Machine Learning manuscript No.
(will be inserted by the editor)

Bake off redux: a review and experimental evaluation of recent time series classification algorithms

Matthew Middlehurst, Patrick Schäfer and Anthony Bagnall

the date of receipt and acceptance should be inserted later

Abstract In 2017, a research paper (Bagnall et al., 2017) compared 18 Time Series Classification (TSC) algorithms on 85 datasets from the University of California, Riverside (UCR) archive. This study, commonly referred to as a 'bake off', identified that only nine algorithms performed significantly better than the Dynamic Time Warping (DTW) and Rotation Forest benchmarks that were used. The study categorised each algorithm by the type of feature they extract from time series data, forming a taxonomy of five main algorithm types. This categorisation of algorithms alongside the provision of code and accessible results for reproducibility has helped fuel an increase in popularity of the TSC field. Over six years have passed since this bake off, the UCR archive has expanded to 112 datasets and there have been a large number of new algorithms proposed. We revisit the bake off, seeing how each of the proposed categories have advanced since the original publication, and evaluate the performance of newer algorithms against the previous best-of-category using an expanded UCR archive. We extend the taxonomy to include three new categories to reflect recent developments. Alongside the originally proposed distance, interval, shapelet, dictionary and hybrid based algorithms, we compare newer convolution and feature based algorithms as well as deep learning approaches. We introduce 30 classification datasets either recently donated to the archive or reformatted to the TSC format, and use these to further evaluate the best performing algorithm from each category. Overall, we find that two recently proposed algorithms, Hydra+MultiROCKET Dempster et al. (2022) and HIVE-COTEv2 Middlehurst et al. (2021), perform significantly better than other approaches on both the current and new TSC problems.

Matthew Middlehurst, m.middlehurst@uea.ac.uk, https://orcid.org/0000-0002-3293-8779
Anthony Bagnall, ajb@uea.ac.uk, https://orcid.org/0000-0003-2360-8994
School of Computing Sciences, University of East Anglia, Norwich, UK
Patrick Schäfer, patrick.schaefer@hu-berlin.de, https://orcid.org/0000-0003-2244-6065
Humboldt-Universität zu Berlin, Berlin, Germany

1 Introduction

Time series classification (TSC) involves fitting a model from a continuous, ordered sequence of real valued observations (a time series) to a discrete response variable. Time series can be univariate (a single variable observed at each time point) or multivariate (multiple variables observed at each time point). For example, we could treat raw audio signals as a univariate time series in a problem such as classifying whale species from their calls and motion tracking co-ordinate data could be a three-dimensional multivariate time series in a human activity recognition (HAR) task. Where relevant, we distinguish between univariate time series classification (UTSC) and multivariate time series classification (MTSC). The ordering of the series does not have to be in time: we could transform audio into the frequency domain or map one dimensional image outlines onto a one dimensional series. Hence, some researchers refer to TSC as data series classification. We retain the term TSC for continuity with past research.

TSC problems arise in a wide variety of domains. Popular TSC archives[1] contain classification problems using: electroencephalograms; electrocardiograms; HAR and other motion data; image outlines; spectrograms; light curves; audio; traffic and pedestrian levels; electricity usage; electrical penetration graph; lightning tracking; hemodynamics; and simulated data. The huge variation in problem domains characterises TSC research. The initial question when comparing algorithms for TSC is whether we can draw any indicative conclusions on performance across a wide range of problems without any prior knowledge as to the underlying common structure of the data. An experimental evaluation of time series classification algorithms, which we henceforth refer to as the *bake off*, was conducted in 2016 and published in 2017 (Bagnall et al., 2017). This bake off, coupled with a relaunch of time series classification archives (Dau et al., 2019), has helped increase the interest in TSC algorithms and applications. Our aim is to summarise the significant developments since 2017. A new MTSC archive (Ruiz et al., 2021) has helped promote research in this field. A variety of new algorithms using different representations, including deep learners (Fawaz et al., 2019), convolution based algorithms (Dempster et al., 2020) and hierarchical meta ensembles (Lines et al., 2018), have been proposed for TSC. Furthermore, the growth in popularity of TSC open source toolkits such as aeon[2] and tslearn[3] have made comparison and reproduction easier. We extend and encompass recent experimental evaluations (e.g. (Ruiz et al., 2021; Bagnall et al., 2020a; Middlehurst et al., 2021; Fawaz et al., 2019)) to provide insights into the current state of the art in the field and highlight future directions. Our contributions can be summarised as follows:

1. We describe a range of new algorithms for TSC and place them in the context of those described in the bake off.

[1] `https://timeseriesclassification.com`
[2] `https://www.aeon-toolkit.org`
[3] `https://tslearn.readthedocs.io/en/stable/`

2. We compare performance of the new algorithms on the current UCR archive datasets in a bake off redux.
3. We contribute 30 new univariate datasets to the TSC archive and evaluate the best in category on this new data.
4. We analyse the factors that drive performance and discuss the merits of different approaches.

To select algorithms, we use the same criteria as the bake off. Firstly, the algorithm must have been published post bake off in a high quality conference or journal (or be an extension of such an algorithm). Secondly, it must have been evaluated on some subset of the UCR/UEA datasets. Thirdly, source code must be available and easily adaptable to the time series machine learning tools we use. Section 2 describes the core terminology relating to TSC. Section 3 summarises how we conduct experimental evaluations of classifiers. We describe the latest TSC algorithms included in this bake off in Section 4. This section also describes the first set of experiments that link to the previous bake off. Section 5 extends the experimental evaluation to include the new datasets. Section 6 investigates variation in performance in more detail. Finally, we conclude and discuss future direction in Section 7.

2 Definitions and Terminology

We define the number of time series in a collection as m, the number of channels/dimensions of any observation as d and length of a series as n.

Definition 1 (Time Series (TS)) : A time series $A = (a_1, a_2, \ldots, a_n)$ is an ordered sequence of n data points. We denote the i-th value of A by a_i.

In the above definition, if every point in $a_i \in A$ in the time series represents a single value ($a_i \in \mathbb{R}$), the series is a *univariate time series (UTS)*. If each point represents the observation of multiple variables at the same time point (e.g., temperature, humidity, pressure, etc.) then each point itself is a vector $a_i \in \mathbb{R}^d$ of length d, and we call it a *multivariate time series (MTS)*:

Definition 2 (Multivariate Time Series (MTS)) : A multivariate time series $A = (a_1, \ldots a_n,) \in \mathbb{R}^{(d \times n)}$ is a list of n vectors with each a_i being a vector of d channels (sometimes referred to as dimensions). We denote the i-th observation of the $k - th$ channel by the scalar $a_{k,i} \in \mathbb{R}$.

Note that it is also possible to view a MTS as a set of d time series, since in practice that is often how they are treated. However, the vector model makes it explicit that we assume that the dimensions are aligned, i.e. we assume that all observations in a_i are observed at the same point in time or space. In the context of supervised learning tasks such as classification, a dataset associates each time series with a label from a predefined set of classes.

Definition 3 (Dataset) : A dataset $D = (X, Y) = \left(A^{(i)}, y^{(i)}\right)_{i \in [1,\ldots,m]}$ is a collection of m time series and a predefined set of discrete class labels C. We denote the size of D by m, and the i^{th} instance by series and its label by $y^{(i)} \in C$.

Many time series classification algorithms make use of subseries of the data.

Definition 4 (Subseries) : A subseries $A_{i,l}$ of a time series $A = (a_1, \ldots, a_n)$, with $1 \leq i < i + l \leq n$, is a series of length l, consisting of the l contiguous points from A starting at offset i: $A_{i,l} = (a_i, a_{i+1}, \ldots, a_{i+l-1})$, i.e. all indices in the right-open interval $[i, i + l)$.

We may extract subseries from a time series by the use of a sliding window.

Definition 5 (Sliding Window) : A time series A of length n has $(n-l+1)$ sliding windows of length l (when increment is 1) given by:

$$sliding\_windows(A) = \{A_{1,l}, \ldots, A_{(n-l+1),l}\}$$

The *dilation technique* is a method that enables a filter, such as a sliding window or convolution filter, to cover a larger portion of the time series data by creating empty spaces between the entries in the filter. These spaces enable the filter to widen its receptive field while maintaining the total number of values constant. To illustrate, a dilation of $d = 2$ would introduce a gap of 1 between each pair of values. This effectively doubles the receptive field's size and enables the filter to analyse the data at various scales, akin to a down-sampling operation.

Definition 6 (Dilated Subseries) : A dilated subseries, denoted by $A_{i,l,d}$, is a sequence extracted from a time series $A = (a_1, \ldots, a_n)$, with $1 \leq i < i + l \times d \leq n$. This subseries has length l and dilation factor d, and it includes l non-contiguous points from A starting at offset i and taking every d-th value as follows:

$$A_{i,l,d} = (a_i, a_{i+d\times1}, \ldots, a_{i+d\times(l-1)})$$

The dilation techniques is used in convolution-based, shapelet-based and dictionary-based models.

In real-world applications, series of D are often unequal length. This is often treated as a preprocessing task, i.e. by appending tailing zeros, although some algorithms have the capability to internally handle this. We further typically assume that all time series of D have the same sampling frequency, i.e., every i^{th} data point of every series was measured at the same temporal distance from its predecessor.

3 Experimental Procedure

The bake off conducted experiments with the 85 UTSC datasets that were in the UCR archive relaunch of 2015. Each dataset was resampled 100 times for training and testing, and test accuracy was averaged over resamples. The evaluation began with 11 standard classifiers (such as random forest), then classifiers in each category were compared, including an evaluation of reproducibility. Finally, the best in class were compared to hybrids (combinations of categories).

We adapt this approach for the bake off redux to reflect the progression of the field. First, we take the previously used benchmark of Dynamic Time Warping using a one nearest neighbour classifier (1-NN DTW) and, if appropriate, the best of each category from the bake off and compare them to new algorithms of that type. We do this stage of experimentation with the 112 equal length problems in the 2019 version of the UCR archive Dau et al. (2019). Performance on these datasets, or some subset thereof, has been used to support every proposed approach, so this allows us to make a fair comparison of algorithms.

Only a subset of the algorithms considered have been adapted for MTSC by their inventors. Furthermore, many algorithms have been proposed solely for MTSC, particularly in the deep learning field. Because of this, we restrict our attention to univariate classification only.

We resample each pair of train/test data 30 times for the redux, stratifying to retain the same class distribution. We do not adopt the bake off strategy of 100 resamples. We have found 30 resamples is sufficient to mitigate small changes in test accuracy over influencing ranks, and is more computationally feasible. Resampling is seeded with the resample ID to aid with reproducibility. Resample 0 uses the original train and test split from the UCR archive.

Our primary performance measure is classification accuracy on the test set. We also compare predictive power with the balanced test set accuracy, to identify whether class imbalance is a problem for an algorithm. The quality of the probability estimates is measured with the negative log likelihood (NLL). The ability to rank predictions is estimated by the area under the receiver operator characteristic curve (AUROC). For problems with two classes, we treat the minority class as a positive outcome. For multiclass problems, we calculate the AUROC for each class and weight it by the class frequency in the train data, as recommended in Provost and Domingos (2003). We present results with diagrams derived from the critical difference plots proposed by Demšar (2006). We average ranks over all datasets and plot them on a line and group classifiers into cliques, within which there is no significant difference in rank. We replace the post-hoc Nemenyi test used to form cliques described in Demšar (2006) with a mechanism built on pairwise tests. We perform pairwise Wilcoxon signed-rank tests and form cliques using the Holm correction for multiple testing as described in García and Herrera (2008); Benavoli et al. (2016).

Critical difference diagrams can be deceptive: they do not display the effective amount of differences, and the linear nature of clique finding can mask rela-

tionships between results. If, for example, three classifiers A, B, C are ordered by rank $A > B > C$, and the test indicates A is significantly better than B, and B is significantly better than C, then we will form no cliques. However, it is entirely possible that A is not significantly different to C, and the diagram cannot display this. Because of this, we expand our results to include pairwise plots, violin plots of accuracy distributions against a base line and tables of test accuracies and unadjusted p-values.

3.1 New Datasets

The 112 equal length TSC problems in the archive constitute a relatively large corpus of problems for comparing classifiers. However, they have been extensively used in algorithm development, and there is always the risk of an implicit overfitting resulting in conclusions that do not generalise well to new problems. Hence, we have gathered new datasets which we use to perform our final comparison of algorithms. These data come from direct donation to the github repository[4], discretised regression datasets[5], a project on audio classification (Flynn and Bagnall, 2019) and reformatting current datasets with unequal length or missing values. We are happy to accept new datasets through the repository associated with the archive.

In total, we have gathered 30 new datasets, summarised in Table 1 and visualised in Figure 2. Datasets with the suffix **Eq** are unequal length series made equal length through truncation to the shortest length series. 11 of these problems (AllGestureWiimote versions, GestureMidAirD1, GesturePebbleZ, PickupGestureWiimoteZ, PLAID and ShakeGestureWiimoteZ) are already in the archive so need no further explanation. Four data sets with the suffix **Nmv** (no missing values) are datasets where the original contains missing values. We have simply removed any cases which contain missing values for these problems (DodgerLoop variants and MelbournePedestrian). These are also from the current archive. The four datasets with the suffix **Discrete** are taken from the TSER archive (Tan et al., 2021). The continuous response variable was discretised manually for each dataset, the original continuous labels and new class values for each dataset are shown in Figure 1.

This leaves 11 datasets that are completely new to the archive. The two **AconityMINIPrinter** data sets are described in (Mahato et al., 2020) and donated by the authors of that paper. The data comes from the AconityMINI 3D printer during the manufacturing of stainless steel blocks with a designed cavity. The problem is to predict whether there is a void in the output of the printer. The time series are temperature data that comes from pyrometers that monitor melt pool temperature. The pyrometers track the scan of the laser to provide a time-series sampled at 100 Hz. The data is sampled from the mid-section of these blocks and is organized into two datasets (large and small). The large dataset covers cubes with large pores (0.4 mm, 0.5 mm, and

[4] `https://github.com/time-series-machine-learning/tsml-repo`

[5] `http://tseregression.org/`

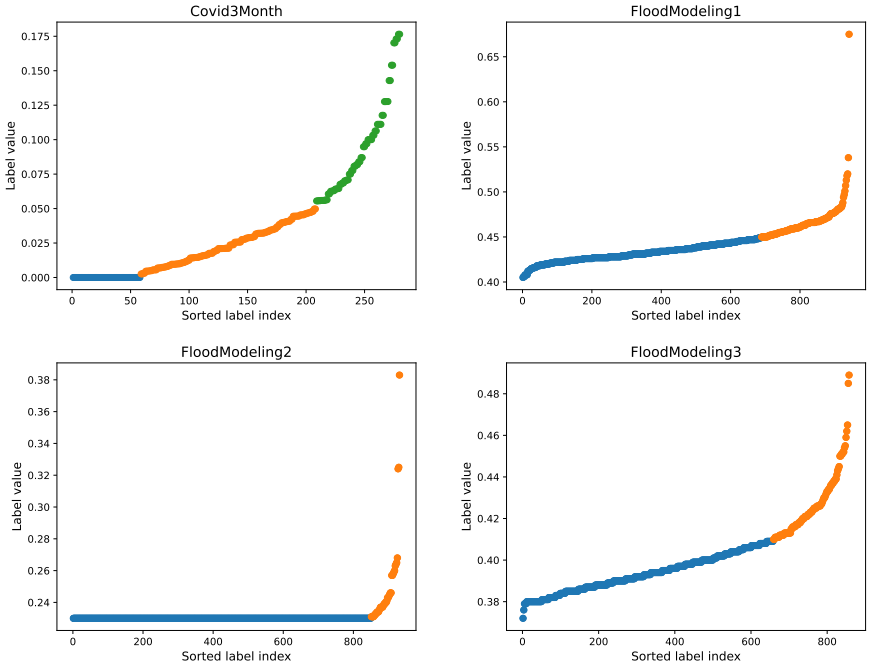


Fig. 1 The sorted original label values for all discretised regression datasets. Each point is a label for a case, and its colour is the class it is part of for the new classification version.

0.6 mm) and the small dataset covers cubes with small pores (0.05 mm and 0.1 mm).

The three **Asphalt** datasets were originally described in (Souza, 2018) and donated by the author of that paper. Accelerometer data was collected on a smartphone installed inside a vehicle using a flexible suction holder near the dashboard. The acceleration forces are given by the accelerometer sensor of the device and are the data used for the classification task. The class values for AsphaltObstacles classes are four common obstacles in the region of data collection: raised cross walk (160 cases); raised markers (187 cases); speed bump (212 cases); and vertical patch (222 cases); flexible pavement (816 cases); cobblestone street (527 cases); and dirt road (768 cases). AsphaltRegularity is a two class problem: Regular (762 cases), where the asphalt is even and the driver's comfort changes little over time; and Deteriorated (740 cases), where irregularities and unevenness in a damaged road surface are responsible for transmitting vibrations to the interior of the vehicle and affecting the driver's comfort.

The **Colposcopy** data is described in (K et al., 2017) and was donated to the repository by the authors[6]. The task is to classify the nature of a diagnosis from a colposcopy. The time series represent the change in intensity values of a

[6] https://github.com/KarinaGF/ColposcopyData

pixel region through a sequence of digital colposcopic images obtained during the colposcopy test that was performed on each patient included in the study.

The **ElectricDeviceDetection** data set (Bagnall et al., 2020b) contains formatted image data for the problem of detecting whether a segment of a 3-D X-Ray contains an electric device or not. The data originates from an unsupervised segmentation of 3-D X-Rays. The data are histograms of intensities, not time series.

KeplerLightCurves was described in (Barbara et al., 2022) and donated by the authors. Each case is a light curve (brightness of an object sampled over time) from NASA's Kepler mission (3-month-long series, sampled every 30 min). There are seven classes relating to the nature of the observed star.

The **SharePriceIncrease** data was formatted as part of a student project. The problem is to predict whether a share price will show an exceptional rise after quarterly announcement of the Earning Per Share based on the price movement of that share price on the preceding 60 days. Daily price data on NASDAQ 100 companies was extracted from a Kaggle data set. Each data represents the percentage change of the closing price from the day before. Each case is a series of 60 days data. The target class is defined as 0 if the price did not increase after company report release by more than five percent or 1 else-wise.

PhoneHeartbeatSound and **Tools** are audio datasets. Tools contains the sound of a chainsaw, drill, hammer, horn and sword, with the task being to match which tool the audio belongs to. PhoneHeartbeatSound contains sounds of the heartbeats recorded on a phone using a digital stethoscope gathered for the 2011 PASCAL classifying heart sounds challenge[7]. The time series represent the change in amplitude over time during an examination of patients suffering from common arrhythmias. The classes are Artifact (40 cases), ExtraStole (46 cases), Murmur (129 cases), Normal (351 cases) and ExtraHLS (40 cases).

