HIVE-COTE 2.0: a new meta ensemble for time series classification

Matthew Middlehurst[1] · James Large[1] · Michael Flynn[1] · Jason Lines[1] · Aaron Bostrom[1] · Anthony Bagnall[1]

Abstract

The Hierarchical Vote Collective of Transformation-based Ensembles (HIVE-COTE) is a heterogeneous meta ensemble for time series classification. HIVE-COTE forms its ensemble from classifiers of multiple domains, including phase-independent shapelets, bag-of-words based dictionaries and phase-dependent intervals. Since it was first proposed in 2016, the algorithm has remained state of the art for accuracy on the UCR time series classification archive. Over time it has been incrementally updated, culminating in its current state, HIVE-COTE 1.0. During this time a number of algorithms have been proposed which match the accuracy of HIVE-COTE. We propose comprehensive changes to the HIVE-COTE algorithm which significantly improve its accuracy and usability, presenting this upgrade as HIVE-COTE 2.0. We introduce two novel classifiers, the Temporal Dictionary Ensemble and Diverse Representation Canonical Interval Forest, which replace existing ensemble members. Additionally, we introduce the Arsenal, an ensemble of ROCKET classifiers as a new HIVE-COTE 2.0 constituent. We demonstrate that HIVE-COTE 2.0 is significantly more accurate on average than the current state of the art on 112 univariate UCR archive datasets and 26 multivariate UEA archive datasets.

Abbreviations

TSC	Time Series Classification
MTSC	Multivariate Time Series Classification
UCR	University of California, Riverside
UEA	University of East Anglia
HIVE-COTE (HC)	Hierarchical Vote Collective of Transformation-based Ensembles
TS-CHIEF	Time Series Combination of Heterogeneous and Integrated Embedding Forest

Editor: Gustavo Batista.

✉ Anthony Bagnall
ajb@uea.ac.uk

Extended author information available on the last page of the article

ROCKET	Random Convolutional Kernel Transform
STC	Shapelet Transform Classifier
EE	Elastic Ensemble
TDE	Temporal Dictionary Ensemble
SFA	Symbolic Fourier Approximation
BOSS	Bag of SFA Symbols
cBOSS	Contractable BOSS
S-BOSS	Spatial BOSS
WEASEL	Word Extraction for Time Series Classification
CIF	Canonical Interval Forest
DrCIF	Diverse Representation CIF
TSF	Time Series Forest
STSF	Supervised TSF
RISE	Random Interval Spectral Ensemble
CAWPE	Cross-validation Accuracy Weighted Probabilistic Ensemble
DTW	Dynamic Time Warping
MCB	Multiple Coefficient Binning
IGB	Information Gain Binning
LOOCV	Leave One Out Cross-validation
catch22	Canonical Time Series Characteristics
AUROC	Area Under the Receiver Operating Characteristic
NLL	Negative Log Likelihood
MSE	Mean Squared Error
HPC	High Performance Computing

1 Introduction

Time series classification (TSC) is the problem of predicting a discrete target variable from a (possibly multivariate) time series. TSC problems are seen in all areas of machine learning applications, including seizure detection (Chaovalitwongse et al. 2006), earthquake monitoring (Arul and Kareem 2021), insect classification (Potamitis 2014) and predictive maintenance (Guillaume et al. 2020). The publication of the University of California, Riverside (UCR) TSC archive resulted in an increased interest into algorithmic research for this type of problem. An experimental study, characterised as a bake off (Bagnall et al. 2017), facilitated the objective and reproducible comparison of learning algorithm performance on the UCR archive. Since then, new classifiers have been proposed in the literature that have advanced the field by significantly outperforming those used in the bake off. There are currently four algorithms with reasonable claim to being state of the art for TSC based on experimentation on the recently expanded UCR archive (Dau et al. 2019). These are: the deep learning approach called InceptionTime (Fawaz et al. 2020); the tree based Time Series Combination of Heterogeneous and Integrated Embedding Forest (TS-CHIEF)

(Shifaz et al. 2020); the Random Convolutional Kernel Transform (ROCKET) (Dempster et al. 2020); and the heterogeneous meta-ensemble Hierarchical Vote Collective of Transformation-based Ensembles (HIVE-COTE) (Lines et al. 2018), the latest version of which is called HIVE-COTE version 1.0 (HC1) (Bagnall et al. 2020). We propose a new version of HIVE-COTE that is significantly more accurate than all four current state-of-the-art algorithms. We call this classifier HIVE-COTE version 2.0, or HC2 for short. The critical difference diagram in Fig. 1 summarises the final results of HC2 against the four leading algorithms on 112 equal length UCR archives, using 30 stratified resamples on each dataset (more detail is provided in Sect. 5). The number associated with each algorithm is the average rank of the classifier on 112 UCR datasets and solid bars group classifiers between which there is no significant difference. HC2 is on average over 1% more accurate per problem than all of the current state of the art.

The key principle behind HIVE-COTE is that TSC problems are best approached by careful consideration of the data representation, and that with no expert knowledge to the contrary, the most accurate algorithm design is to ensemble classifiers built on different representations. The changes from HC1 to HC2 relate to the component classifiers and a redefinition of the underlying data representations used. HC2 contains four component classifiers: the dictionary based Temporal Dictionary Ensemble (TDE) (Middlehurst et al. 2020b); the interval based Diverse Representation Canonical Interval Forest (DrCIF) (Middlehurst et al. 2020a); an adaptation of ROCKET we call the Arsenal and the latest version of the Shapelet Transform Classifier (STC) (Bostrom and Bagnall 2017). Each of these classifiers represents the best in class for a particular representation. Prototype versions of TDE and DrCIF have been presented at conferences and are novel contributions in their own right. Arsenal enhances ROCKET to produce better probability estimates. STC has enhanced usability options that help improve the functionality of the whole ensemble; HC2 is now contractable (i.e. the classifier can be given a maximum run time), checkpointable (i.e. the classifier build can be resumed from a previous run) and works with multivariate time series classification (MTSC). A recent study (Ruiz et al. 2021) concluded that that MTSC is at an earlier stage of development than univariate TSC. The only algorithms significantly better than the standard TSC benchmark, one nearest neighbour with dynamic time warping (DTW), were HC1, ROCKET, InceptionTime and CIF (Middlehurst et al.
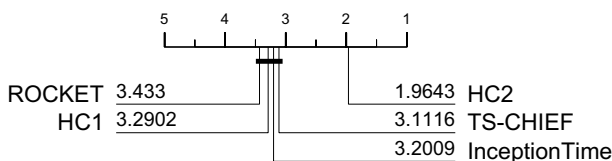

Fig. 1 Critical difference diagram for HC2 against the current state of the art on 112 UCR TSC problems. The average rank for each classifier is shown, and solid lines group classifiers between which there is no significant difference. It demonstrates that there is no difference between HC1 (Bagnall et al. 2020), InceptionTime (Fawaz et al. 2020), ROCKET (Dempster et al. 2020) and TS-CHIEF (Shifaz et al. 2020), but HC2 is significantly higher ranked than all of them. More details are given in Sect. 5

2020a). HC2 is significantly more accurate than all these algorithms on the University of East Anglia (UEA) MTSC archive (Bagnall et al. 2018).

We investigate the effect of how HC estimates test error and the relative importance of each component of HC2 through an ablative study in Sect. 6. We show that there is high variability in performance between the components over datasets and that each component significantly improves the ensemble overall. We also assess alternative ensemble structures, including stacking and selection schemes, and conclude that the simple weighted structure using a tilted distribution, the Cross-validation Accuracy Weighted Probabilistic Ensemble (CAWPE) (Large et al. 2019b), is as good as much more complex approaches.

A common criticism of HC is its run time. This in part is due to out of date received wisdom relating to shapelet search. The initial shapelet search algorithms conducted a computationally expensive exhaustive search. This is not only unnecessary but results in over fitting. Recent versions of STC simply randomise the search. Nevertheless, a full run of HC2 is still computationally expensive on large problems and takes much longer than ROCKET. In Sect. 7 we explore the usability of HC2, including details of open source implementations and assess the performance and its components on two problems with very long series. Our results indicate that in general HC2 performance converges quickly, and that a contracted run time is sufficient to produce reasonable results in a controlled time. Finally, in Sect. 8 we conclude this study by identifying areas of future improvement for HIVE-COTE.

2 Background

Time series classification (TSC) requires a training set of case pairs $\{x, y\}$ with m real-valued ordered observations and a discrete class label y from a range of c possible values. The objective is to create a function that maps from the space of possible input series to the space of possible class labels. This is achieved by using a training set of case pairs to build a model that can output either a predicted class value, or a predicted class distribution, for previously unseen series.

We restrict our attention to problems where series are the same length. In univariate TSC, x is a vector of m observations. The majority of research effort over the last decade has been into developing univariate TSC algorithms. Multivariate time series classification (MTSC) is an extension where the series are multidimensional and a single case is represented by a list of vectors over d dimensions and m observations, $X = <x_1, x_2, ..., x_d>$, and $x_k = <x_{1,k}, x_{2,k}, ..., x_{m,k}>$. When indexing into a dataset, we denote the jth observation of the ith case in dimension k as $x_{i,j,k}$.

One way of categorising algorithms is on the core data representation used. Distance based algorithms rely on elastic distance measures between two series. Dictionary based approaches are based on the frequency of recurring patterns, found through converting real valued time series into a sequence of discrete symbol words. Interval based algorithms derive features on intervals of series to find temporal features that may be otherwise obscured by irrelevant observations. Shapelet based approaches find phase independent discriminatory subseries.

The current approaches to time series classification that exploit one or more of these representations can be grouped into four categories: modular heterogeneous ensembles

where each module consists of a classifier built on a particular transformation type such as HIVE-COTE; tree based homogeneous ensembles where different data representations are embedded within the nodes of the tree Shifaz et al. (2020); deep learning algorithms where the representations are embedded in the network (Fawaz et al. 2019); and transformation/convolution approaches that create massive new feature spaces that are parsed with a linear classifier (Dempster et al. 2020; Nguyen et al. 2019). The most effective algorithms exploit one or more representations. In the remainder of this background, we limit ourselves to describing the most accurate approaches. More complete reviews of TSC algorithms can be found in Bagnall et al. (2017, 2020).

HIVE-COTE 1.0 The original version of HIVE-COTE was first introduced in 2016 (Lines et al. 2016, 2018) and, at the time, was significantly more accurate on average than other known approaches (Bagnall et al. 2017) on the 85 datasets that were then the complete UCR archive (Dau et al. 2019). The first version of HIVE-COTE (later dubbed HIVE-COTE alpha), contained five constituent ensembles that each worked on features from different data transformation domains: the Elastic Ensemble (EE) (Lines and Bagnall 2015); Shapelet Transform Classifier (STC) (Hills et al. 2014); Time Series Forest (TSF) (Deng et al. 2013); Bag of Symbolic-Fourier-Approximation Symbols (BOSS) (Schäfer 2015); and the Random Interval Spectral Ensemble (RISE) that was introduced alongside HIVE-COTE (Lines et al. 2018). Each module was encapsulated and built on the train data independently of the others. For new data, each module passes an estimate of class probabilities to the control unit, which combines them to form a single prediction. It does this by weighting the probabilities of each module by an estimate of its testing accuracy formed from the training data.

