applied sciences

MDPI

Article

SCALE-BOSS-MR: Scalable Time Series Classification Using Multiple Symbolic Representations †

Apostolos Glenis * and George A. Vouros

Department of Digital Systems, University of Piraeus, 185 34 Piraeus, Greece; georgev@unipi.gr

* Correspondence: apostglen46@gmail.com

† This is the extended version of the conference paper "SCALE-BOSS: A framework for scalable time-series classification using symbolic representations" published in the year 2022 as a proceedings of the 12th Hellenic Conference on Artificial Intelligence.

Abstract: Time-Series-Classification (TSC) is an important machine learning task for many branches of science. Symbolic representations of time series, especially Symbolic Fourier Approximation (SFA), have been proven very effective for this task, given their abilities to reduce noise. In this paper, we improve upon SCALE-BOSS using multiple symbolic representations of time series. More specifically, the proposed SCALE-BOSS-MR incorporates into the process a variety of window sizes combined with multiple dilation parameters applied to the original and to first-order differences' time series, with the latter modeling trend information. SCALE-BOSS-MR has been evaluated using the eight datasets with the largest training size of the UCR time series repository. The results indicate that SCALE-BOSS-MR can be instantiated to classifiers that are able to achieve state-of-the-art accuracy and can be tuned for scalability.

Keywords: time series; time series classification; symbolic representations; multiple representations; accuracy; execution time

Citation: Glenis, A.; Vouros, G.A. SCALE-BOSS-MR: Scalable Time Series Classification Using Multiple Symbolic Representations. *Appl. Sci.* **2024**, *14*, 689. https://doi.org/10.3390/app14020689

Academic Editors: Katia Lida Kermanidis, Phivos Mylonas and Manolis Maragoudakis

Received: 6 December 2023
Revised: 6 January 2024
Accepted: 9 January 2024
Published: 13 January 2024

1. Introduction

Time Series Classification (TSC) is an important machine learning task for many branches of science. Time series classifiers incorporate intertwined components on building time series representations, learning predictive models exploiting such representations, and measuring similarities between time series. These implement laborious processes towards improving computational efficiency, especially for training TSC models and achieving accuracy of predictions.

TSC can be applied in many types of data such as ECG in medicine; sensor data in diverse fields, including Internet of Things (IoT); and even imaging data. Indicative applications include seizure detection [1], earthquake monitoring [2], insect classification [3], as well as applications in power systems [4]. In many of these applications, having an algorithm that responds fast and accurately is important. Many algorithms have been proposed to tackle the scalability issue [5–11], which are reviewed subsequently.

This work aims to contribute to this objective by proposing the SCALE-BOSS-MR framework for symbolic TSC that are able to achieve state-of-the-art accuracy with low execution time.

Indeed, the objective of this work is to provide a generic framework for building TSC algorithms that are efficient to learn models of time series, while achieving the accuracy reported by state-of-the-art algorithms. The main ideas behind the proposed framework are as follows: (a) Exploit symbolic representations of time series; (b) leverage TSC with efficient machine learning algorithms, and (c) incorporate techniques to increase efficiency without compromising accuracy. Specifically, we have chosen to use the state-of-the-art SFA symbolic representation of time series, together with the Bag-Of-SFA words (BOSS) approach for encoding the training and test set.

In a previous work, we have proposed the SCALE-BOSS framework [9] to balance between efficiency and accuracy for TSC. The SCALE-BOSS framework has two main variations: One using clustering and the other using classification algorithms for TSC along a pipeline. These variations exploit symbolic time series representations, resulting in concrete TSC algorithms. Specifically, we can build different TSC classifiers with the only requirement that the clustering algorithm uses some form of cluster representatives. Then, simply using a 1-NN classifier, we can classify time series by comparing them only to the (few, compared to the series in the training set) representatives. On the other hand, we can use classification methods that work on the term-frequency vector of the symbolic representation. One of the main conclusions of that study is that we can build very efficient TSC methods using clustering and classification methods that exploit time series' class representatives, such as k-means, mini-batch k-means, or random forests. We have shown that SCALE-BOSS [9] manages to achieve significant improvement on the efficiency of training and testing TSC models, but with room to improve the accuracy of predictions.

Here, we leverage ideas from state-of-the-art algorithms,extending the SCALE-BOSS framework [9] using multiple time series representations to achieve state-of-the-art accuracy, while maintaining high computational efficiency.

Specifically, the contributions of this work are as follows:

1. We incorporate into SCALE-BOSS-MR and explore the use of
   - multiple window sizes;
   - multiple dilation parameters;
   - trend information by means of time series' first-order differences.

   to balance accuracy with the scalability of TSC algorithms exploiting symbolic time series representation.
2. We present extensive results from a multitude of classifiers built using the SCALE-BOSS-MR framework. The proposed classifiers can provide either state-of-the-art accuracy or state-of-the-art scalability. Some of the proposed classifiers provide a balance between accuracy and scalability, meaning that they retain state-of-the-art accuracy while also being significantly faster than other state-of-the-art classifiers that use symbolic representations.

The paper is organized as follows: Section 2 provides an overview of the literature and describes how it relates to SCALE-BOSS-MR. Section 3 describes the pipeline of the framework and its instantiations. Section 4 evaluates different TSC algorithms created using the proposed framework, and finally, Section 5 concludes the paper.

2. Literature Review

As pointed out in the introductory part of this article, one of the main ideas of the proposed approach is to use a symbolic representation of time series. The following paragraphs elaborate on such representations and corresponding TSC methods.

In [12], the authors present the SAX-VSM classification algorithm that uses the SAX representation and Term Frequency—Inverse Document Frequency (TF-IDF) weighting. A tf-idf vector is created for each class, after the training set has been transformed into SAX words. Then, the set of target time series is transformed into SAX words and the Term-Frequency (TF) vector is created for each time series in the target set. To classify a time series, the cosine similarity between the TF vectors of that series and of the classes' centers is computed.

In [13], the authors introduce the Bag of SFA Symbols Ensemble classifier (BOSS) that uses Symbolic Fourier Approximation (SFA) [14] to classify a time series. To compute SFA words, a number of Fourier coefficients are computed, which are grouped based on common prefixes, building histograms per group, discretized, and mapped to an alphabet. The SFA approximation, and thus BOSS, uses a symbolic representation based on the frequency domain, providing information about the whole series. Properties of this representation lead to significant lower training times compared to using the SAX representation. The

BOSS ensemble classifier is based on a 1-NN classification using multiple BOSS models at different time series' substructural sizes. However, BOSS requires the entire training set to be available while classifying target time series.

In [5], the authors present the BOSS-VS classification algorithm where each data point in the training set is transformed into SFA words. Then, a centroid is created for each class and the cosine similarity is computed between centroids and target series, as in [12]. This significantly reduces computational complexity and the memory footprint of the classification algorithm, since now each target time series is only compared to the class centroids. While BOSS-VS improves in scalability, it manages to do so at the expense of accuracy.

In [15], the authors present the K-BOSS-VS algorithm for time series classification, which is based on state-of-the-art symbolic time series classification algorithms, such as BOSS-VS and BOSS [5,13]. The main intuition behind the K-BOSS-VS method is that, by exploiting SFA representations, instead of having a single centroid to represent each class label, we can have K representatives per class. K-BOSS-VS applies the K-means clustering algorithm to each class label of the training set to obtain the K representatives per class. In so doing, it is between using a single representation per class, as in [5], or computing similarities with the entire training set, as in [13]. K-BOSS-VS addresses the scalability problem while at the same time achieving an accuracy greater than BOSS-VS and close to BOSS.

cBOSS [16] aims to speedup BOSS. Due to its grid-search method and the method of retaining ensemble members, BOSS is unpredictable in its time and memory resource usage. cBOSS utilizes an altered random selection of parameters of its ensemble members, allowing the user to control the build using a time contract. In doing so, it manages to significantly speed up BOSS while retaining accuracy.

S-BOSS [17] is a variation of BOSS that takes into account the location of the symbolic words in a series. The intuition is that in some datasets, the locations of certain discriminatory subsequences are important. Some patterns may gain importance only when they occur in a particular location, or a frequently occurring word may be indicative of different classes depending on when/where it occurs.

In [18], the authors propose WEASEL as a middle ground between BOSS-VS [5] and BOSS [13] for time series classification, balancing between accuracy and scalability. It uses SFA, but it does a few novel things: First, WEASEL performs feature discretization to reveal differences between classes; second, it uses windows of variable lengths, also considering the order of windows; and finally, it uses statistical feature selection, leading to significantly reduced runtime. Finally, WEASEL uses unigram and bigrams' symbolic representation of the time series. WEASEL is more scalable and accurate than BOSS, but it is not as scalable as BOSS-VS, as shown in [9].

ROCKET [7] is a non-symbolic time series classification algorithm that uses numerous (10,000) convolution kernels together with a linear classifier (ridge regression or logistic regression). A significantly faster version of ROCKET, called MiniRocket, is presented in [8], which uses fewer parameters than ROCKET to extract features, without sacrificing accuracy. Multi-Rocket [10] uses first differences to further improve accuracy compared to Mini-Rocket: Given a time series x with subsequent points with arithmetic values x_i, $i = 0, 1, \ldots$, first-order differences are computed following $out_i = x_{i+1} - x_i$, revealing time series trend information. Dilation allows for certain elements in the input time series to be downsampled by a factor d, by including every d-th time series value. The ROCKET family of algorithms does not employ a symbolic representation of time series, and it proves to be very scalable while maintaining state-of-the-art accuracy. However, first-order differences and dilation parameters can also be applied on top of algorithms that use symbolic representations to increase their accuracy.

Leveraging these ideas, WEASEL 2.0 [11] builds an ensemble of classifiers, choosing at random window sizes, dilation and the use of first-order differences. Combining windowing and dilation, each window at offset i and of length l includes l time series

elements which are downsampled starting from i by a factor d: This combination increases the receptive field of the algorithms with respect to the windowing process. Using these techniques, WEASEL 2.0 manages to provide state-of-the-art accuracy while maintaining computational efficiency.

Similarly, MR-SQM [19,20] uses different kinds of symbolic representations to improve accuracy.

TDE [21] is a classification algorithm that combines design features of four classifiers (BOSS, WEASEL, S-BOSS, and cBOSS). Like BOSS, TDE is a homogeneous ensemble of nearest neighbor classifiers that uses distance between histograms of word counts and injects diversity via parameter variation. TDE takes the ensemble structure from cBOSS, which is more robust and scalable. The use of spatial pyramids is adapted from S-BOSS and it uses bi-grams like WEASEL. TDE is significantly more accurate than WEASEL and S-BOSS while retaining the scalability of cBOSS.