3.2 Reproducibility

The majority of the classifiers described are available in the aeon time series machine learning toolkit (see Footnote 2) and all datasets are available for download (see Footnote 1). Appendix A gives detailed code examples on how to reproduce these experiments, including parameters used, if they differ from the default. Further guidance on reproducibility and our results files are available in an accompanying webpage[8]

[7] `http://www.peterjbentley.com/heartchallenge/index.html`

[8] `https://tsml-eval.readthedocs.io/en/latest/publications/2023/tsc_bakeoff/tsc_bakeoff_2023.html`

Table 1 A summary of the 30 new univariate datasets used in our experiments with suffix: *Eq, Nmv, Discrete*

Dataset	Train size	Test size	Series length	No. Classes	Category
AconityMINIPrinterLargeEq	2403	1184	300	2	Sensor
AconityMINIPrinterSmallEq	589	292	300	2	Sensor
AllGestureWiimoteXEq	300	700	500	10	Motion
AllGestureWiimoteYEq	300	700	500	10	Motion
AllGestureWiimoteZEq	300	700	500	10	Motion
AsphaltObstaclesUniEq	390	391	736	4	Sensor
AsphaltPavementTypeUniEq	1055	1056	2371	3	Sensor
AsphaltRegularityUniEq	751	751	4201	2	Sensor
Colposcopy	99	101	180	6	Image
Covid3MonthDiscrete	140	140	84	3	Other
DodgerLoopDayNmv	67	77	288	7	Sensor
DodgerLoopGameNmv	17	127	288	2	Sensor
DodgerLoopWeekendNmv	18	126	288	2	Sensor
ElectricDeviceDetection	624	3768	256	2	Image
FloodModeling1Discrete	471	471	266	2	Simulated
FloodModeling2Discrete	466	466	266	2	Simulated
FloodModeling3Discrete	429	429	266	2	Simulated
GestureMidAirD1Eq	208	130	360	26	Motion
GestureMidAirD2Eq	208	130	360	26	Motion
GestureMidAirD3Eq	208	130	360	26	Motion
GesturePebbleZ1Eq	132	172	455	6	Motion
GesturePebbleZ2Eq	146	158	455	6	Motion
KeplerLightCurves	920	399	4767	7	Sensor
MelbournePedestrianNmv	1138	2319	24	10	Sensor
PhoneHeartbeatSound	424	182	3053	5	Other
PickupGestureWiimoteZEq	50	50	361	10	Motion
PLAIDEq	537	537	1345	11	Device
ShakeGestureWiimoteZEq	50	50	385	10	Motion
SharePriceIncrease	965	966	60	2	Other
Tools	310	134	2926	5	Other

4 Time Series Classification Algorithms

The bake off introduced a taxonomy of algorithms based on the representation of the data at the heart of the algorithm. TSC algorithms were classified as either whole series, interval based, shapelet based, dictionary based, combinations or model based. We extend and refine this taxonomy to reflect recent developments.

1. Distance based: classification is based on some time series specific distance measure between series (Section 4.1).
2. Feature based: global features are extracted and passed to a standard classifier in a simple pipeline (Section 4.2).
3. Interval based: features are derived from selected phase dependent intervals in an ensemble of pipelines (Section 4.3).
4. Shapelet based: phase independent discriminatory subseries form the basis for classification (Section 4.4).

Fig. 2 The 30 new univariate datasets showing one representative series for each class.

5. Dictionary based: histograms of counts of repeating patterns are the features for a classifier (Section 4.5).
6. Convolution based: convolutions and pooling operations create the feature space for classification (Section 4.6).
7. Deep learning based: neural network based classification (Section 4.7).
8. Hybrid approaches combine two or more of the above approaches (Section 4.8).

As well as the type of feature extracted, another defining characteristic is the design of the TSC algorithm. The simplest design pattern involves single pipelines where transformation of the series into discriminatory features is followed by the application of a standard machine learning classifier. These algorithms tend to involve an over-production and selection strategy: a large number of features are created, and the classifier determines which features are most useful. The transform can remove time dependency, e.g. by calculating summary features. We call this type series-to-vector transformations. Alternatively, they may be series-to-series, transforming into an alternative time series representation where we hope the task becomes more easily tractable, e.g. transforming to the frequency domain of the series.

The second transformation based design pattern involves ensembles of pipelines, where each base pipeline consists of making repeated, different, transforms and using a homogeneous base classifier. TSC ensembles can also be heterogeneous, collating the classifications from transformation pipelines and ensembles of differing representations of the time series.

The third common pattern involves transformations embedded inside a classifier structure. For example, a decision tree where the data is transformed at each node fits this pattern.

A common theme to all categories of algorithm is ensembling. Another popular method seen in multiple classifiers are transformation pipelines ending with a linear classifier. The most accurate classifiers we find all form homogeneous or heterogeneous ensembles, or extract features prior to a linear RIDGE classifier.

We review each category of algorithms by providing an overview of the approach, review selected classifiers and describe the pattern they use, starting with the best of class from the bake off. We perform a comparison of performance within category on the 112 equal length UTSC problems currently in the UCR archive using 1-NN DTW as a benchmark. More detailed evaluation is delayed until Section 5.

4.1 Distance Based

Distance based classifiers use a distance function to measure the similarity between time series. Historically, distance functions have been mostly used with nearest neighbour (NN) classifiers. Alternative uses of time series distances are described in (Abanda et al., 2019). Prior to the bake off, 1-NN with DTW was considered state of the art for TSC (Rakthanmanon et al., 2013). Figure 3 shows an example of how DTW attempts to align two series, depicted in red and green, to minimise their distance.

In addition to DTW, a wide range of alternative elastic distance measures (distance measures that compensate for possible misalignment between series) have been proposed. These use combinations of warping and editing on series and the derivatives of series. See (Holder et al., 2022) for an overview of elastic distances. Previous studies (Lines and Bagnall, 2014) have shown there is

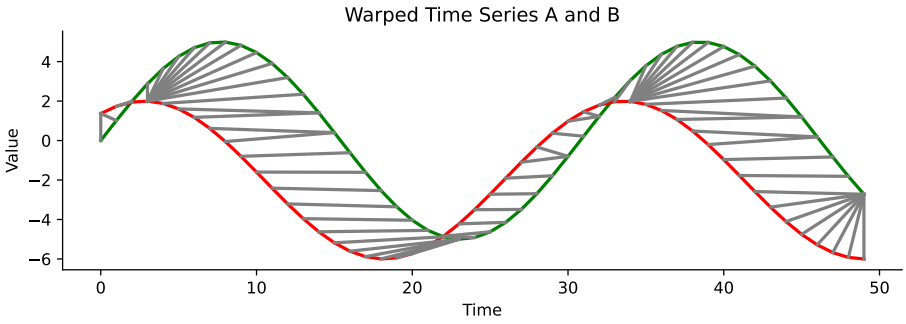


Fig. 3 An example of how DTW compensates for phase shift by realigning two series (in red at the bottom and in green at the top).

little difference in performance between 1-NN classifiers with different elastic distances.

4.1.1 Elastic Ensemble (EE)

The first algorithm to significantly outperform 1-NN DTW on the UCR data was the Elastic Ensemble (EE) (Lines and Bagnall, 2015). EE is a weighted ensemble of 11 1-NN classifiers with a range of elastic distance measures. It was the best performing distance based classifier in the bake off. Elastic distances can be slow, and EE requires cross validation to find the weights of each classifier in the ensemble. A caching mechanism was proposed to help speed up fitting the classifier (Tan et al., 2020) and alternative speed ups were described in (Oastler and Lines, 2019).

4.1.2 ProximityForest (PF)

Proximity Forest (PF) (Lucas et al., 2019) is an ensemble of Proximity Tree based classifiers. PF uses the same 11 distance functions used by EE, but is more accurate and more scalable than EE. At every node of a tree, one of the 11 distances is selected to be applied with a fixed hyperparameter value. An exemplar single series is selected randomly for each class label. At every node, r combinations of distance function, parameter value and class exemplars are randomly selected, and the combination with the highest Gini index split measure is selected. Series are passed down the branch with the exemplar that has the lowest distance to it, and the tree grows recursively until a node is pure.

4.1.3 ShapeDTW

Shape based DTW (ShapeDTW) (Zhao and Itti, 2019) works by extracting a set of shape descriptors over sliding windows of each series. The descriptors

include slope, wavelet transforms and piecewise approximations. These series to series transformed data are then used with a 1-NN classifier with DTW.

4.1.4 Comparison of Distance Based Approaches

Figure 4 shows the relative rank test accuracies of the five distance based classifiers we discuss here, and Table 2 summarises four performance measures over these datasets. The results broadly validate previous findings. EE is significantly better than 1-NN DTW and PF is significantly better than EE. Table 2 shows PF is over 2.5% better in test accuracy and balanced test accuracy, has higher AUROC and lower NLL. Hence, we take PF as best of the distance based category.

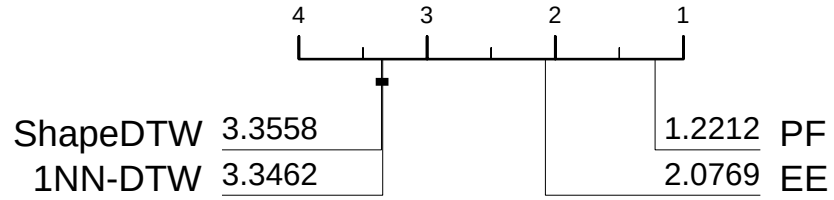


Fig. 4 Ranked test accuracy of four distance based classifiers on 112 UCR UTSC problems. Accuracies are averaged over 30 resamples of train and test splits.

Table 2 Summary performance measures for distance based classifiers on 30 resamples of 112 UTSC problems.

	ACC	BALACC	AUROC	NLL
PF	0.847	0.829	0.943	0.624
EE	0.820	0.802	0.915	0.776
1NN-DTW	0.768	0.751	0.776	1.542
ShapeDTW	0.756	0.740	0.768	1.620

4.2 Feature Based

Feature based classifiers are a popular recent theme. These extract descriptive statistics as features from time series to be used in classifiers. Typically, these features summarise the whole series, so we characterise these as series-to-vector transforms. Most commonly, these features are used in a simple pipeline of transformation followed by a classifier (see Figure 5). Several toolkits exist for extracting features.

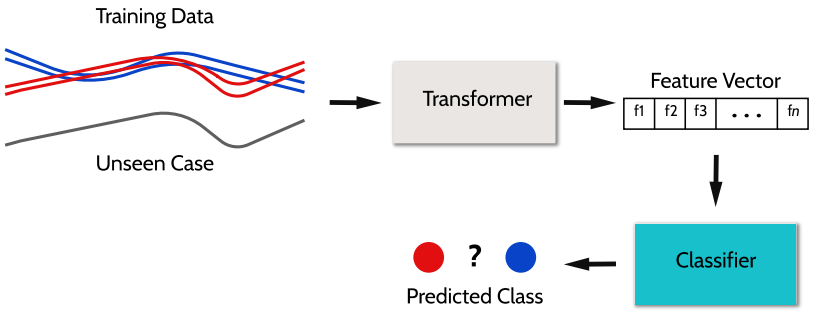


Fig. 5 Visualisation of a pipeline classifier involving feature extraction followed by classification.

4.2.1 The Canonical Time Series Characteristics (Catch22)

The highly comparative time-series analysis (*hctsa*) (Fulcher and Jones, 2017) toolbox can create over 7700 features for exploratory time series analysis. The canonical time series characteristics (Catch22) (Lubba et al., 2019) are 22 *hctsa* features determined to be the most discriminatory of the full set. The Catch22 features were chosen by an evaluation on the UCR datasets. The *hctsa* features were initially pruned, removing those which are sensitive to mean and variance those that could not be calculated on over 80% of the UCR datasets. A feature evaluation was then performed based on predictive performance. Any features which performed below a threshold were removed. For the remaining features, a hierarchical clustering was performed on the correlation matrix to remove redundancy. From each of the 22 clusters formed, a single feature was selected, taking into account balanced accuracy, computational efficiency and interpretability. The Catch22 features cover a wide range of concepts such as basic statistics of time series values, linear correlations, and entropy. Reported results for Catch22 are based on training a decision tree classifier after applying the transform to each time series (Lubba et al., 2019).

4.2.2 Time Series Feature Extraction based on Scalable Hypothesis Tests (TSFresh)

TSFresh (Christ et al., 2018) is a collection of just under 800 features extracted from time series. While the features can be used on their own, a feature selection method called FRESH is provided to remove irrelevant features. FRESH considered each feature using multiple hypotheses tests, including Fisher's exact test, the Kolmogorov-Smirnov test and the Kendal rank test. The Benjamini-Yekutieli procedure is then used to control the false discovery rate caused by comparing multiple hypotheses and features simultaneously.

Results for the base features and after using the FRESH algorithm are reported using both a random forest (Breiman, 2001) and AdaBoost (Freund and Schapire, 1996) classifier. A comparison of alternative pipelines of feature

extractor and classifier found that the most effective approach was the full set of TSFresh features with no feature selection applied, and combined with a rotation forest classifier (Rodriguez et al., 2006). This pipeline was called the FreshPRINCE (Middlehurst and Bagnall, 2022). We include both TSFresh using a random forest the FreshPRINCE classifier in our comparison.

4.2.3 Generalised Signatures

Generalised signatures are a set of feature extraction techniques based on rough path theory. The generalised signature method (Morrill et al., 2020) and the accompanying canonical signature pipeline can be used as a transformation for classification. Signatures are collections of ordered cross-moments. The pipeline begins by applying two augmentations. The basepoint augmentation simply adds a zero at the beginning of the time series, making the signature sensitive to translations of the time series. The time augmentation adds the series timestamps as an extra coordinate to guarantee that each signature is unique and obtain information about the parameterisation of the time series. A hierarchical window is run over the two augmented series, with the signature transform being applied to each window. The output for each window is then concatenated into a feature vector.

4.2.4 Comparison of Feature Based Approaches

Figure 6 shows the relative rank performance, and Table 3 summarises the overall performance statistics. All four pipelines are significantly more accurate than 1-NN DTW. Excluding feature extraction and using rotation forest rather than random forest with TSFresh increases accuracy by over 0.05. This reinforces the findings that rotation forest is the most effective classifier for problems with continuous features (Bagnall et al., 2018).

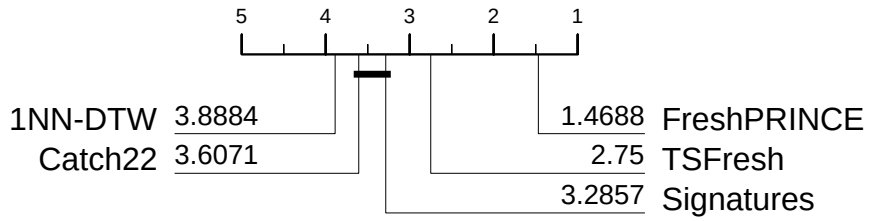


Fig. 6 Ranked test accuracy of four feature based classifiers and the benchmark 1NN-DTW on 112 UCR UTSC problems. Accuracies are averaged over 30 resamples of train and test splits.

Table 3 Summary performance measures for FeatureBased classifiers on 30 resamples of 112 UTSC problems.

	ACC	BALACC	AUROC	NLL
FreshPRINCE	0.855	0.834	0.958	0.678
TSFresh	0.799	0.772	0.859	1.074
Signatures	0.787	0.763	0.919	0.971
Catch22	0.795	0.771	0.929	0.910

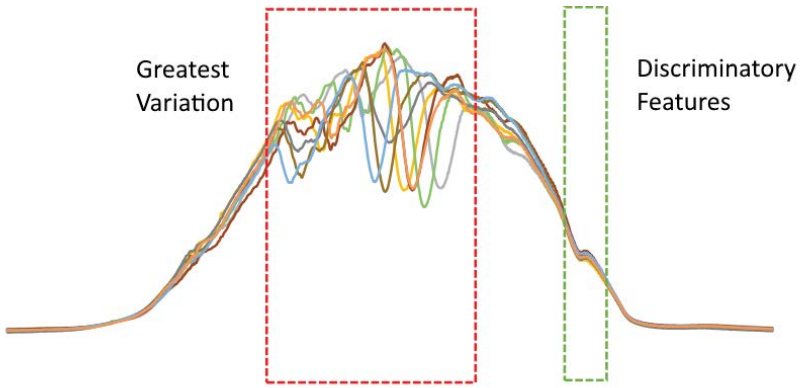


Fig. 7 An example of a problem where interval based approaches may be superior. Each series is a spectrogram from a bottle of alcohol with a different concentration of ethanol. The discriminatory features are in the near infrared interval (green box to the right). However, the confounding factors such as bottle shape, labelling and colouring cause variation in the visible range (red box to the left). Using intervals containing just the near infrared features is likely to make classification easier.

4.3 Interval Based

Interval based classifiers (Deng et al., 2013) extract phase dependent intervals of fixed offsets and compute (summary) statistics on these intervals. A majority of approaches include some form of random selection for choosing intervals, employing essentially a form of randomised feature selection, since the same random interval locations are used across every series, although many of the interval based classifiers combine features from multiple intervals. The motivation for taking intervals is to mitigate for confounding noise. Figure 7 shows an example problem where taking intervals will be better than using features derived from the whole series.

Interval based classifiers adopt a random forest ensemble model, where each base classifier is a pipeline of transformation and a tree classifier (visualised in Figure 8). Diversity is injected through randomising the intervals for each tree.

4.3.1 Time Series Forest (TSF)

TSF (Deng et al., 2013) is the simplest interval based tree based ensemble. For each tree, $\sqrt{n}$ intervals are selected with a random position and length. The

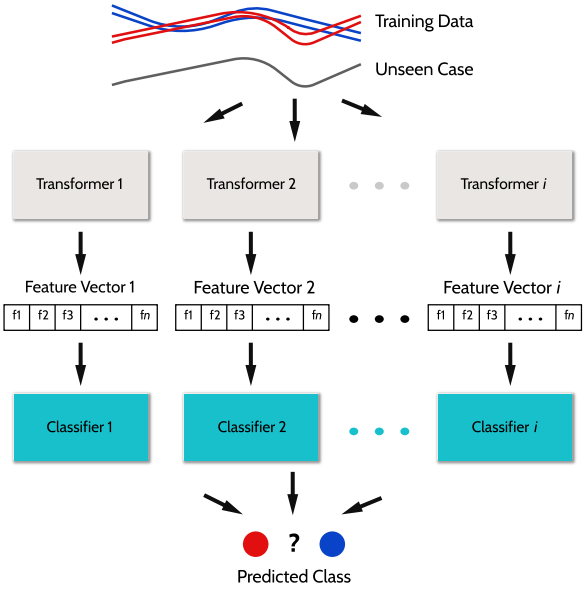


Fig. 8 Visualisation of an ensemble of pipeline classifiers, as used in interval classifiers.

same interval offsets are applied to all series. For each interval, three summary statistics (the mean, variance and slope) are extracted and concatenated into a feature vector. This feature vector is used to build the tree, and features extracted from the same intervals are used to make predictions. The ensemble makes the prediction using a majority vote of base classifiers. The TSF base classifier is a modified decision tree classifier referred to as a time series tree, which considers all attributes at each node and uses a metric called margin gain to break ties.

