Fast and Accurate Time Series Classification Through Supervised Interval Search

Nestor Cabello
The University of Melbourne
Melbourne, Australia
ncabello@student.unimelb.edu.au

Elham Naghizade
RMIT University
Melbourne, Australia
e.naghizade@rmit.edu.au

Jianzhong Qi
The University of Melbourne
Melbourne, Australia
jianzhong.qi@unimelb.edu.au

Lars Kulik
The University of Melbourne
Melbourne, Australia
lkulik@unimelb.edu.au

***Abstract*—*Time series classification* (TSC) aims to predict the class label of a given time series. Modern applications such as *appliance modelling* require to model an abundance of long time series, which makes it difficult to use many state-of-the-art TSC techniques due to their high computational cost and lack of interpretable outputs. To address these challenges, we propose a novel TSC method: the *Supervised Time Series Forest* (STSF). STSF improves the classification efficiency by examining only a (set of) sub-series of the original time series, and its tree-based structure allows for interpretable outcomes. STSF adapts a top-down approach to search for relevant sub-series in three different time series representations prior to training any tree classifier, where the relevance of a sub-series is measured by feature ranking metrics (i.e., supervision signals). Experiments on extensive real datasets show that STSF achieves comparable accuracy to state-of-the-art TSC methods while being significantly more efficient, enabling TSC for long time series.**

***Index Terms*—Time series classification, Interval-based classifier, Feature selection.**

I. Introduction

Time series classification (TSC) aims to predict the class label of a given time series (or parts of it). A time series is an ordered time-stamped sequence of data points (i.e., observations from a variable of interest). Various TSC methods have been proposed for a rich set of application areas such as medicine (e.g., classification of electrocardiograms [1]). Modern applications such as *appliance modelling* and *stress detection* pose significant challenges to the existing TSC methods because (i) their data is usually collected via sensors, resulting in large sets of time series with high update frequencies which are also prone to signal noise (e.g., sensor defects) and (ii) they require interpretable models. For example, a classifier for appliance modelling should report the time intervals that differentiate the time series from different types of appliances, enabling householders to understand their energy consumption.

Ideally, a TSC method fulfills three criteria: it is accurate, efficient and interpretable. However, accurate TSC methods suffer in either efficiency and/or interpretability for modern applications that usually involve long time series. For example, the state-of-the-art TSC methods, *hierarchical vote collective of transformation-based ensembles* (HIVE-COTE) [2] and *time series combination of heterogeneous and integrated embedding forest* (TS-CHIEF) [3] have bi-quadratic and quadratic time complexity in the length of the series, respectively. None of these methods provide *interpretable* classification results.

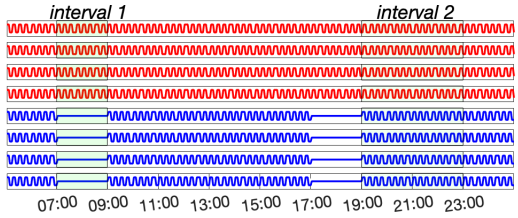

Fig. 1. Discriminatory *interval 1* versus non-discriminatory *interval 2*.

To address *efficiency and interpretability*, novel techniques known as *interval-based* classifiers have focused on intervals rather than the whole signal. *Time series forest* (TSF) [4] is the most accurate interval-based method and one of the fastest TSC methods [5]. TSF randomly selects a number of intervals (i.e., sub-series). Data points within each interval are aggregated with statistical measurements (e.g., mean). The original series are thus transformed into a representation of intervals. The classification is performed using an ensemble of trees, which allows for interpretability.

Motivated by the better efficiency of interval-based methods and the intrinsic capability of providing interpretable classifications through the identification of *discriminatory intervals* (cf. Figure 1), we also build our time series classifier based on intervals. As Fig. 1 shows, *interval 1* is an example of a discriminatory interval. This interval differentiates the blue time series from the red ones. In contrast, *interval 2* is non-discriminatory because it cannot separate the red and the blue series. When classifying large datasets with long series, TSF may become computationally expensive due to the larger number of intervals to be explored. We show that a "supervised" selection of intervals is significantly more efficient through training an ensemble of trees with a pre-computed set of discriminatory intervals.