The goal of HIVE-COTE alpha was to achieve the highest level of accuracy without concern for computational resources. This initial target has since lead to a perception that HIVE-COTE is very slow and does not scale well. A very simple restructure of HIVE-COTE alpha was able to achieve the same level of accuracy in orders of magnitude less time; HIVE-COTE 1.0 (HC1) (Bagnall et al. 2020), was introduced to demonstrate its utility and scalability. (HC1) is based on simple refinements and enchantments to the original HIVE-COTE alpha base constituents. HC1 dropped the distance based EE due to the high computational overhead. STC introduced binary shapelets and a randomised search controlled by a time parameter. HC1 uses the Cross-validation Accuracy Weighted Probabilistic Ensemble (CAWPE) (Large et al. 2019b) ensemble structure. CAWPE uses an accuracy estimate of each classifier formed on the train data to weight the probabilities of each component. It constructs a tilted distribution through exponentiation using a parameter α to extenuate differences in classifiers. Each component's weight is found through an internal estimate for each classifier if capable, else a ten-fold cross-validation of the training data is performed.

TS-CHIEF The Time Series Combination of Heterogeneous and Integrated Embedding Forest (TS-CHIEF) (Shifaz et al. 2020) is the classifier most comparable to HIVE-COTE. TS-CHIEF is made up of an ensemble of trees which embed distance, dictionary and spectral base features. A number of splitting criteria from each representation with randomly initialised parameters are considered at each node. The different types of split criteria are dictionary based splits based on BOSS, similarity based splits based on EE and interval based splits based on RISE. The core distinction is that the usage of base features is embedded in nodes of the tree rather than modularised through separate classifiers.

InceptionTime (Fawaz et al. 2020) is a deep learning ensemble, combining five homogeneous residual networks incorporating inception modules (Szegedy et al. 2015). An individual network is made up of two blocks of three Inception modules which maintain residual connections, followed by global average pooling and softmax layers. Each network in the ensemble is initialised with random weights for stability. It is the best deep learning approach for time series data to our knowledge, and represents deep learning for TSC in our experiments.

ROCKET the Random Convolutional Kernel Transform (ROCKET) (Dempster et al. 2020) produces a large number of summary stats using randomly initialised convolutional kernels, then selects informative ones using a linear classifier. A version of ROCKET is included in the HIVE-COTE 2.0 ensemble, so a more complete description of the algorithm is provided in Sect. 3.3.

Other recent approaches focus on a single representation. Proximity Forest (Lucas et al. 2019) is a tree ensemble that randomly chooses distance functions at each node. Supervised Time Series Forest (STSF) (Cabello et al. 2020) is an interval based tree ensemble that includes a supervised method for extracting intervals and uses summary statistics and spectral features. A number of extensions to the BOSS classifier have been made since the bake off in S-BOSS (Large et al. 2019a), cBOSS (Middlehurst et al. 2019) and WEASEL (Schäfer and Leser 2017a).

There have also been a range of algorithms proposed for MTSC (Ruiz et al. 2021). Dynamic Time Warping with pointwise multivariate distance and a one nearest neighbour classifier, characterised as dependent dynamic time warping (DTW-D) (Shokoohi-Yekta et al. 2017), is the baseline for MTSC. ROCKET, InceptionTime and CIF have multivariate versions which are significantly more accurate.

3 HIVE-COTE 2.0 (HC2)

HIVE-COTE 2.0 replaces three of the four classifiers that make up HIVE-COTE 1.0. The component modules are: the shapelet based Shapelet Transform Classifier (Bostrom and Bagnall 2017); the convolution based ensemble of ROCKET classifiers we call the Arsenal; the dictionary based representation TDE; and the interval based DrCIF. An overview of the updated HC2 structure is displayed in Fig. 2.

Each component is trained independently and in addition to the final model, it is required to produce an estimate of its accuracy on unseen data. For new data, each module produces a probability estimate for each class. The controller constructs a tilted distribution through exponentiation (using $\alpha = 4$ by default) to extenuate differences in classifiers and weighting with the accuracy estimate. Each module of HC2 contains new features and improvements over previous versions. These include novel algorithm improvements, multivariate extensions and contracting improvements. In addition, the method for estimating the accuracy from the train data has been improved. Generally, there are three ways of estimating test accuracy from train data. Firstly, the final model can be assessed directly on the train data. This is likely to be biased and over optimistic, particularly if some form of model selection has occurred without careful regularisation. Secondly, a cross-validation can be performed on the train data in addition to the final build. Whilst this will probably
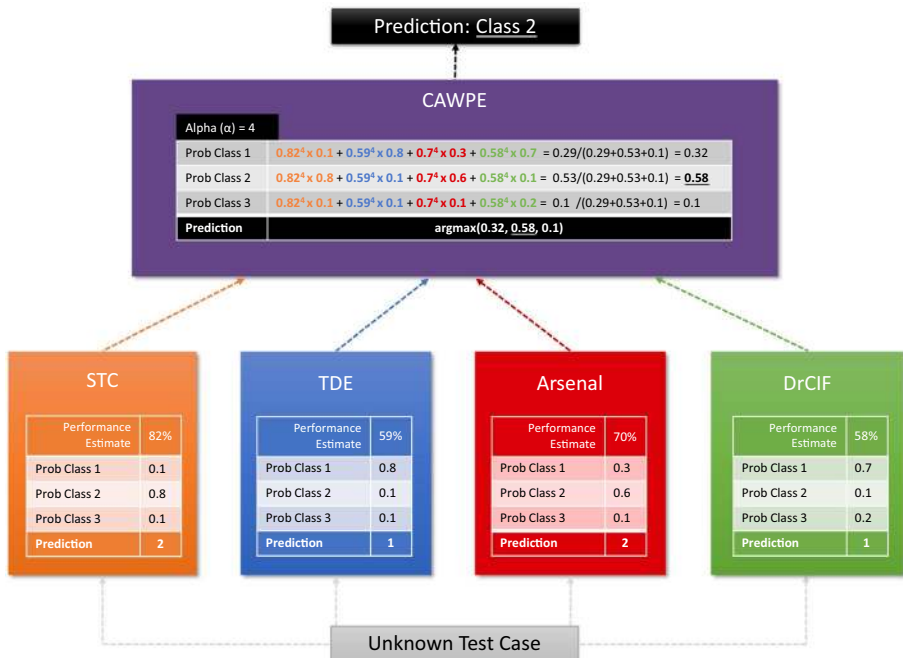
Prediction: Class 2

CAWPE

Alpha (α) = 4

Prob Class 1	0.82^4 x 0.1 + 0.59^4 x 0.8 + 0.7^4 x 0.3 + 0.58^4 x 0.7 = 0.29/(0.29+0.53+0.1) = 0.32
Prob Class 2	0.82^4 x 0.8 + 0.59^4 x 0.1 + 0.7^4 x 0.6 + 0.58^4 x 0.1 = 0.53/(0.29+0.53+0.1) = **0.58**
Prob Class 3	0.82^4 x 0.1 + 0.59^4 x 0.1 + 0.7^4 x 0.1 + 0.58^4 x 0.2 = 0.1 /(0.29+0.53+0.1) = 0.1
Prediction	argmax(0.32, **0.58**, 0.1)

STC

Performance Estimate	82%
Prob Class 1	0.1
Prob Class 2	0.8
Prob Class 3	0.1
Prediction	**2**

TDE

Performance Estimate	59%
Prob Class 1	0.8
Prob Class 2	0.1
Prob Class 3	0.1
Prediction	**1**

Arsenal

Performance Estimate	70%
Prob Class 1	0.3
Prob Class 2	0.6
Prob Class 3	0.1
Prediction	**2**

DrCIF

Performance Estimate	58%
Prob Class 1	0.7
Prob Class 2	0.1
Prob Class 3	0.2
Prediction	**1**

Unknown Test Case

Fig. 2 An overview of the ensemble structure of HIVE-COTE 2.0 for a three class problem. Each module is trained independently and produces an estimate of the probability of membership of each class for unseen data. The control unit (CAWPE) combines these probabilities, weighted by an estimate of the quality of the module found on the train data

be less biased (and generally pessimistic) it is time consuming. Thirdly, some form of hold-out evaluation can be embedded in the full model build, such as bagging. HC1 uses a mixture of approaches based on the classifier. HC2 adopts a standardised bagging approach, which is possible since all four classifiers in HC2 use ensembles. However, we found that whilst using out-of-bag performance produces acceptable estimates of test accuracy, the bagged classifiers themselves were significantly less accurate than those built on the full data. Hence, we adopted a hybrid approach. Rather than build 11 total models for ten-fold cross-validation or a single model using bagging, we construct one bagged model to estimate accuracy for those that require it using out-of-bag error and a full model to predict new cases. Specifics on how each module generates its estimate and the impact of this design choice are discussed in Sect. 6.

HC2 can train each component concurrently. Even so, the components can be slow on large problems. Hence, we allow the user to configure HC2 so that it has a time contract. If contracted, each component ensemble simply builds as many base classifiers as it can in the time provided. This simple form of contracting is an adequate first fix. However, a problem does arise for very large data with short contracts: building a single ensemble

member may exceed the contract. It would be better for the components to self configure when this is likely, by, for example, subsampling cases or series. HC2 is threaded but currently, the components themselves are not. Given they are all ensembles of independent base classifiers, it is in principle easy to do so. This and the inevitable move onto GPU is part of our future work plan.

3.1 Temporal dictionary ensemble (TDE)

HIVE-COTE alpha contains the dictionary based classifier BOSS (Schäfer 2015), which was updated to cBOSS (Middlehurst et al. 2019) in HC1. HC2 uses the Temporal Dictionary Ensemble (TDE) (first introduced in Middlehurst et al. (2020b)), which draws on more recent work on dictionary classifiers (Large et al. 2019a; Schäfer and Leser 2017a) and includes several novel features. Dictionary based approaches aim to capture the repetitions of patterns as discriminatory features rather than solely their presence. These approaches commonly adapt the bag-of-words model used in other domains such as signal processing, computer vision and audio processing for time series data.

TDE is an ensemble of 1-NN classifiers that transforms each series into a histogram of word counts. A sliding window of length w is run along each series, and the subseries is discretised into a word of length l from an alphabet of size α. TDE transforms the window using the Symbolic Fourier Approximation (SFA) (Schäfer and Högqvist 2012) transform proposed for BOSS (Schäfer 2015). Distance between histograms is found using histogram intersection. In addition to word frequencies, TDE also captures the frequencies of bigrams found from non overlapping windows. Thus a transformed case includes a histogram of word counts and bigram counts for a given trio of parameters (w,l,α). TDE also includes some spatial information by the utilisation of spatial pyramids (Lazebnik et al. 2006). This involves splitting a series into h levels each with 2^v disjoint subseries, where v is the current pyramid level. Word counts are found for each subseries independently, then the resulting histograms are concatenated. The distance to histograms of deeper levels with smaller spatial areas in the series are weighted higher than global similarity. Bigrams are only recorded for the first level consisting of the full series. The SFA transform requires a set of breakpoints when creating words. The method of generating these breakpoints b is selected between Multiple Coefficient Binning (MCB) (Schäfer 2015) and Information Gain Binning (IGB) (Schäfer and Leser 2017a). Windows can optionally be normalised during the transform with the p parameter.