4.3.2 Random Interval Spectral Ensemble (RISE)

First developed for the HIVE-COTE ensemble (described in Section 4.8), RISE (Flynn et al., 2019) is an interval based tree ensemble that uses spectral features. Unlike TSF, RISE selects a single random interval for each base classifier. The periodogram and auto-regression function are calculated over each randomly selected interval, and these features are concatenated into a feature vector, from which a tree is built. RISE was primarily designed for use with audio problems, where spectral features are more likely to be discriminatory.

4.3.3 STSF and R-STSF

Supervised Time Series Forest (STSF) (Cabello et al., 2020) is an interval based tree ensemble that includes a supervised method for extracting intervals. Intervals are found and extracted for a periodogram and the first order differences representation as well as the base series. STSF introduces

bagging for each tree and extracts seven simple summary statistics from each interval. For each tree, an initial split point for the series is randomly selected. For both of these splits, the remaining subseries is cut in half, and the half with the higher Fisher score is retained as an interval. This process is then run recursively using higher scored intervals until the series is smaller than a threshold. This is repeated for each of the seven summary statistic features, with the extracted statistic being used to calculate the Fisher score.

Randomised STSF (RSTSF) (Cabello et al., 2021) is an extension of STSF, altering its components with more randomised elements. The split points for interval selection are selected randomly instead of splitting each candidate in half after the first, and a randomised binary tree is used to build the extracted features. Intervals extracted from an autoregressive representation are included alongside the previous additions. Features are extracted multiple times from each representation into a single pool, with a subsample of this pool randomly selected for each tree.

4.3.4 CIF and DrCIF

The Canonical Interval Forest (CIF) (Middlehurst et al., 2020a) is another extension of TSF, that improves accuracy by integrating more informative features and by increasing diversity. Like other interval approaches, CIF is an ensemble of decision tree classifiers built on features extracted from phase dependent intervals. Alongside the mean, standard deviation and slope, CIF also extracts the catch22 features described in Section 4.2. Intervals remain randomly generated, with each tree selecting $\sqrt{m}\sqrt{d}$ intervals. To add additional diversity to the ensemble, a attributes out of the pool of 25 are randomly selected for each tree.The extracted features are concatenated into a $k \cdot a$ length vector for each time series and used to build the tree. For multivariate data, CIF randomly selects the dimension used for each interval.

The Diverse Representation Canonical Interval Forest (DrCIF) (Middlehurst et al., 2021) incorporates two new series representations: the periodograms (also used by RISE and STSF) and first order differences (also used by STSF). For each of the three representations, $(4 + \sqrt{r}\sqrt{d})/3$ phase dependent intervals are randomly selected and concatenated into a feature vector, where r is the length of the series for a representation.

4.3.5 Comparison of Interval Based Approaches

Figure 9 shows the relative ranks of six interval classifiers, with summary performance measures presented in Table 4.

There is no significant difference between DrCIF and RSTSF nor between their precursors CIF and STSF. All are significantly better than TSF, the best in class in the bake off. Figure 10(a) shows the scatter plot of DrCIF vs RSTSF. DRCIF wins on 59, draws 6 and loses 47. Conversely, RSTSF does markedly better on three problems (it is more than 0.08 more accurate on InlineSkate, PigCVP andd PigAirwayPressure). Overall, the two algorithms produce very

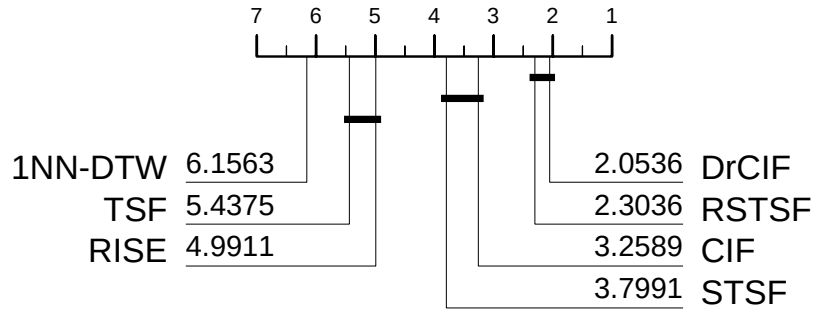


Fig. 9 Ranked test accuracy of six interval based classifiers on 112 UCR UTSC problems. Accuracies are averaged over 30 resamples of train and test splits.

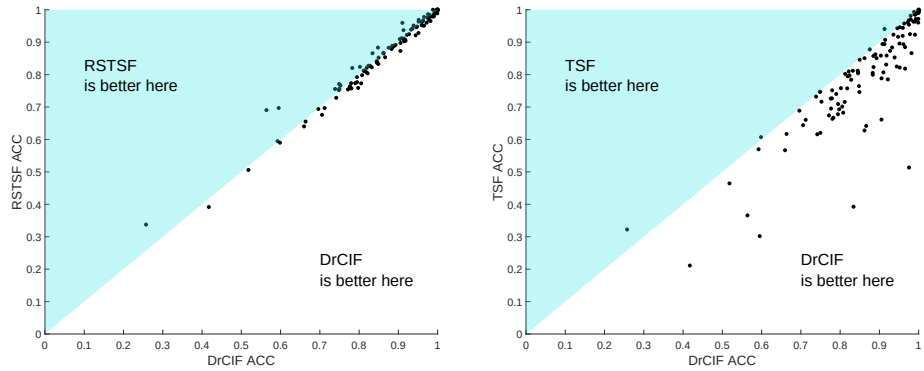
Table 4 Summary performance measures for Interval based classifiers on 30 resamples of 112 UTSC problems.

	ACC	BALACC	AUROC	NLL
DrCIF	0.864	0.841	0.962	0.689
RSTSF	0.864	0.842	0.961	0.693
CIF	0.848	0.824	0.954	0.762
STSF	0.846	0.827	0.955	0.783
RISE	0.806	0.777	0.937	0.834
TSF	0.802	0.780	0.930	1.084

similar results (the test accuracies have a correlation of 98.5%). The balanced test accuracies, area under ROC and negative log likelihood are also very similar, suggesting they rank and produce probability estimates that are not very difference. We choose RSTSF as the best in class because it is significantly faster than DrCIF. Figure 10(b) shows DrCIF against TSF in order to confirm that both DrCIF and RSTSF represent genuine improvements to this type of algorithm over the previous best. Table 4 confirms that on average over 112 problems, the accuracy of DrCIF and RSTSF is over 0.06 higher than TSF.

4.4 Shapelet Based

Shapelets are subseries from the training data that are independent of the phase and can be used to discriminate between classes of time series based on their presence or absence. To evaluate a shapelet, the subseries is slid across the time series, and the z-normalised Euclidean distance between the shapelet and the underlying window is calculated. The distance between a shapelet and any series, $sDist()$, is the minimum distance over all such windows. Figure 11 shows a visualisation of the $sDist()$ process. The shapelet S is shifted along the time series A, and the most similar offset and distance in A are recorded. The distance between a shapelet and the training series is then used as a feature to evaluate the quality of the shapelet.



(a) DrCIF vs RSTSF (b) DrCIF vs TSF

Fig. 10 Scatter plot of test accuracies of DrCIF against RSTSF and TSF. TSF is better than DrCIF on just 6 of the 112 datasets.

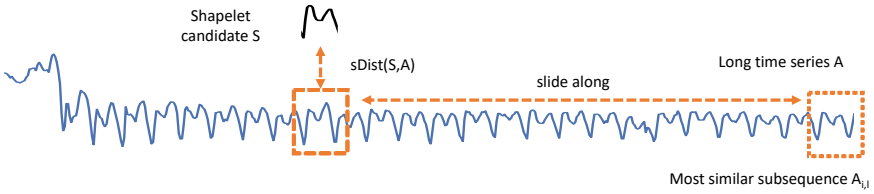


Fig. 11 Visualisation of the shapelet distance operation $sDist()$ between a shapelet S and a series A, which finds the closest distances to the shapelet from all possible subseries of the same length.

Shapelets were first proposed as a primitive in (Ye and Keogh, 2011), and were embedded in a decision tree classifier. There have been four important themes in shapelet research post bake off: The first has concentrated on finding the best way to use shapelets to maximise classification accuracy. The second has focused on overcoming the shortcomings of the original shapelet discovery which required full enumeration of the search space and has cubic complexity in the time series length; the third theme is the progress toward unifying research with convolutions and shapelets; and the fourth theme is the balance between optimisation, randomisation and interpretability when finding shapelets.

4.4.1 The Shapelet Transform Classifier (STC)

The Shapelet Transform Classifier (STC) Hills et al. (2014) is a pipeline classifier which searches the training data for shapelets, transforms series to vectors of $sDist()$ distances to a filtered set of selected shapelets based on information gain, then builds a classifier on the latter. This is in contrast to the decision tree based approaches, which search for the best shapelet at each tree node. The first version of STC performed a full enumeration of all shapelets

from all train cases before selecting the top k. The base classifier used was HESCA (later renamed CAWPE) Large et al. (2019b) ensemble of classifiers, a weighted heterogeneous ensemble of 8 classifiers including a diverse set of linear, tree based and Bayesian classifiers. Due to its full enumeration and large pool of base classifiers requiring weights, the algorithm does not scale well. We call the original full enumeration version **ST-HESCA** to differentiate it from the version described below which we simply call **STC**. It was the best performing shapelet based classifier in the bake off.

The following incremental changes have been made to the STC pipeline, described in Bostrom et al. (2016); Bostrom and Bagnall (2017):

1. Search has been randomised, and the search time is now a parameter. Search time defaults to one hour, and it has been shown that this does not lead to significantly worse performance on the UCR datasets.
2. Shapelets are now binary, in that they represent the class of the origin series and are evaluated against all other classes as a single class using one hot encoding. This facilitates greater use of the early abandon of the order line creation (described in (Ye and Keogh, 2011)), and makes evaluation of split points faster.
3. The heterogeneous ensemble of base classifiers in HESCA has been replaced with a single Rotation Forest (Rodriguez et al., 2006) classifier.

4.4.2 The Generalised Random Shapelet Forest (RSF)

RSF (Karlsson et al., 2016) is a bagging based tree ensemble that attempts to improve the computational efficiency and predictive accuracy of the Shapelet Tree through randomisation and ensembling. At each node of each tree r univariate shapelets are selected from the training set at random. Each shapelet has a randomly selected length between predefined upper and lower limits. The quality of a shapelets is measured in the standard way with $sDist()$ and information gain, and the best is selected. The data is split, and a tree is recursively built until a stopping condition is met. New samples are predicted by a majority vote on the tree's predictions and multiple trees are ensembled.

4.4.3 MrSEQL and MrSQM

The Multiple Representation Sequence Learner (MrSEQL) (Nguyen et al., 2019), is an ensemble classifier that extends previous adaptations of the SEQL classifier (Le Nguyen et al., 2017). MrSEQL looks for the presence or absence of a pattern (shapelet) in the data. Rather than using a distance based approach to measure the presence or not of a shapelet, MrSEQL discretises subseries into words. Words are generated through two symbolic representations, using SAX (Lin et al., 2007) for time domain and SFA (Schäfer and Högqvist, 2012) for frequency domain. A set of discriminative words is selected through Sequence Learner (SEQL) and the output of training is a logistic regression model, which in concept is a vector of relevant subseries and their weights.

Diversification is achieved through the two different symbolic representations and varying the window size.

MrSQM (Le Nguyen and Ifrim, 2022) extends MrSEQL. It also combines two symbolic transformations to create words from subseries and trains a logistic regression classifier. What sets it apart is its innovative strategy for selecting features (substrings).

To begin with, MrSQM uses SFA and SAX to discretize time series subseries into words. It then utilizes a trie to store and rank frequent substrings, and applies either (a) a supervised chi-squared test to identify discriminative words or (b) an unsupervised random substring sampling method to prevent overestimating highly correlated substrings that are likely to be redundant. MrSQM establishes the number of learned representations (SFA or SAX) based on the length of the time series and utilizes an exponential scale for the window size parameter.

4.4.4 Random Dilated Shapelet Transform (RDST)

RDST (Guillaume et al., 2022) is a shapelet-based algorithm that adopts many of the techniques of convolution approaches described in Section 4.6. While traditional shapelet algorithms search for the best shapelets from the train dataset, R-DST takes a different approach by randomly selecting a large number of shapelets from the train data, typically ranging from thousands to tens of thousands, then training a linear RIDGE classifier on features derived from these shapelets.

RDST employs dilation with shapelets. Dilation is a form of down sampling, in that it defines spaces between time points. Hence, a shapelet with dilation d is compared to time points d steps apart when calculating the distance. RDST also uses two features in addition to $sDist()$: it encodes the position of the minimum distance, and records a measure of the frequency of occurrences of the shapelet based on a threshold. Hence the transformed data has $3k$ features for k shapelets.

4.4.5 Comparison of Shapelet Based Approaches

Table 4.4.5 highlights the key differences between the shapelet-based approaches.

Figure 12 shows the relative ranks of the four shapelet classifiers. RDST is the clear winner. Table 6 shows it is, on average more than 1% more accurate than MrSQM, the second-best algorithm. The shapelet based algorithms are more fundamentally different in design than, for example, interval classifiers. This is demonstrated by the spread of test accuracies shown in Figure 13 of the top three algorithms. The grouping may become redundant: RDST is more similar to convolution based algorithms (Section 4.6) in design than STC, and MRSQM has structure in common with dictionary based classifiers (Section 4.5). However, they still retain the key characteristics that, unlike convolutions, they use the training data to find subseries and, unlike dictionary based algorithms, their features include the presence or absence of a pattern.

Table 5 Key differences in shapelet based TSC algorithms

	STC	**RDST**	**RSF**	**MrSQM**
Shapelet Discovery	Random Subsequences	Random Subsequences	Random subseries	Frequent Substrings
Supervised shapelets	yes	no	yes	no
Dilation	no	yes	no	no
Discretisation	no	no	no	yes (SAX/SFA)
Classification	Rotation Forest	Ridge Classifier CV	Random Tree Ensemble	Ridge Classifier CV

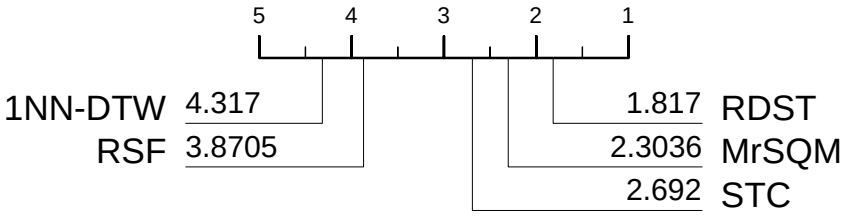
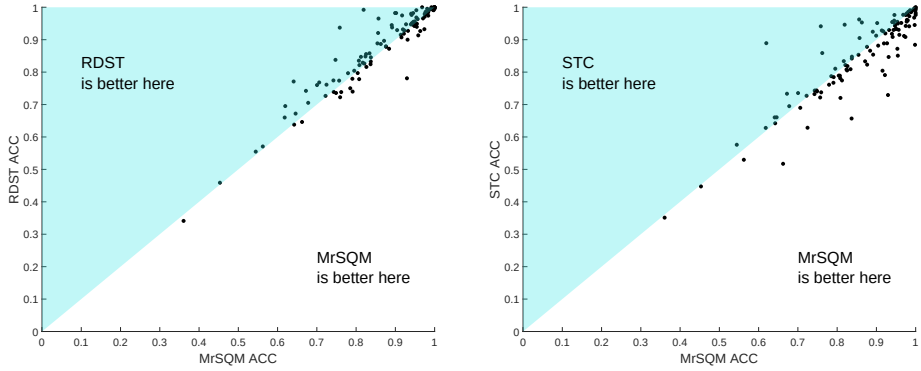


Fig. 12 Ranked test accuracy of four shapelet based classifiers and the benchmark 1NN-DTW on 112 UCR UTSC problems. Accuracies are averaged over 30 resamples of train and test splits.



(a) RDST vs MrSQM (b) STC vs MrSQM

Fig. 13 Scatter plot of test accuracies of shapelet based classifiers.

4.5 Dictionary Based

Similar to shapelet based algorithms, *dictionary approaches* extract phase-independent subseries. However, instead of measuring the distance to a subseries, each window is converted into a short sequence of discrete symbols, commonly known as a word. Dictionary methods differentiate based on word frequency and are often referred to as bag-of-words approaches. Figure 14 il-

Table 6 Summary performance measures for ShapeletBased classifiers on 30 resamples of 112 UTSC problems.

	ACC	BALACC	AUROC	NLL
RDST	0.876	0.856	0.959	0.643
MrSQM	0.863	0.841	0.952	0.689
STC	0.852	0.830	0.926	0.821
RSF	0.801	0.774	0.879	0.936

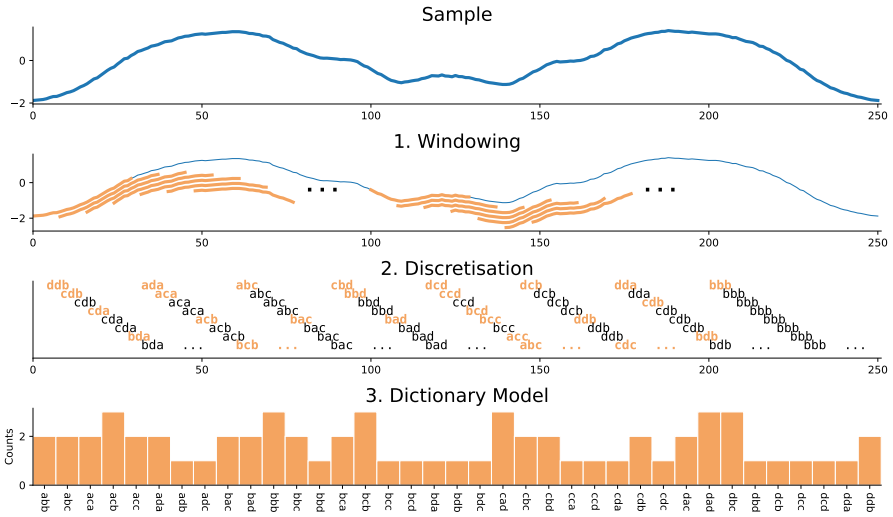


Fig. 14 Transformation of a TS into the dictionary-based model (following (Schäfer and Leser, 2017)) using overlapping windows (second to top), discretisation of windows to words (second from bottom), and word counts (bottom).

lustrates the process that algorithms following the dictionary model take to create a classifier. This process can be summarized as:

1. Extracting subseries, or windows, from a time series;
2. Transforming each window of real values into a discrete-valued *word* (a sequence of symbols over a fixed alphabet);
3. Building a sparse feature vector of histograms of word counts, and
4. Finally, using a classification method from the machine learning repertoire on these feature vectors.