Our algorithm, *Supervised Time Series Forest* (STSF), opportunistically creates a time series forest for classification and feature extraction based on such discriminatory intervals. For a given set of time series with length m, the number of possible interval features is quadratic to the length of the series, i.e., $\mathcal{O}(m^2)$. Using a binary-inspired search strategy jointly with a feature ranking metric (i.e., Fisher score [6]), STSF reduces

the interval feature space to $\mathcal{O}(\log m)$. The main contributions of this paper are summarized as follows:

- We propose STSF, a supervised interval search technique with feature ranking metrics that is a highly efficient and accurate interval-based TSC method. It minimizes unnecessary computations when finding relevant intervals, making it an order of magnitude faster than TSF.
- By extracting relevant intervals from the trained trees, the generated regions of interest, i.e., groups of relevant intervals, highlight the time periods where differences in shape, distribution, or level, between time series of different classes are more pronounced, and thus provide interpretable outcomes of the classification.
- An extensive experimental study on 85 real datasets [7] shows that STSF achieves classification accuracies competitive to state-of-the-art methods but it is at least two orders of magnitude faster than them. This makes STSF suitable for large datasets with long time series.

II. Related Work

Time series classification (TSC) methods can be broadly categorised into *instance-based*, *feature-based*, and *ensemble-based*. Instance-based methods such as 1-Nearest Neighbour classifier with *Dynamic Time Warping* (1-NN DTW) [8] classify a time series according to the similarity of its ordered data points to those of time series with known class labels, where localized distortions of the data points are allowed. These methods tend to be less accurate than feature-based classifiers and computationally inefficient for long time series (i.e., quadratic to the length of the series).

Feature-based methods such as *Shapelet Transform* (ST) [9], *bag-of-SFA-symbols* (BOSS) [10], and *time series forest* (TSF) [4] represent the raw time series with a set of derived properties, i.e., features. ST uses shapelets, i.e., sub-series that are representative of class membership, and time series are classified according to their similarity to the (discriminatory) shapelets. BOSS uses the relative frequency of discriminatory sub-series for classification. It computes the number of times that a discriminatory sub-series (represented as a word) appears in the time series. ST is expensive with a bi-quadratic time complexity in the length of the series whereas BOSS has a word length complexity in the size of the alphabet, which makes it memory intensive for long series. TSF randomly selects a number of intervals to extract discriminatory features using statistical measurements. The idea is that time series from the same class tend to have intervals with similar characteristics. The *random interval spectral ensemble* (RISE) was recently introduced [2] to capture the frequency-domain features. RISE works similar to TSF, but extracts spectral features over each random interval instead of statistical measurements as in TSF. Both TSF and RISE are highly efficient but usually less accurate than other TSC methods.

Ensemble-based methods use ensembles of individual TSC methods. Two representatives are the *elastic ensemble* (EE) [11] and the *hierarchical vote collective of transformation-based ensembles* (HIVE-COTE) [2]. Both approaches are highly accurate but costly. They may become too expensive on long series. EE is based on 11 instance-based TSC methods, hence it has a quadratic time complexity in the length of the time series. HIVE-COTE uses EE, ST, BOSS, TSF, and RISE for classification. HIVE-COTE is the state-of-the-art TSC method in terms of accuracy but is impractical for long series (bi-quadratic time complexity in the length of the time series). *Proximity forest* (PF) [12] and *time series combination of heterogeneous and integrated embedding forest* (TS-CHIEF) [3] focus on providing highly accurate and scalable classifications. PF builds an ensemble of proximity trees using elastic distance measures as splitting criteria. While PF is more scalable (quasi-linear time complexity in the number of time series) than HIVE-COTE, it is significantly less accurate. TS-CHIEF builds on PF and incorporates BOSS and RISE features as splitting criteria. TS-CHIEF is statistically similar to HIVE-COTE in classification accuracy, but more scalable (similar to PF). PF and TS-CHIEF are quadratic to the series length, which make them costly for long series.

Deep learning classifiers such as *fully convolutional networks* (FCN) and *residual networks* (ResNet) [13] obtain competitive classification accuracy and allow interpretations on the model decisions. However, they are also computationally expensive for long series and require GPU.

III. Preliminaries

Interval-based approaches extract sub-series for which aggregates (e.g., mean and standard deviation) are computed and used as features (i.e., interval features). To allow for interpretable classifications, we focus on *phase-dependent intervals*, i.e., discriminatory features located at the same time regions over all time series in a given dataset.