The TDE ensemble is filtered into s total classifiers from k candidates. The accuracy of each candidate is estimated using leave-one-out cross-validation (LOOCV), with the highest s being retained. Diversity is achieved through altering the parameters (w,l,h,b,p) for each new classifier and a 70% sampling of the train data. The first 50 classifiers use randomly selected parameters, while those after are selected using a Gaussian processes regressor. For unseen parameter sets, a prediction of accuracy is made using the parameters of previously built classifiers, with the highest predicted accuracy being chosen for the next classifier build. New cases are classified with a weighted majority vote, using the exponential accuracy weights from CAWPE (Large et al. 2019b). Table 1 shows the range from which individual classifier parameters are subsampled, with m being the series length. The ensemble build process for TDE is described in Algorithm 1. When it comes to replacing BOSS in the HIVE-COTE ensemble, TDE was the only dictionary based approach to significantly improve the ensemble's accuracy (Middlehurst et al. 2020b). Using cBOSS and
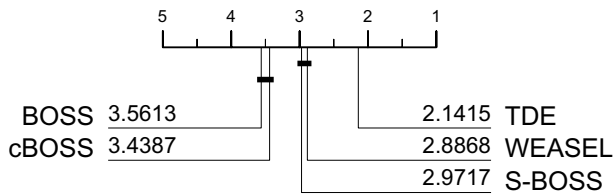
Fig. 3 Results of five dictionary based classifiers on 106 of the UCR datasets. The missing datasets are: ElectricDevices; FordA; FordB; HandOutlines; Non-InvasiveFetalECGThorax1; and NonInvasiveFetalECGThorax2. These are missing due to the long run time of S-BOSS and WEASEL. cBOSS samples 250 parameter sets and has an ensemble size of 50. WEASEL χ is set to 2

Table 1 Parameter ranges for TDE base classifier selection

Parameter	Range
Word lengths	$l = \{16, 14, 12, 10, 8\}$
Window lengths	$w = \{10 \dots m\}$
Normalise	$p = \{true, false\}$
Alphabet size	$\alpha = \{4\}$
No.pyramid levels	$h = \{1, 2, 3\}$
Breakpoints	$b = \{MCB, IGB\}$

S-BOSS as replacements was found to show no significant difference in accuracy, while WEASEL was significantly worse.

We introduce the capability for multivariate time series classification to TDE by making a number of additions to the individual classifier. WEASEL also has a multivariate version, WEASEL-MUSE (Schäfer and Leser 2017b), which shares similarities with our extension. Current dictionary approaches are noticeably memory intensive due to the requirement to store multiple transformed versions of the original data. WEASEL-MUSE can become unusable due to this issue, with over 500GB of memory required on some datasets (Ruiz et al. 2021). We aim to mitigate this issue with the TDE multivariate extension.

For each dimension we extract words using the same process for univariate series, with each dimension having their own set of breakpoints. Words from different dimensions are stored separately in each cases bag. For many multivariate time series, some dimensions hold little or redundant information. Additionally, for problems with many dimensions storing the words extracted from each can cause significant memory issues. As such, prior to creating any bags we take a subsample of dimensions based on an accuracy estimate. We find this estimate using LOOCV on bags created from disjoint windows rather an a sliding one. Any dimension with an accuracy estimate less than 85% of the highest accuracy is not retained for this classifier. Additionally, we set a limit of 20 maximum dimensions retained for each classifier, keeping those with the highest accuracy. To reduce the memory impact of saving features from multiple dimensions, we do not record bigrams for multivariate datasets. The build process for the individual classifiers used in TDE is displayed in Algorithm 2.

Figure 3 provides evidence to support our claim that TDE is the most accurate purely dictionary based algorithm. It shows the results of a comparison of dictionary based classifiers on test accuracy performance on the UCR datasets. TDE is significantly more accurate than WEASEL and S-BOSS, which in turn are more accurate than BOSS and cBOSS.

Algorithm 1 Temporal Dictionary Ensemble(A list of n cases of length m with d dimensions, $\mathbf{T} = (\mathbf{X}, \mathbf{y})$)

Parameters: the number of parameter samples k, the max ensemble size s (default $k = 250$, $s = 50$)
 1: Let w be window length, l be word length, p be normalise/not normalise, α be alphabet size, h be number of pyramid levels and b be MCB or IGB discretisation.
 2: Let $\mathbf{C}$ be a list of s classifiers $(\mathbf{c}_1, \ldots, \mathbf{c}_s)$
 3: Let $\mathbf{E}$ be a list of s classifier weights $(\mathbf{e}_1, \ldots, \mathbf{e}_s)$
 4: Let $\mathbf{G}$ be a list of k parameter and accuracy pairs $(\mathbf{g}_1, \ldots, \mathbf{g}_k)$
 5: Let $\mathbf{R}$ be a set of possible parameter combinations
 6: $i \leftarrow 0$
 7: $lowest\_acc \leftarrow \infty, lowest\_acc\_idx \leftarrow \infty$
 8: **while** $i < k$ AND $|\mathbf{R}| > 0$ **do**
 9:     **if** $i < 50$ **then**
10:         $[l, \alpha, w, p, h, b] \leftarrow$ randomSample($\mathbf{R}$)
11:     **else**
12:         $gp \leftarrow$ buildGaussianProcesses($\mathbf{G}$)
13:         $[l, \alpha, w, p, h, b] \leftarrow$ bestPredictedParameters($\mathbf{R}, gp$)
14:     **end if**
15:     $\mathbf{R} = \mathbf{R} \setminus \{[l, \alpha, w, p, h, b]\}$
16:     $\mathbf{T}' \leftarrow$ subsampleData($\mathbf{T}, 0.7$)
17:     $cls \leftarrow$ buildIndividual($\mathbf{T}', l, \alpha, w, p, h, b$)
18:     $acc \leftarrow$ LOOCV(cls) { *train data accuracy* }
19:     **if** $i < s$ **then**
20:         **if** $acc < lowestAcc$ **then**
21:             $lowestAcc \leftarrow acc, lowestAccIndex \leftarrow i$
22:         **end if**
23:         $c_i \leftarrow cls, e_i \leftarrow acc^4$
24:     **else if** $acc > lowestAcc$ **then**
25:         $c_{lowestAccIndex} \leftarrow cls, e_{lowestAccIndex} \leftarrow acc^4$
26:         $[lowestAcc, lowestAccIndex] \leftarrow$ findNewLowestAcc($\mathbf{C}$)
27:     **end if**
28:     $g_i \leftarrow \{[l, \alpha, w, p, h, b], acc\}$
29:     $i \leftarrow i + 1$
30: **end while**

Algorithm 2 Individual TDE(A list of n cases of length m with d dimensions, $\mathbf{T} = (\mathbf{X}, \mathbf{y})$)


Parameters: word length l, alphabet size α, window length w, normalisation parameter p, No. pyramid levels h, breakpoint function b

1: Let $\mathbf{H}$ be a list of n histograms $(\mathbf{h}_1, \ldots, \mathbf{h}_n)$
2: Let $\mathbf{K}$ be a list of dimension indexes found using LOOCV
3: Let $\mathbf{B}$ be a list of $|\mathbf{K}|$ l by α matrices of breakpoints found by b $(\mathbf{b}_1, \ldots, \mathbf{b}_{|\mathbf{K}|})$
4: Let $\mathbf{O}$ be a matrix of $|\mathbf{K}|$ by n by $m - w + 1$ containing the words for each window
5: **for** $i \leftarrow 1$ to n **do**
6:     **for** $g \leftarrow 1$ to $|\mathbf{K}|$ **do**
7:         **for** $j \leftarrow 1$ to $m - w + 1$ **do**
8:             $\mathbf{s} \leftarrow x_{i,k_g,j} \ldots x_{i,k_g,j+w-1}$
9:             $\mathbf{q} \leftarrow \text{DFT}(\mathbf{s}, l, \alpha)$ { $\mathbf{q}$ *is a vector of the complex DFT coefficients*}
10:             **if** p **then**
11:                 $\mathbf{q}' \leftarrow (q_2 \ldots q_{l/2+1})$
12:             **else**
13:                 $\mathbf{q}' \leftarrow (q_1 \ldots q_{l/2})$
14:             **end if**
15:             $r \leftarrow \text{SFAlookup}(\mathbf{q}', b_g)$
16:             **if** $r \neq o_{g,i,j-1}$ **then**
17:                 **for** $v \leftarrow 1$ to h **do**
18:                     $pos \leftarrow \text{index}(\text{spatialPosition}(r,v),g)$
19:                     $h_{i,pos} \leftarrow h_{i,pos} + 1$
20:                 **end for**
21:                 $bi \leftarrow \text{index}(\text{bigram}(r,o_{g,i,j-w}),g)$
22:                 $h_{i,bi} \leftarrow h_{i,bi} + 1$
23:             **end if**
24:             $o_{g,i,j} \leftarrow r$
25:         **end for**
26:     **end for**
27: **end for**

3.2 Diverse representation canonical interval forest (DrCIF)

The Diverse Representation Canonical Interval Forest (DrCIF) is an interval based ensemble and an extension of its prototype version, the Canonical Interval Forest (CIF) (Middlehurst et al. 2020a). Interval based classifiers extract phase-dependent subseries, aiming to find discriminatory features over different intervals. For time series of length m there are $m(m-1)/2$ possible intervals that can be extracted. The original interval based classifier, the Time Series Forest (Deng et al. 2013), is a component of HC1. It selects multiple intervals for each decision tree base classifier, then concatenates derived features (mean, standard deviation and slope) to form a diverse training set for each ensemble member. The other interval based classifier in HC1, RISE, selects a single interval for each base classifier, then derives spectral features (periodogram and auto-regressive terms) over the single interval. DrCIF replaces both these interval based classifiers, combining and enhancing both feature spaces. It draws on recent ideas presented in the STSF interval based classifier (Cabello et al. 2020) and the feature set method defined as the canonical time series

Fig. 4 Critical difference
diagram for five interval based
classifiers on 112 UCR datasets.
Each classifier builds 500 trees.
TSF and CIF extract sqrt(*m*)
intervals per tree. CIF subsam-
ples 8 attributes per tree
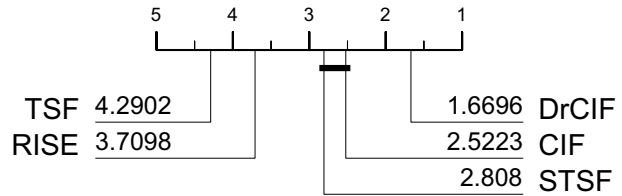
characteristics (catch22) (Lubba et al. 2019). The catch22 features are a set of 22 features
designed for time series data filtered from the 7658 features available in the highly com-
parative time series analysis (hctsa) toolbox (Fulcher and Jones 2017). After a pruning pro-
cess, the catch22 features were derived from a clustering and filtering of the 7658 hctsa
features based on accuracy, scalability and interpretability.

The base classifier for DrCIF is a simple information gain based tree used in TSF, called
the time series tree (Deng et al. 2013). Features from the tree are derived from multiple
intervals taken from the base series, the first order difference series and the periodograms
of the whole series. Intervals from each are randomly selected. Seven basic summary sta-
tistics are part of a pool of possible features extracted from an interval of any one of the
three representations. These are: the mean; standard-deviation; slope; median; inter-quar-
tile range; min; and max. DrCIF adds the catch22 features to this selection of summary
statistics to form a candidate pool of 29 features. *a* out of the 29 features available are
randomly selected for each tree. For each of the 3 representations, *k* phase dependent inter-
vals with randomly selected positions and lengths are extracted. The selected features are
then calculated from each interval. These features are concatenated into a $3 \cdot k \cdot a$ length
vector for each series, and the new dataset is used to build the tree. Diversity is achieved
by providing each base classifier with different intervals and a different subset of the 29
features. Generally, we select *k* as a function of the representation series length *rm*. Each
representation will differ in its length, with the periodogram being half the size of the base
series and the differences having one less value. As such it is likely the number of intervals
selected for each representation will differ. For multivariate data, DrCIF randomly selects
the dimension used for each interval. Replacing TSF with CIF in HIVE-COTE Alpha has
been shown to significantly improve the classifier on univariate data (Middlehurst et al.
2020a). The build process and the default parameter values for the DrCIF ensemble is
described in Algorithm 3. Figure 4 demonstrates that DrCIF represents a new best in class
interval based classifier.