Dictionary-based methods differ in the way they transform a window of real-valued measurements into discrete words. For example, the basis of the BOSS model (Schäfer, 2015) is a representation called Symbolic Fourier Approximation (SFA) (Schäfer and Högqvist, 2012). SFA works as follows:

1. Values in each window of length w are normalized to have standard deviation of 1 to obtain amplitude invariance.
2. Each normalized window of length w is subjected to dimensionality reduction by the use of the truncated Fourier transform, keeping only the first

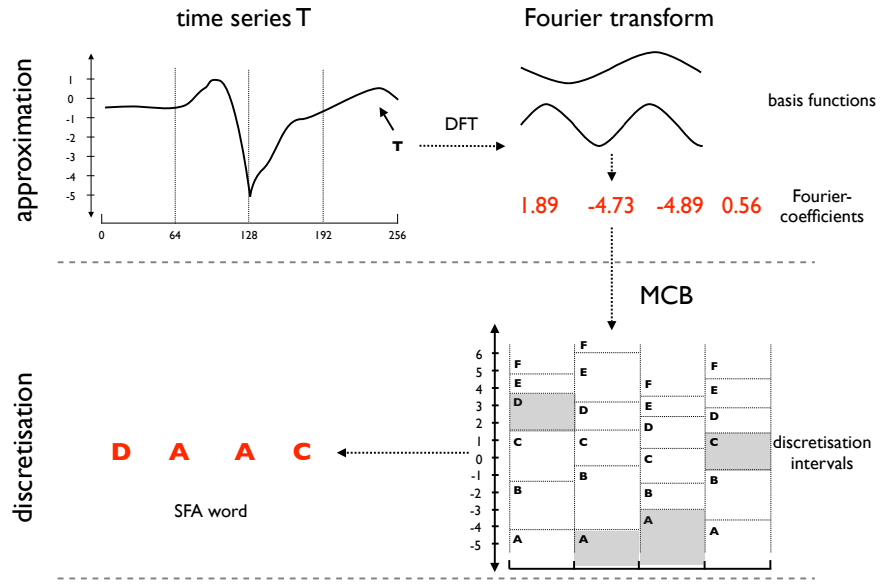


Fig. 15 The Symbolic Fourier Approximation (SFA) (from (Schäfer and Leser, 2017)): A time series (top left) is approximated using the Fourier transform (top right) and discretised to the word $DAAC$ (bottom left) using data adaptive bins (bottom right).

$l < w$ coefficients for further analysis. This step acts as a low pass filter, as higher order Fourier coefficients typically represent rapid changes like dropouts or noise.

3. Discretisation bins are derived through Multiple Coefficient Binning (MCB). It separately records the l distributions of the real and imaginary values of the Fourier transform. These distributions are then subjected to either equi-depth or equi-width binning. The resulting output consists of l sets of bins, corresponding to the target word length of l.
4. Each coefficient is discretized to a symbol of an alphabet of fixed size α to achieve further robustness against noise.

Figure 15 exemplifies this process from a window of length 128 to its DFT representation, and finally the word $DAAC$.

4.5.1 Bag-of-SFA-Symbols (BOSS)

Bag-of-SFA-Symbols (BOSS) (Schäfer, 2015) was among the top-performing algorithms in the initial bake-off study and led to significant further investigation into dictionary-based classifiers. An individual BOSS classifier undergoes the same process described earlier, whereby each sliding window is transformed into a word using SFA. Subsequently, a feature vector is generated by counting the occurrences of each word over all windows. A non-symmetric distance function is then employed with a 1-NN classifier to categorize new instances.

Experiments have shown that when presented with a query and a sample time series, disregarding words that exist solely in the sample time series using the non-symmetric distance function leads to improved performance compared to using the Euclidean distance metric.

The complete BOSS classifier is an ensemble of individual BOSS classifiers. This ensemble is created by exploring a range of parameters, assessing each base classifier through cross-validation, and keeping all base classifiers with an estimated accuracy within 92% of the best classifier. For new instances, the final prediction is obtained through a majority vote of the base classifiers.

4.5.2 Word Extraction for Time Series Classification (WEASEL)

Word Extraction for Time Series Classification (WEASEL) (Schäfer and Leser, 2017) is a pipeline classifier that revolves around identifying words whose frequency count distinguishes between classes and discarding words that lack discriminatory power. The classifier generates histograms of word counts over a broad spectrum of window sizes and word lengths parameters, including bigram words produced from non-overlapping windows. A Chi-squared test is then applied to determine the discriminatory power of each word, and those that fall below a particular threshold are discarded through feature selection. Finally, a linear RIDGE classifier is trained on the remaining feature space. WEASEL utilizes a supervised variation of SFA to create discriminative words, and it leverages an information-gain based methodology for identifying breakpoints that separate the classes.

4.5.3 WEASEL-D (with dilation)

The dictionary-based WEASEL-D, aka WEASEL 2.0 (Schäfer and Leser, 2023), is a complete overhaul of the WEASEL classifier (Schäfer and Leser, 2017). It addresses the problem of the extensive memory footprint of WEASEL by controlling the search space using randomly parameterized SFA transformations. It also significantly improves accuracy. Notably, the most prominent modification is the inclusion of dilation to the sliding window approach. Table 7 presents a comprehensive summary of its alterations.

Dilated Sliding Window: To extract subseries with non-consecutive values from a time series, a dilated sliding window approach is employed, where the dilation parameter maintains a fixed gap between each value. These dilated subseries undergo a Fourier transform, and a word is generated by discretizing them using SFA. The unsupervised learning of bins is achieved using equi-depth and equi-width with an alphabet size of 2. To improve performance, a feature selection strategy based on variance is introduced, which retains only the real and imaginary Fourier values with the highest variance.

Discretisation Parameters: Each of the 50 to 150 SFA transformations is randomly initialized subject to:

1. **Window length** w: Randomly chosen from interval $[w\_min, \ldots, w\_max]$.
2. **Dilation** d: Randomly chosen from interval $[1, \ldots, 2^{\frac{\log(n-1)}{w-1}}]$. The formula is inherited from the ROCKET-family.
3. **Word length** l: Randomly chosen from $\{7, 8\}$.
4. **Binning strategy**: Randomly chosen from {"equi-depth", "equi-width"}.
5. **First order differences**: To extract words from both, the raw time series, and its first order difference, effectively doubling the feature space.

When using an alphabet size of 2 and a length of 8, each SFA transformation creates a dictionary containing only 256 unique words of a fixed size. These dictionaries are then combined to produce a feature vector containing approximately $30k$ to $70k$ features. No feature selection is implemented by default. The resulting features serve as input for training a linear RIDGE classifier.

4.5.4 Contractable BOSS (cBOSS)

The size of the parameter grid searched by BOSS is data dependent, and BOSS uses a method of retaining ensemble members using a threshold of accuracy estimated from the train data. This makes its time and memory complexity unpredictable. BOSS was one of the slower algorithms tested in the bake off and could not be evaluated on the larger datasets in reasonable time. **Contractable BOSS (cBOSS) (Middlehurst et al., 2019)** revises the ensemble structure of BOSS to solve these scalability issues, using the same base transformations as the BOSS ensemble. cBOSS randomly selects k parameter sets of hyper-parameters (w, l and α) for BOSS base classifiers. It retains the best s classifiers (based on a cross validation estimate of accuracy) are retained for the final ensemble. cBOSS allows the k parameter to be replaced by a train time limit t through contraction, allowing the user to better control the training time of the classifier. A subsample of the train data is randomly selected without replacement for each ensemble member and an exponential weighting scheme used in the CAWPE (Large et al., 2019b) ensemble is introduced. The cBOSS alterations to the BOSS ensemble structure showed an order of magnitude improvement in train times with no reduction in accuracy.

4.5.5 SpatialBOSS

BOSS intentionally ignores the locations of words in series, classifying based on the frequency of patterns rather than their location. Spatial Boss (Large et al., 2019a) introduced location information into the design of a BOSS classifier. Spatial pyramids (Lazebnik et al., 2006) are a technique used in computer vision to retain some temporal information back into the bag-of-words paradigm. The core idea, illustrated in Figure 16 is to split the series into different resolutions, then build independent histograms on the spits. The histograms for

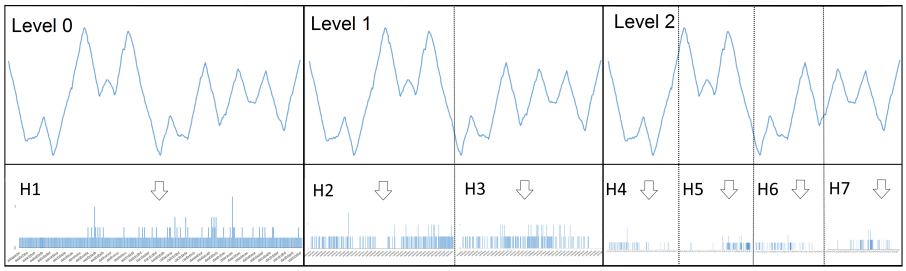


Fig. 16 An example of using a spatial pyramid to form 7 distinct word count histograms.

each level are concatenated into a single feature vector which is used with a 1-NN classifier.

4.5.6 TDE

The Temporal Dictionary Ensemble (TDE) (Middlehurst et al., 2020b) combines the best improvements introduced in WEASEL, SpatialBOSS and cBOSS and also includes several novel features. TDE is an ensemble of 1-NN classifiers which transforms each series into a histogram of word counts using SFA (Schäfer and Högqvist, 2012). From WEASEL, TDE takes the method for finding supervised breakpoints for discretisation, and captures frequencies of bigrams found from non-overlapping windows. The locality information derived from the spatial pyramids used in SpatialBOSS are incorporated. Word counts are found for each spatial subseries independently, with the resulting histograms being concatenated. Bigrams are only found for the full series. The cBOSS ensemble structure is applied with a modified parameter space sampling algorithm. It first randomly samples a small number of parameter sets, then constructs a Gaussian processes regressor on the historic accuracy for unseen parameter sets. The regressor is used to estimate the parameter set for the next candidate, and the model is then updated before the process is repeated. TDE has three additional parameters: the number of levels for the spatial pyramid, the method of generating breakpoints and whether to normalise the window or not.

4.5.7 Comparison of Dictionary Based Approaches

Table 7 shows the key design differences between the dictionary based approaches. Figure 17 shows the ranked test accuracy of five dictionary classifiers we have described, with 1-NN DTW as a benchmark (SpatialBOSS is not included because of technical challenges). WEASEL and TDE are significantly more accurate than BOSS, but WEASEL-D is the most accurate overall. Figure 18(a) illustrates the improvement of WEASEL-D over BOSS, and Figure 18(b) shows the improvement dilation provides over WEASEL. Table 8 summarises the performance of the four new dictionary algorithms.

	BOSS	**TDE**	**WEASEL**	**WEASEL 2.0**
word length	$\{8, 10, 12, 14, 16\}$	$\{8, 10, 12, 14, 16\}$	$\{4, 6\}$	$\{7, 8\}$
alphabet-size	4	4	4	2
FFT Features	First	First	Anova-F-Test	Variance
Binning	equi-depth	equi-depth, IG	IG	equi-depth, equi-width
Bigrams	No	Yes	Yes	No
Pyramids	No	Yes	No	No
Feature Selection	None	None	Chi-squared	optional (default: None)
Window Sizes	variable (Ensemble)	variable (Ensemble)	variable (concatenate)	variable (concatenate)
1st order dif.	no	no	no	yes
Dilation	no	no	no	yes
Classifier	1-NN	1-NN	Ridge Regression	Ridge Regression CV
Feature Vector Size	data dependent	data dependent	data dependent	30 to 70k

Table 7 Key differences in dictionary based TSC algorithms

WEASEL-D is on average 4% more accurate than BOSS and improves balanced accuracy by almost the same amount.

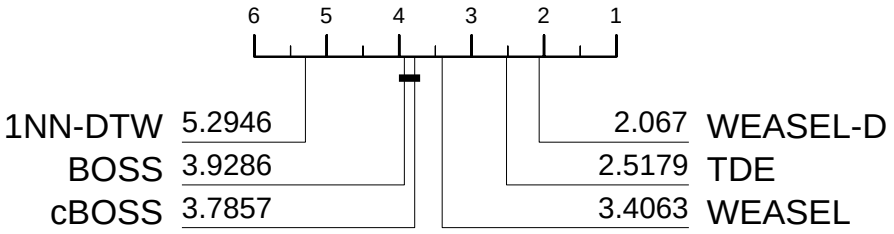
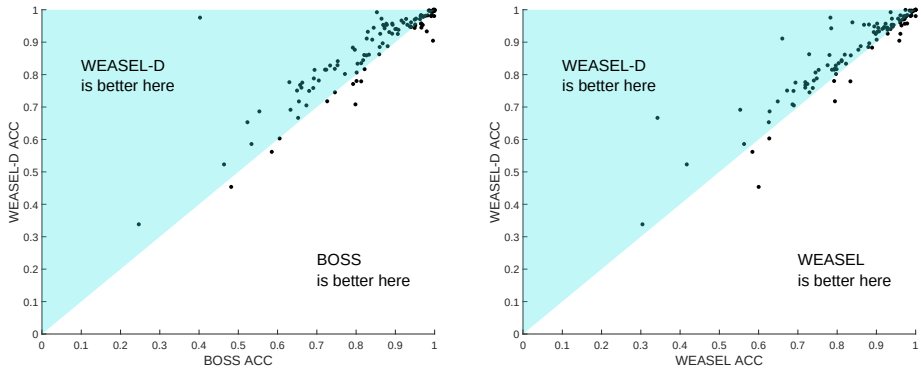


Fig. 17 Ranked test accuracy of five dictionary based classifiers with the benchmark 1-NN DTW on 112 UCR UTSC problems. Accuracies are averaged over 30 resamples of train and test splits.

4.6 Convolution Based

Kernel/Convolution classifiers use convolutions with kernels, which can be seen as subseries used to derive discriminatory features. Each kernel is convolved with a time series through a sliding dot product. Figure 19 shows the windowing process for an input time series and kernel $\omega = [-1, 0, 1]$.



(a) BOSS vs WEASEL-D (b) WEASEL vs WEASEL-D

Fig. 18 Scatter plot of test accuracies of BOSS vs WEASEL-D.

Table 8 Averaged performance statistics for five dictionary classifiers and 1NN-DTW on 30 resamples of 112 UTSC problems.

	ACC	BALACC	AUROC	NLL
WEASEL-D	0.874	0.853	0.877	0.838
TDE	0.861	0.836	0.952	0.580
WEASEL	0.845	0.824	0.945	1.299
cBOSS	0.833	0.806	0.944	0.699
BOSS	0.834	0.813	0.934	0.709

The first entry of the activation map is the result of a dot-product between $T_{1:3} * \omega = T_{1:3} \cdot \omega = 0 + 0 + 3 = 3$. Each convolution creates a series to series transform from time series to activation map. Activation maps are used to create summary features. Convolutions are closely linked to shapelets. Shapelets can be realised through a convolution operation, followed by a min-pooling operation on the array of windowed Euclidean distances. This was first observed by (Grabocka et al., 2014). The main difference between convolutions and shapelets is that shapelets are subseries from the training data whereas convolutions are found from the entire space of possible real-values.

Convolution based TSC algorithms follow a standard pipeline pattern depicted in Figure 20. The activation map is formed for each convolution, followed by pooling operations to extract one relevant feature for each operation. The resulting features are then concatenated to form a single feature vector. Finally, a Ridge classifier is trained on the output to classify the data.

4.6.1 ROCKET

The most well known convolutional approach is the **Random Convolutional Kernel Transform (ROCKET) (Dempster et al., 2020)**. ROCKET is a pipeline classifier. It generates a large number of randomly parameterised convolutional kernels (typically in the range of thousands to tens of thousands), then uses these to transform the data through two pooling operations: the

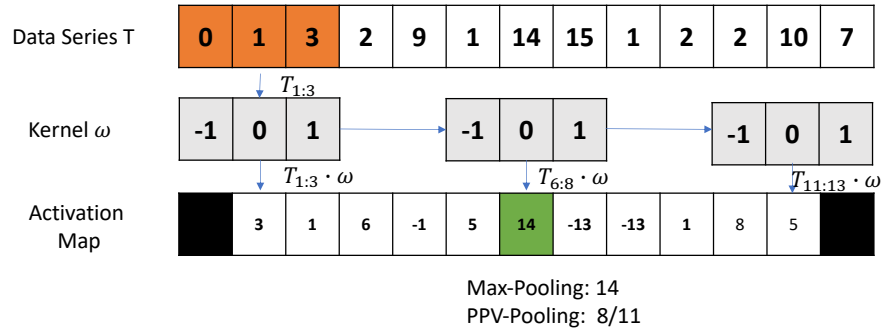


Fig. 19 The convolution operation as a sliding dot-product. The kernel $\omega = [-1, 0, 1]$ is convolved with the input series, producing an activation map. Max-pooling extracts the maximum from this activation map.

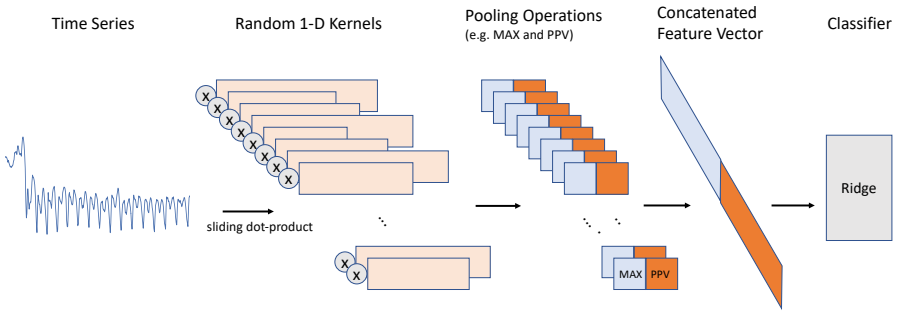


Fig. 20 Pipeline of convolution based approaches such as ROCKET, MiniROCKET or MultiROCKET.

max value and the proportion of positive values (PPV). These two features are concatenated into a feature vector for all kernels. For k kernels, the transformed data has $2k$ features.

Kernel Parameters: In ROCKET, each kernel is randomly initialised with respect to the following parameters:

1. the *kernel length* l, randomly selected from $\{7, 9, 11\}$;
2. the *kernel weights* w, randomly initialised from a normal distribution;
3. a *bias term* b added to the result of the convolution operation;
4. the *dilation* d to define the spread of the kernel weights over the input instance, which allows for detecting patterns at different frequencies and scales. Randomly drawn from an exponential function; and
5. padding p the input series at the start and the end (typically with zeros), such that the activation map has the same length as the input;

The result of applying a kernel ω with dilation d to a time series T at offset i is defined by:

$$T_{i:(i+l)} * \omega = \sum_{j=0}^{l-1} T_{i-(\lfloor m/2 \rfloor)\times d)+(j\times d)} \times w_j$$

The feature vectors are then used to train a Ridge Classifier using cross-validation to train the L_2-regularisation parameter α. A logistic regression classifier is suggested as a replacement for larger datasets. The combination of ROCKET with logistic (RIDGE) regression is conceptually the same as a single-layer CNN with randomly initialised kernels and softmax loss.

