the analysis of feature importance for the dataset  $D_f$ , then  $S_j$  is a shapelet for the dataset  $D$ .*

*Proof.* Let us suppose the  $j^{\text{th}}$  feature is an important feature, and that  $S_j$  is not a shapelet for the dataset. By definition 4, not being a shapelet means that the information gain of  $S_j$  is not high enough, and whether a time series  $T$  is similar or not to  $S_j$  does not give any clue about the class of  $T$ . Therefore, knowing  $\text{dist}(T, S_j)$  doesn't help to classify  $T$ . In other words, the  $j^{\text{th}}$  feature is not correlated to the target variable. Hence, it cannot be an important feature. This proves the statement.  $\square$

### 3.3. Time series classification with SAST

TSC with SAST (Scalable and Accurate Subsequence Transform) is designed with respect to Proposition 1 and Proposition 2. A visual view of the the method is shown in Figure 7. There are two main blocks:

- *The classification block:* this block is actually the SAST algorithm and begins with the random selection of reference time series from which subsequences are then generated. Thereafter, the dataset is transformed by replacing each time series with a vector of its distances to each subsequence. Finally a supervised classifier (illustrated here by a decision tree) is trained on the transformed dataset.
- *The interpretability block:* The role of this block is to explain the SAST algorithm by identifying shapelet candidates associated with the most important features learned by the classifier. For inherently interpretable classifiers such as decision trees, the importance of each feature is computed while fitting the classifier. For other classifiers, eventually not inherently interpretable, an existing post hoc explainer such as LIME (Ribeiro et al., 2016) can be used to find the importance of each feature.

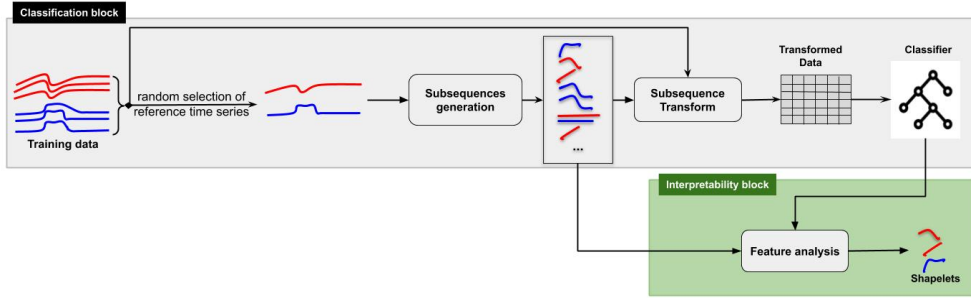


Figure 7: Overview of the SAST method

A pseudo code of the SAST algorithm is given by Algorithm 2. SAST takes as input the time series dataset  $D$ , the number  $k$  of instances to randomly select from each class in order to create the shapelet candidates, the list of lengths to use to generate shapelet candidates, and finally the supervised classifier  $C$  that is going to be trained on the transformed dataset.

---

**Algorithm 2:** ScalableAndAccurateSubsequenceTransform

---

**Input:**  $D$ ,  $k$ : nb of instances per class,  $length\_list$ : subsequence lengths,  $C$ : classifier

---

```

1 begin
2    $D_c \leftarrow \text{randomlySelectInstancesPerClass}(D, k)$ 
3    $S \leftarrow \text{generateShapeletCandidates}(D_c, length\_list)$ 
4    $D_f \leftarrow \emptyset$ 
5   for  $i \leftarrow 1$  to  $n$  do
6      $x_i \leftarrow []$ 
7     for  $j \leftarrow 1$  to  $|S|$  do
8        $x_i[j] \leftarrow \text{dist}(T_i, S_j)$ 
9     end
10     $D_f \leftarrow D_f \cup \{(x_i, c_i)\}$ 
11  end
12   $clf \leftarrow \text{trainClassifier}(C, D_f)$ 
13  return  $(clf, S)$ ;           // the trained classifier and the shapelet candidates
14 end
    
```

---

SAST starts by randomly sampling  $k$  instances per class from the dataset (line 2). By default,  $k$  is set to one. We call the selected instances *reference time series*. The next step is the generation of all subsequences of length in  $length\_list$  from the reference time series (line 3). The dataset transformation is performed from line 4 to 11. Here, the similarity between each time series in the dataset and each shapelet candidate is computed. The classifier taken as input is then trained on the transformed dataset (line 12). The algorithm returns the trained classifier and the shapelet candidates that have been generated. After the training is done, the class labels of a test dataset can be predicted in two steps: firstly the dataset is transformed using the shapelet candidates that have been generated during training, and finally the trained classifier is used to predict the class labels of the transformed test dataset.



Since the number of time series in the dataset and the number of subsequences in a time series are limited, Algorithm 2 always terminates. According to Proposition 2, the returned classifier should have learned to classify time series regarding features that are related shapelets.

### 3.4. SAST time complexity

Each step of the SAST algorithm runs in a finite amount of time, therefore the algorithm always terminates. Selecting  $k$  reference time series is done in  $O(c)$  time complexity,  $c$  is the number of classes in the dataset. There are  $m - l + 1$  subsequences of length  $l$  in a time series of length  $m$ . The total number of subsequences for a time series is  $\frac{m(m+1)}{2}$ . Since there are  $kc$  reference time series in a dataset with  $c$  classes, generating all shapelet candidates is done in  $O(kcm^2)$ . The transformation step requires  $O(nm^2)$  distance computations, each of which requires  $O(l)$  ( $l$  is the length of the subsequence) point wise operations. As the maximum subsequence length is  $m$ , the time complexity of the transformation step is  $O(nm^3)$ . Therefore, the total time complexity of SAST is  $O(c) + O(kcm^2) + O(nm^3) + O(\text{classifier})$ , where  $O(\text{classifier})$  is the time complexity of the classifier used. The overall asymptotic time complexity of the SAST algorithm is therefore  $O(nm^3) + O(\text{classifier})$ . SAST is much faster than the state of the art shapelet transform algorithm (STC) (Hills et al., 2014) whose time complexity is  $O(n^2m^4) + O(\text{classifier})$ .

### 3.5. Ensemble of SAST models

SAST accuracy is highly dependent on the randomly selected reference series. If a reference time series is noisy or not representative of its class, then it could be difficult for SAST to learn the best shapelets for the dataset. Furthermore, the random selection of reference time series could lead to a variance in performance. We use Bagging (Breiman, 1996) to leverage these possible issues and we call the obtained model SASTEnsemble (or SASTEN in reduced form). SASTEN is obtained by ensembling  $r$  SAST models. Each individual model in the ensemble uses randomly selected reference time series and may also have different parameters, especially the parameters controlling the length of shapelet candidates (that is *length\_list* in Algorithm 2). The final prediction is obtained by averaging the predictions of every SAST model in the ensemble. The time complexity of SASTEN is  $r$  times the time complexity of SAST if run sequentially. But this can be reduced using parallelization. SASTEN uses  $r$  times more memory than a regular SAST.

## 4. Experiments

We have implemented STC-k, SAST and SASTEN in Python. Our implementation is based on the scikit-learn machine learning library (Pedregosa, Varoquaux, Gramfort, Michel, Thirion, Grisel, Blondel, Prettenhofer, Weiss, Dubourg, Vanderplas, Passos, Cournapeau, Brucher, Perrot and Duchesnay, 2011). We have also followed scikit-learn design principles so that our models are compatible with any scikit-learn pipeline. We have used the implementation of



STC (Shapelet Transform Classifier) from the sktime library (Löning et al., 2019). The source code of our experiments and all the results we discuss in this paper are publicly available here <sup>1</sup>.

In all our experiments, the number of reference time series per class (that is the parameter  $k$  in Algorithm 2) is always set to one. The supervised classifier used in STC- $k$ , STC and SAST is the Ridge classifier with Leave-One-Out (LOO) cross validation. This classifier is available in the scikit-learn library. The LOO cross validation is used to find the best regularization parameter among 10 log spaced values ranging from  $-3$  to  $3$  (these values are inspired by Dempster et al. (2020)). The other parameters are left to their default values and are not fine tuned.

We have also used the Random Forest classifier in SAST. For this classifier all features are evaluated at each node to find the best split and a split is selected if the impurity decreases by about 0.05, the minimal information gain for a shapelet like in STC. Although it is generally better to evaluate only a subset of the feature space in Random Forest in order to reduce the correlation between the trees, we have not followed this guideline in our work because we want the model to always select the best possible split (that is the best shapelet). However, each tree in the ensemble is trained on a random subset of the training set. This classifier is also available in the scikit-learn library.

We make use of the Wilcoxon significance test with a p-value of 0.05 to compare our models. We give the result of this test as a critical difference diagram on which models that are not significantly different from each other are linked with a bold line. The code used for this test and to draw critical difference diagrams is from Fawaz et al. (2019).

Table 1 describes the models that we use in our experiments.

We experiment using 72 randomly selected datasets from the UEA & UCR repository (Anthony Bagnall and Keogh, 2018). The datasets in the repository are different in terms of series length, number of series, number of classes and application domain. For each dataset, the repository provides a training set and a test set. Since searching shapelets for one hour is not significantly worse than the full search on the UEA & UCR archive (Middlehurst et al., 2020), we used a time contract of one hour for each STC- $k$  models as well as for STC.

## 4.1. Accuracy

In this subsection, we compare the models in terms of accuracy and we use scatter plots and critical difference diagrams to summarize the results. However, the exact accuracies of SAST, STC and STC- $k$ , which are the core models of this work are given in Table 4.

### 4.1.1. STC- $k$ results

We have evaluated STC- $k$  on 72 datasets with different value of the parameters  $k$ . We have considered STC-1, STC-0.5, STC-0.75 and STC. These models are described in Table 1. Figure 8 shows pairwise comparisons of these model accuracies on the test set of each dataset.