Algorithm 3 Diverse Representation Canonical Interval Forest(A list of n cases of length m with d dimensions, $\mathbf{T} = (\mathbf{X}, \mathbf{y})$)

Parameters: the number of trees, r, the number of intervals per representation for each tree, k, and the number of attributes subsampled per tree, a (default $r = 500$, $k = 4 + (\sqrt{d}\sqrt{rm})/3$, and $a = 10$) { *where rm is the length of a representations series*}

1: Let $\mathbf{F} = (\mathbf{F_1} \ldots \mathbf{F_r})$ be the trees in the forest
2: Let $\mathbf{V}$ be a 3 by n by d list of series with variable length, containing the base series list $\mathbf{X}$ and $\mathbf{X}$ transformed to periodograms and first order differences
3: **for** $i \leftarrow 1$ to r **do**
4:     Let $\mathbf{S}$ be a list of n cases $(s_1 \ldots s_n)$ with $a \cdot k$ attributes
5:     Let $\mathbf{U}$ be a list of a randomly selected attribute indices $(u_1 \ldots u_a)$
6:     **for** $y \leftarrow 1$ to 3 **do**
7:         **for** $j \leftarrow 1$ to k **do**
8:             $b = rand(1,|\mathbf{V_y}|-3)$ { *interval position*}
9:             $l = rand(3,|\mathbf{V_y}|/2)$ { *interval length*}
10:            $o = rand(1, d)$ { *interval dimension*}
11:            **for** $t \leftarrow 1$ to n **do**
12:                **for** $c \leftarrow 1$ to a **do**
13:                    $s_{t,a(j-1)+c} \leftarrow summaryStat(u_c, \mathbf{V_{y,t,o}}, b, l)$
14:                **end for**
15:            **end for**
16:        **end for**
17:    **end for**
18:    $F_i.buildTimeSeriesTree([\mathbf{S}, y])$
19: **end for**

3.3 The arsenal: a ROCKET ensemble

The Random Convolutional Kernel Transform (ROCKET) (Dempster et al. 2020) uses a large number of randomly parameterised convolution kernels applied to each case. As each kernel is applied to a series, the max value and proportion of positive values are recorded and concatenated into a feature vector. These features are then used to build a linear ridge regression classifier with built in cross-validation to select the alpha parameter.

For each kernel generated, the parameters are selected from the following spaces: The length, l, is selected such that, $l \in \{7, 9, 11\}$; the value of each weight, w_i, is randomly sampled from a normal distribution $\sim \mathcal{N}(0, 1)$, and are then mean centered; bias b is sampled from a uniform distribution $\sim \mathcal{U}(-1, 1)$; dilation, a, is sampled from an exponential scale up to series length; the binary decision to pad the series p is chosen with equal probability, if true the series is zero padded at the start and end equally such that middle element of the kernel is applied to every point in the input series. Stride is always set to 1. For multivariate datasets, each kernel is assigned a random number of randomly selected dimensions. The kernel for the multivariate case is still one dimensional, but with weighting being different for each dimension. The max and proportion of positive values is calculated across all selected dimensions.

ROCKET is a very fast classifier that has state-of-the-art accuracy, and we believe it is the most important recent development in the field. It represents a different class of approach, and as such is a candidate for assimilation into the collective. However, an issue arises when trying to include ROCKET in HIVE-COTE: the ridge regressor used

by ROCKET is hard to configure to produce useful probability values for each class when making predictions. The CAWPE ensemble structure of HIVE-COTE uses weighted probabilities, and relies on classifiers to produce a distribution representative of the classifiers strength of belief in predictions. One solution would be to replace the ridge regressor with a classifier that does produce representative probability estimates. However, our experimentation with suitable replacement classifiers did not yield a candidate algorithm that was as accurate as the ridge regressor for ROCKET.

To solve this problem, the version of ROCKET we use in HIVE-COTE is an ensemble of smaller ROCKET classifiers. We refer to this fusillade of ROCKETs as the Arsenal. New cases are classified using the CAWPE exponential weighted majority vote, with the weights obtained from the ridge regression classifiers cross-validation. Arsenal is slower to build than ROCKET, but its improved probabilities make it a better candidate for HC2. The build process for Arsenal is described in Algorithm 4.

Algorithm 4 Arsenal(A list of n cases of length m with d dimensions, $\mathbf{T} = (\mathbf{X}, \mathbf{y})$)

Parameters: the ensemble size, r and the number of kernels per classifier, k (default r=25 and $k = 2000$)

1: Let $\mathbf{F}$ be a list of ROCKET classifiers $(\mathbf{f}_1, \ldots, \mathbf{f}_r)$
2: **for** $i \leftarrow 1$ to r **do**
3:     Let $\mathbf{X}'$ be a n by $2k$ list of transformed cases
4:     **for** $j \leftarrow 1$ to k **do**
5:         $[l, w, b, a, p, O] \leftarrow randomKernelParameters(m, d)$
6:         **for** $t \leftarrow 1$ to n **do**
7:             $max \leftarrow -\infty$
8:             $ppv \leftarrow 0$
9:             **for** $g \leftarrow 1$ to $(m + (2p)) - ((l-1)a)$ **do**
10:                 $v \leftarrow 0$
11:                 **for** $c \leftarrow 1$ to $|O|$ **do**
12:                     $v \leftarrow v + applyKernel(\mathbf{X}_t, g, O_c, l, w, b, a)$
13:                 **end for**
14:                 **if** $v > max$ **then**
15:                     $max \leftarrow v$
16:                 **end if**
17:                 **if** $v > 0$ **then**
18:                     $ppv \leftarrow ppv + 1$
19:                 **end if**
20:             **end for**
21:             $\mathbf{X}'_{t,2j-1} \leftarrow max$
22:             $\mathbf{X}'_{t,2j} \leftarrow ppv/((m + (2p)) - ((l-1)a))$
23:         **end for**
24:     **end for**
25:     $f_i.buildRidgeClassifierCV([\mathbf{X}', y])$
26: **end for**

3.4 Shapelet transform classifier (STC)

Shapelets are phase independent subseries found in the training data. The STC approach to classification using shapelets is to construct a pipeline where the search for high quality shapelets is followed by a transformation where the new features represent distances

to retained shapelets. A rotation forest (Rodriguez et al. 2006) is constructed on the transformed features. The shapelet transform is highly configurable: it can use a range of sampling/search techniques in addition to alternative quality measures. We present the default settings and direct the interested reader to the tsml code. The original shapelet based algorithms performed an exhaustive search of all possible shapelets. This of course is very slow. However, subsequent work (Bostrom and Bagnall 2017) identified that exhaustive search can actually lead to over fitting and is never necessary. Instead, we randomly search for shapelets for a given amount of time, which is now a parameter (defaults to one hour). Our version of STC is essentially the same as that used for HC1 (Bagnall et al. 2020), so we direct the interested reader there for more details. The multivariate version searches dimensions independently and is the same version used in the MTSC bake off (Ruiz et al. 2021).

4 Experimental structure

We perform our univariate time series experiments on 112 of the 128 datasets from the UCR time series archive (Dau et al. 2019). We exclude datasets containing series of unequal length or missing values, as we do not want an algorithm's aptitude for these cases to alter results and most implementations are not set up to handle these kinds of data. We additionally remove the Fungi data, which only provides a single train case for each class label. Being unable to properly process data with single case classes is a general limitation for approaches which rely on cross-validation or train accuracy estimates. We use the data as presented by the archive. We do not automatically z-normalise the data. The decision on whether to normalize is delegated to the classifier modules for each representation, which handle it differently. For example, TDE follows BOSS and has normalisation as a parameter, whereas STC always normalises since it also normalises all shapelets. DrCIF does not normalise, since some of its structure summary features measure scale and variance. For our multivariate experiments we use all 26 equal length datasets of the 30 total from the UEA multivariate time series archive (Bagnall et al. 2018). For each dataset we present performance as an average over 30 resamples. Both archives provide a default split into train and test sets which we use for the first resample. The remaining 29 are randomly resampled from the original split in a stratified manner. We seed each classifier and data resample using the fold index to ensure out results are reproducible.

All of our non-deep learning experiments were run using the Java tsml toolkit implementations. For deep learning approaches we use Python sktime companion package sktime-dl.[1] The configuration for each algorithm is provided in Table 2.

Our experiments using algorithms from tsml were conducted on the UEA high performance computing (HPC) cluster. Each job consists of a single dataset, classifier, fold evaluation and runs on a single core. Due to limits on the cluster, a job has a maximum run time of seven days. The maximum memory allowance provided by the cluster stands at 500GB.

sktime-dl experiments were performed on desktops GPUs, one with a Titan XP and one with 4 Titan X Pascals. Each job is run on a single GPU with each GPU running only one job at a time. There is no time limit for running these jobs. However, they are limited by the GPU memory of 12GB per card.

---

[1] https://github.com/sktime/sktime-dl.

Table 2 Classifier configurations for our experiments where m is the series length, d is the number of dimensions and rm is the lengths of DrCIF representations

Algorithm	Configuration
TDE	250 parameter sets sampled, 50 max ensemble size
DrCIF	500 trees, 4+(sqrt(rm)*sqrt(d))/3 intervals per representation, 10 attributes per tree, rm/2 max interval length
Arsenal	2,000 kernels per classifier, 25 ensemble size
STC	1 hour Shapelet Transform train time contract, 200 Rotation Forest trees
ROCKET	10,000 kernels
HIVE-COTE 1.0	α of 4
HIVE-COTE 2.0	α of 4
TS-CHIEF	500 trees, 5 EE splitters, 100 RISE splitters, 100 BOSS splitters
InceptionTime	Epochs 1500, batch size 64, learning rate $1e-3$ and halved after no improvement for 50 epochs.
	Two residual blocks each with three Inception modules with kernel sizes per module [10, 20, 40] plus bottleneck filters for all conv layers of 32
CIF	500 trees, sqrt(m)*sqrt(d) intervals, 8 attributes per tree, m/2 max interval length
DTW_D	Full warping window

We evaluate classifier performance using accuracy, area under the receiver operating characteristic (AUROC) and negative log-likelihood (NLL). This means we can assess classifiers based on predictive performance, ranking predictions and probability estimates. For problems with more than two classes, one-versus-many AUC is averaged over the class values, weighted by class value frequency.

To compare the scores averaged over 30 resamples of two classifiers for multiple datasets, we use a pairwise Wilcoxon signed-rank test. For multiple classifiers over multiple datasets we use an adaptation of the critical difference diagram (Demšar 2006), replacing the post-hoc Nemenyi test with a comparison of all classifiers using pairwise Wilcoxon signed-rank tests, and cliques formed using the Holm correction recommended by (García and Herrera 2008; Benavoli et al. 2016).

5 Results

We have conducted extensive experimentation and produced a high volume of results. There is marginal utility in having massive tables of results. Instead, we present summarised results and have put the raw results and all of the summary analysis on the accompanying website.[2] Details of how to reproduce the results, or some subset of results, are given on the website and the code walk through on the `tsml` github.

5.1 Performance on the UCR archive of 112 univariate TSC problems

Our core result is that HC2 is significantly better than the current state of the art on the 112 UCR equal length datasets. Figure 1 in the Introduction shows the accuracy

---

[2] www.timeseriesclassification.com/HC2.php.