Interval feature. *Given a set of time series* $X = \{\boldsymbol{x}^1; \boldsymbol{x}^2; \boldsymbol{x}^3; \dots; \boldsymbol{x}^n\}$, *where* $\boldsymbol{x}^i = \{x_1^i; x_2^i; \dots; x_m^i\}$, *an aggregation function* $f(\,)$, *and an interval (s,e), an interval feature* $\boldsymbol{a} = f(X; s; e)$ *is a vector of length n, defined as follows:*

$$\boldsymbol{a} = \{f(\boldsymbol{x}^1; s; e); f(\boldsymbol{x}^2; s; e); f(\boldsymbol{x}^3; s; e); \dots; f(\boldsymbol{x}^n; s; e)\}$$

where $f(\boldsymbol{x}^i; s; e) = f(\{x_s^i; x_{s+1}^i; \dots; x_{e\ 1}^i; x_e^i\})_{1 \leqslant s \leqslant e \leqslant m}$.

With s=2, e=4, f=mean, an interval feature $\boldsymbol{a} = \text{mean}(X; 2; 4)$ is represented with the dashed rectangle as follows:

$$X = \begin{pmatrix} x_1^1 & \boxed{x_2^1} & \boxed{x_3^1} & \boxed{x_4^1} & x_5^1 & x_6^1 & \dots & x_{m\ 1}^1 & x_m^1 \\ x_1^2 & \boxed{x_2^2} & \boxed{x_3^2} & \boxed{x_4^2} & x_5^2 & x_6^2 & \dots & x_{m\ 1}^2 & x_m^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_1^n & \boxed{x_2^n} & \boxed{x_3^n} & \boxed{x_4^n} & x_5^n & x_6^n & \dots & x_{m\ 1}^n & x_m^n \end{pmatrix} \begin{matrix} \\ \rightarrow \text{mean}(\{x_2^1; x_3^1; x_4^1\}) \\ \\ \\ \rightarrow \text{mean}(\{x_2^n; x_3^n; x_4^n\}) \end{matrix}$$

Problem statement. We consider a set of n univariate time series $X = \{\boldsymbol{x}^1; \boldsymbol{x}^2; \dots; \boldsymbol{x}^n\}$, where each time series $\boldsymbol{x}^i = \{x_1^i; x_2^i; \dots; x_m^i\}$ has m ordered real-valued observations, sampled at equally-spaced time intervals. Each time series $\boldsymbol{x}^i$ is also associated with a class label y^i. We aim to find the set of interval features that yield the highest time series class

prediction accuracy. Finding such a set of interval features is NP-hard. For a time series of length m, there are $\mathcal{O}(m^2)$ different intervals, and hence $\mathcal{O}(2^{m^2})$ subsets of intervals. For a large m, it is prohibitively expensive to explore all subsets. We present an efficient heuristic to avoid the exhaustive search while retaining a high classification accuracy.

IV. Our Approach

We take a stochastic optimization approach to select a set of interval features with a high discriminating power $\bar{A}$ (i.e., *candidate discriminatory interval features*) from the high dimensional interval feature space A (i.e., all $\mathcal{O}(m^2)$ possible interval features). For a time series of length m, we reduce the interval feature space size to $\mathcal{O}(\log m)$. We search for the best interval feature subset A (i.e., *discriminatory interval features*) through an ensemble of decision trees, which has a time complexity $\mathcal{O}(r\ n\ \log n\ \log m)$, where r is the total number of trees in the ensemble, and n is the number of time series instances. Fig. 2 shows an overview of STSF when training a single tree. For a given training set X, a periodogram representation of X, and a derivative representation of X, the possible number of interval features for each representation is denoted as A_O, A_F, and A_D, respectively. For each interval feature set A_O, A_F, and A_D, STSF selects a group of candidate discriminatory interval features $\bar{A}_O$, $\bar{A}_F$, and $\bar{A}_D$. Lastly, a tree classifier, due to its intrinsic feature selection capability, enables the selection of a set of discriminatory intervals A with which classification is performed on the testing time series. Features from A may be used to add interpretability to the classification task as discussed in Section VI.

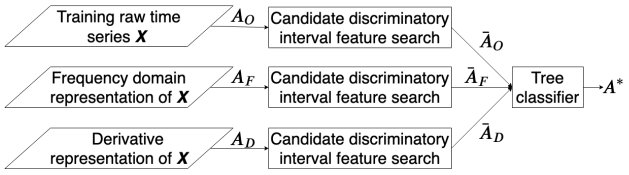

Fig. 2. Overview of STSF. Sets of candidate discriminatory interval features $\bar{A}_O;\bar{A}_F;\bar{A}_D$ are selected from the high dimensional interval feature sets $A_O;A_F;A_D$, respectively. The tree classifier uses the candidate sets to select a set of discriminatory interval features A to further perform the TSC task.