<sup>1</sup><https://github.com/frank11/sast>

**Table 1**

List of models used in our experiments

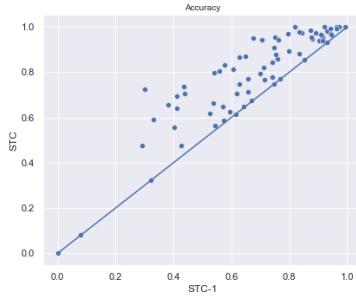
Name	classifier	length_list	Description
STC-1	Ridge classifier	$\{3, 4, \dots, m\}$	STC-k with $k = 1$ , meaning that shapelets are selected from a randomly selected time series per class
STC-0.25	Ridge classifier	$\{3, 4, \dots, m\}$	STC-k that select shapelets from 25% of time series per each class randomly selected
STC-0.5	Ridge classifier	$\{3, 4, \dots, m\}$	STC-k that select shapelets from 50% of time series per each class randomly selected
STC-0.75	Ridge classifier	$\{3, 4, \dots, m\}$	STC-k that select shapelets from 75% of time series per each class randomly selected
STC	Ridge classifier	$\{3, 4, \dots, m\}$	STC-k that select shapelets from every time series in the dataset
SAST-RF	Random Forest	$\{3, 4, \dots, m\}$	SAST model using Random Forest classifier
SAST-Ridge	Ridge classifier	$\{3, 4, \dots, m\}$	SAST model using Ridge classifier with LOO
SAST-Ridge-A	Ridge classifier	$\{9, 13, 15\}, \{7, 11, 15\}, \{7, 9, 15\}$ or $\{9, 11, 15\}$	Approximated SAST-Ridge, that is a SAST-Ridge which considers only some subsequence lengths
SASTEN-Ridge	Ridge classifier	-	Ensemble of 3 SAST-Ridge
SASTEN-Ridge-A	Ridge classifier	-	Ensemble of 3 Approximated SAST-Ridge with <i>length_list</i> $\{3, 4, \dots, 9\}$ , $\{10, 11, \dots, 16\}$ , and $\{17, 18, \dots, 23\}$ respectively

STC is better than any STC-k on almost every dataset. This is because an STC-k model does not search the whole shapelet space, and therefore the shapelets obtained using the minimum information gain are not good enough to classify the dataset. The critical difference diagram in Figure 9 shows that STC-0.75 is not significantly more accurate than STC-0.5, which is significantly more accurate than STC-0.25, which is in turn significantly more accurate than STC-1. Therefore, STC-k accuracy increases with the value of the parameter  $k$ . All STC-k models are considerably less accurate than STC.

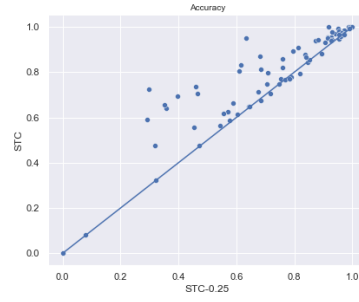
We have observed that STC generally fails at classifying datasets that have few time series in the training set. In particular, STC failed to find shapelets on the Fungi datasets. This dataset has 18 classes with one instance per class in the training set. In this particular case, STC is exactly the same as STC-1.

#### 4.1.2. SAST model results

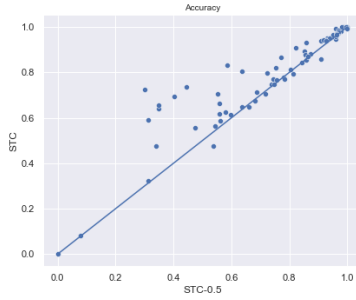
Before comparing SAST to STC, let us see how SAST and ensemble of SAST are compared to each other in terms of accuracy. Figure 11 shows a pairwise comparison of the SAST based models described in Table 1 on the 39 datasets marked with a star in Table 4. The first thing to note is that SAST-Ridge is generally more accurate than SAST-RF on our datasets (Figure 11a). There are many parameters in RF that can be optimized in order to improve SAST-RF, but



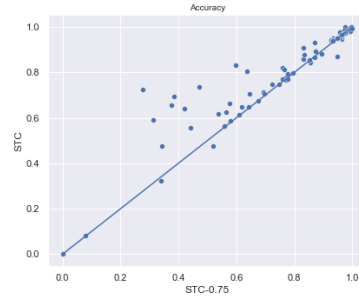
(a) STC (69 wins) vs STC-1 (1 win), 2 draws



(b) STC (58 wins) vs STC-0.25 (12 wins), 2 draws

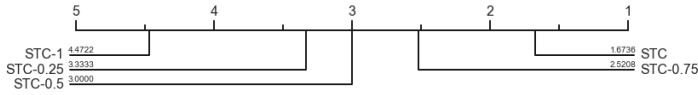


(c) STC (55 wins) vs STC-0.5 (12 wins), 5 draws



(d) STC (50 wins) vs STC-0.75 (16 wins), 6 draws

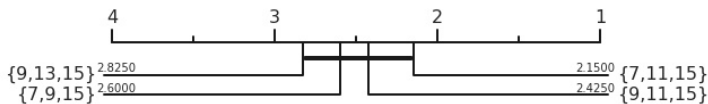
**Figure 8:** Comparison of STC-k to STC in terms of accuracy



**Figure 9:** Critical difference diagram between STC and STC-k

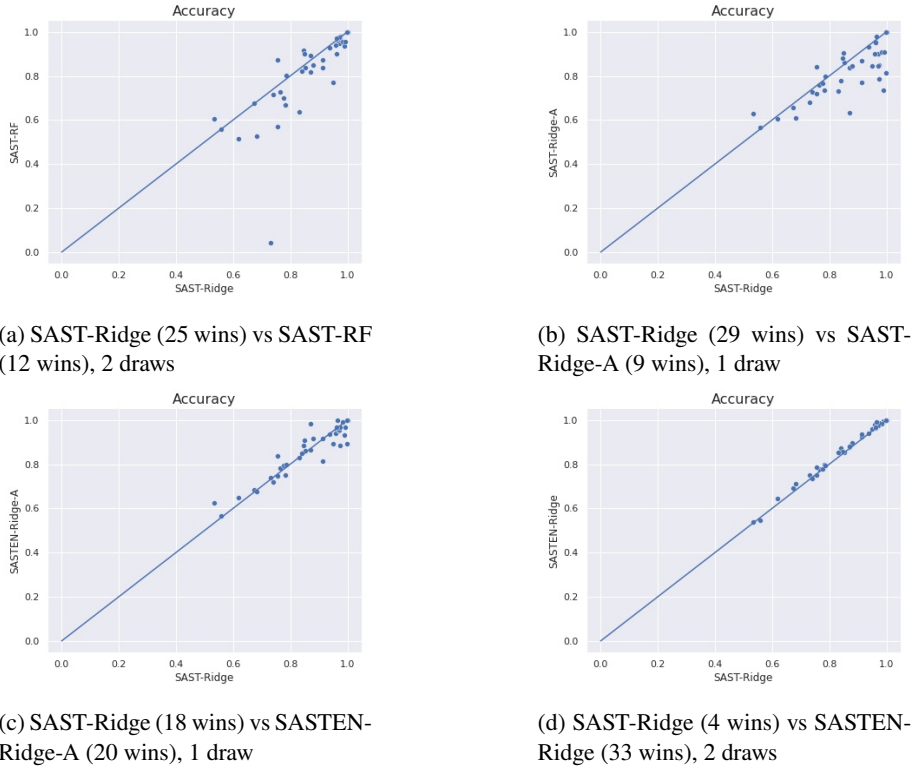
we did not perform parameter tuning in this work and we consider SAST-Ridge as the best model for our experiment. This is why we use SAST-Ridge as the default SAST model. It is important to note that the final results depend on the ability of the classifier to learn a good decision boundary. So, it should be chosen and fine tuned carefully while taking into account the required trade-off between explainability and classification performance

We tried several *length\_list* for the approximated SAST model, and we are presenting here only the four that achieved the best accuracy on our datasets. The critical difference diagram between these four models is given in Figure 10. There is no significant difference between the models, however, the model using *length\_list* = {7, 11, 15} is the best of all. When not clearly precised in the rest of this paper, SAST-Ridge-A is the approximated SAST-Ridge model with *length\_list* = {7, 11, 15}.



**Figure 10:** Critical difference diagram between approximated SAST models

The approximated SAST-Ridge is less accurate than SAST-Ridge in general (Figure 11b). However it is important to note that the approximated model wins on 9 datasets among 30. Therefore, knowing a prior about possible shapelet lengths can be used to train the model faster and without losing accuracy. Furthermore, ensembling approximated SAST models, each one focusing on different shapelet lengths is a possible way to improve accuracy while decreasing the computation time. In fact, SASTEN-Ridge-A is more accurate than SAST-Ridge on 20 datasets and less accurate on 18 (Figure 11c).



**Figure 11:** Pairwise comparison of model accuracies

Figure 11d reveals that ensembling SAST-Ridge models improves accuracy on almost every dataset. But the improvement is slight as even though the reference time series are chosen randomly, SAST-Ridge has very low variance in accuracy over multiple runs. We think that this capability comes from the fact that the model uses only one variant of each shapelet to learn the decision boundaries.

The critical difference diagram between SAST models is shown in Figure 12. SASTEN-Ridge is the best of all, followed by SASTEN-Ridge-A which is not significantly less accurate. SAST-Ridge is the third best model and is not significantly worse than SASTEN-Ridge-A, but is considerably less accurate than SASTEN-Ridge. SAST-Ridge-A and SAST-RF are significantly less accurate.

Although SASTEN-Ridge and SASTEN-Ridge-A are more accurate than SAST-Ridge, we believe that the accuracy

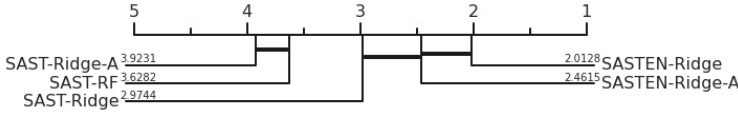


Figure 12: Critical difference diagram between SAST models

gain does not worth the computation time overhead required by SASTEN-Ridge or the engineering work required to find the appropriate *length\_list* to use in SASTEN-Ridge-A. Therefore, in the rest of this paper, we consider only the SAST-Ridge model and call it SAST for simplicity.

#### 4.1.3. SAST vs STC

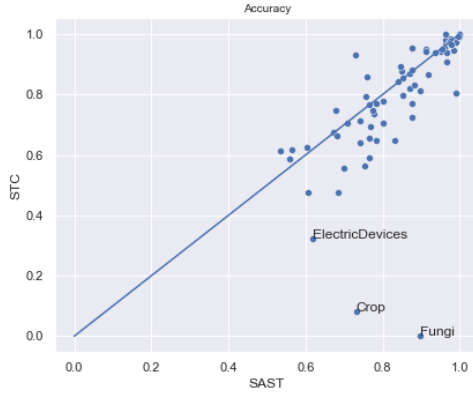
We now compare SAST (i.e SAST-Ridge) to STC, the state of the art shapelet method to our knowledge. This experiment is performed on the same 72 datasets and a pairwise comparison of SAST, STC and STC-1 is shown in Figure 13. SAST is more accurate than STC on 43 datasets, worse on 27 and there are two draws. STC-1 is more accurate than SAST on only 5 datasets among 72, although the only difference between these two models is that the Ridge classifier in SAST is trained using the whole shapelet space while only a subset of the shapelet space is used in STC-1. STC-1, STC and SAST, respectively achieve an average accuracy of  $0.68 \pm 0.21$ ,  $0.79 \pm 0.20$  and  $0.84 \pm 0.12$  on the 72 datasets. The standard deviation of STC and STC-1 models is higher due to the zero score obtained on one dataset (Fungi).