Fig. 5 Test accuracy critical difference diagram for nine classifiers, averaged over 30 resamples for each of the 112 UCR problems

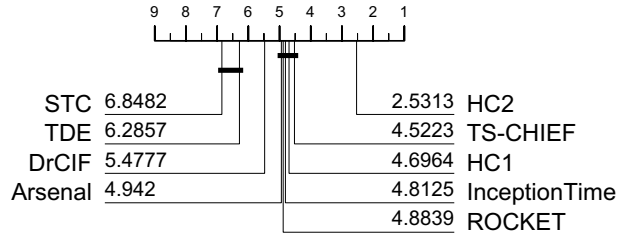|	
STC 6.8482	2.5313 HC2
TDE 6.2857	4.5223 TS-CHIEF
DrCIF 5.4777	4.6964 HC1
Arsenal 4.942	4.8125 InceptionTime
	4.8839 ROCKET

Fig. 6 Test negative log likelihood critical difference diagram for nine classifiers, averaged over 30 resamples for each of the 112 UCR problems
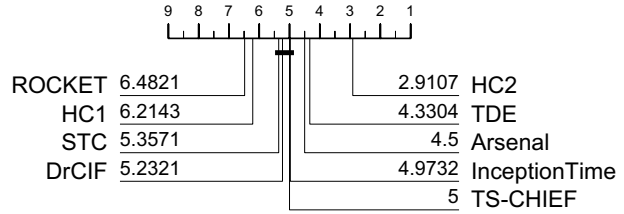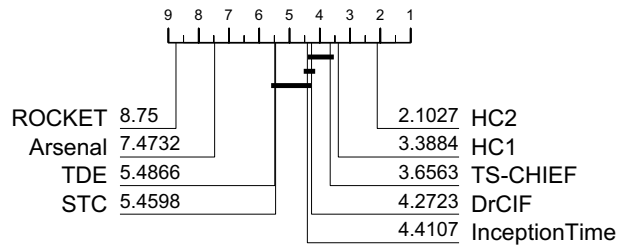
ROCKET 6.4821	2.9107 HC2
HC1 6.2143	4.3304 TDE
STC 5.3571	4.5 Arsenal
DrCIF 5.2321	4.9732 InceptionTime
	5 TS-CHIEF

Fig. 7 The area under the receiver operator curve critical difference diagram for nine classifiers, averaged over 30 resamples for each of the 112 UCR problems

ROCKET 8.75	2.1027 HC2
Arsenal 7.4732	3.3884 HC1
TDE 5.4866	3.6563 TS-CHIEF
STC 5.4598	4.2723 DrCIF
	4.4107 InceptionTime

performance of HC2 versus the four baseline approaches. Figures 5, 6 and 7 shows the critical difference diagrams for accuracy, NLL and AUROC for HC2, its four components and the four state-of-the-art algorithms. They demonstrate that HC2 is significantly better than its components and the four benchmarks using all three metrics. For accuracy (Fig. 5), STC and TDE form the lowest clique. DrCIF is between these and the current state of the art, which contains Arsenal. ROCKET is the worst for probability estimates (Fig. 6), since it only produces 0/1 estimates. The better probability estimates of Arsenal justify our design decision, since HC weights distributions not predictions. TDE is surprisingly good at probabilistic prediction. The poor probability estimates of HC1 highlights one source of improvement in HC2. HC1 does much better at AUROC (Fig. 7), whereas Arsenal does much worse, indicating further calibration may benefit the Arsenal.

Figure 8 shows the accuracy scatter plots of HC2 against each of baseline classifiers and Table 3 summarises the differences in test accuracy between HC2 and the four baselines.

We observe that there is lower variance between HC1 and HC2, but that HC2 consistently outperforms HC1 with an average accuracy of more than 1%. The variation in difference to HC2 is greater with the other three classifiers, in particular ROCKET. The median difference is lower than the mean in all cases. This suggests skew, which supports the core hypothesis that the heterogenous ensemble can compensate for the
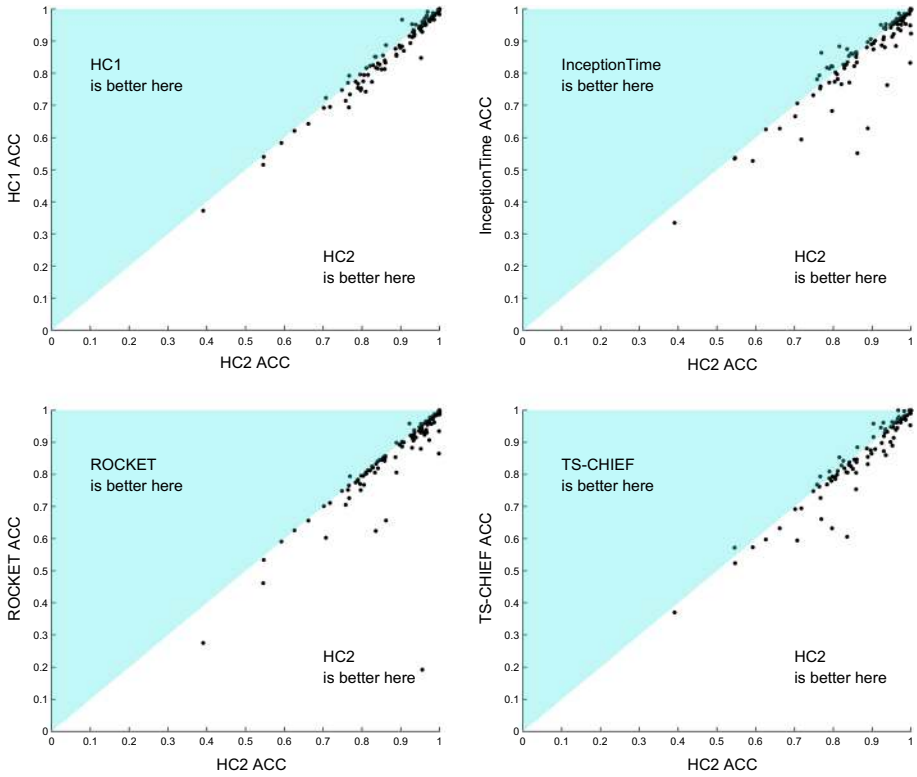
Fig. 8 Scatter plots of HIVE-COTE 2.0 against each of the baseline classifiers

Table 3 Summary of the differences between HC2 and the benchmarks

Classifier	Mean (%)	Median (%)	Max (%)	Min (%)	StDev (%)	HC2 Wins	HC2 Loses
TS-CHIEF	− 1.36	− 0.41	− 22.99	5.50	3.64	77	29
InceptionTime	− 1.69	− 0.37	− 31.04	9.46	5.13	74	32
HC1	− 1.06	− 0.69	− 10.47	6.33	2.10	82	25
ROCKET	− 2.49	− 0.72	− 76.31	3.64	7.92	97	11

A negative value means the HC2 is better

shortcomings of its components. It also suggests that HC2 has a higher representational power, in that it can find a more diverse set of features.

Accuracy is not the only consideration. Table 4 summarises the run time and memory requirements for the classifiers compared in Fig. 8. There are a few caveats to these results. Firstly, all of the results except InceptionTime are run in a single thread on a CPU. Thus InceptionTime time experiments are not really directly comparable, since it runs on a GPU. ROCKET and HC2 are forced to run in a single thread, despite being threadable. The times for the HC2 components are without the time to estimate performance, but these are included in the HC2 times. Memory is the maximum memory used throughout the run, as

Table 4 Run time and memory requirements to train single resample of 112 UCR problems

Algorithm	Total (h)	Average (min)	Max Mem (MB)
ROCKET	2.85	1.53	4349
Arsenal	27.91	14.95	1683
DrCIF	45.40	24.32	920
TDE	75.41	40.40	6565
InceptionTime	86.58	46.38	–
STC	115.88	62.08	4219
HC2	340.21	182.26	6677
HC1	427.18	228.84	4876
TS-CHIEF	1016.87	544.75	26,052

The median of 30 runs is taken for each dataset

obtained from the Java garbage collector, and should be considered approximate. We are not set up to measure the maximum memory used with InceptionTime in practice, but we know it did not exceed 12GB, because that was the memory available on the GPU. Since the run times are sequential, we also use the sequential memory for HC1 and HC2. These would be higher if the classifier were threaded, but of course the run time would be much lower.

With this in mind, we can make the following observations. ROCKET lives up to its name and can build models on all 112 data in under 3 hours, even when not threaded. If speed is the main criteria, ROCKET is a good starting point in any analysis. STC is the slowest component, but this is caused by the configuration rather than an inherent problem: STC defaults to a one hour shapelet search or a full evaluation of the shapelet search if this will take less than an hour. For the very small problems, it takes a lot longer than the other algorithms (although still less than an hour). HC2 is faster than HC1, primarily because of improvements to STC and the change in classifiers. TS-CHIEF is the slowest algorithm by far, and seems to scale less well than the others. On the slowest five problems (HandOutlines, NonInvasiveFetalECGThorax1 and 2, SemgHandMovementCh2 and EthanolLevel), it takes ten times longer than HC2, but the difference is minimal on smaller problems. All of the classifiers are within reasonable bounds for memory. TS-CHIEF has the highest memory requirement, with a max requirement of 26GB on HandOutlines. As with run time, it seems to scale worse than the others. HC2 requires more memory than HC1, but it is is not unreasonable. ROCKET has a worse max memory case (ElectricDevices) than Arsenal. Overall, ROCKET tends to use less memory than Arsenal but appears to scale worse for larger datasets with many cases. Arsenal uses a smaller amount of kernels for each individual classifier, meaning that each transformed set of data is smaller in size and discarded before the next is built. ROCKET on the other hand must transform using its larger amount of kernels at a single point. Figure 9 summarises the accuracy and run time results by plotting the log of the train time against the rank.

5.2 Performance on the UEA archive of 26 multivariate TSC problems

A bake off of TSC algorithms (Ruiz et al. 2021) using the MTSC UEA archive found three algorithms (that could complete all 26 data sets) were significantly more accurate than DTW-D. These were ROCKET, HC1 and CIF. We have repeated these experiments with
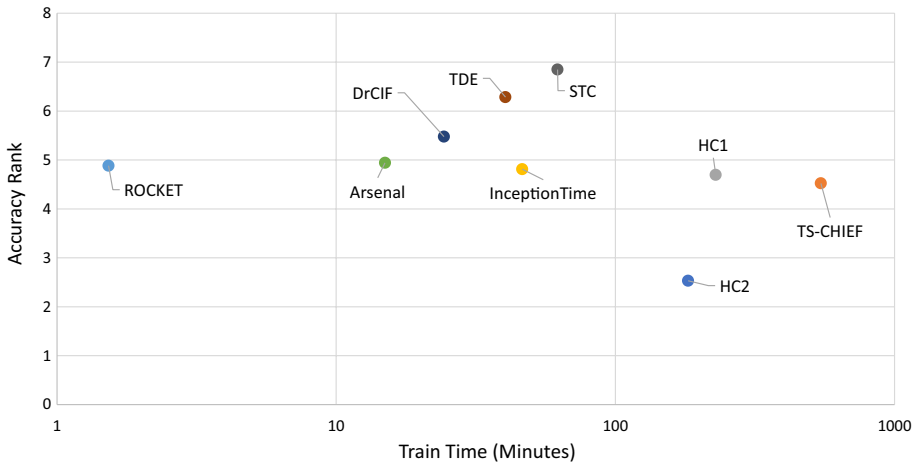
Fig. 9 A comparison of classifiers in terms of accuracy rank and train time. The time and accuracy are averaged over 112 UCR problems. The train time is on a log scale

Fig. 10 Test accuracy critical difference diagram for five classifiers, averaged over 30 resamples for each of the 26 UEA MTSC problems
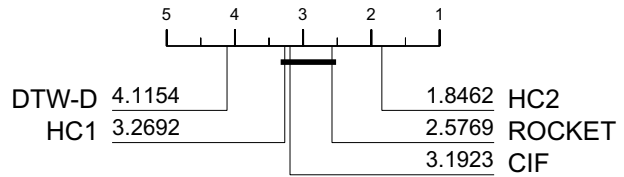