In [22], the authors present Contracted Shapelets, a method to speedup shapelet-based time series classification by performing early abandon. The authors present binary shapelets in order to address three problems of shapelet transform: loss of class information, managing easy vs. hard to classify classes, and addressing multi-class problems.

In [23], the authors propose Proximity Forest, an algorithm that learns accurate models from datasets with millions of time series and classifies a time series in milliseconds. The models are ensembles of highly randomized Proximity Trees. Whereas conventional decision trees branch on attribute values (and usually perform poorly on time series), Proximity Trees branch on the proximity of time series to exemplar time series, leveraging the decades of work into developing relevant measures for time series. The authors show that their multi-resolution multi-domain linear classifier achieves a similar accuracy to the state-of-the-art COTE ensemble, as well as to recent deep learning methods (FCN, ResNet).

TS-CHIEF (Time Series Combination of Heterogeneous and Integrated Embedding Forest) [6] rivals HIVE-COTE in accuracy, but it is significantly faster. TS-CHIEF constructs an ensemble classifier that integrates the most effective methods in the TSC literature such as BOSS and Proximity Forest.

In this paper, we present an improved version of SCALE-BOSS [9], called SCALE-BOSS-MR (i.e., SCALE-BOSS with Multiple Representations). SCALE-BOSS-MR draws inspiration from many diverse state-of-the-art TSC algorithms in order to improve accuracy and bring the framework closer to the state-of-the-art algorithms to which it is compared with in Section 4, without increasing the computational cost of TSC methods. More concretely, in SCALE-BOSS-MR, we explore (a) the use of first-order differences to encode trend information similarly to Multi-Rocket [10], (b) the use of different window sizes over the time series similarly to WEASEL [18], and finally, (c) the use of dilation similarly to Rocket [7] and WEASEL 2.0 [11].

3. Framework Description: From SCALE-BOSS to SCALE-BOSS-MR

As already pointed out, two main ideas behind the SCALE-BOSS framework are as follows: (a) Exploit symbolic representations of time-series, and (b) leverage efficient machine learning algorithms for clustering and classification. Especially, we focus on models that exploit representatives of time series, so as to balance between computational efficiency and prediction accuracy.

As shown in Figure 1, the first step of this framework is to compute the symbolic representation of the training set. Any symbolic representation such as SFA or SAX can be used here.

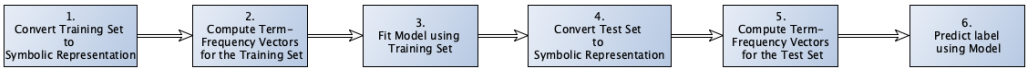

Figure 1. SCALE-BOSS workflow.

Then, the second step computes the Term-Frequency Vectors for the training set, thus creating a Bag-Of-SFA Symbols (BOSS) for the training set.

The third step computes the models of time series, which will be used subsequently for making time series label predictions: To construct such a model, we can use either a clustering or a classification mechanism exploiting the term-frequency vectors. In any case, our interest is on these models that learn representatives of classes/clusters to whom test cases will be compared with, without excluding others.

Having these models, the fourth step computes the symbolic representation for the test set, while the fifth step computes the term-frequency vectors for the test set.

The sixth and final step predicts the label for the test case exploiting the model used. Specifically, in this work, and in case we use a time-series clustering mechanism, a 1-NN classifier classifies the test time series by comparing it to the representatives of each cluster. On the other hand, if a classifier is used, then the test time series is classified using the trained classifier model.

For the different instantiations of the framework, here we succinctly mention all the choices made and alternatives in the TSC pipeline:

1. We have chosen SFA as the symbolic representation. The framework can be tuned to other symbolic representations, but we have chosen SFA because it has been proven superior to others (e.g., SAX) [15,24].
2. Regarding clustering algorithms, we evaluated the use of K-Means, Mini-Batch K-Means, BIRCH, DBSCAN, and K-Medoids.
3. Regarding classifiers, we tested the framework with SGD, LinearSVC, AdaBoost, Multi-Layer Perceptron, Naive Bayes, QDA, RBF-VSM, Decision Trees (DT), and Random Forest (RF).
4. Regarding the distance measure for comparing time series in the clustering approach, we use the cosine similarity of their term-frequency vectors.

It is not our purpose to describe here all alternative clustering and classification methods that have been used, but subsequently,we provide details on the most effective ones, focusing on those that provide a condensed representation of time series in a form of time series class representatives.

Considering the clustering algorithms, all algorithms exploit representatives, except DBSCAN [25,26], which was used as a classic algorithm of density-based time series clustering methods. From the classifiers, we consider that DT and RF provide an elaborated representation of time series along model (decision tree) paths, which can be considered to model training set variations, much like class time series class representatives do.

Close to tree representations, BIRCH [27] is an incremental clustering algorithm using Clustering Feature Trees. BIRCH uses the concept of a *Clustering Feature* (CF): this is a triple that contains the number of data points in the cluster, the linear sum of the points, and the square sum of the points. A *CF tree* is a height-balanced tree with two parameters: the branching factor B and the threshold T. Each non-leaf node contains at most B entries of the form $(CF, child_i)$, and each leaf node consists of L CFs. In addition, each leaf node has two pointers, "prev" and "next", which are used to chain all leaf nodes together for efficient scans. All entries in a leaf node must satisfy a threshold requirement, with respect to a threshold value, and thus, a leaf node represents a cluster made up of all the subclusters represented by its entries.

While the K-BOSS-VS method [15] applies K-means to each class label of the training set to obtain the K representatives per class, we have evaluated the use of clustering variants that increase the computational efficiency of the overall method, such as Mini-Batch K-Means [28], thus resulting in the MB-K-BOSS-VS configuration. Mini-Batch K-Means is an online variant of the K-Means clustering algorithm that converges using only a subset of the dataset and has results very close to the full K-Means clustering algorithm.

In addition to these clustering algorithms, we also evaluated the use of the K-Medoids algorithm [29]. This is a fast and simple algorithm that calculates the distance matrix once and then finds the cluster medoids in each iteration step.

Using decision trees to classify symbolic representations of time series, instead of using similarities between the representatives and the test set, we feed the normalized term-frequency vectors of the training set into a decision tree (DT) classifier or a random forest (RF) classifier. Then, the trained model is used to infer the class label for each time series in the test set. RF is essentially an ensemble variant of DT, allowing us to vary the number of trees, much like varying the number of class representatives in K-BOSS-VS.

In the evaluation reported in [9], we show that amongst the classifiers, Random Forest and MLP provided the best results. For clustering algorithms, Mini-Batch K-Means and K-Means provided the best accuracy. In any case, although computationally efficient, the accuracy achieved by these methods is lower than that of state-of-the-art methods.

SCALE-BOSS-MR refines the SCALE-BOSS framework by fusing multiple representations into a single term-frequency vector that is in turn fed into the regular SCALE-BOSS pipeline. This procedure was mostly inspired by WEASEL [18]. Multiple representations of time series result from the use of first differences of time series in combination with multiple temporal time series' windows and multiple dilation filters. Given any specific dilated window, this is processed to obtain a term-frequency vector using unigrams or bigrams, and all the term vectors from all dilated windows are stacked column-wise to provide the representation of the time series.

More specifically, as we can see in Figure 2, the SCALE-BOSS-MR framework works, stage by stage, as follows:

1. Computes the dilated windows for the training set given a configuration of W window sizes in a set of $window\_configs$ and D dilation filters in a set of $dilation\_filter\_configs$. This results in $DW = W \times D$ dilated windows configurations;
2. Computes the DW first-order differences' dilated windows, i.e., dilated windows for the time series created by the first-order differences of the original time series;
3. Computes the term-frequency vectors of dilated windows for the original time series and of the dilated windows for the first-order differences' time series, then stacks the term-frequency vectors column-wise;
4. Fits the model using the time series representations of the training set;
5. In the next three stages, computes the symbolic representation for the dilated windows of a time series from the test set, as well as the corresponding first-order differences' time series, then computes and stacks the term-frequency vectors.

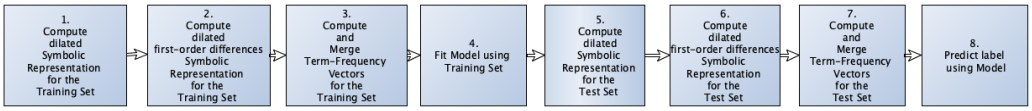

Figure 2. SCALE-BOSS-MR workflow.

Algorithm 1 provides a succinct and comprehensive description of SCALE-BOSS-MR.

More specifically, lines 1–9 describe the main SBMR function: in lines 2 and 3, the algorithm loops over window configurations and dilation sizes configurations. For each window configuration, it computes the dilated time series according to the dilation parameter and produces the term-frequency vector. Line 6 maintains the resulting term-frequency vector by concatenating column-wise term-frequency vectors produced by each window and dilation configuration.

Lines 11 and 12 of the algorithm initialize an empty term-frequency vector for the train and test set, respectively. In Line 13, the algorithm computes the term-frequency vector for the training set.

Algorithm 1: SCALE-BOSS-MR algorithm

```
   Input  : X: the training set
   Input  : y_train: the labels for the training set
   Input  : X_test: the test set
   Input  : window_configs: the window sizes and steps
   Input  : dilation_filter_configs: the dilation filter sizes
   Input  : clf: Classifier that can work on term-frequency vectors
   Output : predicted: the predictions for the test set
 1 Function SBMR-inner(X, window_configs, dilation_filter_configs,
     global_term_frequency_vector):
 2     for curr_window_config in window_configs do
 3         for curr_dilation_filter in dilation_filter_configs do
 4             X_dilated ← dilation (X,curr_dilation_filter)
 5             curr_term_frequency_vector_train← process_time_series
                 (X_dilated,curr_window_config)
 6             global_term_frequency_vector ← hstack
                 (global_term_frequency_vector,curr_term_frequency_vector)
 7         end for
 8     end for
 9     return
10
11 global_term_frequency_vector_train ← array ()
12 global_term_frequency_vector_test ← array ()
13 SBMR-inner (X,window_configs, dilation_filter_configs,
     global_term_frequency_vector_train)
14 if doTrend then
15     X_trend_train ← diff (X)
16     SBMR-inner (X_trend_train,window_configs, dilation_filter_configs,
         global_term_frequency_vector_train)
17 clf.fit (global_term_frequency_vector_train,y_train)
18 SBMR-inner (X_test,window_configs, dilation_filter_configs,
     global_term_frequency_vector_test)
19 if doTrend then
20     X_trend_test ← diff (X_test)
21     SBMR-inner (X_trend_test,window_configs, dilation_filter_configs,
         global_term_frequency_vector_test)
22 predicted ← clf.predict (global_term_frequency_vector_test)
23 return predicted
```

Incorporating first-order differences into the process is an optional alternative, and one can configure the method using the *doTrend* variable. In Lines 14 to 16, the algorithm calls for the SBMR function for the first-order differences' time series of the training set, maintaining the resulting term-frequency vector by concatenating column-wise term-frequency vectors produced by each window and dilation configuration. In Line 17, the classifier is trained on the final term-frequency vectors for the training set. Similarly, in Line 18, the algorithm calls for the SBMR function for the first-order differences' time series of the test set and proceeds in Lines 19 to 21 to compute the term-frequency vector for the test set. Finally, in Line 22, the classifier makes its predictions, taking as input the final concatenated term-frequency vector for the test set.