A. Time series representation

We first discuss time series representations to extract intervals features. We use intervals from original (i.e., time domain), periodogram (i.e., frequency domain), and derivative representations. We focus on the latter two representations.

Periodogram representation: Several TSC methods [2], [14] use the *periodogram representation* when looking for time series similarities in the frequency domain. We adopt this and exploit the periodogram representation of each time series derived from the *discrete Fourier transform*. A side benefit of this representation is that it helps to *indirectly* detect *phase-independent discriminatory intervals,* i.e., discriminatory features located at different time regions of the original series as described in Fig. 3.

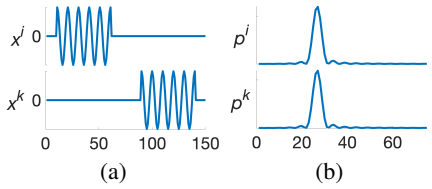

Fig. 3. **(a)** Time series x^i and x^k, with a same class label (i.e., $y^i = y^k$) present a similar sub-series but located at different time regions (i.e., phase-independent). **(b)** The periodogram representation p^i and p^k of series x^i and x^k, respectively. Our algorithm searches for discriminatory phase-dependent interval features and cannot find discriminatory intervals that identify both series as similar. The periodogram representation provides more flexibility as it considers the frequency of the discriminatory sub-series (ignoring its location in time) and thus helps to identify discriminatory sub-series even when they appear at different locations in time across different time series.

Derivative representation: Using a (first-order) difference representation of a given time series rather than the original time series improves the classification accuracy [15] as it provides trend information.

B. Selecting Candidate Discriminatory Intervals

Our approach to find candidate discriminatory intervals first partitions a given time series set X of size $n \quad m$ into two subsets: (i) X_L of size $n \quad u$, and (ii) X_R of size $n \quad m \quad u$. The value of u is randomly selected from the set of numbers $f1;2;:::;mg$. The initial random partition (according to u) enables STSF to explore more diverse sub-series (i.e., at different locations and of different lengths). Next, a set of interval features is extracted in a supervised manner independently from sub-sets X_L and X_R using Algorithm 1. This last step runs for each aggregation function detailed later. Note that the selection of candidate discriminatory intervals runs on each of the time series representations discussed in Section IV-A.

Our supervised search algorithm for the candidate discriminatory intervals is summarized in Algorithm 1. The algorithm recursively breaks a given interval into two equally-sized intervals and computes interval features for the two resultant intervals (Line 4). The *Fisher score* [6] (detailed below) is further computed for each interval feature (Line 5). The interval feature with a higher score is added to the set of candidate discriminatory interval features (the starting and ending time indices for each interval, and the aggregation function used to represent the interval feature are also stored), and the search continues within this interval (Lines 6 to 12). The algorithm stops when the interval cannot be partitioned further (with less than two points, Lines 1 to 3).

Feature ranking metric: The Fisher score of an interval feature indicates how well the feature separates a class of time series from the other classes. For a given vector of class labels $\boldsymbol{y}\ 2\ f1;2;:::;cg^n$, the Fisher score of an interval feature $\boldsymbol{a}$ is computed as $\text{FisherScore}(\boldsymbol{a};\boldsymbol{y}) = \sum_{k=1}^{c} n_k(\mu_k^{\boldsymbol{a}} \ \mu^{\boldsymbol{a}})^2 = \sum_{k=1}^{c} n_k(\sigma_k^{\boldsymbol{a}})^2$. Here, $\mu^{\boldsymbol{a}}$ is the overall mean of the elements in $\boldsymbol{a}$; $\mu_k^{\boldsymbol{a}}$ and $\sigma_k^{\boldsymbol{a}}$ are the mean and standard deviation of the elements in $\boldsymbol{a}$ labelled with the k-th class; and n_k is the number of time series labelled with the k-th class. We use the Fisher score (instead of other metrics) for its fast computation.