There are datasets on which STC and STC-1 hardly achieve 50% accuracy, while SAST performs significantly better. This is the case for the datasets Crop, ElectricDevices and Fungi. These datasets contain respectively 24, 7 and 18 classes. It is difficult to find a subsequence in these datasets that is present in one class and not in the others. A subsequence is generally shared among multiple classes, and therefore is not highly discriminative in terms of information gain by itself. Subsequences need to be combined in order to differentiate classes, and since all the subsequences are available in SAST, this combination is automatically learned by the classifier. Elsewhere SAST achieves 90% accuracy on the dataset Fungi, while STC and STC-1 fail to find any shapelet on it. These results confirm our thought that pruning shapelet candidates, without taking into account the classifier can lead to very inaccurate classification.

The critical difference diagram in Figure 14 reveals that SAST is generally more accurate than STC, but the difference is not highly significant.

#### 4.1.4. SAST vs other shapelet methods

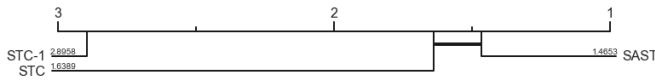
We compare our proposal to Fast Shapelet or FS (Rakthanmanon and Keogh, 2013). We also compare our proposal to methods that learn shapelets, namely Learning time series Shapelets or LS (Grabocka et al., 2014) and ELIS++ (Zhang et al., 2021). The accuracy of ELIS++, FS and LS are taken from the ELIS++ paper and we consider the



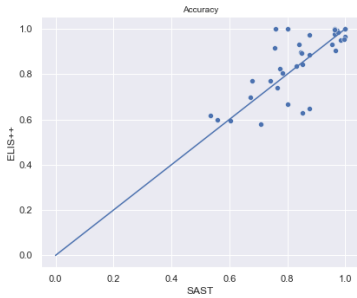
(a) SAST (43 wins) vs STC (27 wins), 2 draws



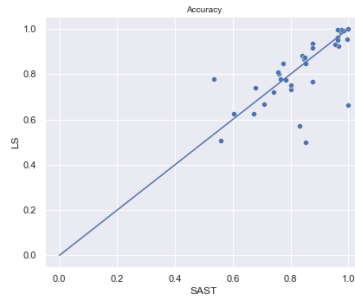
(b) SAST (66 wins) vs STC-1 (5 wins), 1 draw

**Figure 13:** Pairwise comparison of SAST, STC and STC-1

**Figure 14:** Critical difference diagram between SAST, STC and STC-1

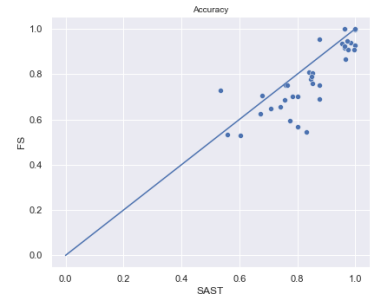
same 35 datasets they used (marked with a *plus* sign in Table 4). The average accuracies of these models on the 35 datasets are  $0.78 \pm 0.14$ ,  $0.81 \pm 0.14$ ,  $0.83 \pm 0.13$  and  $0.85 \pm 0.14$  for FS, LS, SAST and ELIS++ respectively.



(a) SAST (12 wins) vs ELIS++ (22 wins), 1 draw



(b) SAST (17 wins) vs LS (16 wins), 2 draws



(c) SAST (30 wins) vs FS (4 wins), 1 draw

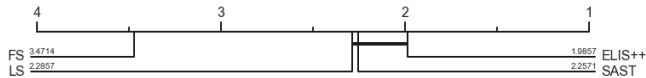
**Figure 15:** Pairwise comparison of SAST, ELIS++, LS and FS

**Figure 16:** Critical difference diagram between SAST, LS, FS and ELIS++

Figure 15 shows a pairwise comparison of these methods and the critical different diagram in Figure 16 shows how significant is each method compared to the others in terms of accuracy. LS, ELIS++ and SAST are not significantly different in terms of accuracy, however, they outperform FS. It is important to note that LS and ELIS++ do not select shapelets from the training set, but learn them through an optimization process. Therefore the shapelet space is un-

limited, the learned shapelets are unpredictable as well as the time required for convergence. Furthermore, finding the hyper-parameters and the appropriate shapelet initialization for these models is challenging. Without any optimization, SAST can achieve accuracies that are not significantly worse than ELIS++ accuracies and that are slightly better than LS accuracies.

#### 4.1.5. SAST vs other types of methods

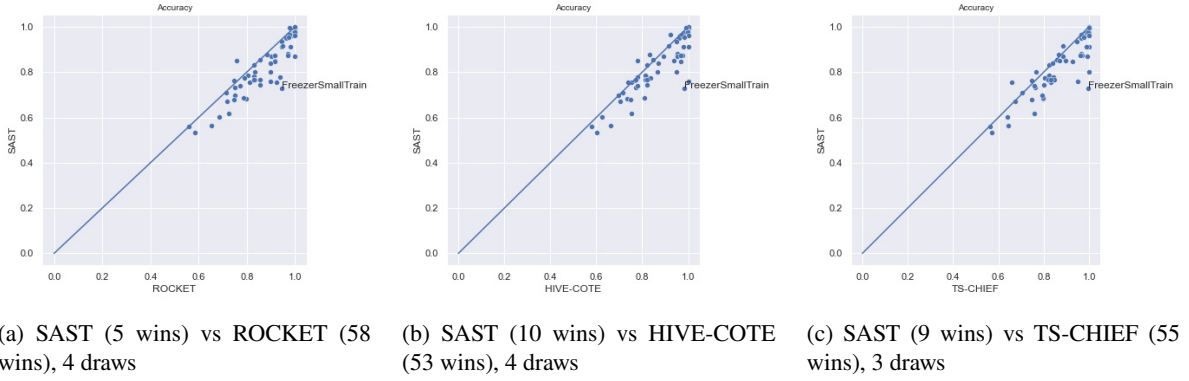
We also compare SAST to other state of the art algorithms that are not necessarily based on shapelets, nor on one type of features. ROCKET (Dempster et al., 2020), HIVE-COTE (Lines et al., 2018) and TS-CHIEF (Shifaz et al., 2020) are to our knowledge the most accurate methods for TSC. The results of these models are taken from the UEA & UCR repository (Anthony Bagnall and Keogh, 2018). Among the 72 datasets on which we have SAST results, ROCKET, HIVE-COTE and TS-CHIEF do not have results for the datasets DodgerLoopDay, DodgerLoopGame, DodgerLoopWeekend, Fungi and MelbournePedestrian; so we excluded these 5 datasets from this comparison. Elsewhere, we believe that the comparison we are doing here is not fair since these methods are not based on only shapelet features. However, considering the *no free lunch theorem* (Wolpert and Macready, 1997), SAST could outperform these models on some datasets and the goal of this experiment is to see how SAST stands w.r.t to these methods that are based on combination of features.

Although our model uses only shapelet features, it manages to outperform ROCKET on 5 among the 67 with 4 draws (Figure 17a). Elsewhere, SAST respectively outperforms HIVE-COTE and TS-CHIEF on 10 and 9 datasets among the 67 with 4 and 3 draws. Since SAST can perform better than HIVE-COTE on some datasets, replacing the shapelet module in HIVE-COTE with a SAST based model could increase HIVE-COTE accuracy and could reduce its time complexity since the shapelet module is the most time consuming one in HIVE-COTE. When TS-CHIEF was proposed, their authors decided not to exploit shapelet features because of their computation time. With the core shapelet recognition task we introduce in this work, we believe that shapelet features could be added in TS-CHIEF at low cost and that this could increase the accuracy of this model.

The Wilcoxon statistical test failed to reject the null hypothesis with a p-value of 0.05, meaning that these four models are not significantly different on the considered 67 datasets. In fact, SAST, ROCKET, HIVE-COTE and TS-CHIEF respectively achieve an average accuracy of  $0.84 \pm 0.12$ ,  $0.88 \pm 0.11$ ,  $.88 \pm 0.11$  and  $0.88 \pm 0.12$ . These average scores clearly show that SAST is comparable to ROCKET and HIVE-COTE in terms of accuracy, and in addition SAST is more interpretable as it is a shapelet based method Ye and Keogh (2009); Bagnall et al. (2017).

#### 4.1.6. Model accuracies per dataset type

The datasets on the UEA & UCR archive are categorized in problem types. Among the 72 datasets we have experimented on, there is 1 electric device problem, 4 ECG problems, 1 High Resolution Melt (HRM) problem, 25 image



**Figure 17:** SAST vs SOTA

problems, 9 motion recognition problems, 1 power consumption problem, 16 sensor reading problems, 7 simulated dataset problems, 6 spectrograph problems and 2 traffic problems.

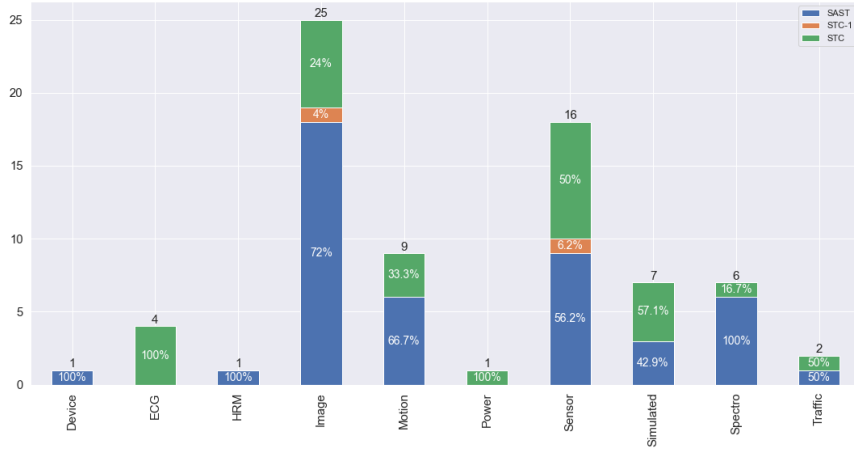
We would like to see the method that is more appropriate for each problem type. However, the observations should be taken wisely since the number of datasets per problem type is relatively small to be representative. We compute these statistics among three groups of methods as in the previous subsections: the first group is SAST, STC-1 and STC; the second group is SAST, ELIS++, FS and LS; and the last group is SAST, ROCKET, TS-CHIEF and HIVE-COTE. For each group and for each problem type, the percentage of times each method achieves the highest accuracy is computed. These statistics are shown as stacked bar plots with problem types on the x-axis and the number of times the highest accuracy is achieved on the y-axis. Above each bar, the number of datasets in the corresponding problem type is displayed. Since more than one model can achieve the highest accuracy on the same dataset, summing the percentage in a bar could be greater than 100% and the value above a bar can be less than the bar height.