Fig. 11 Test negative log likelihood critical difference diagram for five classifiers, averaged over 30 resamples for each of the 26 UEA MTSC problems
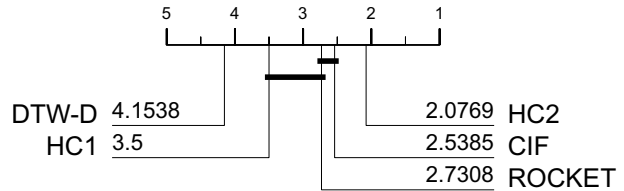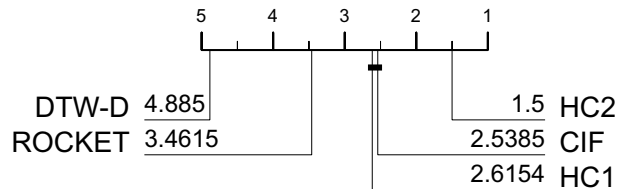

Fig. 12 The area under the receiver operator curve critical difference diagram for five classifiers, averaged over 30 resamples for each of the 26 UEA MTSC problems

HC2. Figures 10, 11 and 12 show that HC2 is significantly better than DTW-D, ROCKET, HC1 and CIF on the 26 datasets for accuracy, NLL and AUROC.
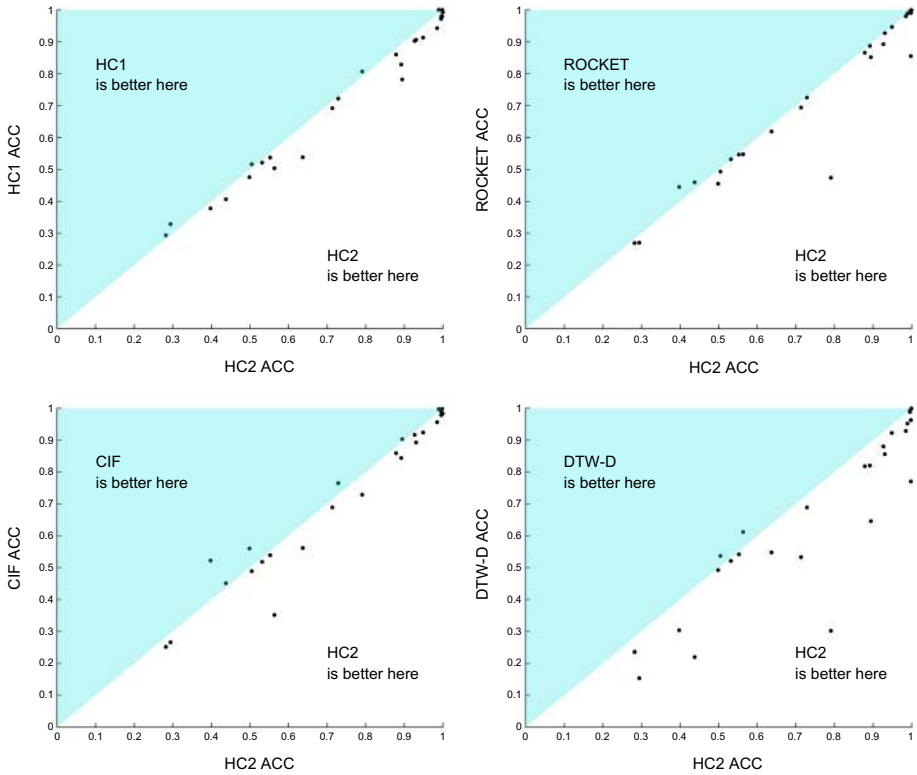
Fig. 13 Scatter plots of HIVE-COTE 2.0 against each of the baseline classifiers

Table 5 Summary of the differences between HC2 and the benchmarks

Classifier	Mean (%)	Median (%)	Max (%)	Min (%)	StDev (%)	HC Wins	HC Loses
ROCKET	− 2.52	− 0.64	− 31.65	4.73	6.75	19	5
CIF	− 1.71	− 1.58	− 21.21	12.43	5.56	19	7
HC1	− 2.25	− 1.92	− 11.27	3.44	3.30	20	6
DTW-D	− 8.22	− 4.66	− 48.94	4.87	11.39	23	3

A negative value means the HC2 is better

Figure 13 shows the accuracy scatter plots and Table 5 summarises the differences of HC2 against the benchmarks. We think these results strongly support the assertion that HC2 represents a new state of the art for multivariate time series classification.

6 Inside HC2: an ablative study

We address the question of why HC2 works so well, and evaluate design decisions made in the change from HC1 to HC2. HC1 uses cross-validation to estimate the test accuracy from the train data for each component. HC2 modules are all ensembles, and so it was natural to attempt to use bagging and the resulting out-of-bag error estimate to speed up

Table 6 Summary of the difference between estimated and observed test accuracy for HC2 and its components

Classifier	Mean (%)	Median (%)	Min (%)	Max (%)	MSE (%)
DrCIF	− 2.15	− 0.93	− 46.78	9.64	0.40
Arsenal	− 1.17	− 0.40	− 23.13	9.23	0.15
STC	1.24	0.78	− 55.54	26.88	0.86
TDE	− 1.14	− 0.77	− 18.89	10.02	0.14
HC2	0.47	0.11	− 19.81	19.17	0.13

A positive figure means that the classifier is overestimating accuracy from the train data
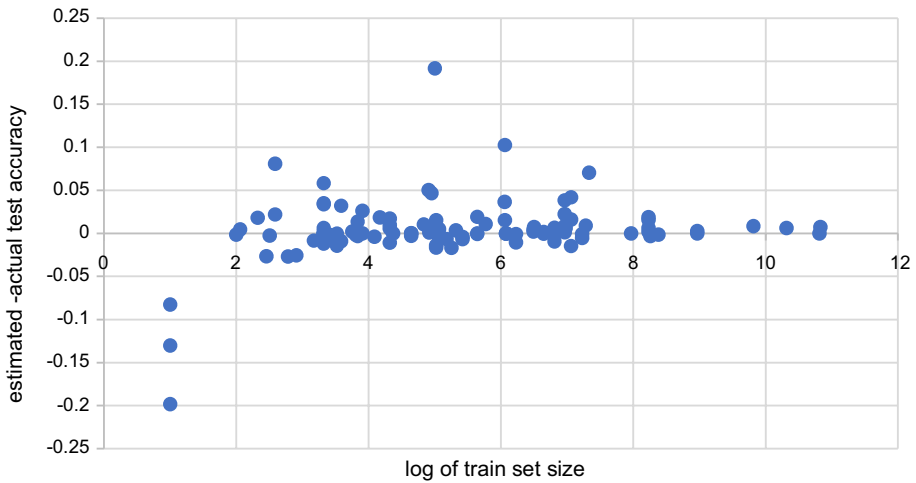
Fig. 14 Difference in estimated and observed test set accuracy against the log of the train set size for 112 UCR datasets

HIVE-COTE training. However, whilst this produces good estimates of the test accuracy, the models were less accurate on unseen data for every module. Hence, we made the decision to fit a separate bagging model for the estimation stage for those that need it, thus providing an order of magnitude speed up compared to cross-validation. DrCIF and Arsenal both create separate models with bagging to generate their estimates. STC builds a new Rotation Forest model with bagging for its estimate, but uses the same transformed shapelet data for both. TDE naturally takes a 70% subsample when creating its ensemble, as such a new model is not required to generate its out-of-bag error. However, we were concerned that these estimates may be biased and/or not consistent. Table 6 summarises the distributions of the differences between estimated and observed test accuracy for HC2 and its components.

Whilst there is small bias for each component, HC2 ensemble method compensates for this and has the lowest average deviation (and MSE deviation) between estimated and observed test accuracy. This is due to the averaging ensemble effect, and the biasing effect of reusing estimates from the components: a full nested cross-validation estimate would be computationally demanding and is not necessary. STC is the only component that is over optimistic. This is to be expected. STC performs a random search on the whole train data then bags rotation forest. This introduces bias, and is a possible area for
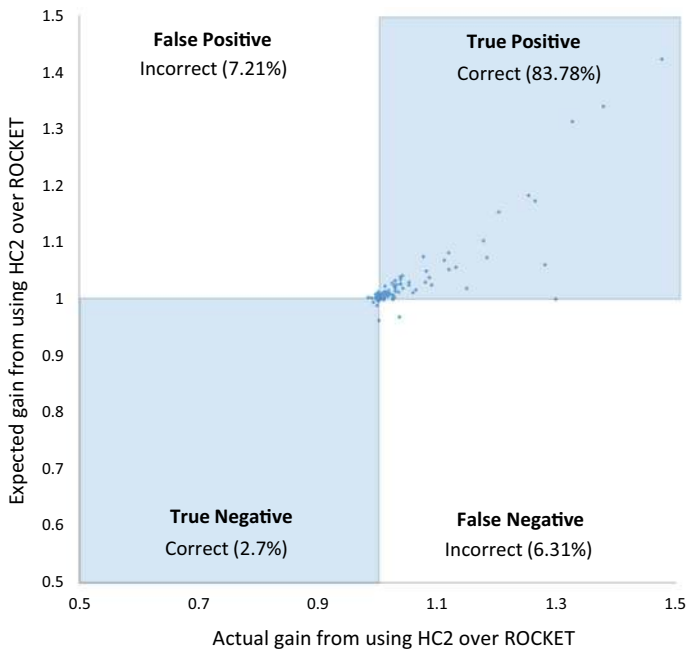
Fig. 15 Texas sharpshooter plot for HC2 versus ROCKET. Each point represents a single dataset. The x-axis is the ratio of HC2 and ROCKET actual test accuracy and the y-axis is the ratio or predicted test accuracy

Table 7 Possible variants of HC2 components

Component	HC-1	HC-2	HC-3	HC-4	HC-5	HC-6	HC-7	HC-8	HC-9	HC-10
DrCIF	X	X	X				X	X	X	
Arsenal	X			X	X		X	X		X
STC		X		X		X	X		X	X
TDE			X		X	X		X	X	X

future improvement. The min and the max show that there are some very large differences between estimate and observed. These primarily arise in problems where there are very few cases per class, such as PigAirwayPressure, PigCVP and PigArtPressure, which each have only two cases per class. Every classifier underestimates the test accuracy by over 10% on these problems. Figure 14 shows the difference in the test accuracy estimate and actual plotted against the log of the train set size for HC2. The picture is not conclusive, but it could be argued that the variance of the difference is decreasing, which is encouraging evidence for the consistency of the HC estimate.