4.6.2 Mini-ROCKET

ROCKET has two extensions. The first extension is MiniROCKET (Dempster et al., 2021), which speeds up ROCKET by over an order of magnitude with no significant difference in accuracy. MiniROCKET removes many of the random components of ROCKET, making the classifier almost deterministic. The kernel length is fixed to 9, only two weight values are used, and the bias value is drawn from the convolution output. Only the PPV is extracted, discarding the max. These changes alongside general optimisations taking advantage of the new fixed values provide a considerable speed-up to the algorithm. MiniROCKET generates a total of 10k features from 10k kernels and PPV pooling.

4.6.3 Multi-ROCKET

MultiROCKET (Tan et al., 2022) further extends the MiniROCKET improvements, extracting features from first order differences and adding three new pooling operations extracted from each kernel: mean of positive values (MPV), mean of indices of positive values (MIPV) and longest stretch of positive values (LSPV). MultiROCKET generates a total of 50k features from 10k kernels and 5 pooling operations.

4.6.4 Hydra and Hydra-MultiROCKET

HYbrid Dictionary–ROCKET Architecture (Hydra) (Dempster et al., 2022) is a model that combines dictionary-based and convolution-based models. It begins by utilizing random convolutional kernels to calculate the activation of time series. These kernels, unlike ROCKET, are arranged into g groups of k kernels each. In each group of k kernels, the activation of a kernel with the input time series is calculated, and we record how frequently this kernel is the best match (counts the highest activation). This results in a k-dimensional count vector for each of the g groups, resulting in a total of $g \times k$ features.

To implement Hydra, the time series is convolved with the kernels, and the resulting activation maps are organized into g groups. Next, an (arg)max

	ROCKET	**MiniR**	**MultiR**	**Hydra**
kernel length	{7,9,11}	9	9	9
kernel weights	$N(0,1)$	$-1,2$	$-1,2$	$N(0,1)$
bias	$U(-1,1)$	from output	from output	none
dilation	random	fixed (relative to input)	fixed (relative to input)	random
padding	random	fixed	fixed	always
pooling operators	MAX, PPV	PPV	PPV, MPV, MIPV, LSPV	Response per Kernel/Group
1st order difference	no	no	yes	yes
feature vector size	20k	10k	50k	relative to input

Table 9 Key Differences in approaches from ROCKET to MiniROCKET to MultiROCKET.

operation is performed to count the number of best matches, and the counts for each group's dictionary are increased. The main hyperparameters to consider are the number of groups and the number of kernels per group, with default values of $g = 64$ and $k = 8$. Hydra is applied to both the time series and its first-order differences. The best results in (Dempster et al., 2022) come from combining features from Hydra with features from MultiROCKET to form an ensemble. We call this classifier Hydra-MultiROCKET.

4.6.5 Comparison of Convolution Based Approaches

Table 9 highlights the key differences between the convolution based approaches.

Hydra-MultiROCKET is the top performer, and is significantly better ranked than the next best, Multi-ROCKET. Table 10 and Figure 22(a) show that the actual difference between the algorithms is small. Progress in the field is demonstrated by Figure 22(b). Hydra-MR is nearly 2% better on average than ROCKET, which itself was considered state of the art as recently as 2020 Bagnall et al. (2020a).

Table 10 Summary performance measures for ConvolutionBased classifiers on 30 resamples of 112 UTSC problems.

	ACC	BALACC	AUROC	NLL
Hydra-MR	0.884	0.866	0.908	0.737
Multi-R	0.881	0.863	0.887	0.770
Hydra	0.870	0.850	0.884	0.787
ROCKET	0.868	0.850	0.875	0.862
Mini-R	0.874	0.856	0.878	0.834
Arsenal	0.866	0.846	0.875	0.875

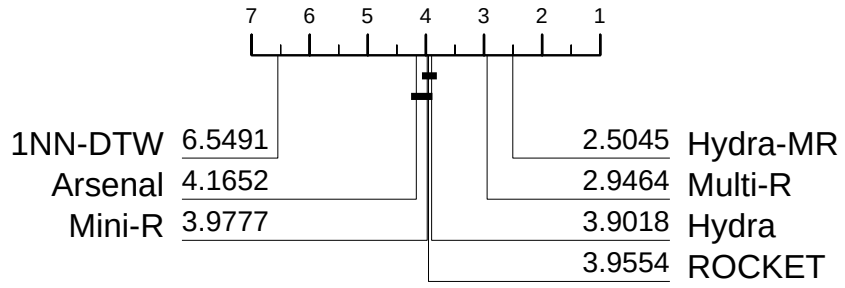
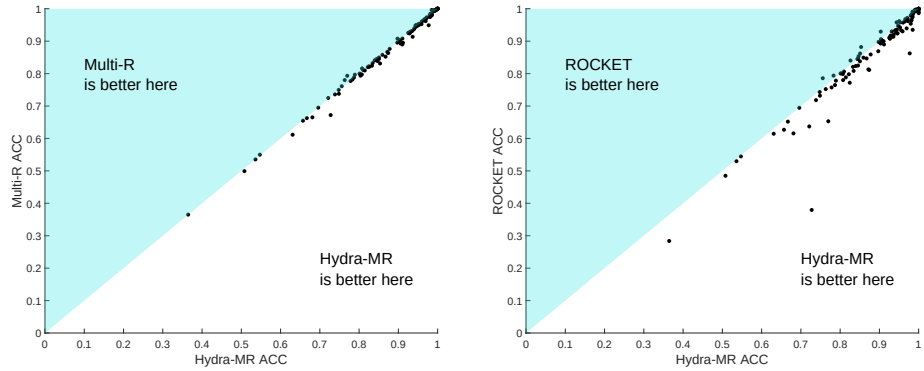


Fig. 21 Ranked test accuracy of four shapelet based classifiers and the benchmark 1NN-DTW on 112 UCR UTSC problems. Accuracies are averaged over 30 resamples of train and test splits.



(a) (b)

Fig. 22 Scatter plot of test accuracies of convolution based classifiers.

4.7 Deep Learning

Deep learning has been the most active area of TSC research since the bake off in terms of the number of publications. It was thought by many that the impact deep learning had on fields such as vision and speech would be replicated in TSC research. In a paper with *"Finding AlexNet for time series classification"* in the title, Fawaz et al. (2020) discuss the impact AlexNet had on computer vision and observe that this lesson indicates that *"given the similarities in the data, it is easy to suggest that there is much potential improvement for deep learning in TSC."*. A highly cited survey paper Fawaz et al. (2019) found that up to that point, ResNet (Wang et al., 2017) was the most accurate TSC deep learner. Subsequently, the same group proposed InceptionTime (Fawaz et al., 2020), which is not significantly different to HC1 in terms of accuracy (Bagnall et al., 2020a). Since InceptionTime there have been a huge number of deep learning papers proposing TSC algorithms: a recent survey (Foumani et al., 2023) references 246 papers, most of which have been published in the

Table 11 Overview of recently proposed deep learning classifiers.

Name	Year	Code	Uni/Mul	Benchmark
Disjoint-CNN	2021	y	M	MTCS-26
Inception-FCN	2021	y	U	UTCS-85
KDCTime	2022	n	U	UTCS-113
Multi-Stage-Att	2020	n	M	own
CT_CAM	2020	n	M	15 MTCS
CA-SFCN	2020	y	M	14
RTFN	2021	n	U/M	UTCS-85, MTCS-30
LAXCAT	2021	n	M	4
MACNN	2021	y	U	UTCS-85
T2	2021	y	M	own
GTN	2021	y	M	MTCS-13
TRANS	2021	n	M	own
FMLA	2022	n	U	UTCS-85
AutoTransformer	2022	n	U	UTCS-85
BENDER	2021	y	M	5 EEG
TST	2021	y	M	MTCS-11
TARNET	2022	y	M	MTCS/UCI-34

last three years. Table 4.7 summarises some recently proposed deep learning classification algorithms. Without naming and shaming, there are several concerning trends in the deep learning TSC research thread. Most seriously, there is a tendency to perform model selection on test data, i.e. maximize the test accuracy over multiple epochs. This is obviously biased, yet seems to happen even with publications in highly selective venues. Secondly, many papers do not make their source code available. Given all these algorithms are based on standard tools like TensorFlow and PyTorch, this seem inexcusable. Thirdly, they often evaluate on subsets of the archive without any clear rationale as to why. Most are evaluated only on the multivariate archive. Whilst cherry-picking data is questionable, using just MTSC data is not. However, it puts them beyond the scope of this paper. Fourthly, they frequently only compare against other deep learning classifiers, often set up as weak straw men. Finally, they often do not seem to offer any advance on previous research. We have not seen any algorithm that can realistically claim to outperform InceptionTime (Fawaz et al., 2020). Because of this, we restrict our attention to three deep learning algorithms. We include a standard CNN implementation as a baseline. We use the same CNN structure as used in the deep learning bake off (Fawaz et al., 2019). We evaluate ResNet since it was best performing in (Fawaz et al., 2019). InceptionTime Fawaz et al. (2020) is included since it is, to the best our knowledge, best in category for deep learning.

4.7.1 CNN

Convolution Neural Networks (CNN), were first introduced in (Fukushima, 1980), and have gained widespread use in image recognition. Their popularity has increased significantly since AlexNet won the ImageNet competition in

2012. CNNs comprise three types of layers: convolutional, pooling, and fully connected. The convolutional layer slides a filter over a time series, extracting features that are unique to the input. Convolving a one-dimensional filter with the input produces an activation or feature map.

The result of applying one filter ω to a time series T at offset t is defined by:

$$C_t = f\left(\omega * T_{t:(t+l)} + b\right) \forall t \in [1 \dots n - l + 1]$$

Where the filter ω is of length l, the bias parameter is b, and f is a non-linear activation function such as ReLu applied to the result of the convolution. One significant advantage of CNNs is that the filter weights are shared across each convolution, reducing the number of weights that must be learned when compared to fully connected neural networks. But instead of manually setting filter weights, these are learned by the CNN directly from the training data.

As multiple learned filters are applied to the input, each resulting in one activation map of roughly the same size as the input, a pooling layer is used in-between every two convolution layers. A pooling layer, such as Max or Min-pooling, reduces the number of features in each map to i.e. the maximum value, thus providing phase-invariance. After several blocks of convolutional and pooling layers, one or more fully connected layers follow. Finally, a softmax layer with one output neuron per class is used in the final layer.

4.7.2 ResNet

The Residual Network (ResNet) Wang et al. (2017), is a deep learning architecture that has been successfully adapted for time series analysis. ResNet is composed of three residual blocks, each comprising two main components: (a) three convolutional layers that extract features from the input data followed by batch normalization and a ReLu non-linear activation function, and (b) a shortcut connection that allows the direct propagation of information from earlier layers to later ones.

The shortcut connection is designed to mitigate the vanishing gradients problem for deep neural networks, and the convolutional layers extract features from time series data. At the end of the model, the features are passed through one Global Average Pooling (GAP) and one fully-connected softmax layer is used with the number of neurons equal to the number of classes.

4.7.3 InceptionTime

InceptionTime is a deep learning model proposed by (Fawaz et al., 2020). It is an ensemble of five deep learning classifiers, each with the same architecture built on cascading Inception modules (Szegedy et al., 2015). Diversity is achieved through randomising initial weight values in each of the five models.

The network, illustrated in Figure 24, is composed of two consecutive residual blocks. Where each residual block is composed of three inception modules. The input of the residual block is connected via a shortcut connection to the

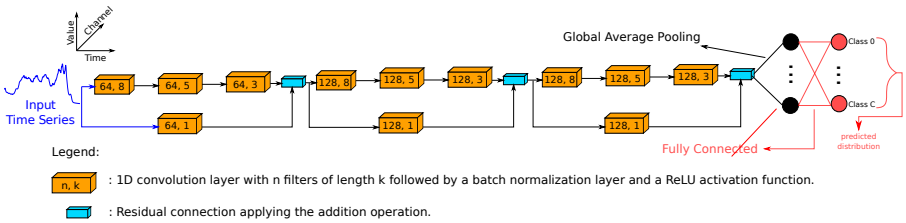


Fig. 23 Overview of the ResNet structure, image taken from (Ismail-Fawaz et al., 2022) with permission.

block's output, to address the vanishing gradient problem. A Global Average Pooling (GAP) layer follows the two residual blocks. Finally, a fully-connected softmax output layer is used with the number of neurons equal to the number of classes. An inception module first applies a *bottleneck layer*, to transform an input multivariate TS to a lower dimensional TS. It then applies multiple convolutional filters of varying kernel sizes to capture temporal features at different scales.

Key design differences to ResNet are ensembling of models, the use of bottleneck layers and the use of only two residual blocks, as opposed to three in ResNet.

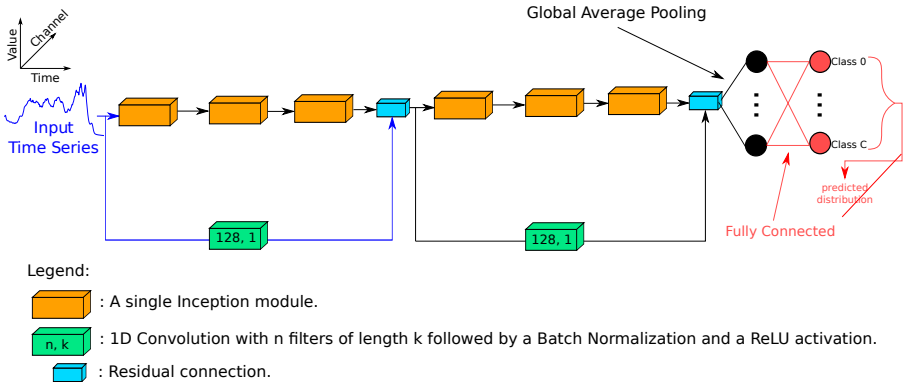


Fig. 24 Overview of the InceptionTime structure, image taken from (Ismail-Fawaz et al., 2022) with permission

4.7.4 Comparison of Deep Learning Based Approaches

Figure 25 shows the relative performance of the three deep learning algorithms and 1NN-DTW. Averaged statistics are shown in Table 12. The results confirm our prior belief: CNN is no better than 1NN-DTW, ResNet is significantly better than CNN, and InceptionTime is significantly better than ResNet.

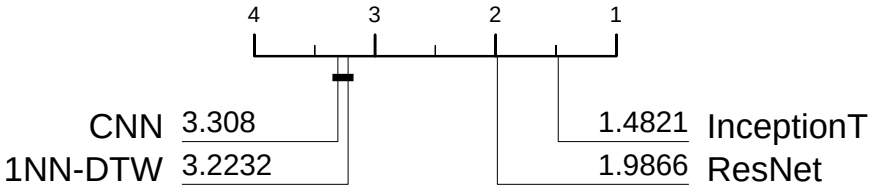


Fig. 25 Ranked test accuracy of three deep learning based classifiers and the benchmark 1NN-DTW on 112 UCR UTSC problems. Accuracies are averaged over 30 resamples of train and test splits.

Table 12 Summary performance measures for Deep Learning classifiers on 30 resamples of 112 UTSC problems.

	ACC	BALACC	AUROC	NLL
InceptionTime	0.874	0.859	0.959	0.743
ResNet	0.833	0.818	0.939	1.630
CNN	0.727	0.727	0.848	3.175

4.8 Hybrid

The nature of the data and the problem dictate which category of algorithm is most appropriate. The most accurate algorithms on average, with no apriori knowledge of the best approach, combine multiple transformation types in a hybrid algorithm. We define a hybrid algorithm as one which by design encompasses or ensembles multiple of the discriminatory representations we have previously described. Some algorithms will naturally include multiple transformation characteristics, but are not classified as hybrid approaches. For example, many interval approaches extract unsupervised summary statistics from the intervals they select, but as the focus of the algorithm is on generating features from intervals we would not consider it a hybrid.

The overall best performing approach in the bake off by a significant margin was the **Collective of Transformation Ensembles (COTE) Bagnall et al. (2015)**, which at the time was the only algorithm that explicitly ensembles over different representations. It has been subsequently renamed Flat-COTE due to its structure: it is an ensemble of 35 time series classifiers built in the time, auto-correlation, power spectrum and shapelet domains. The components of the ST-HESCA Hills et al. (2014) and EE Lines and Bagnall (2015) ensembles are pooled with classifiers built on autocorrelation (ACF) and power spectrum (PS) representation. All together, this includes the eight classifiers built on the shapelet transform from ST-HESCA, the 11 elastic distance 1-NN classifiers from EE and the eight HESCA classifiers built on ACF and PS transformed series. A weighed vote is used to label new cases, with each classifier being weighted using its train set cross-validation accuracy.

The COTE family of classifiers has evolved since Flat-COTE, and new hybrid algorithms have been produced following the success shown by ensembling multiple representations.

4.8.1 HIVE-COTE (HCα)

The Hierarchical Vote Collective of Transformation Ensembles (HIVE-COTE) Lines et al. (2018) was proposed to overcome some of the problems with Flat-COTE. This first version of HIVE-COTE, subsequently called HIVE-COTE$_\alpha$ (HC$_\alpha$), is a heterogeneous ensemble containing five modules each from a different representation: EE from the distance based representation; TSF from interval based methods; BOSS from dictionary based approaches and ST-HESCA from shapelet based techniques and the spectral based RISE. The five modules are ensembled using the Cross-validation Accuracy Weighted Probabilistic Ensemble (CAWPE, known at the time as HESCA) Large et al. (2019b). CAWPE employs a tilted probability distribution using exponential weighing of probabilities estimated for each module found through cross-validation on the train data. The weighted probabilities from each module are summed and standardised to produce the HIVE-COTE probability prediction.

4.8.2 HIVE-COTE version 1 (HC1)

Whilst state-of-the-art in terms of accuracy, HC$_\alpha$ scales poorly. A range of improvements to make HIVE-COTE more usable were introduced in HIVE-COTE v1.0 (HC1) (Bagnall et al., 2020a). HC1 has four modules: it drops the computationally intensive EE algorithm without loss of accuracy. BOSS is replaced by the more configurable cBOSS (Middlehurst et al., 2019). The improved randomised version of STC (Bostrom and Bagnall, 2017) is included with a default one hour shapelet search and the Rotation Forest classifier. TSF and RISE had usability improvements. HC1 is designed to be contractable, in that you can specify a maximum train time.

4.8.3 HIVE-COTE version 2 (HC2)

In 2021, HIVE-COTE was again updated to further address scalability issues and reflect recent innovations to individual TSC representations and HIVE-COTE v2.0 (HC2) (Middlehurst et al., 2021) was proposed. In HC2, RISE, TSF and cBOSS are replaced, with only STC retained. TDE (Middlehurst et al., 2020b) replaces cBOSS as the dictionary classifier. DrCIF replaces both TSF and RISE for the interval and frequency representations. An ensemble of ROCKET classifiers called the Arsenal is introduced as a new convolutional based approach. Estimation of test accuracy via cross-validation is replaced by an adapted form of out-of-bag error, although the final model is still built using all training data. Unlike previous versions, HC2 is capable of classifying multivariate time series. Figure 26 illustrates the structure of HC2.