Figure 18 shows the percentage of times SAST, STC-1 and STC achieve the highest accuracy per problem type. STC-1 achieves the highest accuracy on the image dataset MiddlePhalanxOutlineAgeGroup and on the sensor dataset Earthquake. STC is the only method that achieves the highest accuracies for ECG and Power. Elsewhere STC seems more appropriate for simulated datasets. SAST tends to be generally the best choice for electric device, HRM, image, motion recognition, sensor and is always the best for spectrograph problems compared to STC approaches.

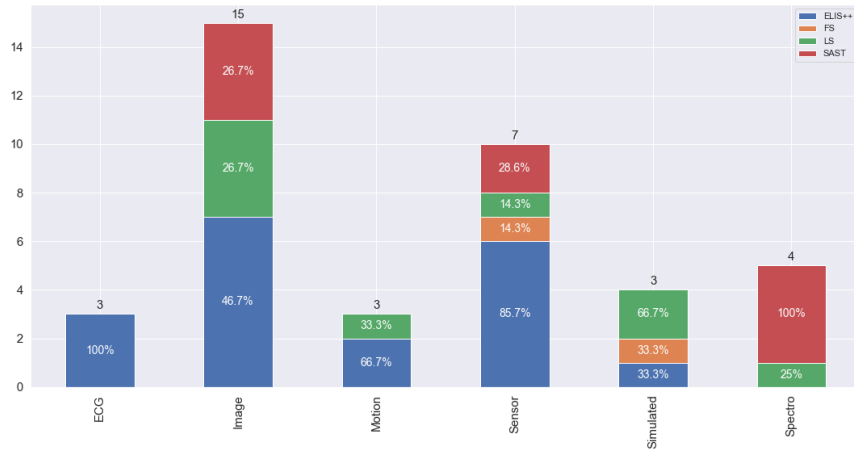
When comparing SAST to other shapelet methods (ELIS++, FS and LS), we can see in Figure 19 that SAST always achieves the highest accuracy on spectrograph problems and is therefore a good choice for this problem type. Elsewhere it achieves the highest accuracy on more than 25% of image and sensor datasets. ELIS++ is more suitable for ECG, image, motion, and sensor problem types. LS is a good choice for simulated datasets.

Finally, Figure 20 reveals that ROCKET, TS-CHIEF and HIVE-COTE win on more datasets than SAST, but with a relatively small difference in accuracy. ROCKET seems to be the most promising method for ECG, motion, sensor, simulated, spectrograph and traffic datasets, while HIVE-COTE is a good choice for image and power datasets. TS-





**Figure 18:** SAST, STC-1 and STC percentage of wins per dataset type



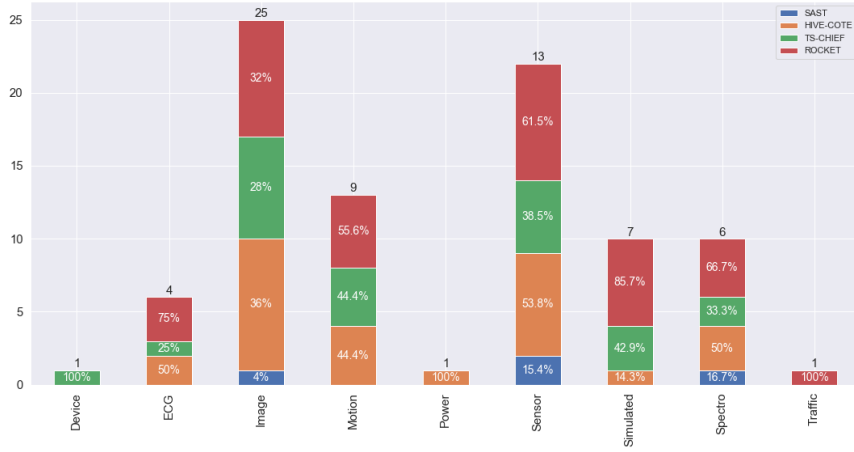
**Figure 19:** SAST, ELIS++, LS and FS wins per dataset type considering the 35 datasets used in ELIS++ paper.

CHIEF is a fair option for device.

Although SAST achieves the highest accuracy than ROCKET, HIVE-COTE and TS-CHIEF on some datasets, it sometimes obtains the same average accuracy as these methods. In fact, Table 2 gives the mean and standard deviation of each model accuracy per dataset type. We can see that SAST achieves the same average accuracy as the state of the art methods on spectrograph and is on average relatively closed on many other data types, except device (but there is only one dataset of that type). This results emphasize the fact that SAST can achieve accuracy equal to or closed to the state of the art method accuracy while offering easier interpretability.

## 4.2. Scalability

The scalability of SAST based models and STC is assessed regarding two criteria: the time series length and the number of time series in the dataset. In this experiment, the time contract is not used for STC, and therefore the full



**Figure 20:** SAST, HIVE-COTE, TS-CHIEF and ROCKET percentage of wins per dataset type

**Table 2**

Average accuracy of each model per problem type

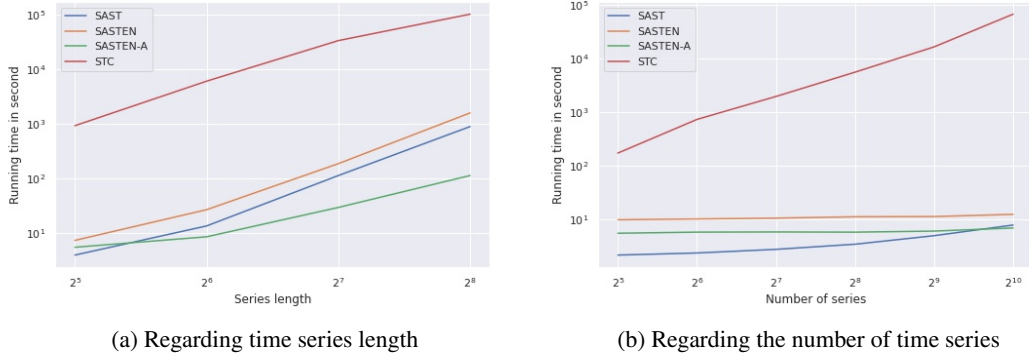
	HIVE-COTE	ROCKET	SAST	TS-CHIEF	Number of datasets
Device	$0.75 \pm 0.0$	$0.73 \pm 0.0$	$0.62 \pm 0.0$	<b><math>0.76 \pm 0.0</math></b>	1
ECG	$0.95 \pm 0.06$	<b><math>0.96 \pm 0.05</math></b>	$0.93 \pm 0.07$	$0.94 \pm 0.07$	4
Image	$0.82 \pm 0.12$	$0.82 \pm 0.12$	$0.78 \pm 0.12$	<b><math>0.83 \pm 0.12</math></b>	25
Motion	<b><math>0.93 \pm 0.09</math></b>	<b><math>0.93 \pm 0.07</math></b>	$0.88 \pm 0.1$	<b><math>0.93 \pm 0.09</math></b>	9
Power	<b><math>1.0 \pm 0.0</math></b>	$0.94 \pm 0.0$	$0.91 \pm 0.0$	$0.99 \pm 0.0$	1
Sensor	$0.89 \pm 0.12$	<b><math>0.9 \pm 0.11</math></b>	$0.85 \pm 0.14$	$0.89 \pm 0.13$	13
Simulated	$0.99 \pm 0.02$	<b><math>1.0 \pm 0.01</math></b>	$0.95 \pm 0.04$	<b><math>1.0 \pm 0.01</math></b>	7
Spectro	<b><math>0.87 \pm 0.11</math></b>	<b><math>0.87 \pm 0.12</math></b>	<b><math>0.87 \pm 0.11</math></b>	<b><math>0.87 \pm 0.11</math></b>	6
Traffic	<b><math>0.98 \pm 0.0</math></b>	<b><math>0.98 \pm 0.0</math></b>	$0.96 \pm 0.0$	$0.97 \pm 0.0$	1
Average	$0.88 \pm 0.11$	$0.88 \pm 0.11$	$0.84 \pm 0.12$	$0.88 \pm 0.12$	67

search is performed. Elsewhere, the training set is considered as the test set during the test phase. For each model, the time taken to fit the model on the training set and then predict the test set is recorded.

#### 4.2.1. Time series length

Here we use the dataset HouseTwenty from the UEA & UCR repository (Anthony Bagnall and Keogh, 2018). It is a binary dataset of electricity usage in houses. The training set has 34 time series and of length 3000 each. We vary the series length starting at 32 and only the first time steps up to the current length are used to train our models. More precisely, we consider the HouseTwenty dataset with time series truncated at length  $2^5$ ,  $2^6$ ,  $2^7$  and finally  $2^8$ . The running time of each model is given in Figure 21a.

For each of the four models, the running time increases with the length of time series in the dataset. However, SAST models are much more scalable than STC, and SASTEN-A is the most scalable of all, since it uses a fixed number of shapelet candidates whatever the length of time series. For SASTEN-A, increasing the length of the time series only increases the computation time of the similarity between time series and shapelet candidates. More specifically, STC



**Figure 21:** Running time (in second) of each model

takes about 1 hour and 40 minutes to train on a dataset of 34 time series of length 64, while SAST, SASTEN and SASTEN-A take about 13 seconds, 27 seconds and 8 seconds respectively. For the same number of time series but now of length 256, STC takes a bit more than a day, while SAST, SASTEN and SASTEN-A take about 14 minutes, 26 minutes and 2 minutes respectively. Therefore, even our slowest method SASTEN is 55 times faster than STC. SASTEN-A and SAST are respectively 1440 times and 102 times faster than STC.

#### 4.2.2. Training set size

The Chinatown dataset is used here. It is a binary dataset with time series of length 24. There are 20 instances in the training set and we use random oversampling to create bigger versions of this dataset. Figure 21b shows the running time of each model.

The running time of each model increases near linearly with the number of time series in the dataset. STC running time starts higher and increases much faster compared to other models. This is not surprising since the training time of shapelet methods is extremely related to the number of shapelet candidates, and the number of shapelet candidates in STC increases with the number of time series while the number of shapelet candidates in a SAST model increases with the number of classes. More precisely, STC takes about 12 minutes on a dataset of 64 time series of length 24, while SAST takes only 2 seconds, SASTEN requires 10 seconds and SASTEN-A needs about 6 seconds. For a dataset with 1024 time series of length 24, SASTEN, SAST and SASTEN-A are respectively about 5000 times, 8000 times and 9000 times faster than STC.