Algorithm 1: SupervisedSearch

Input: X^0: set of n time series of length m^0; y: class label vector; f: aggregation function; fr: feature ranking metric; $\bar{A}$: set of candidate discriminatory intervals.

```
if m^0 < 2 then
    return Ā;
else
    a_L  f(X^0; 1; m^0=2); a_R  f(X^0; m^0=2; m);
    score_L  fr(a_L; y); score_R  fr(a_R; y);
    if score_L >= score_R then
        Ā  Ā [ {a_L};
        SupervisedSearch(X^0(1 : m^0=2); y; f; fr; Ā);
    else
        Ā  Ā [ {a_R};
        SupervisedSearch(X^0(m^0=2 : m); y; f; fr; Ā);
    end
end
return Ā;
```

C. Classification with Discriminatory Intervals

STSF uses an ensemble of trees for classification. To train a tree, STSF uses the set of $\mathcal{O}(\log m)$ candidate discriminatory intervals previously extracted from the series (detailed in Section IV-B). In each tree node, we split the node according to the feature with the highest *information gain* (IG). If the entropy of the node is 0, we label the node as a leaf node. The training process of STSF is less expensive than that in TSF. In TSF, as stated in the original paper: "In each time series tree node, we consider randomly sampling $\mathcal{O}(\sqrt{m})$ interval sizes and $\mathcal{O}(\sqrt{m})$ starting positions". Thus, in TSF, every node (of each tree) has to compute $\mathcal{O}(\sqrt{m})$ new features to split the node, whereas in STSF each node uses the same $\mathcal{O}(\log m)$ features for the split. When classifying a new time series, the class is predicted by a majority vote from the tree ensemble.

V. Experiments

We evaluate our approach in two sets of experiments: classification accuracy (i.e., effectiveness) and training time (i.e., computational efficiency). For each set of experiments, we describe the experimental setup and present the results. STSF is implemented in Matlab. All experiments are run on a macOS 10.14.3 system with a dual core CPU (i5, 2.3GHz) using a single thread. We provide the full raw results and our code at `https://github.com/stevcabello/STSF`.

A. Classification Accuracy

Experimental Setup: STSF is tested with a set of 85 benchmark datasets [7]. Each dataset provides a default train and test split. We compare with TSF [4], RISE [2], HIVE-COTE [2], TS-CHIEF [3], PF [12], BOSS [10], FCN [13], ResNet [13], and 1NN-DTW [8]. STSF and TSF are trained with 500 trees. STSF uses the mean, standard deviation (std), median, interquartile range (iqr), minimum (min), maximum (max), and slope aggregation functions. Similar to every non-deterministic competitor (e.g., TSF, PF, and TS-CHIEF), we compute the average classification accuracy over 10 runs of STSF for each dataset.

Results: Table I shows the classification accuracy for STSF and the competitors. STSF shows a competitive average accuracy of 82.60, which is 4% higher than the best existing interval-based method TSF. STSF ranks higher (on average) than 1NN-DTW, TSF, RISE, PF, FCN and BOSS. Only ResNet, HIVE-COTE, and TS-CHIEF rank higher than STSF. However, as the critical difference diagram (Fig. 4) shows, STSF is *not* significantly different from ResNet. Despite trading off accuracy for interpretability (only considering *phase-dependent* discriminatory intervals), STSF achieves a highly competitive accuracy close to the state-of-the-art. Meanwhile, as our next set of experiments will show, STSF is *at least* two order of magnitude faster than state-of-the-art TSC methods.

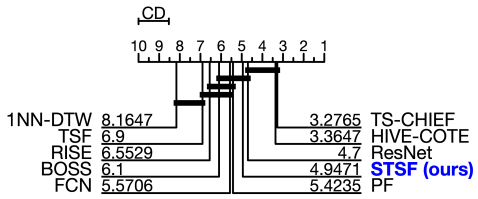

Fig. 4. Critical difference diagram of average ranks on 85 benchmark datasets. Bold lines show groups of statistically similar methods. STSF is competitive to exhaustive and state-of-the-art methods.

B. Training Time

Experimental Setup: We measured the training times of TSF, TS-CHIEF, PF, ResNet, and 1NN-DTW over 45 datasets (underlined in Table I). These methods (except for ResNet and TSF) are implemented in Java. While comparing running times of TSC methods implemented in different programming languages may seem biased, STSF is implemented purely in Matlab which is – as an interpreted language – slower than Java (used for the baselines). Parameters of each competitor are set as suggested in its original paper.