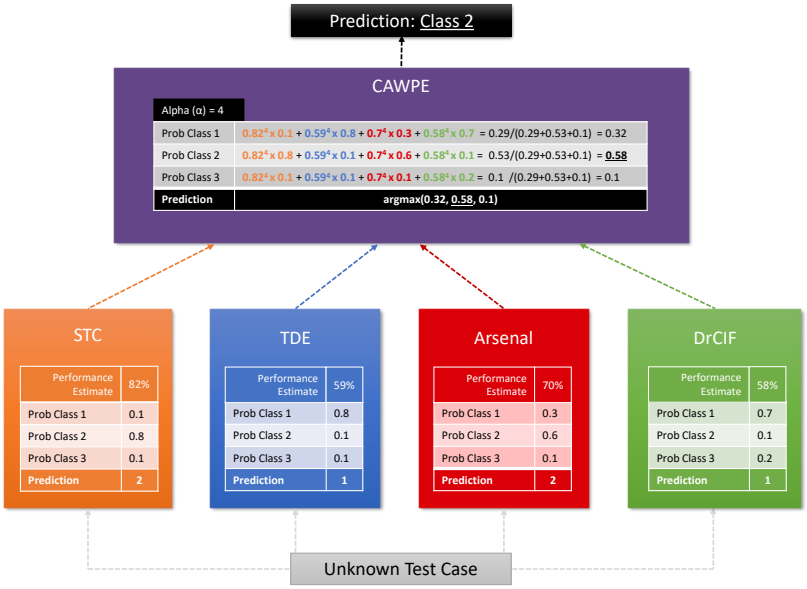


Fig. 26 Overview of the HIVE-COTE version 2 ensemble structure.

4.8.4 TS-CHIEF

The Time Series Combination of Heterogeneous and Integrated Embedding Forest (TS-CHIEF) (Shifaz et al., 2020) is a homogeneous ensemble where hybrid features are embedded in tree nodes rather than modularised through separate classifiers. The TS-CHIEF is made up of an ensemble of trees which embed distance, dictionary and spectral base features. At each node, a number of splitting criteria from each of these representations are considered. These splits use randomly initialised parameters to help maintain diversity in the ensemble. The dictionary based splits are based on BOSS, distance splits based on EE and interval splits based on RISE. The goal of TS-CHIEF was to obtain the benefits of multiple representations without the massive processing requirement of the original HIVE-COTE.

4.8.5 Comparison of Hybrid Approaches

Figure 27 shows that HC2 is significantly more accurate than the other two classifiers on all four metrics shown in Table 13. The scatter plots in Figure 28 shows it is consistently better, with higher variance compared to TS-CHIEF.

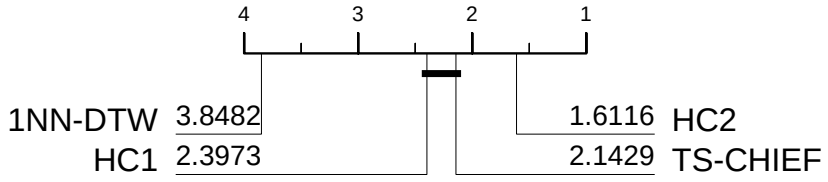
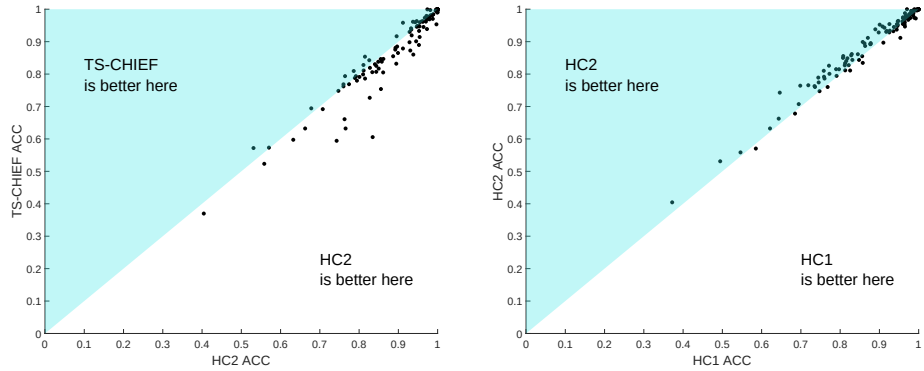


Fig. 27 Ranked test accuracy of four shapelet based classifiers and the benchmark 1NN-DTW on 112 UCR UTSC problems. Accuracies are averaged over 30 resamples of train and test splits.



(a) (b)

Fig. 28 Scatter plot of test accuracies of hybrid classifiers.

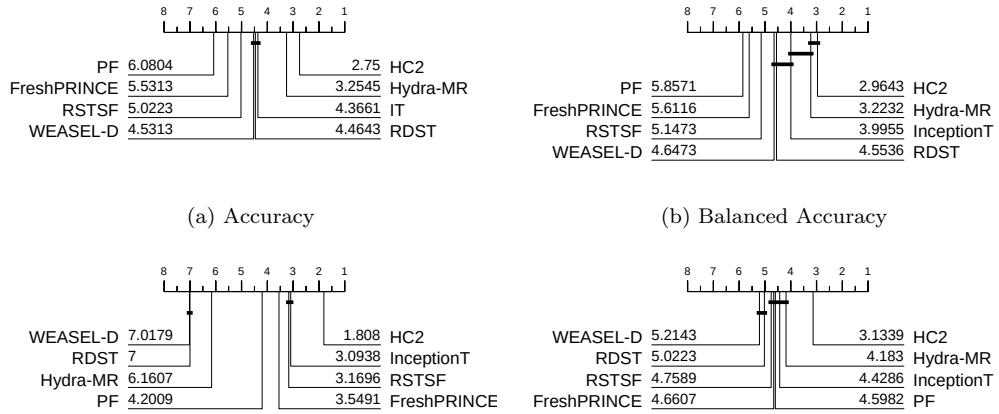
Table 13 Summary performance measures for Hybrid classifiers on 30 resamples of 112 UTSC problems.

	ACC	BALACC	AUROC	NLL
HC2	0.892	0.871	0.968	0.518
HC1	0.878	0.857	0.960	0.610
TS-CHIEF	0.879	0.854	0.964	0.677

5 Results

To keep the analysis tractable, we restrict further analysis of performance to the best classifier in each of the eight categories. Figure 29 shows the ranking of these classifiers on the 112 UCR data for 30 resamples of train/tests splits. HC2 is significantly better than all others on all metrics except balanced accuracy, where there is no difference between HC2 and Hydra-MR. This suggests that Hydra-MR may be better at problems where there is class imbalance, indicating a possible area of improvement of HC2. There is no significant difference between IT, RDST and WEASEL-D in terms of test accuracy. The picture for AUROC and NLL is more confused lower down the order, indic-

ating that many of the classifiers are bad at ranking tests cases (AUROC) or estimating probabilities.

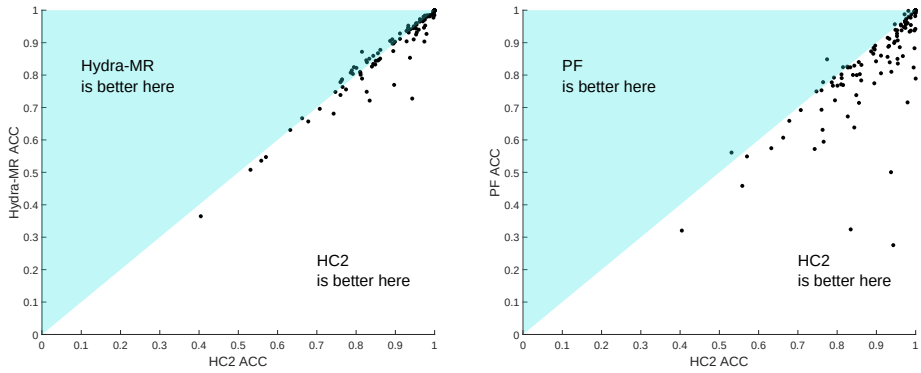


(a) Accuracy (b) Balanced Accuracy

(c) AUROC (d) Negative Log Likelihood

Fig. 29 Averaged ranked performance statistics for eight best of category algorithms on 112 UCR UTSC problems. Statistics are averaged over 30 resamples of train and test splits.

For context, Figure 30 shows the scatter plot of HC2 against the next best (Hydra-MR) and the worst performing (PF). It is worth reiterating that PF is significantly better than both EE and 1-NN DTW, both of which were considered state of the art until recently.



(a) HC2 W/D/L = 66/7/44 (b) W/L/D = 95/3/14

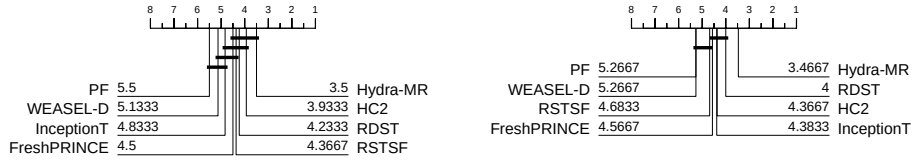
Fig. 30 Scatter plot of test accuracies of state of the art classifiers.

For convenience, Table 14 summarises the summary statistics presented in Section 4. HC2 is on average about 0.5% more accurate than Hydra-MR, over 6% more accurate than PF and over 12% more accurate than 1NN-DTW.

Table 14 Summary performance measures for best in category classifiers on 30 resamples of 112 UTSC problems.

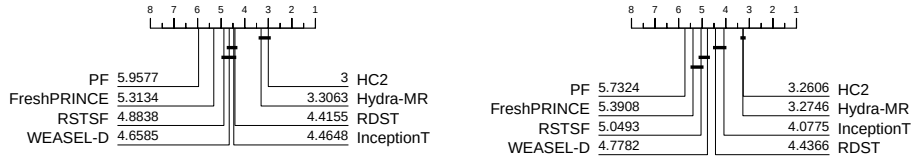
	ACC	BALACC	AUROC	NLL
HC2	0.892	0.871	0.968	0.518
Hydra-MR	0.884	0.866	0.887	0.770
InceptionT	0.874	0.859	0.959	0.743
RDST	0.876	0.856	0.879	0.821
WEASEL-D	0.874	0.853	0.877	0.838
RSTSF	0.864	0.842	0.961	0.693
FreshPRINCE	0.855	0.834	0.958	0.678
PF	0.837	0.819	0.941	0.680
1NN-DTW	0.765	0.739	0.771	1.627

5.1 Performance on New TSC Datasets



(a) Accuracy (b) Balanced Accuracy

Fig. 31 Averaged ranked performance statistics for eight best of category algorithms on 30 new UTSC problems. Statistics are averaged over 30 resamples of train and test splits.



(a) Accuracy (b) Balanced Accuracy

Fig. 32 Averaged ranked performance statistics for eight best of category algorithms on 142 TSC problems. Statistics are averaged over 30 resamples of train and test splits.

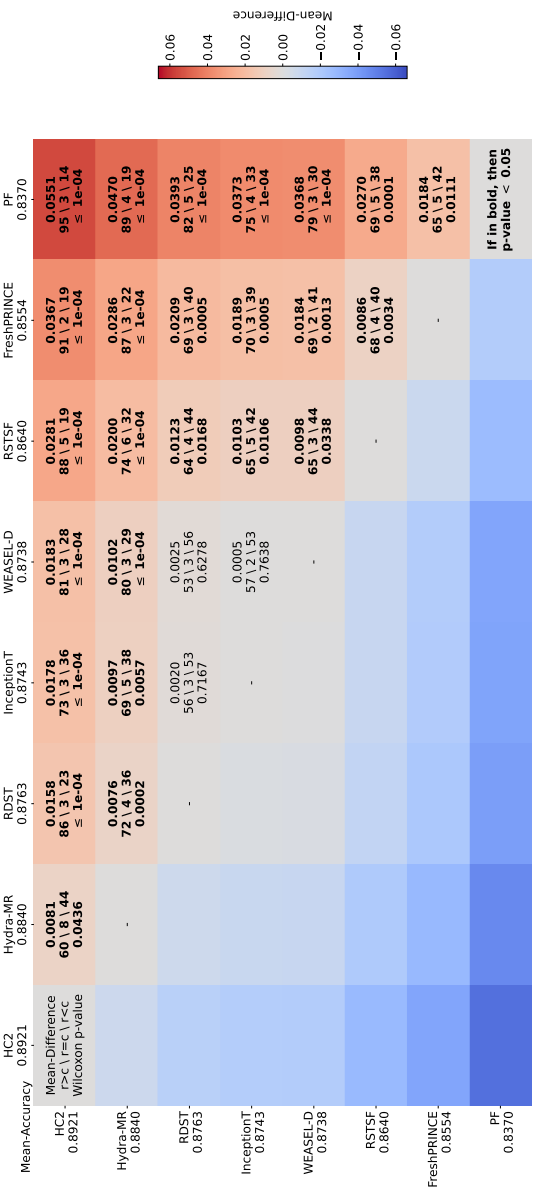


Fig. 33 Summary performance statistics for eight classifiers on 142 datasets, generated with a multiple comparison tool.

HC2 performs the best on the 112 datasets. However, HIVE-COTE has been in development for over five years, and all advances were judged by evaluation on these data. As acknowledged in Middlehurst et al. (2021), there is always the risk of the introduction of subconscious bias in the design decisions that lead to the new algorithms. To counter this, we have assembled 30 new

datasets, as described in Section 3.1. Figure 31 shows the ranks for the eight best of category on these data sets. The top clique for accuracy contains Hydra-MR, HC2, RDST and RSTSF. In terms of balanced accuracy, Hydra-MR is significantly better than other approaches. This mirrors the results on the original 112, where HC2 performs worse on balanced accuracy than it does with accuracy, suggesting it does not handle class imbalance well. Figure 32 shows the results for the combined 142 datasets. The performance of HC2 and Hydra-MR is now similar, and they are in a clique that is significantly better than the other six classifiers. Figure 33 summarises the relative performance of the eight classifiers using a new heatmap tool[9].

6 Analysis

Relative performance on test suites is important when evaluating classifiers, but it does not necessarily generalise to new problems. There will be problem domains and specific applications where different classifiers will be the most effective. Furthermore, characteristics such as the variability in performance and the run time complexity of algorithms are also of great interest to the practitioner.

We model the approach used in Bagnall et al. (2017) by comparing performance by data characteristics using all 142 datasets. Tables 15, 16 and 17 break performance down by series length, train set size and number of classes. HC2 and Hydra-MR are first or second on average in each category. HC2 seems to do better with longer series. Hydra-MR performs better with larger train set sizes. Table 18 breaks down performance by problem type. HC2 and Hydra-MR are the top two ranked in all categories. Hydra-MR does particularly well on image outlines, whereas HC2 excels at electric devices and spectrograms.

Table 15 Average rank of classifiers on 30 resamples of 142 TSC problems split by series length.

	1-199 (44)	200-499 (44)	500-999 (27)	1000+ (27)
HC2	**3.239 (1)**	3.034 (2)	**3.111 (1)**	**2.444 (1)**
Hydra-MR	3.511 (2)	**2.841 (1)**	3.407 (2)	3.630 (2)
InceptionT	4.648 (4)	4.682 (5)	3.389 (3)	4.889 (5)
RDST	4.795 (6)	3.807 (3)	4.944 (5)	4.259 (3)
WEASEL-D	4.443 (3)	4.420 (4)	4.833 (4)	5.222 (7)
RSTSF	4.648 (4)	5.261 (6)	5.056 (6)	4.481 (4)
FreshPRINCE	4.784 (5)	6.034 (8)	5.204 (7)	5.111 (6)
PF	5.932 (7)	5.920 (7)	6.056 (8)	5.963 (8)

Run time is clearly an important consideration. The speed of the ROCKET family of classifiers is a significant feature. It was stated in the bake off that *"[a]n algorithm that is faster than [the current state of the art] but not significantly less accurate would be a genuine advance in the field"*. ROCKET

[9] `https://github.com/MSD-IRIMAS/Multi_Comparison_Matrix`

Table 16 Average rank of classifiers on 30 resamples of 142 TSC problems split by train set size.

	1-99 (42)	100-299 (32)	300-699 (47)	700+ (21)
HC2	**2.607 (1)**	**2.891 (1)**	3.191 (2)	3.524 (2)
Hydra-MR	3.857 (2)	3.359 (2)	**3.064 (1)**	**2.667 (1)**
InceptionT	4.810 (5)	4.250 (4)	4.468 (3)	4.095 (3)
RDST	4.583 (4)	3.547 (3)	4.723 (5)	4.714 (5)
WEASEL-D	4.345 (3)	4.656 (5)	4.702 (4)	5.190 (7)
RSTSF	4.583 (4)	5.031 (6)	5.085 (7)	4.810 (6)
FreshPRINCE	5.607 (6)	6.156 (8)	4.957 (6)	4.238 (4)
PF	5.607 (6)	6.109 (7)	5.809 (8)	6.762 (8)

Table 17 Average rank of classifiers on 30 resamples of 142 TSC problems split by number of classes.

	2 (50)	3-5 (38)	6-10 (30)	11+ (24)
HC2	3.360 (2)	**2.618 (1)**	**3.017 (1)**	2.833 (2)
Hydra-MR	**3.180 (1)**	3.737 (2)	3.617 (2)	**2.500(1)**
InceptionT	4.450 (3)	4.645 (4)	5.167 (7)	3.333 (3)
RDST	4.870 (6)	4.474 (3)	4.050 (3)	3.833 (4)
WEASEL-D	4.600 (5)	4.684 (5)	4.717 (4)	4.667 (5)
RSTSF	4.480 (4)	4.684 (5)	5.017 (6)	5.875 (7)
FreshPRINCE	4.910 (7)	5.447 (6)	4.967 (5)	6.375 (8)
PF	6.150 (8)	5.711 (7)	5.450 (8)	6.583 (6)

Table 18 Average rank of classifiers on 30 resamples of 142 TSC problems split by problem type.