#### 4.3. Interpretability

The predictions of a SAST model trained on a dataset are explained by identifying and visualizing the shapelets that have been learned for that dataset. This is how the explanation of shapelet methods is given in the literature (Ye and Keogh, 2009; Wang, Emonet, Fromont, Malinowski and Tavenard, 2020). This is done using feature importance

analysis (see Proposition 2). Each feature is related to a shapelet candidate extracted from a time series whose class label is known. Shapelet candidates related to the most important features are the top best shapelets. We say that any shapelet candidate is from the class of the time series from which it has been extracted. Therefore, the class label of a time series can be interpreted by looking at the class labels of the shapelet candidates to which it is the most similar. Let us interpret the predictions of SAST-RF and SAST-Ridge trained on the Chinatown dataset. We consider this dataset because it has only two classes and time series of length 24, it is therefore easy to visualize this dataset. However, what we are doing here is applicable to any dataset.

Since SAST-RF uses a tree based classifier, information gain is used as feature importance. With SAST-Ridge, the importance of feature is given by the absolute value of the corresponding learned weight. Although feature importance is computed differently for both models, we show that their predictions are interpretable in the same manner.

Figures 22 and 23 show the top-5 best shapelets plotted on the reference time series for the Chinatown dataset with respect to SAST-Ridge and SAST-RF respectively. The top rows of the figures are the reference time series selected from class 1, while the second rows are the reference

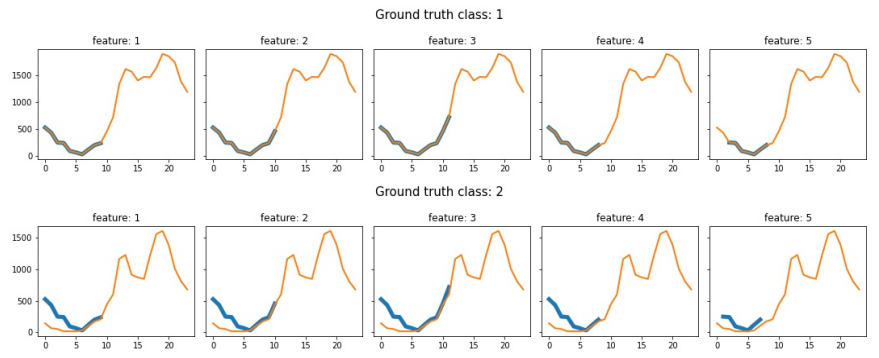


Figure 22: Top 5 shapelets learned by SAST-Ridge on Chinatown.

time series selected from class 2. A perfect match between a shapelet candidate and a reference time series means that the shapelet has been extracted from that reference time series. Hence, the top-5 best shapelets learned by SAST-Ridge are from class 1. The second of the top-5 best shapelets learned by SAST-RF is from class 1, while the four others are from class 2.

In order to predict the class label of a test time series, SAST identifies the most important features similar to the time series. In other words, SAST checks if the time series contains subsequences that are similar to the most important features. Figure 24 shows the matches between the top-5 most important features learned by SAST-Ridge and two randomly selected test time series. We can note that the model correctly predicts the class labels. Since the top-5 shapelets learned by SAST-Ridge are from class 1, there are near perfect matches with the test instance from class 1 (see Figure 24 top). A near perfect match between a subsequence and shapelet candidate means that the subsequence is a variant of that shapelet candidate. No good match is found with the test instance from class 2 (see Figure 24 bottom). Therefore, we have an explanation (i.e the most important features that triggered the predicted class label) of why the first instance is predicted as coming from class 1, while the second one is predicted as coming from class 2.

The same analysis is shown for SAST-RF in Figure 25. Like SAST-Ridge, SAST-RF also predicted the class labels correctly. The first test time series has a near perfect match with the second top best shapelet candidate (see Figure 25 top) which is a shapelet candidate of class 1.

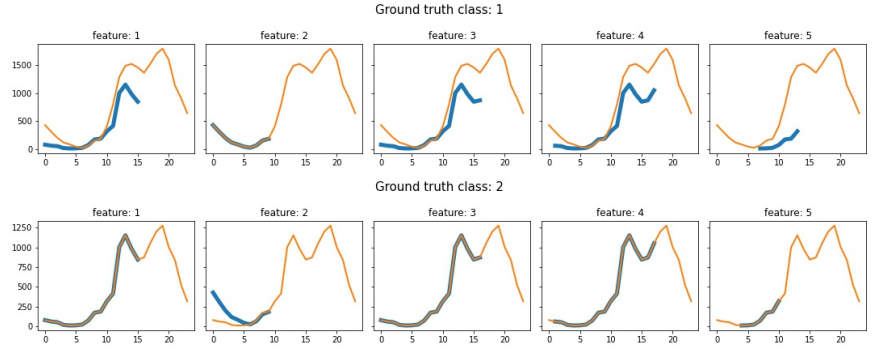


Figure 23: Top 5 shapelets learned by SAST-RF on Chinatown.

The other top best shapelet candidates, which are all from class 2 do not match with the first time series. This explains why the predicted class label for the first time series is class 1 and not class 2. The first, third, fourth and fifth top best shapelet candidates, which are all from class 2 have near perfect matches with the second time series, while the second top best shapelet candidate, which is from class 1 does not match (see Figure 25 bottom). Hence, we can interpret why the class label of the second instance is predicted as class 2 and not class 1.

Therefore, we have proved experimentally in this Section that SAST-RF and SAST-Ridge automatically learn to put more attention on the subsequences that are shapelets for any given dataset. More generally a SAST model automatically learns to put more attention on shapelet

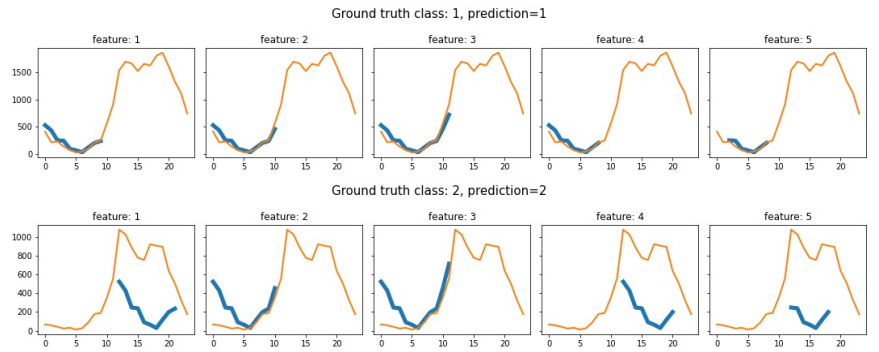
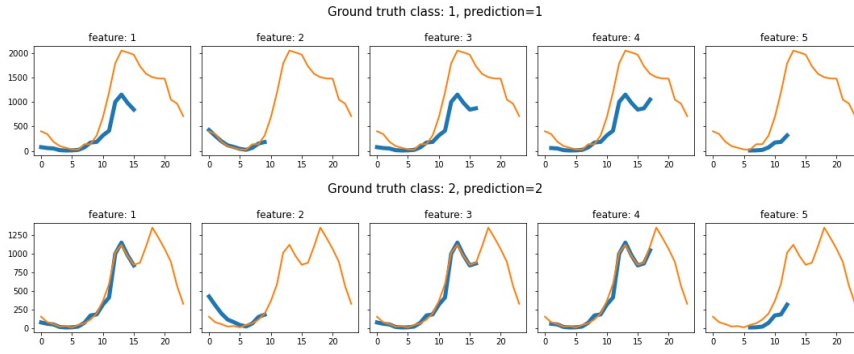


Figure 24: Explanation of SAST-Ridge predictions on two random test instances candidates that are actually shapelets during its training. It also automatically learns to put less attention on the shapelet candidates that are not shapelets.

We also note that the top best shapelets learned by SAST-Ridge and SAST-RF on the Chinatown dataset are the same as the ones selected by STC (see Figure 5).

## 5. Conclusion

In this paper, we have shown that the number of shapelet candidates in a shapelet algorithm can be reduced considerably without losing accuracy. We have also shown that it is not always necessary to learn shapelets beforehand of classification. We introduced the Scalable and Accurate Subsequence Transform (SAST) algorithm which is in-



**Figure 25:** Explanation of SAST-RF predictions on two test instances

terpretable, accurate and a more scalable alternative to the Shapelet Transform algorithm. Furthermore, SAST is comparable in terms of accuracy to the state of the art methods ROCKET, HIVE-COTE and TS-CHIEF, especially for the spectrograph dataset type, while offering easier interpretability. Our experiments have revealed that a good trade-off between accuracy and scalability can be found by ensembling different SAST models, each one focusing on different shapelet candidates. We have also introduced the *core shapelet recognition* task which consists of learning a shapelets model using only few variants of each shapelet candidate. SAST achieves this task accurately and we hope future shapelet methods will follow the idea presented in this paper. We plan to do many improvements into the SAST algorithm in the future. Particularly, distance computation could be speed up using lower bounding and early abandoning techniques. Different variants of the same shapelet can be present in the same time series, therefore similar subsequences can be pruned in order to further reduce the number of shapelet candidates. Elsewhere, we plan to replace the shapelet module in HIVE-COTE with a SAST method with the intent of decreasing its computation time while keeping a high accuracy performance. We are also planing to explore how core shapelet recognition can be applied in TS-CHIEF in order to take shapelet features into account. In parallel, we will run our experiment on the remaining datasets on the UEA & UCR archive. Although we focused on TSC in this paper, we hope that the same idea can be used in the near future to increase the scalability of shapelet-based time series clustering (Siyu Fotso et al., 2020).

## A. Effect of the used classifier on SAST's accuracy

Throughout this work we have considered Random Forest and Ridge, a linear and a tree-based classifiers. Combining SAST with Ridge significantly achieves the best performance overall (11a). We have chosen these classifiers because they are known to be “white box” models, making our approach explainable by-design. Our approach's accuracy and explainability depend on the classifier used. The parameters of the classifier also affect the result. The most suitable classifier (and its parameters) should be selected regarding the specificities of the application, the desired

tradeoff between accuracy and explainability, and the computation resource.