Results: In Fig. 5, we present the average ranks of each TSC method and their scaled training times. STSF is two orders of magnitude faster than ResNet and TS-CHIEF, and it is also significantly faster than PF. STSF preserves its efficiency even when classifying large datasets with long time series, whereas PF, TS-CHIEF, and ResNet scale poorly. PF uses only 100 trees instead of 500 as TSF, TS-CHIEF, and STSF do. Since PF's training time scales linearly with the number of trees, using 500 trees will make PF's time five times larger than as shown in Fig. 5. However, PF's average rank is unlikely to change. As the PF paper suggests: "It is unlikely that more trees would provide a very significant improvement, because the ratio of error-rates between 100 and 50 (trees) is already close to 1 (i.e., the errors are only slightly reduced)". Due the expensive training times of HIVE-COTE, we report the training times of HIVE-COTE when classifying 21/45 small to medium datasets. The results suggest that STSF is two orders

TABLE I

AVERAGE CLASSIFICATION ACCURACY OVER 10 RUNS OF STSF AND TSF FOR EACH DATASET. OTHER METHODS ARE AS REPORTED IN THEIR PAPERS.

Datasets	# classes	# train	# test	length	1NN-DTW	TSF	RISE	BOSS	FCN	ResNet	HIVE-COTE	PF	TS-CHIEF	STSF
Adiac	37	390	391	176	60.87	76.32	78.01	76.47	**84.40**	82.90	81.07	73.40	79.80	82.79
ArrowHead	3	36	175	251	80.00	72.57	79.43	83.43	84.30	84.50	86.29	**87.54**	83.27	67.49
Beef	5	30	30	470	66.67	86.33	83.33	80.00	69.70	75.30	**93.33**	72.00	70.61	84.00
BeetleFly	2	20	20	512	65.00	75.00	75.00	90.00	86.00	85.00	**95.00**	87.50	91.36	94.00
BirdChicken	2	20	20	512	70.00	80.00	95.00	95.00	**95.50**	88.50	85.00	86.50	90.91	90.00
Car	4	60	60	577	76.67	76.67	80.00	83.33	90.40	**92.50**	86.67	84.67	85.45	81.50
CBF	3	30	900	128	99.44	97.27	95.11	99.78	99.40	99.50	**99.89**	99.33	99.79	97.90
ChlorineConcentration	3	467	3840	166	65.00	74.93	76.85	66.09	81.40	**84.40**	71.20	63.39	71.67	78.04
CinCECGTorso	4	40	1380	1639	93.04	95.07	98.62	88.70	82.40	82.60	**99.64**	93.43	98.32	98.49
Coffee	2	28	28	286	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**
Computers	2	250	250	720	62.40	72.00	76.40	75.60	**82.20**	81.50	76.00	64.44	70.51	75.60
CricketX	12	390	390	300	77.95	64.97	69.74	73.59	79.20	79.10	**82.31**	80.21	81.38	68.33
CricketY	12	390	390	300	75.64	70.87	71.79	75.38	78.70	80.30	**84.87**	79.38	80.19	74.77
CricketZ	12	390	390	300	73.59	66.62	70.51	74.62	81.10	81.20	83.08	80.10	**83.40**	72.18
DiatomSizeReduction	4	16	306	345	93.46	94.80	92.81	93.14	31.30	30.10	94.12	96.57	**97.30**	96.63
DistalPhalanxOutlineAgeGroup	3	400	139	80	62.59	74.82	**76.26**	74.82	71.00	71.70	**76.26**	73.09	74.62	72.81
DistalPhalanxOutlineCorrect	2	600	276	80	72.46	77.17	77.54	72.83	76.00	77.10	77.17	**79.28**	78.23	78.84
DistalPhalanxTW	6	400	139	80	63.31	66.91	67.63	67.63	**69.00**	66.50	68.35	65.97	67.04	68.27
Earthquakes	2	322	139	512	72.66	74.82	74.82	74.82	72.70	71.20	74.82	75.40	74.82	**76.91**
ECG200	2	100	100	96	88.00	85.50	88.00	87.00	88.90	87.40	85.00	**90.90**	86.18	88.00
ECG5000	5	500	4500	140	92.51	93.89	93.69	94.13	94.00	93.40	**94.62**	93.65	94.54	94.21