The framework has been implemented on top of the pyts [30] Time Series Classification library. The source code of the SCALE-BOSS-MR implementation is provided in https://github.com/aglenis/scale_boss_mr (accessed on 8 January 2024).

We have chosen the word length to be equal to four in all the configurations we explore, given that it is the "default" for the BOSS-VS implementation of pyts.

Additionally, when computing the symbolic representation, we use unigrams, or unigrams and bigrams: Using bigrams and unigrams helps retain sequence information that is inherently lost in the bag-of-words model.

As a final note, it is clear from the specified algorithm that trend information is split in parts by the windowing process applied on first-order differences' time series, according to the windowing and dilation configurations.

For the different clustering algorithms and classifiers, we have used the implementations provided by the Sci-kit Learn [31] python library.

4. Evaluation

Aiming to balance between accuracy of predictions and efficiency in training and testing, we evaluate instantiations of the SCALE-BOSS-MR framework in comparison to the state-of-the-art methods, in terms of the mean total time and mean accuracy.

The mean total time is the total execution time (both train and test time) for all datasets divided by the total number of datasets. The mean accuracy is the average accuracy across all datasets. We also provide the standard deviation (std) of the total execution time and of accuracy.

Accuracy and execution time are the commonly used measures to report performance in most works in the literature, e.g., [11]. This choice allows us to be directly comparable to other algorithms. However, to comprehensively present the strengths and limitations of SCALE-BOSS-MR, we directly compare SCALE-BOSS-MR configurations with state-of-the-art algorithms such as WEASEL 2.0, Rocket, MiniRocket, and MRSQM.

While Appendix A reports on accuracy and total execution time for individual time series datasets, we report on mean accuracy and mean total execution time in the main part of the paper, comprehensively providing the strengths and limitations of the proposed framework.

Table 1 shows the characteristics of the UCR datasets used in the evaluation. UCR time-series datasets are the de facto datasets for time series classification, and we have chosen to use the eight datasets from the UCR time series repository with the largest training set. *Train_size* denote the number of time series in the training set, *test_size* denote the number of time series in the test set, *n_classes* denotes the number of classes in the dataset, and *n_timestamps* denotes the length of each time series in the dataset. The datasets are split into train and test time series by the dataset providers.

Table 1. Characteristics of the datasets.

Name	Train_Size	Test_Size	n_Classes	n_Timestamps
Crop	7200	16,800	24	46
FordB	3636	810	2	500
FordA	3601	1320	2	500
NonInvasiveFetalECGThorax2	1800	1965	42	750
NonInvasiveFetalECGThorax1	1800	1965	42	750
PhalangesOutlinesCorrect	1800	858	2	80
HandOutlines	1000	370	2	2709
TwoPatterns	1000	4000	4	128

Table 2 specifies the configurations of window sizes and window steps. When the value is a float, the float represent the percentage of the size of time series. If the value is an integer, then this denotes the number of subsequent time series points included in the window. Integer values of the window sizes are those used by WEASEL in [18]; however, with sufficiently large differences between subsequent window configuration sizes (i.e., four). Similarly, float values are those provided by pyts as initial values, and we add to them to these further configurations for exploration.

Table 2. Window configurations.

Name	Window_Configs (Window Sizes)	Window Step
W0	24	1
W1	0.1, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9	0.0125
W2	0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9	0.0125
W3	0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9	0.00625
W4	0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9	0.003125
W5	0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9	0.003125
W6	4, 12, 28	1
W7	4, 8, 12, 16, 20, 24, 28	1
W8	4, 8, 12, 16, 20, 24, 28, 32, 36, 40	1
W9	12, 16, 20, 24, 28, 32, 36, 40	1
W10	12, 16, 20, 24, 28, 32, 36, 40	2
W11	12, 16, 20, 24, 28, 32, 36, 40	4
W12	4, 8, 12, 16, 20, 24, 28, 32, 36, 40	4
W13	4, 8, 12, 16, 20, 24, 28, 32, 36, 40	2
W14	12, 16, 20, 24, 28, 32	2
W15	12, 16, 20, 24, 28, 32, 36, 40, 32	8

Table 3 shows dilation configurations in terms of the dilation factors. These dilation configurations are those mostly suggested in [11]. We subsequently perform a thorough evaluation to determine the best dilation configuration parameters in order to balance between accuracy and execution time.

Table 3. Dilation configurations.

Name	Dilation_Filter_Configs
D0	1 (no dilation)
D1	1, 5, 7, 9, 11
D2	1, 7, 9, 11
D3	1, 7, 11
D4	1, 11
D5	1, 7
D6	1, 9

To present the results, we use the following SCALE-BOSS-MR configurations' naming convention:

NAME-CLS_METHOD-WINDOW_CONFIG-USE_OF_TREND-DILATION_CONFIG-NGRAM,

where

- NAME is the name of the method used; for example, SBMR corresponds to SCALE-BOSS-MR;
- CLS METHOD refers to the classifier used, e.g., Random Forest (RF), MLP, or MB-K-BOSS-VS. The MB-K-BOSS-VS classifier exploits the concatenated term-frequency vector resulting from the SCALE-BOSS-MR workflow, etc.;
- WINDOW CONFIG refers to the window configuration as described in Table 2;
- USE OF TREND denotes whether or not we use first-order differences to encode trend information. When the configuration uses first-order differences, this is denoted by *"trend"*, whereas if not, we denoted it as *"noTrend"*;
- DILATION CONFIG denotes the dilation filter values as described in Table 3;
- NGRAM denotes whether we use unigrams or unigrams together with bigrams. Unigrams are denoted as UG whereas unigrams and bigrams are denoted as BG.

Table 4 shows the results from different SBMR configurations, including also BOSS-RF, BOSS-MLP, and MB-K-BOSS-VS, all with trend information and no multiple window configurations, but with alternative dilation configurations and n-grams.

Table 4. Accuracy and total execution time of the classifiers using the W0 windows configuration.

Algorithm	Accuracy_Mean	Accuracy_std	Total_Time_Mean	Total_Time_std
SBMR-RF-W0-trend-D1-BG	0.857	0.089	111.7	57.8
SBMR-RF-W0-trend-D3-BG	0.857	0.085	70.0	36.4
SBMR-RF-W0-trend-D2-BG	0.856	0.085	92.1	47.0
SBMR-RF-W0-trend-D6-BG	0.849	0.082	52.4	26.7
SBMR-RF-W0-trend-D1-UG	0.848	0.083	101.6	57.3
SBMR-RF-W0-trend-D4-BG	0.847	0.078	46.0	23.6
SBMR-RF-W0-trend-D5-BG	0.843	0.085	51.2	26.5
SBMR-RF-W0-trend-D6-UG	0.839	0.082	38.8	21.6
SBMR-RF-W0-trend-D0-UG	0.827	0.080	19.1	11.6
SBMR-MLP-W0-trend-D0-UG	0.816	0.090	21.9	10.6
BOSS-RF	0.796	0.088	9.7	5.5
BOSS-MLP	0.778	0.102	12.5	5.4
SBMR-MB-K-BOSS-VS-trend-D0-UG	0.766	0.098	19.0	11.7
MB-K-BOSS-VS	0.727	0.101	11.5	6.9

Additionally, Table 5 shows results from different SBMR configurations, with and without trend information and no dilation, but with different classifiers, window configurations, and n-grams. Tables 4 and 5 show, among others, that RF are the most competitive classifiers.

Table 5. Accuracy and execution time of configurations with multiple window sizes.

Algorithm	Accuracy_Mean	Accuracy_std	Total_Time_Mean	Total_Time_std
SBMR-RF-W8-trend-D0-UG	0.869	0.079	204.0	130.6
SBMR-RF-W8-trend-D0-BG	0.863	0.081	216.8	123.2
SBMR-RF-W9-trend-D0-UG	0.862	0.080	154.9	96.8
SBMR-RF-W10-trend-D0-UG	0.862	0.078	72.7	45.5
SBMR-RF-W8-noTrend-D0-BG	0.861	0.080	104.6	60.6
SBMR-RF-W13-trend-D0-UG	0.861	0.074	91.8	55.9
SBMR-RF-W8-noTrend-D0-UG	0.859	0.078	93.8	59.1
SBMR-RF-W14-trend-D0-UG	0.850	0.073	54.8	33.4
SBMR-RF-W11-trend-D0-UG	0.846	0.074	36.5	21.3
SBMR-RF-W12-trend-D0-UG	0.845	0.070	44.9	25.8
SBMR-RF-W7-noTrend-D0-UG	0.843	0.072	60.8	36.6
SBMR-RF-W6-trend-D0-UG	0.842	0.081	52.8	32.3
SBMR-RF-W4-trend-D0-UG	0.841	0.097	264.9	203.1
SBMR-RF-W1-trend-D0-BG	0.832	0.103	58.3	36.5
SBMR-RF-W1-trend-D0-UG	0.830	0.101	49.6	31.9
SBMR-RF-W4-noTrend-D0-UG	0.830	0.100	132.1	105.1
SBMR-RF-W5-noTrend-D0-UG	0.825	0.098	80.1	67.9
SBMR-RF-W3-noTrend-D0-UG	0.824	0.106	75.7	50.8
SBMR-RF-W2-noTrend-D0-UG	0.814	0.110	42.3	23.4
SBMR-RF-W1-noTrend-D0-BG	0.814	0.106	29.8	17.8
SBMR-RF-W1-noTrend-D0-UG	0.813	0.105	25.8	17.2
SBMR-MB-K-BOSS-VS-W1-trend-D0-BG	0.794	0.098	67.6	56.8
SBMR-MB-K-BOSS-VS-W1-trend-D0-UG	0.788	0.112	43.5	27.9

Table 5. *Cont.*

Algorithm	Accuracy_Mean	Accuracy_std	Total_Time_Mean	Total_Time_std
SBMR-MB-K-BOSS-VS-W1-noTrend-D0-BG	0.767	0.106	32.5	24.9
SBMR-MB-K-BOSS-VS-W1-noTrend-D0-UG	0.760	0.113	23.1	15.4

Table 6 shows different configurations with RF classifiers, but also with different combinations of window and dilation configurations.