Another benefit of accurate estimates from the train data is that they can be used to compare classifiers with a Texas Sharpshooter plot (Batista et al. 2014). These compare two classifiers by comparing the ratio of estimates from the train data with those of the test data to form a kind of contingency table. Computing train estimates through cross-validation for TS-CHIEF and InceptionTime is unpractical due to run times. However, it is

Critical difference diagram:

HC-6	7.25
HC-1	7.2232
HC-4	7.0179
HC-2	6.9375
HC-5	6.7991

3.8661	Full HC2
5.0848	HC-8
5.2054	HC-10
5.3214	HC-7
5.375	HC-9
5.9196	HC-3

Critical difference diagram:

HC2-FS	4.808
HC2-PB	4.4955
HC2-ES	3.9821

1.7902	HC2-Oracle
2.442	HC2
3.4821	HC2-RandF
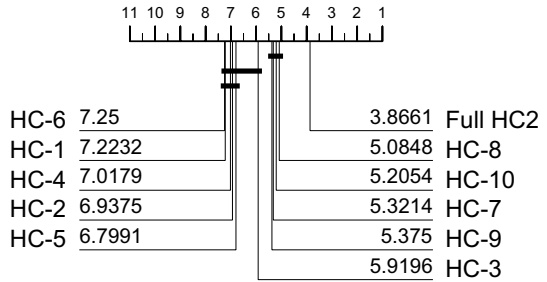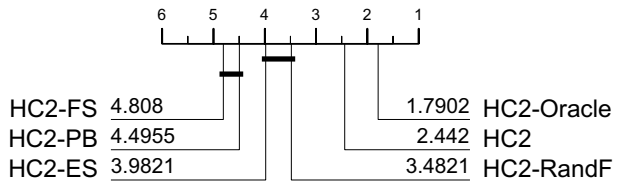
easy with ROCKET, since it is so fast. Figure 15 shows the plot for ROCKET versus HC2. Using the train estimates would lead to the correct decision of choosing HC2 on 94 of the 112 datasets.

The next issue is to quantify what impact each component has on the overall performance. Ignoring single component variants (which are presented in Fig. 5), there are 11 possible combinations, identified as HC-1 to HC-10 in Table 7, with the eleventh being the Full HC2, referred to as just HC2 elsewhere. Figure 16 shows the relative performance of the 11 possible variants. The two component models (HC-1 to HC-6) form a clear clique, followed by another clique of three component versions. However, the full four component classifier is significantly more accurate than all of the other variants. This demonstrates that each element contributes to the overall whole.

6.1 Ensembling methods

In extending HC1 into HC2, the two main factors are what representations should be included in the ensemble, and how the predictions drawn from each representation should be combined. Here we investigate the latter. HC1 uses the CAWPE ensembling scheme, which was found to be the best combination method for small sets of diverse classifiers across two different dataset archives with limited domain specialisation or prior knowledge (Large et al. 2019b). It was also shown that it improved HC1's performance relative to the previous simple majority voting. With updated components, which may be more or less specialised into their own representation formats with different degrees of overlap in their expertise, does this still hold true, or would a different scheme be better? We compare various ensemble selection and stacking schemes to assess whether a more complex scheme than CAWPE could improve HC2. To avoid suspicions of overfitting, we make it clear that we performed this analysis after generating the results presented in Sect. 5 using the design we selected a priori.

For context, we also compare to the individual model selection schemes of picking the best classifier per dataset resample based on the train estimates, and picking the best based on the test data (i.e. cheating) as an oracle scheme. In general, a reasonable ensembling scheme will on average perform somewhere between these two landmarks across arbitrarily
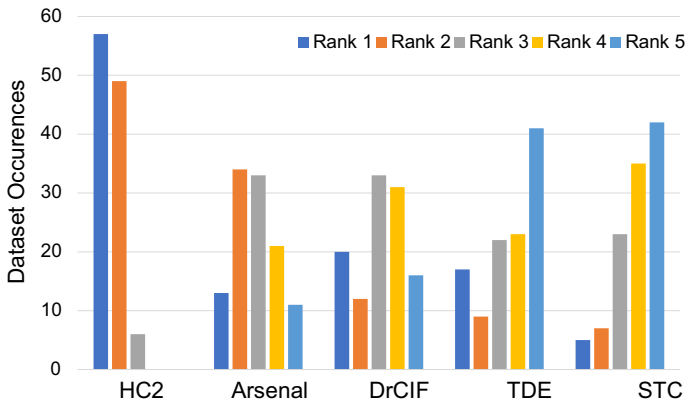
Fig. 18 Histograms of ranks between HC2 and its components over the 112 univariate datasets

large dataset spaces. Combining the predictions of the classifier pool has a beneficial averaging effect by accounting for the imperfections of performance estimation mechanisms from the train data. However, they may have an overly conservative averaging effect compared to looking at test performance to pick the best.

Figure 17 summarises our comparison of alternate ensembling schemes over the HC2 components. HC2-Oracle cheats by picking the single best component based on test accuracy, while HC2-PB picks the best based on the train estimate of test accuracy. HC2-FS executes a forward selection of components per dataset, ranking the train estimates and continuing to include components into the ensemble in order while the ensemble's own train estimate continues to improve. HC2-ES uses ensemble selection per dataset as described in (Caruana and Niculescu-Mizil 2004), which selects and weights components based on a repeated bagging with replacement strategy. Lastly, HC2-RandF stacks a random forest classifier onto the meta-data of the components' predicted probability distributions.

We can see that, unsurprising, the oracle selection scheme is still the best on average, and that perfectly selecting the best representation per dataset would still be better than combining them. If there is enough training data to produce reliable train estimates that are unbiased and have very low variance, this may be achievable in practice. However, the reality on our data is that picking the best on the train data (HC-PB) is significantly worse than HC2 and HC2-Oracle. It is worth stressing that selecting the single best component on test accuracy is not always the best. In accordance with the original hypothesis for HIVE-COTE, there are 31 datasets where combining representations is outright better than picking the best, even with perfect hindsight of stochastic differences brought about by resampling.

For many problems, discriminatory features may exist in multiple domains. This is often counter to received wisdom, it is always tempting to think a single type of model is the best approach. HC2 can discover complex interactions between domains. Figure 18 compares the ranks of HC2 (with its default CAWPE) and its individual constituents in isolation. HC2 is in fact best or tied for best on 57 of the 112 datasets, and rarely if ever ranked worse than second. This shows that beyond the requirement for perfect domain knowledge being required to beat ensembling on average, on many individual datasets more representations increase the accuracy outright.

Otherwise, Fig. 17 shows that using the CAWPE scheme over the HC2 components is significantly better than the alternatives on average. Most popular ensemble schemes in the
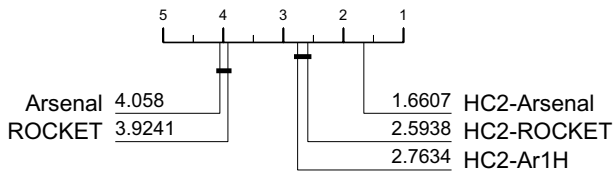
Fig. 19 Critical difference diagram for both versions of ROCKET and versions of HIVE-COTE using them on 112 UCR datasets. HC2-Ar1H represents HIVE-COTE using the Arsenal classifier with probabilities generated in the same way as ROCKET

literature assume a large pool of potentially homogeneous classifiers. We have a small pool of heterogeneous classifiers, and evidence from these experiments and an extensive study on standard classification problems (Large et al. 2019b) suggest that CAWPE is the best ensemble scheme for this scenario.

HC2 has a single parameter, the weighing factor α. We set α to four when first developing the CAWPE algorithm with the UCI datasets and have kept it the same to avoid the danger of parameter selection bias. However, it is worthwhile considering how sensitive HC2 is to this parameter. We evaluated HC2 for $\alpha = \{1, 2, \ldots, 10\}$. We found that in fact $\alpha = 8$ is the best overall, but that there was very little difference between all values greater than three.

Finally, we explore the effect of using Arsenal instead of ROCKET within HC2. Figure 19 shows both versions of ROCKET and three versions of HC2, two with a single version of both included in the ensemble and one containing a version of Arsenal where the probability of the predicted class is set to 1 rather than generated through the ensemble (Ar1H). Arsenal makes no improvement over default ROCKET in terms of accuracy, and Arsenal using the same method for generating probabilities as ROCKET makes no improvement in HC2. However, the HC2 version including an unaltered Arsenal is significantly better. Even with probabilities estimated through the votes of a small sized ensemble, a large difference is made over having none at all in HIVE-COTE.

To investigate the type of datasets that most benefit from using Arsenal rather than ROCKET, we looked at the datasets where HC2-Arsenal is on average more accurate than HC2-ROCKET by more than 0.5%, but where we do not see the same difference in the difference between Arsenal and ROCKET. Twenty two data sets fulfill this criteria. These 22 have, on average, more class values than the other 90 datasets. The mean number of classes for the 22 HC-Arsenal winners is 16.22 (median 6.5), whereas for the rest the mean is 6.34 (median 3). The 22 are also on average longer. The mean series length for the 22 is 860, whereas for the rest it is 480.

7 HC2 usability

All our code is open source and our experiments are simple to reproduce. Two implementations of HC2 are available in toolkits we help maintain and develop. `tsml`[3] is a Java based time series toolkit compatible with Weka and our primary development platform for TSC. We also implement our algorithms in `sktime`,[4] a Python based time series toolkit

---

[3] https://github.com/uea-machine-learning/tsml.

[4] https://github.com/alan-turing-institute/sktime.

Table 8 Train times in hours on problems where a sequential run of HC2 takes longer than 12 hours

Problem	Rocket	InceptionTime	TS-CHIEF	Hc1	HC2
ElectricDevices	0.43	6.15	7.94	56.80	54.55
Crop	0.22	4.37	4.59	73.54	28.41
FordB	0.19	6.06	43.33	19.33	24.67
FordA	0.19	6.01	40.88	20.25	23.52
HandOutlines	0.23	7.11	166.76	7.77	18.53

Table 9 Table showing the attributes of the large datasets used in checkpointing experiments

	FruitFlies	InsectSound
Train Size	17,255	25,000
Test Size	17,256	25,000
No. Classes	3	10
Series Length	5000	15,883
Total size (Gb)	2.44	14.75

compatible with scikit-learn. Where possible, we have verified the consistency of results in both toolkits. Both offer an easy to use interface, and we have provided example code on the website associated with this paper. All datasets are available in a format directly usable in `tsml` and `sktime`, and we have also provided details of how to recreate all our experiments.

Table 4 shows that when run sequentially, HC2 is slower than the current state of the art, particularly ROCKET. If speed is more important than a small accuracy gain, this is an argument against using HC2. HC2 is simply not designed to be trained in seconds, and we would not recommend its use in scenarios where models need to trained incredibly quickly. However, we have designed HC2 so that the run time can be controlled by the user through a time contract. We make the assumption that when time is a serious constraint, the problem must be fairly large. There are only five problems where a sequential build of HC2 would take more than half a day with a single processor run. The list of problems, and the time taken by TS-CHIEF, InceptionTime and ROCKET, are listed in Table 8. ROCKET is very fast, InceptionTime hard to compare to and TS-CHIEF is similar to HC2 on average but is unpredictable. If we run HC2 with a four hour contract on these problems we achieve 98% of the final accuracy, and if we run it for 12 hours we achieve 99% of the full build accuracy. If a reasonable model on bigger problems is required in hours, then contracting HC2 offers a good solution. However, if the problem is truly large, then all TSC have usability issues. TS-CHIEF can require massive amounts of memory and/or time and the memory usage of ROCKET can rapidly increase with the number of cases used.