In order to make this statement more explicit, we executed our method using 5 different classifiers (Naive Bayes, Nearest Neighbor, SVM, Random Forest and Ridge) on 3 datasets (Chinatown, PenDigits and Smoothsubspace) which are from different types (resp. Traffic, Motion and

Table 3: SAST's accuracy using different classifiers

Model	Chinatown	PenDigits	Smoothsubspace
SAST-KNN	$0.91 \pm 0.06$	$0.93 \pm 0.00$	$0.91 \pm 0.01$
SAST-NB	$0.93 \pm 0.02$	$0.94 \pm 0.02$	$0.93 \pm 0.03$
SAST-SVM	$0.96 \pm 0.00$	$0.95 \pm 0.01$	$0.95 \pm 0.01$
SAST-Ridge	$0.96 \pm 0.00$	$0.96 \pm 0.01$	$0.96 \pm 0.00$
SAST-RF	$0.97 \pm 0.00$	$0.97 \pm 0.00$	$0.98 \pm 0.00$

Simulated), different size, different time series length and different number of classes. The results listed in the table 3 confirm that using the right classifier has a significant impact on the overall approach. In fact, Nearest Neighbor and Naïve Bayes achieve lower accuracy than the remaining classifiers.

## B. Scalability of SAST, SASTEN and SASTEN-A regarding the dataset size

Figure 26 shows a zoom in Figure 21b. We can now clearly see that SASTEN is slower than SAST and SASTEN-A whatever the number of time series in the dataset. SASTEN-A running time is quite linear because the shapelet space is constant and only the transformation time increases with the number of series.

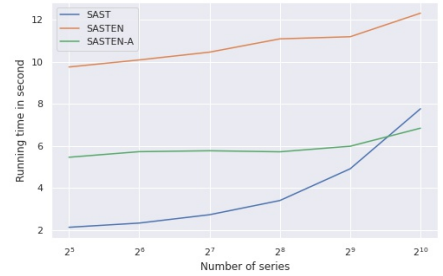


Figure 26: SAST's Running time

## C. Models performance on each dataset

Table 4 gives the average accuracy obtained by each STC-k models, STC and SAST on each dataset. For each dataset, each model is run 5 times and accuracy mean and standard deviation on the test set are recorded. The type of each dataset is written in parenthesis after the name. The last row in the table gives the mean and standard deviation of each method accuracy on the 72 datasets. Datasets marked with a star are those used for the experiment whose results are presented in Section 4.1.2 and datasets marked with a *plus* sign are those used for the experiment whose results are given in Section 4.1.4.

Table 4: Accuracy of models on 72 UEA &amp; UCR datasets (average over 5 runs). The numbers are rounded at 2 decimals.

	STC-1	STC-0.25	STC-0.5	STC-0.75	STC	SAST
Adiac (Image)	0.29 $\pm$ 0.06	0.32 $\pm$ 0.03	0.34 $\pm$ 0.04	0.34 $\pm$ 0.02	0.48 $\pm$ 0.04	<b>0.68 <math>\pm</math> 0.0</b>
ArrowHead+ (Image)	0.63 $\pm$ 0.06	0.71 $\pm$ 0.08	0.74 $\pm$ 0.03	0.72 $\pm$ 0.04	0.75 $\pm$ 0.03	<b>0.77 <math>\pm</math> 0.02</b>
BME* (Simulated)	0.65 $\pm$ 0.1	0.68 $\pm$ 0.05	0.86 $\pm$ 0.05	<b>0.95 <math>\pm</math> 0.02</b>	0.87 $\pm$ 0.03	0.87 $\pm$ 0.02
Beef+ (Spectro)	0.44 $\pm$ 0.05	0.47 $\pm$ 0.09	0.55 $\pm$ 0.09	0.65 $\pm$ 0.06	0.71 $\pm$ 0.09	<b>0.8 <math>\pm</math> 0.02</b>
BeetleFly+ (Image)	0.74 $\pm$ 0.06	0.79 $\pm$ 0.09	0.78 $\pm$ 0.09	0.77 $\pm$ 0.07	0.78 $\pm$ 0.05	<b>0.8 <math>\pm</math> 0.03</b>
BirdChicken+ (Image)	0.76 $\pm$ 0.1	0.76 $\pm$ 0.1	<b>0.91 <math>\pm</math> 0.04</b>	0.83 $\pm$ 0.14	0.86 $\pm$ 0.1	0.76 $\pm$ 0.1
CBF*+ (Simulated)	0.88 $\pm$ 0.09	0.95 $\pm$ 0.01	0.96 $\pm$ 0.01	0.96 $\pm$ 0.01	0.95 $\pm$ 0.01	<b>0.98 <math>\pm</math> 0.01</b>
Car+ (Sensor)	0.66 $\pm$ 0.04	0.75 $\pm$ 0.02	0.74 $\pm$ 0.02	0.76 $\pm$ 0.03	0.77 $\pm$ 0.06	<b>0.88 <math>\pm</math> 0.01</b>
Chinatown* (Traffic)	0.91 $\pm$ 0.08	0.95 $\pm$ 0.02	0.95 $\pm$ 0.03	0.96 $\pm$ 0.01	<b>0.97 <math>\pm</math> 0.01</b>	0.96 $\pm$ 0.01
ChlorineConcentration* (Sensor)	0.54 $\pm$ 0.02	0.54 $\pm$ 0.01	0.54 $\pm$ 0.01	0.56 $\pm$ 0.0	0.56 $\pm$ 0.0	<b>0.75 <math>\pm</math> 0.04</b>
Coffee+ (Spectro)	0.96 $\pm$ 0.03	0.99 $\pm$ 0.02	0.99 $\pm$ 0.03	<b>1.0 <math>\pm</math> 0.0</b>	<b>1.0 <math>\pm</math> 0.0</b>	<b>1.0 <math>\pm</math> 0.0</b>
CricketX (Motion)	0.38 $\pm$ 0.04	0.35 $\pm$ 0.03	0.35 $\pm$ 0.01	0.38 $\pm$ 0.03	0.66 $\pm$ 0.02	<b>0.77 <math>\pm</math> 0.01</b>
CricketY (Motion)	0.41 $\pm$ 0.04	0.36 $\pm$ 0.04	0.35 $\pm$ 0.03	0.42 $\pm$ 0.05	0.64 $\pm$ 0.02	<b>0.74 <math>\pm</math> 0.01</b>
CricketZ (Motion)	0.41 $\pm$ 0.06	0.4 $\pm$ 0.06	0.4 $\pm$ 0.03	0.38 $\pm$ 0.03	0.69 $\pm$ 0.01	<b>0.77 <math>\pm</math> 0.01</b>
Crop* (Image)	0.08 $\pm$ 0.0	0.08 $\pm$ 0.0	0.08 $\pm$ 0.0	0.08 $\pm$ 0.0	0.08 $\pm$ 0.0	<b>0.73 <math>\pm</math> 0.0</b>
DiatomSizeReduction+ (Image)	0.9 $\pm$ 0.03	0.87 $\pm$ 0.06	0.91 $\pm$ 0.04	0.93 $\pm$ 0.02	0.94 $\pm$ 0.04	<b>0.97 <math>\pm</math> 0.0</b>
DistalPhalanxOutlineAgeGroup*+ (Image)	0.71 $\pm$ 0.02	<b>0.77 <math>\pm</math> 0.01</b>	0.76 $\pm$ 0.01	<b>0.77 <math>\pm</math> 0.01</b>	<b>0.77 <math>\pm</math> 0.01</b>	0.76 $\pm$ 0.02
DistalPhalanxOutlineCorrect*+ (Image)	0.66 $\pm$ 0.01	0.68 $\pm$ 0.02	0.69 $\pm$ 0.01	0.69 $\pm$ 0.03	0.71 $\pm$ 0.01	<b>0.74 <math>\pm</math> 0.01</b>
DistalPhalanxTW*+ (Image)	0.67 $\pm$ 0.01	<b>0.68 <math>\pm</math> 0.01</b>	<b>0.68 <math>\pm</math> 0.01</b>	<b>0.68 <math>\pm</math> 0.01</b>	<b>0.68 <math>\pm</math> 0.0</b>	0.67 $\pm$ 0.02
DodgerLoopDay (Sensor)	0.43 $\pm$ 0.03	0.47 $\pm$ 0.02	0.54 $\pm$ 0.02	0.52 $\pm$ 0.04	0.47 $\pm$ 0.04	<b>0.61 <math>\pm</math> 0.04</b>
DodgerLoopGame (Sensor)	0.61 $\pm$ 0.09	0.68 $\pm$ 0.11	0.8 $\pm$ 0.02	0.77 $\pm$ 0.03	0.81 $\pm$ 0.03	<b>0.9 <math>\pm</math> 0.02</b>
DodgerLoopWeekend (Sensor)	0.94 $\pm$ 0.05	0.97 $\pm$ 0.0	0.96 $\pm$ 0.0	0.97 $\pm$ 0.01	0.97 $\pm$ 0.01	<b>0.98 <math>\pm</math> 0.01</b>
ECG200*+ (ECG)	0.74 $\pm$ 0.07	<b>0.85 <math>\pm</math> 0.02</b>	0.84 $\pm$ 0.01	<b>0.85 <math>\pm</math> 0.01</b>	0.84 $\pm$ 0.0	0.84 $\pm$ 0.03
ECG5000* (ECG)	0.91 $\pm$ 0.01	0.93 $\pm$ 0.01	0.93 $\pm$ 0.01	0.93 $\pm$ 0.01	<b>0.94 <math>\pm</math> 0.0</b>	<b>0.94 <math>\pm</math> 0.0</b>
ECGFiveDays*+ (ECG)	0.92 $\pm$ 0.05	<b>1.0 <math>\pm</math> 0.0</b>	<b>1.0 <math>\pm</math> 0.0</b>	<b>1.0 <math>\pm</math> 0.0</b>	<b>1.0 <math>\pm</math> 0.0</b>	<b>1.0 <math>\pm</math> 0.0</b>
Earthquakes+ (Sensor)	<b>0.75 <math>\pm</math> 0.0</b>	<b>0.75 <math>\pm</math> 0.0</b>	<b>0.75 <math>\pm</math> 0.0</b>	<b>0.75 <math>\pm</math> 0.0</b>	<b>0.75 <math>\pm</math> 0.0</b>	0.68 $\pm$ 0.04
ElectricDevices* (Device)	0.32 $\pm$ 0.04	0.32 $\pm$ 0.07	0.31 $\pm$ 0.05	0.34 $\pm$ 0.05	0.32 $\pm$ 0.03	<b>0.62 <math>\pm</math> 0.01</b>
FaceAll* (Image)	0.43 $\pm$ 0.04	0.46 $\pm$ 0.03	0.44 $\pm$ 0.04	0.47 $\pm$ 0.03	0.74 $\pm$ 0.01	<b>0.78 <math>\pm</math> 0.01</b>
FaceFour+ (Image)	0.96 $\pm$ 0.06	0.95 $\pm$ 0.03	0.96 $\pm$ 0.06	0.99 $\pm$ 0.01	0.99 $\pm$ 0.01	<b>1.0 <math>\pm</math> 0.01</b>
FacesUCR* (Image)	0.71 $\pm$ 0.04	0.88 $\pm$ 0.01	0.92 $\pm$ 0.01	0.93 $\pm$ 0.01	0.94 $\pm$ 0.0	<b>0.95 <math>\pm</math> 0.0</b>
FiftyWords (Image)	0.33 $\pm$ 0.06	0.29 $\pm$ 0.03	0.31 $\pm$ 0.02	0.31 $\pm$ 0.03	0.59 $\pm$ 0.01	<b>0.77 <math>\pm</math> 0.0</b>
FreezerRegularTrain (Sensor)	0.95 $\pm$ 0.09	0.97 $\pm$ 0.03	0.96 $\pm$ 0.03	0.96 $\pm$ 0.03	0.97 $\pm$ 0.01	<b>0.98 <math>\pm</math> 0.01</b>
FreezerSmallTrain (Sensor)	<b>0.93 <math>\pm</math> 0.1</b>	0.91 $\pm$ 0.13	0.86 $\pm$ 0.17	0.87 $\pm$ 0.08	<b>0.93 <math>\pm</math> 0.03</b>	0.73 $\pm$ 0.01
Fungi (HRM)	0.0 $\pm$ 0.0	0.0 $\pm$ 0.0	0.0 $\pm$ 0.0	0.0 $\pm$ 0.0	0.0 $\pm$ 0.0	<b>0.9 <math>\pm</math> 0.0</b>
GunPoint*+ (Motion)	0.79 $\pm$ 0.05	0.94 $\pm$ 0.06	<b>0.97 <math>\pm</math> 0.02</b>	0.96 $\pm$ 0.03	<b>0.97 <math>\pm</math> 0.02</b>	<b>0.97 <math>\pm</math> 0.02</b>
GunPointAgeSpan* (Motion)	0.84 $\pm$ 0.03	0.96 $\pm$ 0.01	0.97 $\pm$ 0.0	<b>0.98 <math>\pm</math> 0.0</b>	0.97 $\pm$ 0.0	0.97 $\pm$ 0.0