ECGFiveDays	2	23	861	136	79.67	93.69	99.88	**100.00**	98.70	97.50	**100.00**	84.92	**100.00**	97.77
ElectricDevices	7	8926	7711	96	63.08	69.25	66.35	**79.92**	70.20	72.90	77.03	70.60	75.53	74.06
FaceAll	14	560	1690	131	80.77	76.83	76.15	78.17	**94.50**	83.90	80.30	89.38	84.14	78.85
FaceFour	4	24	88	350	89.77	98.41	89.77	**100.00**	92.80	95.50	95.45	97.39	**100.00**	97.73
FacesUCR	14	200	2050	131	90.78	90.01	87.51	95.71	94.60	95.50	96.29	94.59	**96.63**	88.59
FiftyWords	50	450	450	270	76.48	73.28	69.23	70.55	67.90	72.70	80.88	83.14	**84.50**	77.05
Fish	7	175	175	463	83.43	85.26	84.57	98.86	95.80	97.90	98.86	93.49	**99.43**	90.34
FordA	2	3601	1320	500	66.52	81.52	94.09	92.95	90.40	92.00	**96.44**	85.46	94.10	96.30
FordB	2	3636	810	500	59.88	68.77	81.11	82.00	87.80	**91.30**	82.35	71.49	82.96	79.42
GunPoint	2	50	150	150	91.33	95.07	98.00	**100.00**	**100.00**	99.10	**100.00**	99.73	**100.00**	92.00
Ham	2	109	105	431	60.00	74.29	68.57	66.67	71.80	**75.70**	66.67	66.00	71.52	73.81
HandOutlines	2	1000	370	2709	87.84	91.89	88.38	91.10	80.60	91.10	**93.24**	92.14	93.22	92.03
Haptics	5	155	308	1092	41.56	43.57	45.78	46.10	48.00	51.90	**51.95**	44.45	51.68	50.75
Herring	2	64	64	512	53.12	60.94	64.06	54.69	60.80	61.90	**68.75**	57.97	58.81	62.97
InlineSkate	7	100	550	1884	38.73	32.24	34.91	51.64	33.90	37.30	50.00	54.18	52.69	**55.47**
InsectWingbeatSound	11	220	1980	256	57.37	63.28	65.51	52.32	39.30	50.70	65.51	61.87	64.29	**66.56**
ItalyPowerDemand	2	67	1029	24	95.53	97.00	95.34	90.86	96.10	96.30	96.31	96.71	**97.06**	**97.06**
LargeKitchenAppliances	3	375	375	720	79.47	57.07	63.73	76.53	**90.20**	90.00	86.40	78.19	80.68	79.39
Lightning2	2	60	61	637	**86.89**	79.51	70.49	83.61	73.90	77.00	81.97	86.56	74.81	72.46
Lightning7	7	70	73	319	71.23	74.11	69.86	68.49	82.70	**84.50**	73.97	82.19	76.34	76.99
Mallat	8	55	2345	1024	91.43	96.46	92.15	93.82	96.70	97.20	96.20	95.76	**97.50**	96.88
Meat	3	60	60	448	93.33	93.33	93.33	90.00	85.30	**96.80**	93.33	93.33	88.79	93.17
MedicalImages	10	381	760	99	74.74	78.00	66.18	71.84	77.90	77.00	77.76	75.82	**79.58**	78.59
MiddlePhalanxOutlineAgeGroup	3	400	154	80	51.95	57.79	**59.74**	54.55	55.30	56.90	**59.74**	56.23	58.32	56.82
MiddlePhalanxOutlineCorrect	2	600	291	80	76.63	82.82	82.13	78.01	80.10	80.90	83.16	83.64	**85.35**	82.27
MiddlePhalanxTW	6	399	154	80	50.65	56.49	**59.09**	54.55	51.20	48.40	57.14	52.92	55.02	58.90
MoteStrain	2	20	1252	84	88.63	86.90	87.22	87.86	93.70	92.80	93.29	90.24	**94.75**	92.36
NonInvasiveFetalECGThorax1	42	1800	1965	750	82.90	89.97	90.13	83.82	**95.60**	94.50	93.03	90.66	91.13	93.27
NonInvasiveFetalECGThorax2	42	1800	1965	750	87.02	91.13	91.55	90.08	**95.30**	94.60	94.45	93.99	94.50	94.06
OliveOil	4	30	30	570	86.67	90.67	90.00	86.67	72.30	83.00	90.00	86.67	88.79	**93.33**
OSULeaf	6	200	242	427	59.92	58.39	64.88	95.45	97.70	97.90	97.93	82.73	**99.14**	79.83
PhalangesOutlinesCorrect	2	1800	858	80	76.11	80.30	81.35	77.16	82.00	83.90	80.65	82.35	**84.50**	83.17
Phoneme	39	214	1896	1024	22.68	21.20	35.55	26.48	32.50	33.40	**38.24**	32.01	36.91	32.52