	DEVICE (11)	ECG (7)	IMAGE (34)	MOTION (27)
HC2	**2.000 (1)**	**2.143 (1)**	3.441 (2)	**2.759 (1)**
Hydra-MR	2.455 (2)	2.571 (2)	**2.912(1)**	3.056 (2)
InceptionT	4.455 (4)	4.286 (3)	4.676 (5)	3.833 (4)
RDST	3.909 (3)	4.571 (5)	3.971 (4)	3.722 (3)
WEASEL-D	5.364 (7)	4.286 (4)	3.706 (3)	4.741 (5)
RSTSF	5.091 (6)	5.429 (6)	5.324 (6)	5.926 (6)
FreshPRINCE	4.909 (5)	5.571 (7)	5.676 (7)	6.019 (8)
PF	7.818 (8)	7.143 (8)	6.294 (8)	5.944 (7)

	SENSOR (35)	SIMULATED (12)	SPECTRO (12)
HC2	**3.414 (1)**	3.333 (2)	**2.167 (1)**
Hydra-MR	3.957 (2)	**3.083 (1)**	3.792 (3)
InceptionT	4.200 (4)	3.917 (4)	5.958 (8)
RDST	4.871 (5)	5.667 (7)	5.083 (5)
WEASEL-D	4.900 (6)	6.833 (8)	4.083 (4)
RSTSF	4.071 (3)	5.167 (6)	3.417 (2)
FreshPRINCE	5.171 (7)	3.750 (3)	5.667 (6)
PF	5.414 (8)	4.250 (5)	5.833 (7)

and the subsequent refinements fulfil this criteria and represent an important advance. Table 19 shows the total run time for classifiers on the 142 problems and Figure 34 shows the plot of rank against run time (on a logarithmic scale). HC2 is clearly much slower than Hydra-MR. This is at least in parts the result of the configuration of HC2. For example, STC, a component of HC2, performs at least an hour long randomised search for shapelets. This is clearly unnecessary for many problems, and the algorithm could be better configured.

Nevertheless, there is no doubt that Hydra-MR offers the best accuracy/train time trade off: it is on average as accurate as HC2 but orders of magnitude faster. If results are required very quickly or train set sizes are large, Hydra-MR would seem to be the better option. However, for smaller train set sizes (see Table 16), or if probabilities or orderings are required (see Table 14), the results indicate that HC2 is the better option.

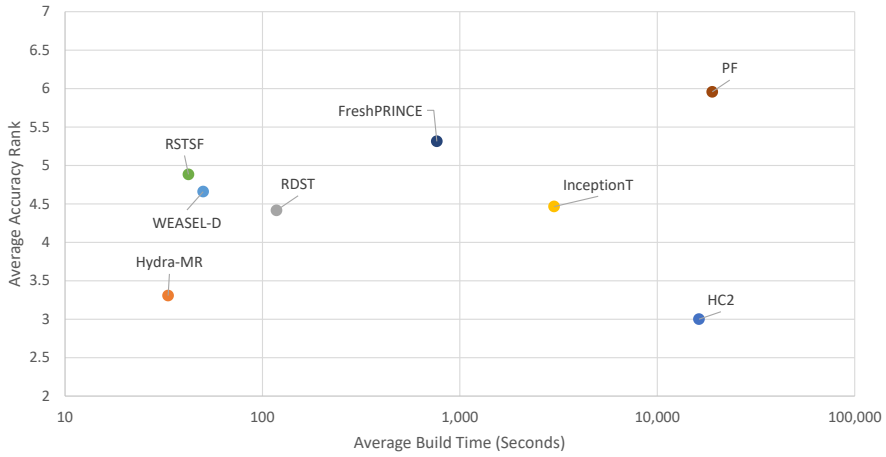


Fig. 34 Averaged accuracy rank against build time on 142 UTSC problems.

Table 19 Build time statistics on 142 TSC problems.

	Total (Hours)	Mean (Minutes)	Min (Seconds)	Max (Hours)
Hydra-MR	1.274	0.538	0.086	0.134
RSTSF	1.626	0.687	0.690	0.064
WEASEL-D	1.937	0.818	0.233	0.171
RDST	4.610	1.948	15.222	0.235
FreshPRINCE	30.115	12.725	12.343	4.176
InceptionT	118.426	50.039	159.949	7.419
HC2	641.764	271.168	7316.688	44.610
PF	903.823	381.897	0.523	287.647

A further consideration when choosing a classifier is the variability of performance, for a single problem (i.e. variation between resamples) and over different problems. We resample each problem 30 times, and record the standard deviation over these resamples for each dataset. Table 20 summarises this within problem variation over all problems. It presents the mean. median and max of the standard deviation for the 30 resamples over all problems. We observe that HC2 has the lowest variation on average. Given it is a heterogeneous meta-ensemble, we would expect this. Hydra-MR, InceptionTime

and WEASEL-D have similar standard deviation. RDST, 1NN-DTW, FreshPRINCE and PF have the highest variation between resamples.

Table 20 Summary of the within problem variation of classifiers over 142 problems. We measure the standard deviation over 30 resamples for each problem, then summarise over problems.

Classifier	Mean	Median	Max
HC2	1.86%	1.38%	10.99%
Hydra-MR	1.92%	1.57%	7.65%
InceptionTime	1.91%	1.58%	7.32%
RDST	2.36%	1.64%	16.58%
WEASEL-D	1.93%	1.62%	8.40%
RSTSF	2.11%	1.89%	9.62%
FreshPRINCE	2.20%	1.85%	8.57%
ProximityForest	2.24%	1.82%	8.46%
1NN-DTW	2.43%	2.08%	8.68%

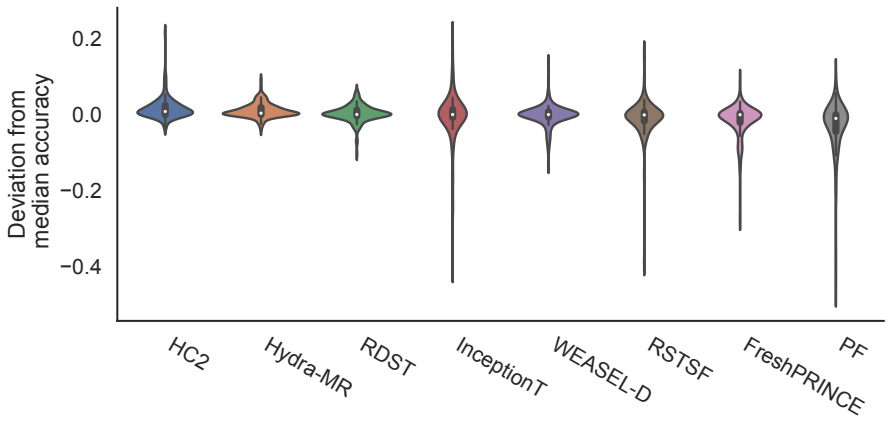


Fig. 35 Distributions of the deviation of test accuracies from the median value of eight classifiers

Figure 35 shows the violin plot of the deviation of each classifier from the median performing algorithm. It shows that both HC2 and Hydra-MR have tightly grouped distributions, with HC2 having a wider spread of positive values. InceptionTime has a very wide spread, both positive and negative, reflecting the wide variation in performance we have previously observed. RSTSF and PF also have wide distributions, but perform relatively poorly more often than they do well.

7 Conclusions

Research into algorithms for TSC has seen genuine progress in the last ten years, and the volume of research has dramatically increased. We have provided a particular view of this research landscape by grouping algorithms into eight categories defined by the core representation/transformation. We have compared the best in each category on 112 TSC problem and introduced 30 new datasets to counter any possible bias from over fitting. We evidence progress by benchmarking against algorithms previously considered the best performing, and show that two classifiers, Hydra-MR and HC2, generally perform the best. HC2 performs significantly better on the current UCR archive, but there is less observable difference when we compare them on 30 new problems we have introduced. This could be due to the smaller sample size, the nature of the data sets or reflect some embedded bias in algorithm design. We note that HC2 does worse than Hydra-MR on imbalanced data and with larger train set sizes, but is better with more class balance, smaller train set sizes and with long series.

We are not claiming that these results should be taken to mean practitioners should always use Hydra-MR and/or HC2. There are strengths and weakness to all the algorithms we have described. Understanding when it is appropriate to use which algorithm for a specific problem is an active research area. However, we suggest that, in the absence of any prior information these two algorithms make a sensible starting point for a new TSC problem. Despite significant research effort, there has not been an Alexnet for TSC, i.e. a deep learning approach that has dominated all others. It may be because the problems in the archives are relatively small compared to other archives used for deep learning evaluation. However, we think the core reason deep learning has not provided the gains many expected is that, unlike specific applications such as image classification or natural language processing, there is not one common underlying structure for the neural networks to exploit. Nevertheless, there is no doubt scope for improvements in deep learning algorithms for TSC. InceptionTime performs well overall, but Figure 35 demonstrates its limitations: it often performs terribly, and this makes its overall performance worse. If this tendency could be corrected, possibly by some automated structural optimisation, it seems likely that InceptionTime could match HC2 and Hydra-MR.

The ROCKET and HIVE-COTE family of classifiers work well because they combine convolution/shapelet approaches with dictionary based ones, i.e. they look for the presence of or the frequency of subseries. A key component of ROCKET based classifiers is dilation. We have shown that using dilation has significantly improved the single representation classifiers RDST and WEASEL-D. Incorporation of dilation could well benefit other algorithms, such as interval based classifiers.

We believe there is great scope for improving time series specific classifiers: none scale particularly well for large data, particularly in terms of memory: we are constructing a set of larger problems but none of the classifiers could be

built in them in reasonable time and/or memory; there is a lack of principled work flows for using these classifiers to help understand the mechanisms for forming classifiers; multivariate TSC is less understood and many of the classifiers described have not been designed to be used in this way; and there has been little research into how best to handle unequal length series. WE believe there are many unanswered questions in the field of TSC and predict it will remain as active and productive for the next 10 years.

Acknowledgements

This work is supported by the UK Engineering and Physical Sciences Research Council (EPSRC) grant number EP/W030756/1. The experiments were carried out on the High Performance Computing Cluster supported by the Research and Specialist Computing Support service at the University of East Anglia. We would like to thank all those responsible for helping maintain the time series classification archives and those contributing to open source implementations of the algorithms.

References

Abanda A, Mori U, Lozano J (2019) A review on distance based time series classification. Data Mining and Knowledge Discovery 33(2):378–412

Bagnall A, Lines J, Hills J, Bostrom A (2015) Time-series classification with COTE: The collective of transformation-based ensembles. IEEE Transactions on Knowledge and Data Engineering 27:2522–2535

Bagnall A, Lines J, Bostrom A, Large J, Keogh E (2017) The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. Data Mining and Knowledge Discovery 31(3):606–660

Bagnall A, Bostrom A, Cawley G, Flynn M, Large J, Lines J (2018) Is rotation forest the best classifier for problems with continuous features? ArXiv e-prints arXiv:1809.06705, URL `http://arxiv.org/abs/1809.06705`

Bagnall A, Flynn M, Large J, Lines J, Middlehurst M (2020a) On the usage and performance of HIVE-COTE v1.0. In: proceedings of the 5th Workshop on Advanced Analytics and Learning on Temporal Data, Lecture Notes in Artificial Intelligence, vol 12588

Bagnall A, Southam P, Large J, Harvey R (2020b) Detecting electric devices in 3d images of bags. arXiv preprint arXiv:200502163

Barbara NH, Bedding TR, Fulcher BD, Murphy SJ, Van Reeth T (2022) Classifying Kepler light curves for 12000 A and F stars using supervised feature-based machine learning. Monthly Notices of the Royal Astronomical Society 514(2):2793–2804

Batista G, Keogh E, Tataw O, deSouza V (2014) CID: an efficient complexity-invariant distance measure for time series. Data Mining and Knowledge Discovery 28(3):634–669

Baydogan M, Runger G (2016) Time series representation and similarity based on local autopatterns. Data Mining and Knowledge Discovery 30(2):476–509

Baydogan M, Runger G, Tuv E (2013) A bag-of-features framework to classify time series. IEEE Transactions on Pattern Analysis and Machine Intelligence 25(11):2796–2802

Benavoli A, Corani G, Mangili F (2016) Should we really use post-hoc tests based on mean-ranks? Journal of Machine Learning Research 17:1–10

Bostrom A, Bagnall A (2017) Binary shapelet transform for multiclass time series classification. Transactions on Large-Scale Data and Knowledge Centered Systems 32:24–46

Bostrom A, Bagnall A, Lines J (2016) Evaluating improvements to the shapelet transform. in Workshop on Mining and Learning from Time Series

Breiman L (2001) Random forests. Machine Learning 45(1):5–32

Cabello N, Naghizade E, Qi J, Kulik L (2020) Fast and accurate time series classification through supervised interval search. In: IEEE International Conference on Data Mining

Cabello N, Naghizade E, Qi J, Kulik L (2021) Fast, accurate and interpretable time series classification through randomization. arXiv preprint arXiv:210514876

Christ M, Braun N, Neuffer J, Kempa-Liehr AW (2018) Time series feature extraction on basis of scalable hypothesis tests (tsfresh–a python package). Neurocomputing 307:72–77

Dau H, Bagnall A, Kamgar K, Yeh M, Zhu Y, Gharghabi S, Ratanamahatana C, Chotirat A, Keogh E (2019) The UCR time series archive. IEEE/CAA Journal of Automatica Sinica 6(6):1293–1305

Dempster A, Petitjean F, Webb G (2020) ROCKET: Exceptionally fast and accurate time series classification using random convolutional kernels. Data Mining and Knowledge Discovery 34:1454–1495

Dempster A, Schmidt D, Webb G (2021) Minirocket: A very fast (almost) deterministic transform for time series classification. In: proceedings of the 27th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining

Dempster A, Schmidt DF, Webb GI (2022) HYDRA: Competing convolutional kernels for fast and accurate time series classification. arXiv preprint arXiv:220313652

Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research 7:1–30

Deng H, Runger G, Tuv E, Vladimir M (2013) A time series forest for classification and feature extraction. Information Sciences 239:142–153

Fawaz H, Forestier G, Weber J, Idoumghar L, Muller P (2019) Deep learning for time series classification: a review. Data Mining and Knowledge Discovery 33(4):917–963

Fawaz H, Lucas B, Forestier G, Pelletier C, Schmidt D, Weber J, Webb G, Idoumghar L, Muller P, Petitjean F (2020) InceptionTime: finding AlexNet for time series classification. Data Mining and Knowledge Discovery 34(6):1936–1962

Flynn M, Bagnall A (2019) Classifying flies based on reconstructed audio signals. In: proceedings of the Intelligent Data Engineering and Automated Learning, Lecture Notes in Computer Science, vol 11872, pp 249–258

Flynn M, Large J, Bagnall A (2019) The contract random interval spectral ensemble (c-RISE): The effect of contracting a classifier on accuracy. In: proceedings of the Hybrid Artificial Intelligence Systems, Lecture Notes in Computer Science, vol 11734, pp 381–392

Foumani N, Miller L, Tan C, Webb G, Forestier G, Salehi M (2023) Deep learning for time series classification and extrinsic regression: A current survey. arXiv preprint arXiv:230202515

Freund Y, Schapire R (1996) Experiments with a new boosting algorithm. In: Proc. International Conference on Machine Learning, vol 96, pp 148–156

Fukushima K (1980) Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biological cybernetics 36(4):193–202

Fulcher B, Jones N (2017) hctsa: A computational framework for automated time-series phenotyping using massive feature extraction. Cell Systems 5(5):527–531

García S, Herrera F (2008) An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. Journal of Machine Learning Research 9:2677–2694

Górecki T, Łuczak M (2013) Using derivatives in time series classification. Data Mining and Knowledge Discovery 26(2):310–331

Górecki T, Łuczak M (2014) Non-isometric transforms in time series classification using DTW. Knowledge-Based Systems 61:98–108

Grabocka J, Schilling N, Wistuba M, Schmidt-Thieme L (2014) Learning time-series shapelets. In: proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining

Guillaume A, Vrain C, Elloumi W (2022) Random dilated shapelet transform: A new approach for time series shapelets. In: Pattern Recognition and Artificial Intelligence: Third International Conference, ICPRAI 2022, Paris, France, June 1–3, 2022, Proceedings, Part I, Springer, pp 653–664

Hills J, Lines J, Baranauskas E, Mapp J, Bagnall A (2014) Classification of time series by shapelet transformation. Data Mining and Knowledge Discovery 28(4):851–881

Holder C, Middlehurst M, Bagnall A (2022) A review and evaluation of elastic distance functions for time series clustering. arXiv preprint arXiv:220515181

Ismail-Fawaz A, Devanne M, Weber J, Forestier G (2022) Deep learning for time series classification using new hand-crafted convolution filters. In: 2022 IEEE International Conference on Big Data (IEEE BigData 2022), pp 972–981

Jeong Y, Jeong M, Omitaomu O (2011) Weighted dynamic time warping for time series classification. Pattern Recognition 44:2231–2240

K KGF, H HAM, Cruz-Ramírez N, Hernández-Jiménez R (2017) Optimization of classification strategies of acetowhite temporal patterns towards improving diagnostic performance of colposcopy. Computational and Mathematical

Methods in Medicine (4)
Karlsson I, Papapetrou P, Boström H (2016) Generalized random shapelet forests. Data Mining and Knowledge Discovery 30(5):1053–1085
Kate R (2016) Using dynamic time warping distances as features for improved time series classification. Data Mining and Knowledge Discovery 30(2):283–312
Large J, Bagnall A, Malinowski S, Tavenard R (2019a) On time series classification with dictionary-based classifiers. Intelligent Data Analysis 23(5):1073–1089
Large J, Lines J, Bagnall A (2019b) A probabilistic classifier ensemble weighting scheme based on cross validated accuracy estimates. Data Mining and Knowledge Discovery 33(6):1674—1709
Lazebnik S, Schmid C, Ponce J (2006) Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In: proceeding of the IEEE conference on computer vision and pattern recognition, IEEE, vol 2, pp 2169–2178
Le Nguyen T, Ifrim G (2022) Fast time series classification with random symbolic subsequences. In: AALTD, Springer
Le Nguyen T, Gsponer S, Ifrim G (2017) Time series classification by sequence learning in all-subsequence space. In: 2017 IEEE 33rd international conference on data engineering (ICDE), IEEE, pp 947–958
Lin J, Keogh E, Wei L, Lonardi S (2007) Experiencing SAX: a novel symbolic representation of time series. Data Mining and Knowledge Discovery 15(2):107–144
Lin J, Khade R, Li Y (2012) Rotation-invariant similarity in time series using bag-of-patterns representation. Journal of Intelligent Information Systems 39(2):287–315
Lines J, Bagnall A (2014) Ensembles of elastic distance measures for time series classification. In: proceedings of the 14th SIAM International Conference on Data Mining
Lines J, Bagnall A (2015) Time series classification with ensembles of elastic distance measures. Data Mining and Knowledge Discovery 29:565–592
Lines J, Taylor S, Bagnall A (2018) Time series classification with HIVE-COTE: The hierarchical vote collective of transformation-based ensembles. ACM Transactions Knowledge Discovery from Data 12(5):1–36
Lubba C, Sethi S, Knaute P, Schultz S, Fulcher B, Jones N (2019) catch22: canonical time-series characteristics. Data Mining and Knowledge Discovery 33(6):1821–1852
Lucas B, Shifaz A, Pelletier C, O'Neill L, Zaidi N, Goethals B, Petitjean F, Webb G (2019) Proximity forest: an effective and scalable distance-based classifier for time series. Data Mining and Knowledge Discovery 33(3):607–635
Mahato V, Obeidi MA, Brabazon D, Cunningham P (2020) Detecting voids in 3d printing using melt pool time series data. Journal of Intelligent Manufacturing