Table 6. Accuracy and execution time for combinations of window and dilation configurations.

Algorithm	Accuracy_Mean	Accuracy_std	Total_Time_Mean	Total_Time_std
SBMR-RF-W8-trend-D6-BG	0.875	0.079	489.0	234.5
SBMR-RF-W14-trend-D3-BG	0.865	0.077	230.8	105.2
SBMR-RF-W10-trend-D6-BG	0.865	0.073	217.9	100.4
SBMR-RF-W14-trend-D6-BG	0.863	0.074	149.2	68.3
SBMR-RF-W11-trend-D3-BG	0.859	0.073	201.1	103.7
SBMR-RF-W11-trend-D6-BG	0.857	0.073	116.3	54.1

Subsequently, we describe the major findings from the results reported in these tables and provide further results that allow us to show the potential of the proposed approach.

4.1. Evaluation of SBMR with D0 and W0 Configurations

From Table 4 and Figure 3, we can see the following.

1. BOSS-RF starts with an approximately 0.8 accuracy and 9.7 s of mean total execution time;
2. SBMR-RF-W0-trend-D0-UG achieves a 0.827 accuracy with 19.7 s of mean total execution time. This means that adding trend information gives a boost in accuracy, but at the expense of doubling the mean total execution time;
3. SBMR-RF-W0-trend-D6-BG achieves a 0.849 accuracy with 52 s of mean total execution time. This means that even moderate dilation helps to boost accuracy at the expense of further increasing the total execution time;
4. SBMR-RF-W0-trend-D3-BG achieves an accuracy of 0.857 with 70 s of mean total execution time.

From the above, we can conclude the following:

1. Bigrams with dilation help to improve the accuracy of the method;
2. D6 (described in Table 3) performs better than D4 (with little difference) and D5 (with slightly bigger difference) configurations;
3. However, D3 (as described in Table 3) performs best (but with slightly worse total execution time compared to D6 and D4 configurations). This is expected, as the representations from D3 are more in number compared to those of D4 and D6, while it seems that the receptive field introduced by D4 and D6 is not sufficient for the algorithms to discriminate between time series' classes effectively. It must be noted that in these configurations, dilation applies once to the time series, as these use the W0 window configuration. This is further supported by the results reported by D1 and D2, which incorporate the representations produced by D3 and D6, thus achieving nearly the same accuracy, but with considerably increased total execution time;
4. An additional finding is that representations with bigrams (BG) are in general more effective compared to those using unigrams (UG), in these configurations of the framework.

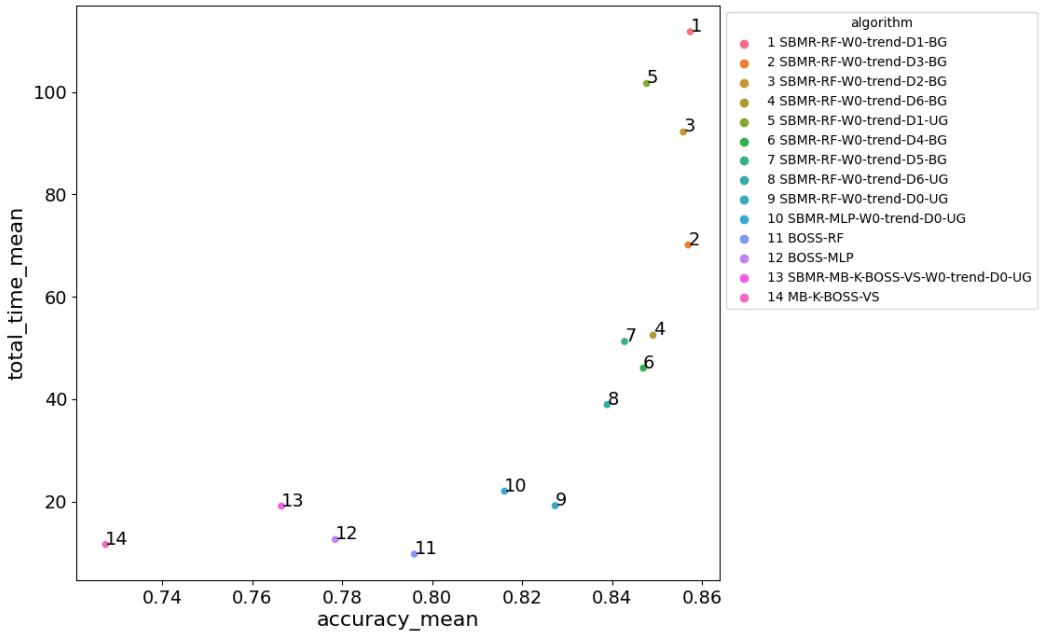

Figure 3. Total execution time (in seconds) and accuracy for single window representation.

Table 5 provides evidence on window configurations with no dilation. Contrary to what is reported in Table 4, here, representations with unigrams seem to be more effective compared to those with bigrams, while it is clear that adding representations with trend increases the mean classification accuracy. The W8 configuration reports the best mean accuracy among others, although with increased total execution time. This is so, given that W8 incorporates many window configurations, all with step 1. In contrast, W10 is an indicative case with slightly fewer window configurations, and step 2 achieves worst accuracy, but with significantly reduced mean total execution time.

4.2. Evaluation of SBMR-RF with Multiple Windows and Dilation Configuration

Table 6 and Figure 4 report on configurations combing window and dilation configurations, with representations including bigrams, including trend information. The results show the following:

1. SBMR-RF-W8-D6-trend-BG achieves an accuracy of 0.875 with 489 s of mean total execution time compared to the undilated 0.863 with 216 s of mean total execution time reported in Table 5;
2. SBMR-RF-W14-D6-trend-BG achieves a slightly worst accuracy of 0.863 with significantly reduced mean total execution time of 149 s versus 0.850 with 54 s for the undilated case reported in Table 5;
3. SBMR-RF-W11-D6-trend-BG achieves an accuracy of 0.857 with 116 s compared to 0.846 with 36 s for the undilated case reported in Table 5;
4. SBMR-RF-W10-D6-trend-BG achieves an accuracy of 0.865 with 217 s compared to 0.862 with 72 s for the undilated case reported in Table 5.

From the above, we can conclude the following:

1. Dilation combined with window configurations has a considerable impact on total execution time, as expected, but significantly increases the accuracy of predictions for any of the windowing configurations;
2. Dilation configuration D3 is slightly better in terms of accuracy compared to configuration D6, which is significantly faster than D3. This is due to the fact that D3 increases the time series representations when combined with window configurations, resulting in the worst total execution time compared to D6. This leads us to believe that D6 achieves a good balance between total execution time and accuracy when it is combined with suitable window configurations.

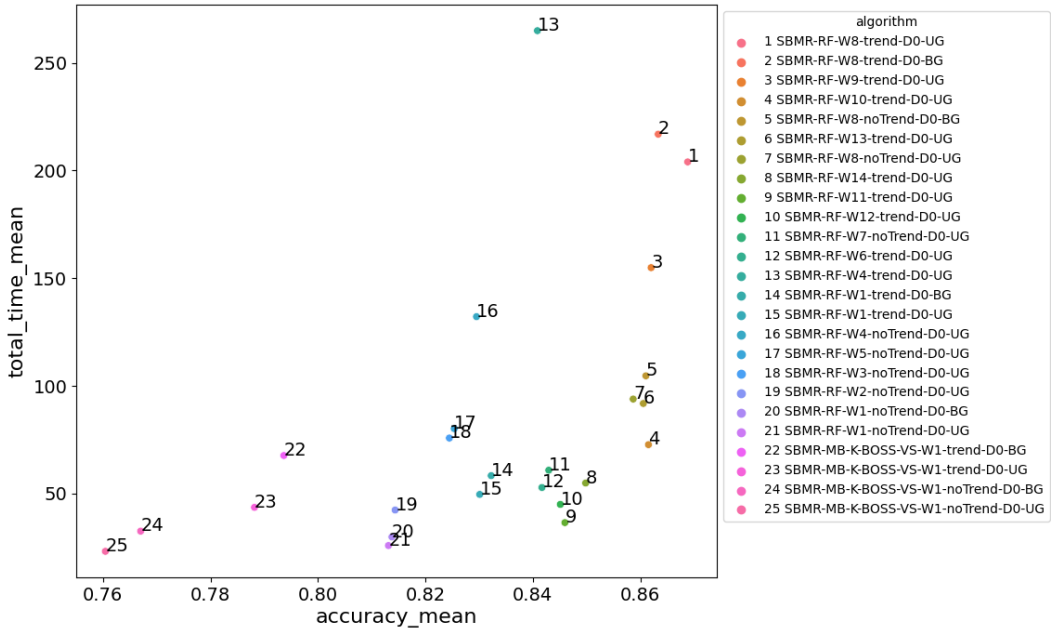

Figure 4. Average total execution time (in seconds) and accuracy for multiple window configurations.

4.3. Evaluation of SBMR with the Ridge Regression with Cross-Validation Classifier

As part of the evaluation, we used a Ridge Regression with a Cross-Validation classifier denoted by RidgeCV. We chose to run experiments using RidgeCV because it is the classifier used in ROCKET, miniRocket, and WEASEL 2.0.

The results with the RidgeCV and W0 configuration can be found in Table 7, showing, at a great extent, consistency with the results reported for RF.

From Table 8 and Figure 5, we can see the following:

1. The best configuration in terms of accuracy is SBMR-RidgeCV-W8-trend-D6-BG, which achieves an accuracy of 0.892 with 517.5 s of mean total execution time.
2. Other window configurations can improve the execution time, but at the expense of lower accuracy:
    - SBMR-RidgeCV-W14-trend-D6-BG achieves an accuracy of 0.882 with 160.4 s of mean total execution time;
    - SBMR-RidgeCV-W11-trend-D6-BG achieves an accuracy of 0.881 with 129 s of mean total execution time;
    - SBMR-RidgeCV-W0-trend-D6-BG achieves an accuracy of 0.844 with 70.2 s of mean total execution time.
3. Opting for no dilation but with different window configurations can further improve the total execution time, but at the expense of even lower accuracy:
    - SBMR-RidgeCV-W11-trend-D0-BG achieves an accuracy of 0.844 with 63.5 s of mean total execution time;
    - SBMR-RidgeCV-W15-trend-D0-BG achieves an accuracy of 0.831 with 40.6 s of mean total execution time;
    - SBMR-RidgeCV-W11-trend-D0-UG achieves an accuracy of 0.827 with 43.7 s of mean total execution time.