For genuinely large data where HC2 may takes weeks for a full run, it is worthwhile considering how long it would take to converge. With many algorithms, it is often the case that most of the gain in accuracy is made relatively quickly. On large datasets this can equate to days of processing that contributes relatively little to the overall performance. Furthermore, a practitioners concern may not be accuracy and an understanding of the

Table 10 Table showing both accuracy achieved by last checkpoint and variance in accuracy between first and last checkpoint for each HC2 constituent on the FruitFlies dataset

	Accuracy	Variance	Constituents
TDE	0.8016	− 0.0822	20
STC	0.8901	− 0.0431	N/A
DrCIF	0.9278	− 0.1084	65
Arsenal	0.6433	− 0.0131	25

Table 11 Table showing both accuracy achieved by last checkpoint and variance in accuracy between first and last checkpoint for each HC2 constituent on the InsectSound dataset

	Accuracy	Variance	Constituents
TDE	0.2746	− 0.0735	14
STC	0.7357	− 0.1509	N/A
DrCIF	–	–	–
Arsenal	0.2951	− 0.0313	10
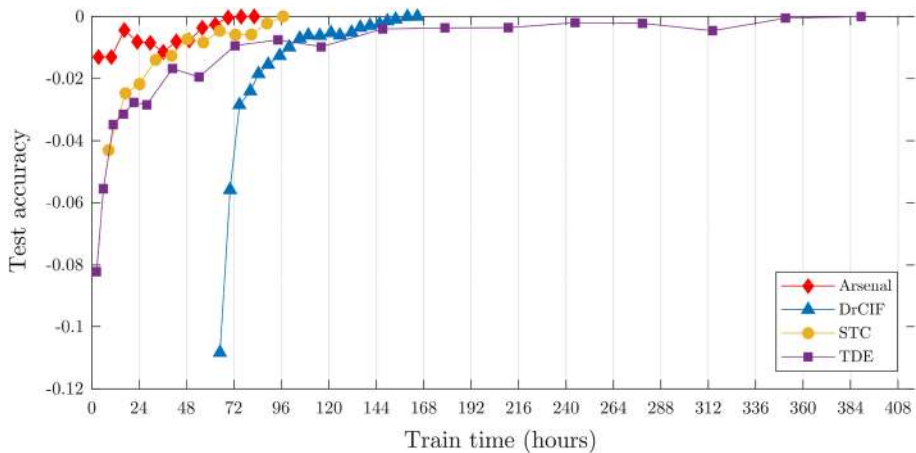
Fig. 20 Accuracy as a function of train time for HC2 components on the FruitFlies dataset

evolution of performance over time provides a foundation on which paramerterisation decisions can be made.

In this Section we comment on the evolution of accuracy over time for each of the HC2 components. Experiments were run on 2 large datasets, described in Table 9. The datasets used are notoriously problematic for complex approaches. This is typically because internal transforms are sensitive to series length, number of cases or both. These datasets were deliberately chosen to explore the limitations of the HC2 constituents.

In order to overcome the imposed runtime limitations on the UEA HPC cluster a checkpointing mechanism was utilised to periodically save an approaches state during the training phase. This allowed both the continuation of training beyond what would usually be feasible, via reloading and continuing training from the saved point, and the opportunity to assess the approach at each saved state, via invoking the test phase after reloading the saved state. Figures 20 and 21 show how relative accuracy changes with respect to time
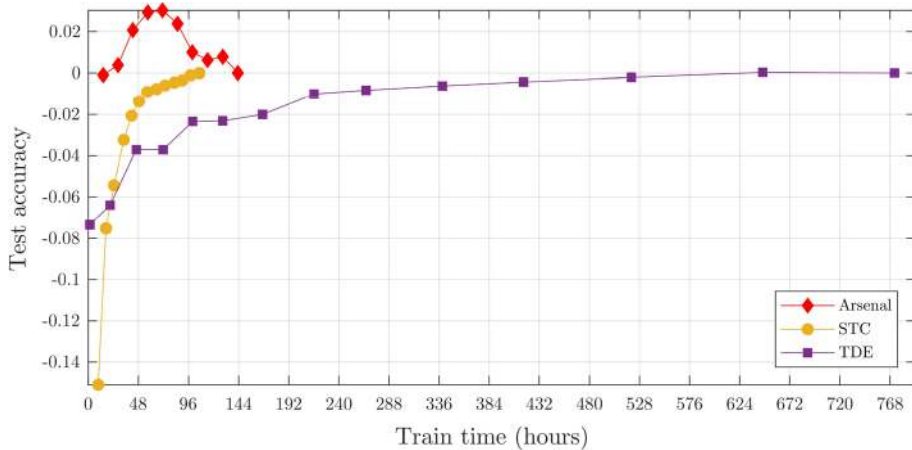
Fig. 21 Accuracy as a function of train time for HC2 components on the InsectWingbeat dataset

for the FruitFlies and InsectSound datasets. Each data point shows the relative difference of the accuracy achieved at each checkpoint with respect to the last checkpoint recorded. Tables 10 and 11 present the real accuracy achieved with the last checkpoint processed, the variance in accuracy between the first and last recorded checkpoint, and the number of constituents built in by the last checkpoint.

Figures 20 and 21 show the accuracy of DrCIF, STC and TDE follow a similar trend with respect to time. In the case of these approaches accuracy increases quickly as the number of constituents present increases, before reaching the point of diminishing returns. For these approaches 80% of the accuracy achieved is done so in less than 50% of the train time used by the last recorded checkpoint. The Arsenal approach does not follow this trend and instead the total variance in accuracy throughout the training period is not as pronounced. Also, the changes in accuracy appear to be more erratic with additional constituents producing decreases in accuracy as well as increases.

Furthermore, of the 8 combinations presented only the Arsenal approach on the Fruit-Flies dataset was able to complete. In most cases the outstanding experiments were prohibited by inflation in the time taken to test. Checkpointing during the test phase is not implemented and as a result the entire test process is subject to a hard time limit of 7 days. This effected STC and DrCIF on the FruitFlies dataset and STC and TDE on the InsectSounds dataset. Aditionally, the Arsenal approach was limited by its memory requirement on the InsectSounds dataset, for which we are limited to 700Gb.

8 Conclusion

HIVE-COTE version 2.0 is a meta ensemble of four very different classifiers, each of which is designed to capture different discriminatory features. It represents a new state of the art in terms of time series classification, significantly outperforming the previous best on both univariate and multivariate problems in terms of accuracy. Our ablative study showed that HC2 is better than any one of its constituents, and that each component makes a significant contribution to the overall performance. We believe its strength lies in the fact that many problems have discriminatory features in multiple data

domains; a shapelet might be indicative of one class value, whereas a repeating pattern may characterise another. HC2 uses a simple yet highly effective ensemble scheme to combine this information which we demonstrated was significantly better than alternatives such as stacking or a selection strategy.

HC2's weakness is that it does not scale well to very large problems. We showed that for problems with thousands of series of length in the tens of thousands, build times can be excessive, but that with contacting an estimate be obtained at least. We note that the current state of the art has similar limitations. Even ROCKET, which is by far the fastest algorithm, can struggle to scale in terms of memory with increasing number of cases.

There is room for further improvement with HC2. The STC design remains the same in HC1 and HC2, and we believe there are ways it could be improved. Contracting could be enhanced so that components could produce bespoke estimates of the time, and reconfigure themselves more intelligently to make best use of the available run time. Individual components could be threaded. Variability in estimates from the train data could be incorporated into ensemble process, and weights per case might improve predictions.

Over the last six years we have developed COTE classifiers using the UCR data sets for evaluation. There is a danger that we have simply overfit the archives. We believe the diversity and number of datasets used, and the repeated resampling, make this unlikely. However, it is a genuine concern, in particular for future development. We are compiling new datasets for both archives, and once this process is complete we will repeat all experiments with new data. However, we need to wait until the new archive datasets are publicly available to avoid any suspicion of selection bias. Hence, we classify this as future work.

HC2 is available in two open source toolkits and has improved usability features such as contracting, which allow the user to specify an approximate maximum run time. Our experiments are easily reproducible, and an accompanying website contains complete results and more information on how to use HC2.

Availability of data and material (data transparency) All time series classification data is available from www.timeseriesclassification.com. Our results are available at the supporting website www.timeseriesclassification.com/HC2.php.

Code availability (software application or custom code) The time series classification code is available at https://github.com/alan-turing-institute/sktime (Python) and https://github.com/uea-machine-learning/tsml (Java). Both of these toolkits are under constant development, so we have frozen a branch containing the code used in our experiments specifically for this paper: https://github.com/uea-machine-learning/tsml/tree/HC2.

Declarations

Conflict of interest All the authors declared that they have no conflict of interest.

References

Arul, M., & Kareem, A. (2021). Applications of shapelet transform to time series classification of earthquake, wind and wave data. *Engineering Structures, 228*, 111564.

Bagnall, A., Dau, H., Lines, J., Flynn, M., Large, J., Bostrom, A., Southam, P., & Keogh, E. (2018). The UEA multivariate time series classification archive, 2018. ArXiv e-prints arXiv:1811.00075.

Bagnall, A., Flynn, M., Large, J., Lines, J., & Middlehurst, M. (2020). On the usage and performance of HIVE-COTE v1.0. In *Proceedings of the 5th workshop on advances analytics and learning on temporal data, lecture notes in artificial intelligence* (Vol. 12588).

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.

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.

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.

Cabello, N., Naghizade, E., Qi, J., & Kulik, L. (2020). Fast and accurate time series classification through supervised interval search. In *Proceedings of the IEEE international conference on data mining*.

Caruana, R., & Niculescu-Mizil, A. (2004). Ensemble selection from libraries of models. In *Proceedings of the 21st international conference on machine learning*.

Chaovalitwongse, W. A., Prokopyev, O. A., & Pardalos, P. M. (2006). Electroencephalogram (EEG) time series classification: Applications in epilepsy. *Annals of Operations Research, 148*(1), 227–250.

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.

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.

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.

Guillaume, A., Vrain, C., & Wael, E. (2020). Time series classification for predictive maintenance on event logs. arXiv preprint arXiv:201110996.

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.

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 *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)* (Vol. 2, pp. 2169–2178). IEEE.

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. (2016). HIVE-COTE: The hierarchical vote collective of transformation-based ensembles for time series classification. In *Proceedings of 16th IEEE international conference on data mining*.

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. et al. (2019). Proximity forest: An effective and scalable distance-based classifier for time series. *Data Mining and Knowledge Discovery, 33*(3), 607–635.

Middlehurst, M., Large, J., & Bagnall, A. (2020a). The canonical interval forest (CIF) classifier for time series classification. In *2020 IEEE international conference on big data (big data)* (pp. 188–195). IEEE.

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*.

Middlehurst, M., Vickers, W., & Bagnall, A. (2019). Scalable dictionary classifiers for time series classification. In *Proceedings of intelligent data engineering and automated learning, lecture notes in computer science* (Vol. 11871, pp. 11–19).

Nguyen, T. L., 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.

Potamitis, I. (2014). Classifying insects on the fly. *Ecological Informatics, 21*, 40–49.

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, A. P., 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. (2017a). Fast and accurate time series classification with WEASEL. In *Proceedings of the ACM on conference on information and knowledge management* (pp. 637–646).

Schäfer, P., & Leser, U. (2017b). Multivariate time series classification with weasel+ muse. arXiv preprint arXiv:171111343.

Shifaz, A., Pelletier, C., Petitjean, F., & Webb, G. (2020). TS-CHIEF: A scalable and accurate forest algorithm for time series classification. *Data Mining and Knowledge Discovery, 34*, 742–775.

Shokoohi-Yekta, M., Hu, B., Jin, H., Wang, J., & Keogh, E. (2017). Generalizing DTW to the multi-dimensional case requires an adaptive approach. *Data Mining and Knowledge Discovery, 31*(1), 1–31.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going deeper with convolutions. In *The IEEE conference on computer vision and pattern recognition (CVPR)*.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Authors and Affiliations

Matthew Middlehurst[1] · **James Large[1]** · **Michael Flynn[1]** · **Jason Lines[1]** · **Aaron Bostrom[1]** · **Anthony Bagnall[1]**

Matthew Middlehurst
m.middlehurst@uea.ac.uk

James Large
james.large@uea.ac.uk

Michael Flynn
michael.flynn@uea.ac.uk

Jason Lines
j.lines@uea.ac.uk

Aaron Bostrom
a.bostrom@uea.ac.uk

[1]   School of Computing Sciences, University of East Anglia, Norwich, UK