Marteau P (2009) Time warp edit distance with stiffness adjustment for time series matching. IEEE Transactions on Pattern Analysis and Machine Intelligence 31(2):306–318

Middlehurst M, Bagnall A (2022) The freshprince: A simple transformation based pipeline time series classifier. In: International Conference on Pattern Recognition and Artificial Intelligence, Springer, pp 150–161

Middlehurst M, Vickers W, Bagnall A (2019) Scalable dictionary classifiers for time series classification. In: proceedings of the Intelligent Data Engineering and Automated Learning, Lecture Notes in Computer Science, vol 11871, pp 11–19

Middlehurst M, Large J, Bagnall A (2020a) The canonical interval forest (CIF) classifier for time series classification. In: IEEE International Conference on Big Data, pp 188–195

Middlehurst M, Large J, Cawley G, Bagnall A (2020b) The temporal dictionary ensemble (TDE) classifier for time series classification. In: proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, Lecture Notes in Computer Science, vol 12457, pp 660–676

Middlehurst M, Large J, Flynn M, Lines J, Bostrom A, Bagnall A (2021) HIVE-COTE 2.0: a new meta ensemble for time series classification. Machine Learning 110:3211–3243

Morrill J, Fermanian A, Kidger P, Lyons T (2020) A generalised signature method for multivariate time series feature extraction. arXiv preprint arXiv:200600873

Nguyen TL, Gsponer S, Ilie I, O'Reilly M, Ifrim G (2019) Interpretable time series classification using linear models and multi-resolution multi-domain symbolic representations. Data Mining and Knowledge Discovery 33(4):1183–1222

Oastler G, Lines J (2019) A significantly faster elastic-ensemble for time-series classification. In: proceedings of the Intelligent Data Engineering and Automated Learning, Lecture Notes in Computer Science, vol 11871, pp 446–453

Provost F, Domingos P (2003) Tree induction for probability-based ranking. Machine Learning 52(3):199–215

Rakthanmanon T, Keogh E (2013) Fast-shapelets: A fast algorithm for discovering robust time series shapelets. In: proceedings of the 13th SIAM International Conference on Data Mining

Rakthanmanon T, Bilson J, Campana L, Mueen A, Batista G, Westover B, Zhu Q, Zakaria J, Keogh E (2013) Addressing big data time series: Mining trillions of time series subsequences under dynamic time warping. ACM Transactions on Knowledge Discovery from Data 7(3)

Rodriguez J, Kuncheva L, Alonso C (2006) Rotation forest: A new classifier ensemble method. IEEE Transactions on Pattern Analysis and Machine Intelligence 28(10):1619–1630

Ruiz AP, Flynn M, Large J, Middlehurst M, Bagnall A (2021) The great multivariate time series classification bake off: a review and experimental evaluation of recent algorithmic advances. Data Mining and Knowledge Discovery

35(2):401—449
Schäfer P (2015) The BOSS is concerned with time series classification in the presence of noise. Data Mining and Knowledge Discovery 29(6):1505–1530
Schäfer P, Högqvist M (2012) 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, pp 516–527
Schäfer P, Leser U (2017) Fast and accurate time series classification with weasel. In: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, pp 637–646
Schäfer P, Leser U (2023) Weasel 2.0 - a random dilated dictionary transform for fast, accurate and memory constrained time series classification. arXiv preprint arXiv:230110194
Senin P, Malinchik S (2013) SAX-VSM: interpretable time series classification using sax and vector space model. In: proceedings of the 13th IEEE International Conference on Data Mining
Shifaz A, Pelletier C, Petitjean F, Webb GI (2020) TS-CHIEF: a scalable and accurate forest algorithm for time series classification. Data Mining and Knowledge Discovery 34(3):742–775
Souza V (2018) Asphalt pavement classification using smartphone accelerometer and complexity invariant distance. Engineering Applications of Artificial Intelligence 74:198–211
Stefan A, Athitsos V, Das G (2013) The Move-Split-Merge metric for time series. IEEE Transactions on Knowledge and Data Engineering 25(6):1425–1438
Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A (2015) Going deeper with convolutions. In: proceeding of the IEEE Conference on Computer Vision and Pattern Recognition
Tan CW, Petitjean F, Webb G (2020) FastEE: Fast ensembles of elastic distances for time series classification. Data Mining and Knowledge Discovery 34:1–42
Tan CW, Bergmeir C, Petitjean F, Webb G (2021) Time series extrinsic regression. Data Mining and Knowledge Discovery 35:1032—1060
Tan CW, Dempster A, Bergmeir C, Webb G (2022) MultiRocket: multiple pooling operators and transformations for fast and effective time series classification. Data Mining and Knowledge Discovery 36:1623–1646
Wang Z, Yan W, Oates T (2017) Time series classification from scratch with deep neural networks: A strong baseline. In: 2017 International joint conference on neural networks (IJCNN), IEEE, pp 1578–1585
Ye L, Keogh E (2011) Time series shapelets: a novel technique that allows accurate, interpretable and fast classification. Data Mining and Knowledge Discovery 22(1-2):149–182
Zhao J, Itti L (2019) shapeDTW: Shape dynamic time warping. Pattern Recognition 74:171–184

A Reproducibility

The majority of the algorithms we have evaluated in this work are available in the aeon toolkit (see Footnote 2), and those that aren't we plan to contribute over time. aeon is a Python based toolbox for time series analysis which contains a developed classification module. The aeon toolkit is compatible with scikit-learn[10], and aims to follow its interface and enable usage of its tools where possible.

The version of aeon classifiers used in this work is aeon v0.1.0, with this version and later releases available on PyPi[11]. In Listing 1 we show a usage example for a time series classifier in aeon. While we use ROCKET in the example as it is fast, the same interface applied to other aeon classifiers[12].

```
import numpy as np

from aeon.classification.convolution_based import RocketClassifier
from aeon.datasets import load_from_tsfile

if __name__ == "__main__":
    # 1a. Load data
    X_train, y_train = load_from_tsfile("data_TRAIN.ts")
    X_test, y_test = load_from_tsfile("data_TEST.ts")

    # 1b. Alternatively, format your data as a 3D numpy array and
    labels as a 1D array
    # shape == (n_instances, n_channels, series_length)
    X_train = 2 * np.random.uniform(size=(100, 1, 100))
    y_train = X_train[:, 0, 0].astype(int)
    X_test = 2 * np.random.uniform(size=(50, 1, 100))
    y_test = X_test[:, 0, 0].astype(int)

    # 2. Call fit() to build the classifier
    clf = RocketClassifier()
    clf.fit(X_train, y_train)

    # 3a. To predict the class label for new cases, use predict()
    predictions = clf.predict(X_test)

    # 3b. If probabilities are required, use predict_proba()
    probabilities = clf.predict_proba(X_test)

    # 3c. To just calculate accuracy, use score()
    accuracy = clf.score(X_test, y_test)
```

Listing 1 Loading data, building an estimator and making predictions using a time series classifier in aeon.

To run our experiments, we use the tsml-eval[13] package to produce results files for aeon and scikit-learn estimators. The version used for our experiments is tsml-eval v0.1.0, which contains additional estimators not currently available in aeon. Listing 2 gives an example for running an experiment using an aeon classifier loading from .ts files. Alternatively, you can input already loaded data as shown in Listing 3. For more guidance on producing our results using tsml-eval, visit our accompanying webpage hosted on the repository (see Footnote 8).

[10] https://scikit-learn.org/stable/

[11] https://pypi.org/project/aeon/

[12] https://www.aeon-toolkit.org/en/latest/api_reference.html

[13] https://github.com/time-series-machine-learning/tsml-eval

```python
from aeon.classification.convolution_based import RocketClassifier

from tsml_eval.experiments import
    load_and_run_classification_experiment
from tsml_eval.experiments.set_classifier import set_classifier

if __name__ == "__main__":
    # The directory where the data is stored. This directory
    should contain a directory with the name *dataset*, holding a
    *dataset*.TS file or *dataset*_TRAIN.TS and *dataset*_TEST.TS
    files
    data_dir = "../"
    # The directory to write the results file to
    results_dir = "../"
    # The name of the dataset to load
    dataset = "ItalyPowerDemand"
    # The resample id to use for random resampling, 0 uses the
    original train/test split if available
    resample_id = 0
    # The name of the classifier to use in the experiment
    classifier_name = "ROCKET"

    # The classifier to use
    classifier = RocketClassifier(random_state=resample_id)
    # Alternatively, use the set_classifier function to select a
    predefined classifier
    classifier = set_classifier(classifier_name)

    # Run the experiment!
    load_and_run_classification_experiment(
        data_dir,
        results_dir,
        dataset,
        classifier,
        resample_id=resample_id,
        classifier_name=classifier_name,
    )
```

Listing 2 Running a classification experiment using tsml-eval with data loaded from file.

```python
import numpy as np
from aeon.classification.convolution_based import RocketClassifier

from tsml_eval.experiments import run_classification_experiment

if __name__ == "__main__":
    # shape == (n_instances, n_channels, series_length)
    X_train = 2 * np.random.uniform(size=(100, 1, 100))
    y_train = X_train[:, 0, 0].astype(int)
    X_test = 2 * np.random.uniform(size=(50, 1, 100))
    y_test = X_test[:, 0, 0].astype(int)

    results_dir = "../"
    dataset = "ItalyPowerDemand"
    resample_id = 0
    classifier_name = "ROCKET"
    classifier = RocketClassifier(random_state=resample_id)

```

```
    run_classification_experiment(
        X_train,
        y_train,
        X_test,
        y_test,
        classifier,
        results_dir,
        classifier_name=classifier_name,
        dataset_name=dataset,
        resample_id=resample_id,
    )
```

Listing 3 Running a classification experiment using tsml-eval with pre=loaded data.

Table 21 Algorithms used in the 2017 bake off.

Standard classifiers	
Logistic	Logisitic regression
C45	Decision Tree
NB	Naive Bayes
BN	Bayesian Network
SVML	Linear kernel Support Vector Machine
SVMQ	Quadratic kernel Support Vector Machine
MLP	Multilayer Perceptron
RandF	Random Forest
RotF	Rotation Forest
Distance based	
ED	Euclidean Distance
DTW	Dynamic Time Warping
WDTW	Weighted DTW (Jeong et al., 2011)
TWE	Time Warp Edit (Marteau, 2009)
MSM	Move-Split-Merge (Stefan et al., 2013)
CID_DTW	Complexity Invariant Distance with DTW (Batista et al., 2014)
DD_DTW	Derivative DTW (Górecki and Łuczak, 2013)
DTD_C	Derivative Transform Distance (Górecki and Łuczak, 2014)
EE	Elastic Ensemble (Lines and Bagnall, 2015)
Interval Based	
TSF	Time Series Forest (Deng et al., 2013)
TSBF	Time Series Bag of Features (Baydogan et al., 2013)
LPS	Learned Pattern Similarity (Baydogan and Runger, 2016)
Shapelet Based	
FS	Fast Shapelets (Rakthanmanon and Keogh, 2013)
ST	Shapelet Transform (Hills et al., 2014)
LS	Learned Shapelets (Grabocka et al., 2014)
Dictionary Based	
BoP	Bag of Patterns (Lin et al., 2012)
SAXVSM	Symbolic Aggregate Approximation-vector Space Model (Senin and Malinchik, 2013)
BOSS	Bag of Symbolic Fourier Approximation Symbols (Schäfer, 2015)
Hybrid	
COTE/flat-COTE	Collective of Transformation-based Ensembles (Bagnall et al., 2015)
DTW_F	DTW Features (Kate, 2016)

B Algorithms

Table 21 shows the algorithms used in the original classification bake off Bagnall et al. (2017). Table 22 shows the algorithms used in our experiments.

Table 22 Algorithms used in the Redux bake off.

Distance based	
DTW	Dynamic Time Warping
ShapeDTW	Shape Based DTW (Zhao and Itti, 2019)
EE	Fast Elastic Ensemble (Oastler and Lines, 2019)
PF	Proximity Forest (Lucas et al., 2019)
Feature Based	
Catch22	Canonical Time Series Characteristics (Lubba et al., 2019)
TSFresh	Time Series Feature Extraction Based on Scalable Hypothesis Tests (Christ et al., 2018)
FreshPRINCE	Fresh Pipeline with Rotation Forest Classifier (Middlehurst and Bagnall, 2022)
Shapelet Based	
RSF	Random Shapelet Forest (Karlsson et al., 2016)
STC	Binary Shapelet Transform Classifier (Bostrom and Bagnall, 2017)
MrSQM	Multiple Representations Sequence Miner (Le Nguyen and Ifrim, 2022)
RDST	Random Dilated Shapelet Transform (Guillaume et al., 2022)
Interval Based	
TSF	Time Series Forest (Deng et al., 2013)
CIF	Canonical Interval Forest (Middlehurst et al., 2020a)
DrCIF	Diverse Representation CIF (Middlehurst et al., 2021)
STSF	Supervised TSF (Cabello et al., 2020)
R-STSF	Randomised STSF (Cabello et al., 2021)
Dictionary Based	
BOSS	Bag of Symbolic Fourier Approximation Symbols (Schäfer, 2015)
cBOSS	Contractable BOSS (Middlehurst et al., 2019)
WEASEL	Word Extraction for Time Series Classification (Schäfer and Leser, 2017)
TDE	Temporal Dictionary Ensemble (Middlehurst et al., 2020b)
WEASEL-D	WEASEL with Dilation (Schäfer and Leser, 2023)
Kernel/convolution Based	
ROCKET	Random Convolutional Kernel Transform (Dempster et al., 2020)
Arsenal	The Arsenal (Middlehurst et al., 2021)
MultiROCKET	MultiROCKET (Tan et al., 2022)
MiniROCKET	MiniROCKET (Dempster et al., 2021)
Hydra	Hybrid Dictionary–ROCKET Architecture (Dempster et al., 2022)
Hydra-MultiROCKET	Hydra + MultiROCKET (Dempster et al., 2022)
Deep Learning Based	
CNN	Convolution Neural Network (Fukushima, 1980)
ResNet	Residual Network (Wang et al., 2017)
InceptionTime	Inception Time (Fawaz et al., 2020)
Hybrid	
TS-CHIEF	Time Series Combination of Heterogeneous and Integrated Embedding Forest (Shifaz et al., 2020)
HC1	Hierarchical Vote Collective of Transformation-based Ensembles (Bagnall et al., 2020a)
HC2	HIVE-COTE version 2 (Middlehurst et al., 2021)

C Results

Table C shows the accuracy results of the best performing algorithms on the 30 new classification datasets.

Table 23 Accuracy of classifiers on the 30 new datasets (averaged over 30 resamples)

	Hydra-MR	HC2	RDST	RSTSF	FreshPRINCE	InceptionT	WEASEL-D	PF	1NN-DTW
AconityMINIPrinterLargeEq	95.70%	95.47%	95.82%	94.33%	94.61%	96.55%	95.34%	91.04%	85.85%
AconityMINIPrinterSmallEq	97.75%	97.80%	97.56%	97.44%	97.19%	97.28%	97.63%	96.40%	95.00%
AllGestureWiimoteXEq	76.53%	73.64%	71.03%	68.74%	68.82%	81.83%	70.08%	76.68%	68.67%
AllGestureWiimoteYEq	80.63%	77.93%	73.28%	70.70%	71.43%	84.63%	73.16%	75.34%	67.65%
AllGestureWiimoteZEq	73.07%	72.70%	67.76%	68.12%	67.49%	78.11%	69.17%	74.19%	68.22%
AsphaltObstaclesUniEq	88.52%	88.83%	88.00%	85.14%	85.72%	91.18%	88.37%	88.87%	80.49%
AsphaltPavementTypeUniEq	91.76%	89.57%	89.24%	93.01%	92.93%	93.25%	79.96%	89.26%	59.97%
AsphaltRegularityUniEq	98.69%	97.90%	97.69%	98.96%	98.66%	98.70%	93.27%	98.30%	69.24%
Colposcopy	37.59%	39.64%	39.60%	43.37%	41.19%	37.69%	38.28%	32.90%	28.65%
Covid3MonthDiscrete	78.29%	81.48%	79.02%	79.14%	78.90%	55.69%	79.29%	78.67%	74.86%
DodgerLoopDayNmv	54.81%	59.52%	61.08%	62.68%	57.45%	52.25%	60.39%	58.83%	41.95%
DodgerLoopGameNmv	86.54%	84.33%	80.84%	85.51%	84.88%	80.39%	84.25%	88.06%	86.77%
DodgerLoopWeekendNmv	98.10%	98.36%	98.36%	98.54%	97.88%	98.04%	97.94%	98.47%	95.53%
ElectricDeviceDetection	90.18%	88.86%	90.02%	89.85%	89.68%	73.16%	89.31%	88.99%	85.84%
FloodModeling1Discrete	96.16%	92.80%	93.04%	96.72%	96.94%	95.99%	89.41%	96.34%	94.15%
FloodModeling2Discrete	98.34%	95.94%	95.85%	97.85%	98.30%	98.10%	95.91%	97.39%	97.45%
FloodModeling3Discrete	96.55%	92.70%	92.63%	96.43%	97.44%	95.74%	90.37%	96.44%	94.84%
GestureMidAirD1Eq	77.36%	77.77%	74.26%	69.95%	70.67%	75.05%	73.33%	64.46%	45.62%
GestureMidAirD2Eq	63.67%	65.31%	62.21%	59.72%	61.36%	58.21%	63.33%	53.95%	31.79%
GestureMidAirD3Eq	48.46%	54.08%	51.92%	49.95%	47.46%	38.51%	44.72%	34.54%	18.41%
GesturePebbleZ1Eq	95.72%	95.70%	97.13%	92.93%	94.40%	97.11%	96.10%	89.96%	71.10%
GesturePebbleZ2Eq	96.79%	96.75%	97.49%	93.92%	94.41%	96.81%	96.92%	90.78%	73.35%
KeplerLightCurves	92.46%	96.54%	92.84%	95.12%	96.57%	74.85%	92.39%	89.47%	80.14%
MelbournePedestrianNmv	96.31%	93.80%	95.95%	96.97%	96.44%	96.77%	92.09%	95.28%	88.55%
PhoneHeartbeatSound	65.07%	64.52%	66.87%	65.86%	65.26%	65.22%	64.08%	63.81%	54.27%
PickupGestureWiimoteZEq	84.67%	76.73%	82.20%	77.07%	79.80%	76.47%	82.07%	80.93%	70.00%
PLAIDEq	93.92%	88.76%	91.78%	88.27%	88.87%	49.50%	89.85%	87.99%	84.97%
ShakeGestureWiimoteZEq	92.87%	94.33%	93.73%	85.40%	90.47%	85.40%	92.73%	89.73%	85.47%
SharePriceIncrease	66.48%	68.70%	66.02%	69.30%	69.44%	65.21%	68.62%	69.13%	62.05%
Tools	86.52%	88.58%	86.27%	84.80%	86.99%	84.50%	88.26%	76.74%	69.33%