Table 7. Accuracy and execution time of RidgeCV classifier without multiple windows.

Algorithm	Accuracy_Mean	Accuracy_std	Total_Time_Mean	Total_Time_std
SBMR-RidgeCV-W0-trend-D1-BG	0.858	0.095	135.9	66.4
SBMR-RidgeCV-W0-trend-D2-BG	0.855	0.094	100.2	48.8
SBMR-RidgeCV-W0-trend-D3-BG	0.852	0.090	78.2	38.0
SBMR-RidgeCV-W0-trend-D6-BG	0.844	0.087	59.6	37.4
SBMR-RidgeCV-W0-trend-D5-BG	0.841	0.095	59.8	33.8
SBMR-RidgeCV-W0-trend-D4-BG	0.840	0.089	60.2	36.4
SBMR-RidgeCV-W0-trend-D6-UG	0.818	0.101	38.7	21.9
SBMR-RidgeCV-W0-trend-D0-UG	0.780	0.101	21.5	11.4
SBMR-RidgeCV-W0-noTrend-D0-UG	0.743	0.100	11.2	6.0

Table 8. Accuracy and execution time of RidgeCV classifier with multiple windows and dilation configurations.

Algorithm	Accuracy_Mean	Accuracy_std	Total_Time_Mean	Total_Time_std
SBMR-RidgeCV-W8-trend-D6-BG	0.892	0.079	517.5	247.6
SBMR-RidgeCV-W14-trend-D6-BG	0.882	0.077	160.4	76.3
SBMR-RidgeCV-W11-trend-D6-BG	0.881	0.076	129.0	68.3
SBMR-RidgeCV-W15-trend-D6-BG	0.871	0.071	78.9	51.7
SBMR-RidgeCV-W0-trend-D6-BG	0.844	0.087	70.2	43.6
SBMR-RidgeCV-W11-trend-D0-BG	0.844	0.075	63.5	35.9
SBMR-RidgeCV-W15-trend-D0-BG	0.831	0.072	40.5	30.0
SBMR-RidgeCV-W11-trend-D0-UG	0.827	0.085	43.6	22.6

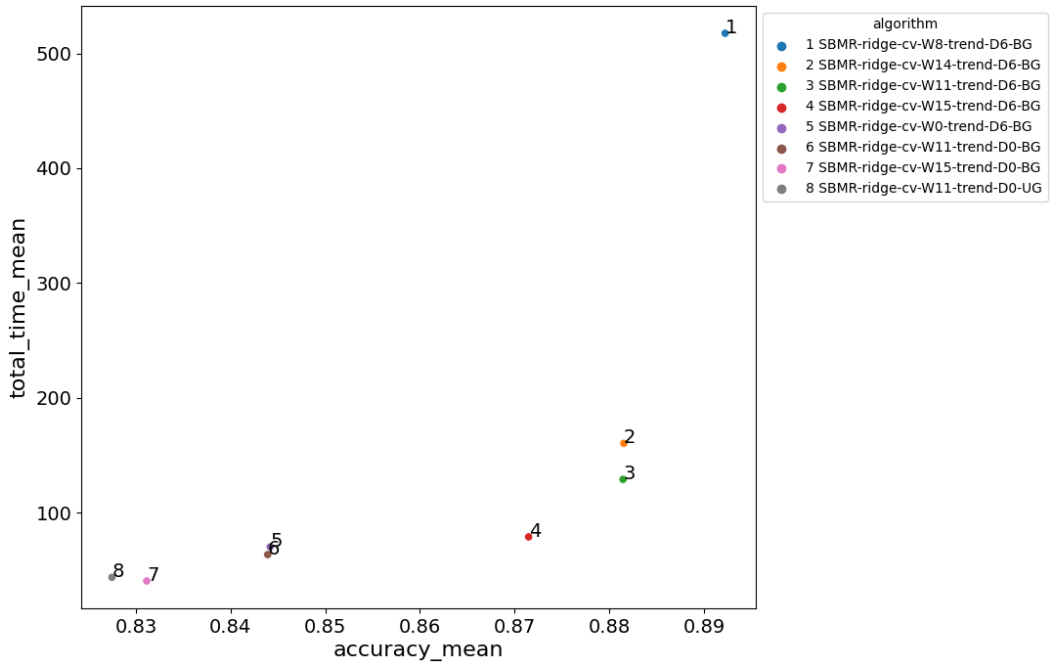

Figure 5. Average total execution time (in seconds) and accuracy using ridge regression.

From the above, we can see that using RidgeCV SCALE-BOSS-MR gives better results in terms of accuracy when using both multiple windows representation and dilation, compared to RF, without a significant increase in total execution time.

4.4. Comparison with the State-of-the-Art Algorithms

From the state-of-the-art algorithms, WEASEL 2.0 and MR-SQM exploit symbolic representations and are thus closer to our method. Table 9 and Figure 6 report on state-of-the-art algorithms:

1. WEASEL V2.0 achieves the best accuracy of 0.904 with 339 s of mean total execution time compared to 309 of Rocket and 18.7 s of Mini-Rocket;
2. MR-SQM with five SFA representations achieves a 0.889 accuracy with 418 s of mean total execution time. With single SFA representation and single SAX representation, it achieves 0.878 with 176 s of exec time. Using only a single SFA, it achieves an accuracy of 0.861 with 117 s of mean total execution time.

Given the above results, we can conclude the following:

1. SBMR-RidgeCV-W8-trend-D6-BG achieves an almost state-of-the-art accuracy, but with a significant increase in total execution time compared with the state-of-the-art methods. SBMR-RidgeCV-W8-trend-D6-BG is only 1% less accurate than WEASEL 2.0, Rocket, and MiniRocket. This suggests that the added representations help achieve a state-of-the-art accuracy;
2. SBMR-RidgeCV-W11-trend-D6-BG achieves an accuracy of 0.881, very close to the accuracy achieved by state-of-the-art algorithms, and with 129 s of mean total execution time, which is significantly faster than state-of-the-art algorithms exploiting symbolic representations;
3. SBMR-RidgeCV-W15-trend-D6-BG achieves an accuracy of 0.871, which is still close to the state of the art with 79 s of mean total execution time, being even faster than SBMR-RidgeCV-W11-trend-D6-BG.

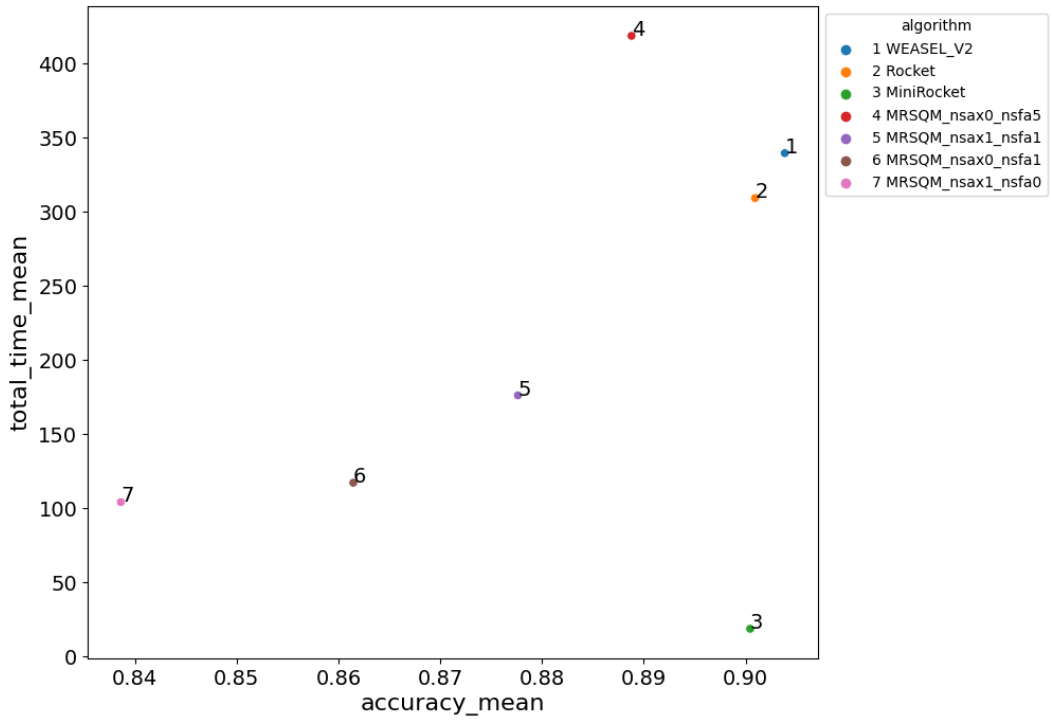

Figure 6. Average total execution time (in seconds) and accuracy for the state-of-the-art representations.

These results show that the SCALE-BOSS-MR can be configured to achieve high accuracy, with low total execution time, compared to state-of-the-art methods exploiting symbolic representations.

Table 9. Accuracy and execution time of state-of-the-art algorithms.

Algorithm	Accuracy_Mean	Accuracy_Pop_std	Total_Time_Mean	Total_Time_Pop_std
WEASEL_V2	0.904	0.078	339.6	189.9
Rocket	0.901	0.084	309.3	167.7
MiniRocket	0.900	0.082	18.7	20.1
MRSQM_nsax0_nsfa5	0.889	0.093	418.8	239.4
MRSQM_nsax1_nsfa1	0.878	0.101	176.2	101.1
MRSQM_nsax0_nsfa1	0.861	0.109	117.2	76.1
MRSQM_nsax1_nsfa0	0.839	0.113	104.2	59.0

5. Conclusions

In this paper, we extend the SCALE-BOSS framework by incorporating into the process multiple time series representations using multiple time series' windows' sizes combined with multiple dilation parameters and applied to the original time series, as well as to the first-order differences' time series encoding trend information. Specifically, SCALE-BOSS-MR presents an improved version of SCALE-BOSS [9], incorporating multiple time series symbol-only representations, drawing inspiration from many diverse state-of-the-art TSC algorithms, with the objective to achieve state-of-the-art accuracy without increasing the computational cost of TSC methods. In so doing, SCALE-BOSS-MR incorporates (a) first-order differences to encode trend information similarly to Multi-Rocket [10], (b) different window sizes over the time series similarly to WEASEL [18], and (c) multiple dilation parameters similarly to Rocket [7] and WEASEL 2.0 [11].