Continued on next page



Table 4: Accuracy of models on 72 UEA &amp; UCR datasets (average over 5 runs). The numbers are rounded at 2 decimals.

	STC-1	STC-0.25	STC-0.5	STC-0.75	STC	SAST
GunPointMaleVersusFemale* (Motion)	0.89 ± 0.05	0.97 ± 0.01	0.97 ± 0.01	0.98 ± 0.01	0.97 ± 0.01	<b>0.99 ± 0.01</b>
GunPointOldVersusYoung* (Motion)	0.75 ± 0.03	0.93 ± 0.02	0.95 ± 0.01	<b>0.96 ± 0.01</b>	0.95 ± 0.01	<b>0.96 ± 0.02</b>
Ham+ (Spectro)	0.62 ± 0.06	<b>0.72 ± 0.02</b>	<b>0.72 ± 0.04</b>	0.7 ± 0.01	0.7 ± 0.02	0.71 ± 0.03
Herring+ (Image)	0.59 ± 0.07	0.57 ± 0.04	0.58 ± 0.02	0.57 ± 0.06	<b>0.62 ± 0.03</b>	0.6 ± 0.04
InsectWingbeatSound (Sensor)	0.53 ± 0.02	0.56 ± 0.01	0.56 ± 0.01	0.54 ± 0.01	<b>0.62 ± 0.0</b>	0.56 ± 0.01
ItalyPowerDemand*+ (Sensor)	0.91 ± 0.06	<b>0.96 ± 0.0</b>	<b>0.96 ± 0.01</b>	<b>0.96 ± 0.0</b>	<b>0.96 ± 0.0</b>	<b>0.96 ± 0.01</b>
Meat (Spectro)	0.63 ± 0.1	0.84 ± 0.04	0.77 ± 0.02	0.87 ± 0.03	0.87 ± 0.06	<b>0.92 ± 0.02</b>
MedicalImages* (Image)	0.54 ± 0.02	0.59 ± 0.03	0.56 ± 0.01	0.58 ± 0.02	0.66 ± 0.01	<b>0.68 ± 0.01</b>
MelbournePedestrian (Traffic)	0.3 ± 0.04	0.3 ± 0.07	0.3 ± 0.03	0.28 ± 0.01	0.72 ± 0.02	<b>0.87 ± 0.0</b>
MiddlePhalanxOutlineAgeGroup*+ (Image)	<b>0.62 ± 0.02</b>	0.6 ± 0.01	0.6 ± 0.02	0.61 ± 0.02	0.61 ± 0.02	0.53 ± 0.02
MiddlePhalanxOutlineCorrect*+ (Image)	0.57 ± 0.0	0.65 ± 0.05	0.66 ± 0.04	0.62 ± 0.04	0.65 ± 0.06	<b>0.83 ± 0.01</b>
MiddlePhalanxTW*+ (Image)	0.57 ± 0.02	0.58 ± 0.02	0.56 ± 0.01	0.58 ± 0.01	<b>0.59 ± 0.02</b>	0.56 ± 0.02
MoteStrain*+ (Sensor)	0.8 ± 0.06	0.79 ± 0.05	0.85 ± 0.02	0.87 ± 0.02	<b>0.89 ± 0.01</b>	0.85 ± 0.03
PhalangesOutlinesCorrect* (Image)	0.64 ± 0.01	0.64 ± 0.01	0.64 ± 0.0	0.64 ± 0.0	0.65 ± 0.01	<b>0.78 ± 0.01</b>
Plane*+ (Sensor)	0.97 ± 0.02	0.99 ± 0.0	<b>1.0 ± 0.0</b>	<b>1.0 ± 0.0</b>	<b>1.0 ± 0.0</b>	<b>1.0 ± 0.0</b>
PowerCons* (Power)	0.76 ± 0.03	0.93 ± 0.02	0.92 ± 0.02	0.93 ± 0.02	<b>0.94 ± 0.02</b>	0.91 ± 0.02
ProximalPhalanxOutlineAgeGroup*+ (Image)	0.85 ± 0.01	0.85 ± 0.01	<b>0.86 ± 0.01</b>	0.85 ± 0.01	<b>0.86 ± 0.01</b>	0.85 ± 0.0
ProximalPhalanxOutlineCorrect* (Image)	0.71 ± 0.03	0.76 ± 0.03	0.75 ± 0.04	0.76 ± 0.01	0.82 ± 0.02	<b>0.87 ± 0.01</b>
ProximalPhalanxTW*+ (Image)	0.77 ± 0.02	<b>0.78 ± 0.01</b>	<b>0.78 ± 0.01</b>	<b>0.78 ± 0.0</b>	0.77 ± 0.01	<b>0.78 ± 0.01</b>
ShapeletSim+ (Simulated)	0.82 ± 0.15	0.92 ± 0.06	0.98 ± 0.01	0.98 ± 0.02	<b>1.0 ± 0.0</b>	0.96 ± 0.01
SmoothSubspace* (Simulated)	0.68 ± 0.05	0.91 ± 0.01	0.94 ± 0.01	<b>0.95 ± 0.01</b>	<b>0.95 ± 0.01</b>	0.91 ± 0.02
SonyAIBORobotSurface1*+ (Sensor)	0.7 ± 0.14	<b>0.82 ± 0.03</b>	0.81 ± 0.02	0.78 ± 0.05	0.79 ± 0.04	0.76 ± 0.05
SonyAIBORobotSurface2*+ (Sensor)	0.75 ± 0.05	0.84 ± 0.03	0.86 ± 0.05	0.83 ± 0.03	<b>0.88 ± 0.01</b>	0.85 ± 0.04
Strawberry (Spectro)	0.75 ± 0.05	0.81 ± 0.02	0.82 ± 0.03	0.83 ± 0.02	0.91 ± 0.03	<b>0.97 ± 0.01</b>
SwedishLeaf* (Image)	0.58 ± 0.05	0.61 ± 0.02	0.59 ± 0.07	0.6 ± 0.09	0.83 ± 0.02	<b>0.88 ± 0.02</b>
Symbols+ (Image)	0.91 ± 0.05	0.63 ± 0.07	0.93 ± 0.02	0.93 ± 0.01	<b>0.95 ± 0.01</b>	<b>0.95 ± 0.0</b>
SyntheticControl*+ (Simulated)	0.87 ± 0.02	0.97 ± 0.01	0.97 ± 0.01	0.97 ± 0.01	<b>0.98 ± 0.0</b>	<b>0.98 ± 0.0</b>
ToeSegmentation1+ (Motion)	0.88 ± 0.04	0.91 ± 0.02	0.94 ± 0.01	<b>0.95 ± 0.0</b>	<b>0.95 ± 0.0</b>	0.88 ± 0.04
ToeSegmentation2+ (Motion)	0.84 ± 0.08	<b>0.9 ± 0.04</b>	0.87 ± 0.02	0.89 ± 0.02	0.88 ± 0.03	0.88 ± 0.03
Trace (Sensor)	0.94 ± 0.07	0.99 ± 0.01	<b>1.0 ± 0.0</b>	<b>1.0 ± 0.0</b>	0.99 ± 0.0	<b>1.0 ± 0.0</b>
TwoLeadECG*+ (ECG)	0.93 ± 0.04	0.95 ± 0.02	0.98 ± 0.03	<b>0.99 ± 0.01</b>	0.98 ± 0.01	0.96 ± 0.03
TwoPatterns* (Simulated)	0.56 ± 0.07	0.61 ± 0.05	0.64 ± 0.07	0.64 ± 0.03	0.81 ± 0.03	<b>0.99 ± 0.0</b>
UMD* (Simulated)	0.83 ± 0.07	0.93 ± 0.03	0.97 ± 0.03	0.96 ± 0.03	<b>0.98 ± 0.01</b>	<b>0.98 ± 0.01</b>
Wafer* (Sensor)	0.99 ± 0.01	<b>1.0 ± 0.0</b>	<b>1.0 ± 0.01</b>	<b>1.0 ± 0.0</b>	<b>1.0 ± 0.0</b>	<b>1.0 ± 0.0</b>
Wine+ (Spectro)	0.54 ± 0.06	0.71 ± 0.1	0.72 ± 0.07	0.8 ± 0.04	0.8 ± 0.06	<b>0.85 ± 0.06</b>
WordSynonyms (Image)	0.4 ± 0.01	0.45 ± 0.02	0.47 ± 0.02	0.44 ± 0.01	0.56 ± 0.01	<b>0.7 ± 0.01</b>

Continued on next page

Table 4: Accuracy of models on 72 UEA & UCR datasets (average over 5 runs). The numbers are rounded at 2 decimals.