Plane	7	105	105	144	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**
ProximalPhalanxOutlineAgeGroup	3	400	205	80	78.54	84.88	85.37	83.41	83.10	85.30	**85.85**	84.63	84.97	84.44
ProximalPhalanxOutlineCorrect	2	600	291	80	79.04	82.82	87.63	84.88	90.30	**92.10**	87.97	87.32	88.82	90.52
ProximalPhalanxTW	6	400	205	80	76.10	81.46	80.98	80.00	76.70	78.00	81.46	77.90	**81.86**	76.49
RefrigerationDevices	3	375	375	720	44.00	**58.93**	54.40	49.87	50.80	52.50	55.73	53.23	55.83	58.03
ScreenType	3	375	375	720	41.07	45.60	52.80	46.40	**62.50**	62.20	58.93	45.52	50.81	53.33
ShapeletSim	2	20	180	500	69.44	47.78	78.33	**100.00**	72.40	77.90	**100.00**	77.61	**100.00**	98.33
ShapesAll	60	600	600	512	80.17	79.17	83.33	90.83	89.50	92.10	90.50	88.58	**93.00**	85.22
SmallKitchenAppliances	3	375	375	720	67.20	81.07	81.07	72.53	78.30	78.60	**85.33**	74.43	82.21	83.44
SonyAIBORobotSurface1	2	20	601	70	69.55	75.64	82.20	63.23	**96.00**	95.80	76.54	84.58	82.64	90.67
SonyAIBORobotSurface2	2	27	953	65	85.94	81.86	91.08	85.94	**97.90**	97.80	92.76	89.63	92.48	83.25
StarLightCurves	3	1000	8236	1024	89.83	96.40	97.50	97.78	96.10	97.20	98.15	98.13	**98.24**	97.84
Strawberry	2	613	370	235	94.59	96.49	96.49	97.57	97.20	**98.10**	97.03	96.84	96.63	96.38
SwedishLeaf	15	500	625	128	84.64	89.57	93.60	92.16	**96.90**	95.60	95.36	94.66	96.55	94.29
Symbols	6	25	995	398	93.77	88.56	93.27	96.68	95.50	90.60	97.39	96.16	**97.66**	88.39
SyntheticControl	6	300	300	60	98.33	97.57	66.67	96.67	99.00	**100.00**	99.67	99.53	99.79	99.03
ToeSegmentation1	2	40	228	277	75.00	74.12	90.79	93.86	96.10	96.30	**98.25**	92.46	96.53	84.43
ToeSegmentation2	2	36	130	343	90.77	81.54	90.00	**96.15**	88.00	90.60	95.38	86.23	95.38	88.46
Trace	4	100	100	275	99.00	97.80	96.00	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	99.00
TwoLeadECG	2	23	1139	82	86.83	90.39	88.76	98.07	**100.00**	**100.00**	99.65	98.86	99.46	98.72
TwoPatterns	4	1000	4000	128	99.85	94.67	43.50	99.30	87.10	**100.00**	**100.00**	99.96	**100.00**	99.77
UWaveGestureLibraryAll	8	896	3582	945	96.23	95.73	92.13	93.89	81.70	86.00	96.85	**97.23**	96.89	95.48
UWaveGestureLibraryX	8	896	3582	315	77.44	78.96	61.86	76.21	75.40	78.00	83.98	82.86	**84.11**	81.10
UWaveGestureLibraryY	8	896	3582	315	70.18	71.15	66.69	68.51	63.90	67.00	76.55	76.15	**77.23**	74.16
UWaveGestureLibraryZ	8	896	3582	315	67.50	73.58	64.99	69.49	72.60	75.00	78.31	76.40	**78.44**	75.86
Wafer	2	1000	6164	152	99.59	99.50	99.55	99.48	99.70	99.90	99.94	99.55	99.91	**99.98**
Wine	2	57	54	234	61.11	62.96	64.81	74.07	58.70	74.40	77.78	56.85	**89.06**	66.85
WordSynonyms	25	267	638	270	74.92	62.43	58.93	63.79	56.40	62.20	73.82	77.87	**78.74**	63.64
Worms	5	181	77	900	53.25	61.04	66.23	55.84	76.50	79.10	55.84	71.82	**80.17**	76.75
WormsTwoClass	2	181	77	900	58.44	62.34	**83.12**	**83.12**	72.60	74.70	77.92	78.44	81.58	79.09
Yoga	2	300	3000	426	84.30	84.14	81.83	**91.83**	83.90	87.00	91.77	87.86	83.47	82.80
Average accuracy					75.91	78.19	78.84	81.16	80.92	82.48	**84.71**	81.94	**84.64**	**82.60**
Average rank					8.16	6.90	6.55	6.10	5.57	**4.70**	**3.36**	5.42	**3.28**	**4.95**