The major findings are as follows: Adding trend information improved TSC accuracy, while maintaining scalability in terms of execution time. Using both multiple window sizes and trend information, the produced algorithms reach state-of-the-art accuracy.

Specifically:

1. Trend encoding helps both algorithms in the single window case and when using multiple window sizes;
2. Adding dilation configurations to the SCALE-BOSS-MR workflow helps significantly, but it increases the execution time considerably when combined with multiple window configurations;
3. Adding multiple window sizes to the SCALE-BOSS-MR workflow also helps in significantly increasing accuracy;
4. Using multiple window sizes, trend information, and dilation, we can achieve state-of-the-art accuracy. In addition, the resulting method can be tuned to be as efficient as other state-of-the-art methods exploiting symbolic time series representations;
5. Adding a few representations resulting from different dilation configurations improves accuracy while at the same time retaining scalability. This is validated since D6 is only marginally worse than D3 in terms of accuracy, but it is significantly faster and is also combined with suitable window configurations;
6. Ridge regression with cross validation performed the best in terms of accuracy compared to the Random Forest classifier;
7. SCALE-BOSS-MR is only 1% less accurate compared to state-of-the-art algorithms when tuned for accuracy, while it can be significantly faster than state-of-the-art algorithms with minimal loss in accuracy (2–3%) when tuned for scalability.

As future work, we plan to port SCALE-BOSS-MR on a parallel processing framework such as Apache Spark [32], which is similar to [15]. This will allow for (a) incorporating additional time series representations to further increase accuracy in more efficient ways; (b) investigating using multiple instantiations of SCALE-BOSS-MR in an ensemble without compromising efficiency; and (c) providing experimental results with real-world datasets, with big data characteristics and application/domain-related characteristics (e.g., delay).

Furthermore, while SCALE-BOSS-MR shows that it can be tuned to accuracy vs. computational efficiency, we need effective ways of determining the appropriate configurations to address real-world requirements: This is an important part of future work as well.

Finally, we do plan to investigate the use of convolutional neural networks on top of the symbolic representations to further advance accuracy while significantly reducing the total execution time and well as adapting the SCALE-BOSS-MR algorithm for multivariate time series classification.

Author Contributions: Conceptualization, A.G. and G.A.V.; methodology, A.G. and G.A.V.; software, A.G.; validation, A.G. and G.A.V.; formal analysis, A.G. and G.A.V.; investigation, A.G.; resources, not applicable; data curation, A.G.; writing—original draft preparation, A.G. and G.A.V.; writing—review and editing, A.G. and G.A.V.; visualization, A.G. and G.A.V.; supervision, G.A.V.; project administration, G.A.V. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Publicly available datasets were analyzed in this study. The UCR datasets used in this study can be found at: http://www.timeseriesclassification.com/dataset.php (accessed on 8 January 2024).

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

Appendix A

The appendix shows detailed accuracy and total execution time results for all the methods and datasets. The detailed tables help compare the proposed methods with other methods proposed in the literature, as well as compare the proposed methods against each other on a per-dataset basis.

Table A1. Accuracy of the classifiers without multiple window configurations.

Dataset Algorithm	Crop	FordA	FordB	HandOutlines	NonInvasiveFetalECGThorax1	NonInvasiveFetalECGThorax2	PhalangesOutlinesCorrect	TwoPatterns
SBMR-RF-W0-trend-D1-BG	0.705	0.911	0.728	0.914	0.900	0.930	0.818	0.952
SBMR-RF-W0-trend-D1-UG	0.710	0.914	0.740	0.835	0.894	0.922	0.814	0.952
SBMR-RF-W0-trend-D3-BG	0.709	0.917	0.738	0.911	0.893	0.925	0.817	0.945
SBMR-RF-W0-trend-D4-BG	0.709	0.910	0.741	0.895	0.884	0.915	0.805	0.916
SBMR-RF-W0-trend-D5-BG	0.708	0.919	0.742	0.792	0.898	0.924	0.812	0.947
SBMR-RF-W0-trend-D6-BG	0.703	0.914	0.741	0.892	0.894	0.919	0.805	0.924
SBMR-RF-W0-trend-D6-UG	0.707	0.912	0.759	0.784	0.907	0.920	0.793	0.929
BOSS-MLP	0.576	0.872	0.695	0.776	0.795	0.798	0.774	0.941
SBMR-MLP-W0-trend-D0-UG	0.641	0.892	0.794	0.749	0.859	0.866	0.779	0.950
BOSS-RF	0.634	0.883	0.717	0.838	0.780	0.823	0.765	0.928
SBMR-RF-W0-trend-D0-UG	0.683	0.924	0.768	0.811	0.845	0.878	0.774	0.936
MB-K-BOSS-VS	0.550	0.833	0.668	0.681	0.702	0.771	0.712	0.902
SBMR-MB-K-BOSS-VS-W0-trend-D0-UG	0.608	0.883	0.720	0.686	0.773	0.828	0.717	0.918

Table A2. Total execution time without multiple window configurations.

Dataset Algorithm	Crop	FordA	FordB	HandOutlines	NonInvasiveFetalECGThorax1	NonInvasiveFetalECGThorax2	PhalangesOutlinesCorrect	TwoPatterns
SBMR-RF-W0-trend-D1-BG	87.343	135.100	122.949	194.351	152.298	151.977	11.589	38.516
SBMR-RF-W0-trend-D1-UG	58.626	122.240	127.212	184.951	140.944	139.074	10.026	30.200
SBMR-RF-W0-trend-D2-BG	79.353	108.209	104.967	158.977	125.302	121.167	9.591	29.938
SBMR-RF-W0-trend-D3-BG	55.140	86.998	77.606	122.502	94.733	93.438	7.314	23.052
SBMR-RF-W0-trend-D4-BG	37.555	55.917	52.452	79.548	60.430	62.730	4.875	14.792
SBMR-RF-W0-trend-D5-BG	40.507	62.404	56.891	87.589	69.401	71.217	5.389	16.574
SBMR-RF-W0-trend-D6-BG	40.689	63.691	57.149	90.239	72.025	70.745	5.749	19.403
SBMR-RF-W0-trend-D6-UG	25.173	48.654	43.773	70.978	53.165	53.914	3.921	11.617
BOSS-MLP	14.537	12.924	13.113	18.980	16.656	16.805	2.836	4.653
SBMR-MLP-W0-trend-D0-UG	19.615	24.757	25.050	36.260	30.015	29.316	2.973	7.908
BOSS-RF	5.879	11.837	10.547	18.049	13.826	13.815	0.931	2.741
SBMR-RF-W0-trend-D0-UG	8.869	23.781	21.487	36.326	28.164	27.698	1.695	5.371
MB-K-BOSS-VS	5.991	13.669	12.064	21.532	17.655	17.501	0.979	3.261
SBMR-MB-K-BOSS-VS-W0-trend-D0-UG	8.150	23.272	21.998	35.862	27.673	29.041	1.482	5.154

Table A3. Accuracy using different windows' configurations.

Dataset Algorithm	Crop	FordA	FordB	HandOutlines	NonInvasiveFetalECGThorax1	NonInvasiveFetalECGThorax2	PhalangesOutlinesCorrect	TwoPatterns
SBMR-MB-K-BOSS-VS-W1-D0-BG	0.673	0.730	0.616	0.886	0.737	0.785	0.739	0.969
SBMR-MB-K-BOSS-VS-W1-trend-D0-BG	0.688	0.791	0.644	0.911	0.792	0.833	0.738	0.953
SBMR-MB-K-BOSS-VS-W1-trend-D0-UG	0.686	0.777	0.609	0.892	0.804	0.827	0.723	0.988
SBMR-MB-K-BOSS-VS-W1-noTrend-D0-UG	0.664	0.719	0.588	0.868	0.752	0.786	0.730	0.979
SBMR-RF-W1-noTrend-D0-BG	0.713	0.811	0.626	0.922	0.810	0.850	0.790	0.988
SBMR-RF-W1-trend-D0-BG	0.723	0.856	0.638	0.914	0.851	0.882	0.807	0.987
SBMR-RF-W1-trend-D0-UG	0.729	0.828	0.644	0.914	0.853	0.880	0.801	0.992
SBMR-RF-W1-noTrend-D0-UG	0.717	0.789	0.632	0.916	0.818	0.851	0.788	0.994
SBMR-RF-W10-trend-D0-UG	0.713	0.933	0.815	0.835	0.901	0.912	0.812	0.971
SBMR-RF-W11-trend-D0-UG	0.708	0.926	0.795	0.816	0.887	0.889	0.802	0.945
SBMR-RF-W12-trend-D0-UG	0.716	0.924	0.788	0.830	0.883	0.888	0.802	0.931
SBMR-RF-W13-trend-D0-UG	0.724	0.930	0.817	0.835	0.903	0.908	0.803	0.963
SBMR-RF-W14-trend-D0-UG	0.710	0.929	0.804	0.838	0.884	0.880	0.803	0.951
SBMR-RF-W2-noTrend-D0-UG	0.718	0.793	0.615	0.924	0.822	0.861	0.790	0.992
SBMR-RF-W3-noTrend-D0-UG	0.719	0.810	0.628	0.916	0.844	0.873	0.810	0.996
SBMR-RF-W4-trend-D0-UG	0.732	0.848	0.672	0.916	0.867	0.893	0.803	0.996
SBMR-RF-W4-noTrend-D0-UG	0.721	0.814	0.662	0.911	0.853	0.883	0.796	0.997
SBMR-RF-W5-noTrend-D0-UG	0.717	0.805	0.670	0.919	0.838	0.863	0.795	0.996
SBMR-RF-W6-trend-D0-UG	0.715	0.908	0.737	0.827	0.905	0.896	0.796	0.950
SBMR-RF-W7-noTrend-D0-UG	0.714	0.912	0.774	0.851	0.860	0.883	0.800	0.949
SBMR-RF-W8-noTrend-D0-BG	0.712	0.930	0.801	0.859	0.894	0.900	0.809	0.984
SBMR-RF-W8-trend-D0-BG	0.724	0.930	0.804	0.830	0.917	0.923	0.800	0.980
SBMR-RF-W8-trend-D0-UG	0.729	0.934	0.804	0.859	0.921	0.914	0.808	0.982
SBMR-RF-W8-noTrend-D0-UG	0.718	0.922	0.805	0.849	0.891	0.898	0.803	0.985
SBMR-RF-W9-trend-D0-UG	0.720	0.927	0.811	0.824	0.914	0.916	0.803	0.981

Table A4. Total execution time using different windows' configurations.