	STC-1	STC-0.25	STC-0.5	STC-0.75	STC	SAST
Average	$0.68 \pm 0.22$	$0.73 \pm 0.24$	$0.74 \pm 0.24$	$0.75 \pm 0.24$	$0.79 \pm 0.2$	<b><math>0.84 \pm 0.13</math></b>

## References

- Anthony Bagnall, Jason Lines, W.V., Keogh, E., 2018. The UEA & UCR time series classification repository. [www.timeseriesclassification.com](http://www.timeseriesclassification.com) (visited on Oct. 2020).
- Bagnall, A., Janacek, G., 2014. A run length transformation for discriminating between auto regressive time series. *Journal of classification* 31, 154–178.
- Bagnall, A., Lines, J., Bostrom, A., Large, J., Keogh, E., 2017. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery* 31, 606–660.
- Bostrom, A., Bagnall, A., 2015. Binary shapelet transform for multiclass time series classification, in: *International Conference on Big Data Analytics and Knowledge Discovery*, Springer. pp. 257–269.
- Breiman, L., 1996. Bagging predictors. *Machine Learning* 24, 123–140.
- Cabello, N., Naghizade, E., Qi, J., Kulik, L., 2020. Fast and accurate time series classification through supervised interval search, in: *2020 IEEE International Conference on Data Mining (ICDM)*, pp. 948–953.
- Dash, M., Liu, H., 1997. Feature selection for classification. *Intelligent Data Analysis* 1, 131–156.
- Dempster, A., Petitjean, F., Webb, G.I., 2020. Rocket: Exceptionally fast and accurate time series classification using random convolutional kernels. *Data Mining and Knowledge Discovery* , 1–42.
- Deng, H., Runger, G., Tuv, E., Vladimir, M., 2013. A time series forest for classification and feature extraction. *Information Sciences* 239, 142–153.
- DiCarlo, J.J., Zoccolan, D., Rust, N.C., 2012. How does the brain solve visual object recognition? *Neuron* 73, 415–434.
- Fang, Z., Wang, P., Wang, W., 2018. Efficient learning interpretable shapelets for accurate time series classification, in: *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, pp. 497–508.
- Fawaz, H.I., Forestier, G., Weber, J., Idoumghar, L., Muller, P.A., 2019. Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery* 33, 917–963.
- Fawaz, H.I., Lucas, B., Forestier, G., Pelletier, C., Schmidt, D.F., Weber, J., Webb, G.I., Idoumghar, L., Muller, P.A., Petitjean, F., 2020. Inception-time: Finding alexnet for time series classification. *Data Mining and Knowledge Discovery* 34, 1936–1962.
- Flynn, M., Large, J., Bagnall, T., 2019. The contract random interval spectral ensemble (c-rise): the effect of contracting a classifier on accuracy, in: *International Conference on Hybrid Artificial Intelligence Systems*, Springer. pp. 381–392.
- Grabocka, J., Schilling, N., Wistuba, M., Schmidt-Thieme, L., 2014. Learning time-series shapelets, in: *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 392–401.
- Gupta, A., Gupta, H.P., Biswas, B., Dutta, T., 2020. An unseen fault classification approach for smart appliances using ongoing multivariate time series. *IEEE Transactions on Industrial Informatics* , 1–1.

- Heeger, D., 2002-2014. Center for neural science, lecture notes: Perception (undergraduate). URL: <http://www.cns.nyu.edu/~david/courses/perception/lecturenotes/recognition/recognition.html>. Last visited on May 2021.
- Hills, J., Lines, J., Baranauskas, E., Mapp, J., Bagnall, A., 2014. Classification of time series by shapelet transformation. *Data Mining and Knowledge Discovery* 28, 851–881.
- Ismail, A.A., Gunady, M., Corrada Bravo, H., Feizi, S., 2020. Benchmarking deep learning interpretability in time series predictions, in: *Advances in Neural Information Processing Systems*, pp. 6441–6452.
- Karlsson, I., Papapetrou, P., Boström, H., 2016. Generalized random shapelet forests. *Data Mining and Knowledge Discovery* 30, 1053–1085.
- Large, J., Bagnall, A., Malinowski, S., Tavenard, R., 2019. On time series classification with dictionary-based classifiers. *Intelligent Data Analysis* 23, 1073–1089.
- Lines, J., Taylor, S., Bagnall, A., 2018. Time series classification with hive-cote: The hierarchical vote collective of transformation-based ensembles. *ACM Transactions on Knowledge Discovery from Data* 12.
- Löning, M., Bagnall, A., Ganesh, S., Kazakov, V., Lines, J., Király, F.J., 2019. sktime: A Unified Interface for Machine Learning with Time Series, in: *Workshop on Systems for ML at NeurIPS 2019*.
- Lubba, C.H., Sethi, S.S., Knaute, P., Schultz, S.R., Fulcher, B.D., Jones, N.S., 2019. catch22: Canonical time-series characteristics. *Data Mining and Knowledge Discovery* 33, 1821–1852.
- Middlehurst, M., Large, J., Bagnall, A., 2020. The canonical interval forest (cif) classifier for time series classification. 2020 IEEE International Conference on Big Data (Big Data) doi:10.1109/bigdata50022.2020.9378424.
- Middlehurst, M., Vickers, W., Bagnall, A., 2019. Scalable dictionary classifiers for time series classification, in: *International Conference on Intelligent Data Engineering and Automated Learning*, Springer. pp. 11–19.
- Molnar, C., 2022. *Interpretable Machine Learning*. 2 ed. URL: <https://christophm.github.io/interpretable-ml-book>.
- Moss, A., 2018. Improved photometric classification of supernovae using deep learning. *arXiv:1810.06441*.
- Murdoch, W.J., Singh, C., Kumbier, K., Abbasi-Asl, R., Yu, B., 2019. Definitions, methods, and applications in interpretable machine learning. *Proceedings of the National Academy of Sciences* 116, 22071–22080.
- Papagiannopoulou, C., Parchen, R., Waegeman, W., 2019. Investigating time series classification techniques for rapid pathogen identification with single-cell maldi-tof mass spectrum data, in: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer. pp. 416–431.
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E., 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12, 2825–2830.
- Rakthanmanon, T., Keogh, E., 2013. Fast shapelets: A scalable algorithm for discovering time series shapelets, in: *proceedings of the 2013 SIAM International Conference on Data Mining*, SIAM. pp. 668–676.
- Renard, X., Rifqi, M., Erray, W., Detyniecki, M., 2015. Random-shapelet: an algorithm for fast shapelet discovery, in: *2015 IEEE international conference on Data Science and Advanced Analytics (DSAA)*, pp. 1–10.
- Ribeiro, M.T., Singh, S., Guestrin, C., 2016. " why should i trust you?" explaining the predictions of any classifier, in: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1135–1144.
- Samek, W., Montavon, G., Lapuschkin, S., Anders, C.J., Müller, K.R., 2020. Toward interpretable machine learning: Transparent deep neural networks and beyond. *arXiv:arXiv:2003.07631*.
- Sanchez, E.H., Serrurier, M., Ortner, M., 2019. Learning disentangled representations of satellite image time series, in: *Joint European Conference*

- on Machine Learning and Knowledge Discovery in Databases, Springer. pp. 306–321.
- Schäfer, P., 2015. The boss is concerned with time series classification in the presence of noise. *Data Mining and Knowledge Discovery* 29, 1505–1530.
- Schäfer, P., Leser, U., 2017. Fast and accurate time series classification with weasel, in: *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pp. 637–646.
- Shifaz, A., Pelletier, C., Petitjean, F., Webb, G.I., 2020. Ts-chief: A scalable and accurate forest algorithm for time series classification. *Data Mining and Knowledge Discovery* , 1–34.
- Siyou Fotso, V.S., Mephu Nguifo, E., Vaslin, P., 2020. Frobenius correlation based u-shapelets discovery for time series clustering. *Pattern Recognition* , 107301.
- Smilkov, D., Thorat, N., Kim, B., Viégas, F., Wattenberg, M., 2017. Smoothgrad: removing noise by adding noise. *arXiv:arXiv:1706.03825*.
- Tan, C.W., Petitjean, F., Webb, G.I., 2020. Fastee: Fast ensembles of elastic distances for time series classification. *Data Mining and Knowledge Discovery* 34, 231–272.
- Wang, Y., Emonet, R., Fromont, E., Malinowski, S., Tavenard, R., 2020. Adversarial regularization for explainable-by-design time series classification, in: *2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 1079–1087.
- Wang, Z., Yan, W., Oates, T., 2017. Time series classification from scratch with deep neural networks: A strong baseline, in: *2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 1578–1585.
- Wistuba, M., Grabocka, J., Schmidt-Thieme, L., 2015. Ultra-fast shapelets for time series classification. *arXiv:1503.05018* .
- Wolpert, D.H., Macready, W.G., 1997. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation* 1, 67–82.
- Ye, L., Keogh, E., 2009. Time series shapelets: a new primitive for data mining, in: *Proceedings of the 15th ACM SIGKDD international conference on Knowledge Discovery and Data Mining*, pp. 947–956.
- Zhang, H., Wang, P., Fang, Z., Wang, Z., Wang, W., 2021. Elis++: a shapelet learning approach for accurate and efficient time series classification. *World Wide Web* 24, 511–539.

**Michael Franklin MBOUOPDA** is a PhD student the University Clermont Auvergne (UCA), France. He is currently working on the classification of uncertain time series. During, his Master studies, he worked on named entities recognition for low resource language and published a paper with his supervisor. After, his PhD, he would like to continuous his career as a pro researcher.

**Engelbert MEPHU NGUIFO** is a full professor of computer science at University Clermont Auvergne (UCA), France. At UCA, he has served as Vice-President of the mathematical and computer science department (2012-2016). He is leading research on complex data mining and machine learning in the joined University-CNRS laboratory LIMOS (Laboratory of Computer Science, Modelisation and Optimization), where he is co-chair of the Information and Communication Systems research group. His research interests include formal concept analysis, artificial intelligence, machine learning, complex data mining, pattern recognition, bioinformatics, big data, and knowledge representation. Mephu Nguifo has a Ph.D. in computer science from the University of Montpellier. He has published more than hundred technical papers in majors journals and conferences, and was advisor of more than ten PhD students currently in academic position. He was principal investigator of several international research project grants. He is member of the steering committee of international conference on Concept Lattices and their Applications (CLA), and has served as PC Chair of CLA in 2006, as well as French conference on Machine Learning (CAp) in 2010. He has co-organized several workshops of majors conferences (ECAI, IJCAI, ECML/PKDD, VLDB, ICML) on several research topics (Bioinformatics and AI, Evolving Graphs, Concept Lattices). He is an ACM Senior member, and has been a member of the executive board of the French Association on Artificial Intelligence (AFIA). He is also member of the executive board of the French CNRS research group on Artificial Intelligence (GDR IA).