Dataset Algorithm	Crop	FordA	FordB	HandOutlines	NonInvasiveFetalECGThorax1	NonInvasiveFetalECGThorax2	PhalangesOutlinesCorrect	TwoPatterns
SBMR-MB-K-BOSS-VS-W1-D0-BG	86.356	20.894	18.920	59.325	25.624	26.825	6.214	16.277
SBMR-MB-K-BOSS-VS-W1-trend-D0-BG	200.633	41.447	37.514	111.834	51.802	50.080	12.117	35.905
SBMR-MB-K-BOSS-VS-W1-trend-D0-UG	63.766	34.822	32.983	105.891	40.918	41.350	9.537	19.486
SBMR-MB-K-BOSS-VS-W1-noTrend-D0-UG	32.922	17.800	16.825	58.226	22.518	22.307	5.061	9.755
SBMR-RF-W1-noTrend-D0-BG	53.764	26.274	21.370	64.255	23.518	25.175	8.189	16.317
SBMR-RF-W1-trend-D0-BG	123.621	48.227	44.306	113.608	44.698	47.865	12.850	31.346
SBMR-RF-W1-trend-D0-UG	67.551	41.563	39.191	123.410	46.112	44.985	11.648	22.346
SBMR-RF-W1-D0-UG	38.886	22.346	19.385	64.721	23.268	21.385	6.192	10.634
SBMR-RF-W10-trend-D0-UG	28.821	90.495	82.290	141.950	105.976	106.154	6.003	20.042
SBMR-RF-W11-trend-D0-UG	19.471	45.179	41.175	69.448	51.510	50.943	3.553	10.818
SBMR-RF-W12-trend-D0-UG	25.852	55.476	49.940	85.222	62.440	62.680	4.545	13.631
SBMR-RF-W13-trend-D0-UG	40.024	113.978	104.673	176.298	133.941	131.057	8.028	26.638
SBMR-RF-W14-trend-D0-UG	24.444	69.885	61.702	106.377	77.706	78.166	4.887	15.989
SBMR-RF-W2-noTrend-D0-UG	66.429	34.471	32.837	86.851	45.062	45.409	9.464	18.323
SBMR-RF-W3-noTrend-D0-UG	54.580	64.919	57.721	190.971	95.157	95.112	9.929	37.914
SBMR-RF-W4-trend-D0-UG	115.171	272.809	248.441	712.521	338.855	337.439	19.800	74.526
SBMR-RF-W4-noTrend-D0-UG	55.183	135.333	123.038	369.460	164.462	163.198	10.083	36.445
SBMR-RF-W5-noTrend-D0-UG	34.064	91.950	82.449	239.261	83.306	85.705	5.608	18.888
SBMR-RF-W6-trend-D0-UG	25.724	66.809	50.068	103.188	78.103	77.710	5.087	15.794
SBMR-RF-W7-noTrend-D0-UG	27.869	76.878	68.603	116.077	86.834	87.152	5.637	17.672
SBMR-RF-W8-noTrend-D0-BG	58.735	129.312	117.890	196.222	148.158	146.636	9.260	31.182
SBMR-RF-W8-trend-D0-BG	130.044	264.569	241.091	404.723	305.697	300.872	20.342	67.718
SBMR-RF-W8-trend-D0-UG	70.762	251.439	226.320	408.547	302.159	297.023	17.237	58.612
SBMR-RF-W8-D0-UG	34.509	117.547	106.560	183.777	137.370	136.876	8.005	26.503
SBMR-RF-W9-trend-D0-D0-UG	55.596	197.256	200.075	294.347	220.346	219.082	11.669	40.845

Table A5. Accuracy results with different windows and dilation configurations.

Dataset Algorithm	Crop	FordA	FordB	HandOutlines	NonInvasiveFetalECGThorax1	NonInvasiveFetalECGThorax2	PhalangesOutlinesCorrect	TwoPatterns
SBMR-RF-W10-trend-D6-BG	0.722	0.930	0.793	0.914	0.899	0.924	0.814	0.924
SBMR-RF-W11-trend-D3-BG	0.724	0.926	0.774	0.916	0.888	0.915	0.810	0.916
SBMR-RF-W11-trend-D6-BG	0.723	0.926	0.775	0.924	0.894	0.917	0.803	0.891
SBMR-RF-W14-trend-D3-BG	0.719	0.932	0.783	0.911	0.890	0.928	0.818	0.941
SBMR-RF-W14-trend-D6-BG	0.725	0.920	0.780	0.911	0.900	0.924	0.816	0.929
SBMR-RF-W8-trend-D6-BG	0.730	0.928	0.788	0.911	0.922	0.942	0.819	0.959

Table A6. Total execution time results with different windows and dilation configurations.

Dataset Algorithm	Crop	FordA	FordB	HandOutlines	NonInvasiveFetalECGThorax1	NonInvasiveFetalECGThorax2	PhalangesOutlinesCorrect	TwoPatterns
SBMR-RF-W10-D6-trend-bigram	279.795	250.885	230.631	333.031	270.118	274.105	22.243	83.030
SBMR-RF-W11-D3-trend-bigram	402.083	206.374	187.194	255.903	219.389	212.974	20.059	104.887
SBMR-RF-W11-D6-trend-bigram	196.900	128.388	117.495	158.459	132.424	129.837	12.071	55.198
SBMR-RF-W14-D3-trend-bigram	328.484	257.580	235.637	343.221	281.788	279.821	24.589	95.934
SBMR-RF-W14-D6-trend-bigram	179.336	176.598	156.272	233.769	186.544	186.198	16.465	58.888
SBMR-RF-W8-D6-trend-bigram	548.205	564.149	514.789	815.115	619.852	621.876	50.155	178.285

Table A7. Accuracy using ridge regression classifier without multiple window configurations.

Dataset Algorithm	Crop	FordA	FordB	HandOutlines	NonInvasiveFetalECGThorax1	NonInvasiveFetalECGThorax2	PhalangesOutlinesCorrect	TwoPatterns
SBMR-RidgeCV-W0-noTrend-D0-UG	0.579	0.885	0.665	0.668	0.743	0.796	0.733	0.877
SBMR-RidgeCV-W0-trend-D0-UG	0.622	0.914	0.743	0.649	0.830	0.843	0.752	0.891
SBMR-RidgeCV-W0-trend-D1-BG	0.717	0.936	0.721	0.886	0.921	0.940	0.787	0.960
SBMR-RidgeCV-W0-trend-D2-BG	0.711	0.923	0.715	0.886	0.917	0.939	0.791	0.958
SBMR-RidgeCV-W0-trend-D3-BG	0.701	0.923	0.749	0.857	0.910	0.933	0.781	0.960
SBMR-RidgeCV-W0-trend-D4-BG	0.696	0.915	0.728	0.838	0.896	0.912	0.779	0.953
SBMR-RidgeCV-W0-trend-D5-BG	0.701	0.934	0.748	0.770	0.909	0.929	0.779	0.959
SBMR-RidgeCV-W0-trend-D6-BG	0.699	0.924	0.741	0.832	0.907	0.926	0.788	0.936
SBMR-RidgeCV-W0-trend-D6-UG	0.648	0.921	0.737	0.746	0.892	0.913	0.755	0.929

Table A8. Total execution time using ridge regression classifier without multiple window configurations.

Dataset Algorithm	Crop	FordA	FordB	HandOutlines	NonInvasiveFetalECGThorax1	NonInvasiveFetalECGThorax2	PhalangesOutlinesCorrect	TwoPatterns
SBMR-RidgeCV-W0-noTrend-D0-UG	8.016	13.956	12.343	19.992	15.528	15.329	1.144	3.432
SBMR-RidgeCV-W0-trend-D0-UG	15.590	27.439	24.766	37.587	28.384	29.850	2.043	6.809
SBMR-RidgeCV-W0-trend-D1-BG	151.967	158.446	145.818	209.656	182.595	186.436	13.105	39.796
SBMR-RidgeCV-W0-trend-D2-BG	121.808	117.256	105.820	159.941	128.159	128.951	9.795	29.982
SBMR-RidgeCV-W0-trend-D3-BG	99.919	90.811	82.360	124.798	97.516	99.381	8.204	22.887
SBMR-RidgeCV-W0-trend-D4-BG	132.663	63.136	60.237	79.282	63.763	61.836	6.260	14.825
SBMR-RidgeCV-W0-trend-D5-BG	121.146	67.777	61.386	81.091	63.431	62.849	5.928	15.386
SBMR-RidgeCV-W0-trend-D6-BG	135.884	60.460	54.292	80.885	62.658	62.774	5.674	14.681
SBMR-RidgeCV-W0-trend-D6-UG	23.037	47.292	42.238	72.090	54.201	54.692	3.789	12.554

Table A9. Accuracy using ridge regression classifier with different windows' configurations.

Dataset Algorithm	Crop	FordA	FordB	HandOutlines	NonInvasiveFetalECGThorax1	NonInvasiveFetalECGThorax2	PhalangesOutlinesCorrect	TwoPatterns
SBMR-RidgeCV-W11-trend-D6-BG	0.750	0.944	0.821	0.941	0.922	0.944	0.790	0.941
SBMR-RidgeCV-W11-trend-D0-BG	0.729	0.938	0.817	0.743	0.885	0.903	0.812	0.924
SBMR-RidgeCV-W11-trend-D0-UG	0.704	0.932	0.810	0.714	0.868	0.884	0.779	0.930
SBMR-RidgeCV-W14-trend-D6-BG	0.729	0.942	0.820	0.927	0.920	0.940	0.819	0.956
SBMR-RidgeCV-W15-trend-D6-BG	0.741	0.938	0.794	0.935	0.912	0.929	0.817	0.906
SBMR-RidgeCV-W15-trend-BG	0.721	0.931	0.807	0.754	0.869	0.897	0.775	0.895
SBMR-RidgeCV-W8-trend-D6-BG	0.747	0.939	0.810	0.938	0.936	0.955	0.830	0.984
SBMR-RidgeCV-W0-trend-D6-BG	0.699	0.924	0.741	0.832	0.907	0.926	0.788	0.936

Table A10. Total execution time using ridge regression classifier with different windows' configurations.

Dataset Algorithm	Crop	FordA	FordB	HandOutlines	NonInvasiveFetalECGThorax1	NonInvasiveFetalECGThorax2	PhalangesOutlinesCorrect	TwoPatterns
SBMR-RidgeCV-W11-trend-D6-BG	259.304	139.684	128.264	164.312	137.236	135.372	12.777	55.314
SBMR-RidgeCV-W11-trend-D0-BG	134.422	68.572	62.969	82.938	64.841	64.785	6.752	22.797
SBMR-RidgeCV-W11-trend-D0-UG	35.636	52.451	48.510	76.236	59.065	58.964	5.114	13.579
SBMR-RidgeCV-W14-trend-D6-BG	247.075	179.955	164.707	237.269	190.337	189.244	16.639	58.201
SBMR-RidgeCV-W15-trend-D6-BG	200.779	80.733	75.727	80.190	72.676	72.613	7.627	41.114
SBMR-RidgeCV-W15-trend-D0-BG	112.878	41.061	38.141	41.757	34.921	35.401	4.238	16.034
SBMR-RidgeCV-W8-trend-D6-BG	577.872	604.649	545.924	845.690	664.549	665.130	52.951	183.486
SBMR-RidgeCV-W0-trend-D6-BG	158.734	71.131	64.672	91.027	76.762	75.718	6.973	16.592

Table A11. Accuracy of state-of-the-art algorithms.

Dataset Algorithm	Crop	FordA	FordB	HandOutlines	NonInvasiveFetalECGThorax1	NonInvasiveFetalECGThorax2	PhalangesOutlinesCorrect	TwoPatterns
MRSQM_nsax0_nsfa1	0.652	0.944	0.815	0.927	0.903	0.929	0.733	0.989
MRSQM_nsax0_nsfa5	0.703	0.948	0.816	0.943	0.935	0.950	0.816	0.999
MRSQM_nsax1_nsfa0	0.623	0.874	0.736	0.911	0.901	0.919	0.759	0.987
MRSQM_nsax1_nsfa1	0.679	0.942	0.812	0.954	0.916	0.933	0.786	0.998
MiniRocket	0.748	0.948	0.819	0.943	0.949	0.965	0.834	0.998
Rocket	0.753	0.943	0.805	0.949	0.951	0.968	0.838	1.000
WEASEL_V2	0.760	0.961	0.840	0.959	0.927	0.955	0.832	0.997

Table A12. Total execution time of state-of-the-art algorithms.

Dataset Algorithm	Crop	FordA	FordB	HandOutlines	NonInvasiveFetalECGThorax1	NonInvasiveFetalECGThorax2	PhalangesOutlinesCorrect	TwoPatterns
MRSQM_nsax0_nsfa1	264.978	104.805	90.142	138.350	153.971	153.768	9.630	22.638
MRSQM_nsax0_nsfa5	689.978	380.430	360.333	669.216	584.349	562.895	38.789	65.097
MRSQM_nsax1_nsfa0	211.168	104.785	100.683	137.459	121.358	120.781	11.651	25.750
MRSQM_nsax1_nsfa1	306.893	158.834	149.386	277.161	235.267	234.790	15.556	32.008
MiniRocket	70.201	18.755	18.277	11.860	11.730	12.059	3.624	3.249
Rocket	221.754	368.653	347.407	555.628	444.713	411.419	35.938	89.637
WEASEL_V2	251.385	420.989	389.773	651.314	445.368	433.097	35.586	90.074

References

1. Chaovalitwongse, W.A.; Prokopyev, O.A.; Pardalos, P.M. Electroencephalogram (EEG) time series classification: Applications in epilepsy. *Ann. Oper. Res.* **2006**, *148*, 227–250. [CrossRef]
2. Arul, M.; Kareem, A. Applications of shapelet transform to time series classification of earthquake, wind and wave data. *Eng. Struct.* **2021**, *228*, 111564. [CrossRef]
3. Potamitis, I. Classifying insects on the fly. *Ecol. Inform.* **2014**, *21*, 40–49. [CrossRef]
4. Susto, G.A.; Cenedese, A.; Terzi, M. Chapter 9—Time-Series Classification Methods: Review and Applications to Power Systems Data. In *Big Data Application in Power Systems*; Arghandeh, R., Zhou, Y., Eds.; Elsevier: Amsterdam, The Netherlands, 2018; pp. 179–220. [CrossRef]
5. Schäfer, P. Scalable time series classification. *Data Min. Knowl. Discov.* **2016**, *30*, 1273–1298. [CrossRef]
6. Shifaz, A.; Pelletier, C.; Petitjean, F.; Webb, G.I. TS-CHIEF: A scalable and accurate forest algorithm for time series classification. *Data Min. Knowl. Discov.* **2020**, *34*, 742–775. [CrossRef]
7. Dempster, A.; Petitjean, F.; Webb, G.I. ROCKET: Exceptionally fast and accurate time series classification using random convolutional kernels. *Data Min. Knowl. Discov.* **2020**, *34*, 1454–1495. [CrossRef]
8. Dempster, A.; Schmidt, D.F.; Webb, G.I. Minirocket: A very fast (almost) deterministic transform for time series classification. In Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, Singapore, 14–18 August 2021; pp. 248–257.
9. Glenis, A.; Vouros, G.A. SCALE-BOSS: A framework for scalable time-series classification using symbolic representations. In Proceedings of the 12th Hellenic Conference on Artificial Intelligence, Corfu, Greece, 7–9 September 2022; pp. 1–9.
10. Tan, C.W.; Dempster, A.; Bergmeir, C.; Webb, G.I. MultiRocket: Multiple pooling operators and transformations for fast and effective time series classification. *Data Min. Knowl. Discov.* **2022**, *36*, 1623–1646. [CrossRef]
11. Schäfer, P.; Leser, U. WEASEL 2.0—A Random Dilated Dictionary Transform for Fast, Accurate and Memory Constrained Time Series Classification. *arXiv* **2023**, arXiv:2301.10194.
12. Senin, P.; Malinchik, S. Sax-vsm: Interpretable time series classification using sax and vector space model. In Proceedings of the 2013 IEEE 13th International Conference on Data Mining, Dallas, TX, USA, 7–10 December 2013; pp. 1175–1180.
13. Schäfer, P. The BOSS is concerned with time series classification in the presence of noise. *Data Min. Knowl. Discov.* **2015**, *29*, 1505–1530. [CrossRef]
14. Schäfer, P.; Högqvist, M. SFA: A symbolic fourier approximation and index for similarity search in high dimensional datasets. In Proceedings of the 15th International Conference on Extending Database Technology, Berlin, Germany, 27–30 March 2012; pp. 516–527.
15. Glenis, A.; Vouros, G.A. Balancing between scalability and accuracy in time-series classification for stream and batch settings. In Proceedings of the International Conference on Discovery Science, Thessaloniki, Greece, 19–21 October 2020; pp. 265–279.
16. Middlehurst, M.; Vickers, W.; Bagnall, A. Scalable dictionary classifiers for time series classification. In Proceedings of the International Conference on Intelligent Data Engineering and Automated Learning, Manchester, UK, 14–16 November 2019; pp. 11–19.
17. Large, J.; Bagnall, A.; Malinowski, S.; Tavenard, R. On time series classification with dictionary-based classifiers. *Intell. Data Anal.* **2019**, *23*, 1073–1089. [CrossRef]
18. Schäfer, P.; Leser, U. Fast and accurate time series classification with weasel. In Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, Singapore, 6–10 November 2017; pp. 637–646.
19. Nguyen, T.L.; Ifrim, G. MrSQM: Fast time series classification with symbolic representations. *arXiv* **2021**, arXiv:2109.01036.
20. Nguyen, T.L.; Ifrim, G. Fast time series classification with random symbolic subsequences. In Proceedings of the International Workshop on Advanced Analytics and Learning on Temporal Data, Grenoble, France, 19–23 September 2022; pp. 50–65.
21. Middlehurst, M.; Large, J.; Cawley, G.; Bagnall, A. The temporal dictionary ensemble (TDE) classifier for time series classification. In Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Ghent, Belgium, 14–18 September 2020; pp. 660–676.
22. Bostrom, A.; Bagnall, A. Binary shapelet transform for multiclass time series classification. In *Transactions on Large-Scale Data-and Knowledge-Centered Systems XXXII*; Springer: Berlin/Heidelberg, Germany, 2017; pp. 24–46.
23. Lucas, B.; Shifaz, A.; Pelletier, C.; O'Neill, L.; Zaidi, N.; Goethals, B.; Petitjean, F.; Webb, G.I. Proximity forest: An effective and scalable distance-based classifier for time series. *Data Min. Knowl. Discov.* **2019**, *33*, 607–635. [CrossRef]
24. Mahato, V.; O'Reilly, M.; Cunningham, P. A Comparison of k-NN Methods for Time Series Classification and Regression. In Proceedings of the AICS, Dublin, Ireland, 6–7 December 2018; pp. 102–113.
25. Ester, M.; Kriegel, H.P.; Sander, J.; Xu, X. A density-based algorithm for discovering clusters in large spatial databases with noise. In Proceedings of the kdd, Portland, OR, USA, 2–4 August 1996; Volume 96, pp. 226–231.
26. Schubert, E.; Sander, J.; Ester, M.; Kriegel, H.P.; Xu, X. DBSCAN revisited, revisited: Why and how you should (still) use DBSCAN. *ACM Trans. Database Syst.* **2017**, *42*, 1–21. [CrossRef]
27. Zhang, T.; Ramakrishnan, R.; Livny, M. BIRCH: An efficient data clustering method for very large databases. *ACM Sigmod Rec.* **1996**, *25*, 103–114. [CrossRef]
28. Sculley, D. Web-scale k-means clustering. In Proceedings of the 19th International Conference on World Wide Web, Raleigh, NC, USA, 26–30 April 2010; pp. 1177–1178.

29. Park, H.S.; Jun, C.H. A simple and fast algorithm for K-medoids clustering. *Expert Syst. Appl.* **2009**, *36*, 3336–3341. [CrossRef]
30. Faouzi, J.; Janati, H. pyts: A Python Package for Time Series Classification. *J. Mach. Learn. Res.* **2020**, *21*, 1–6.
31. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
32. Zaharia, M.; Xin, R.S.; Wendell, P.; Das, T.; Armbrust, M.; Dave, A.; Meng, X.; Rosen, J.; Venkataraman, S.; Franklin, M.J.; et al. Apache spark: A unified engine for big data processing. *Commun. ACM* **2016**, *59*, 56–65. [CrossRef]