Time Series Classification with HIVE-COTE: The Hierarchical Vote Collective of Transformation-based Ensembles

JASON LINES, University of East Anglia
SARAH TAYLOR, University of East Anglia
ANTHONY BAGNALL, University of East Anglia

A recent experimental evaluation assessed 19 time series classification (TSC) algorithms and found that one was significantly more accurate than all others: the Flat Collective of Transformation-based Ensembles (Flat-COTE). Flat-COTE is an ensemble that combines 35 classifiers over four data representations. However, while comprehensive, the evaluation did not consider deep learning approaches. Convolutional neural networks (CNN) have seen a surge in popularity and are now state of the art in many fields and raises the question of whether CNNs could be equally transformative for TSC.

We implement a benchmark CNN for TSC using a common structure and use results from a TSC-specific CNN from the literature. We compare both to Flat-COTE and find that the collective is significantly more accurate than both CNNs. These results are impressive, but Flat-COTE is not without deficiencies. We significantly improve the collective by proposing a new hierarchical structure with probabilistic voting, defining and including two novel ensemble classifiers built in existing feature spaces, and adding further modules to represent two additional transformation domains. The resulting classifier, the Hierarchical Vote Collective of Transformation-based Ensembles (HIVE-COTE), encapsulates classifiers built on five data representations. We demonstrate that HIVE-COTE is significantly more accurate than Flat-COTE (and all other TSC algorithms that we are aware of) over 100 resamples of 85 TSC problems and is the new state of the art for TSC. Further analysis is included through the introduction and evaluation of 3 new case studies and extensive experimentation on 1000 simulated datasets of 5 different types.

CCS Concepts: • **Computing methodologies** → **Machine learning**; **Supervised learning by classiftcation**; **Ensemble methods**;

Additional Key Words and Phrases: time series classification, heterogeneous ensembles, meta ensembles, deep learning

1 INTRODUCTION

Time series classification (TSC) problems arise across a rich and diverse range of domains. We may consider any ordered data to be a time series, which allows the definition to encompass data from various fields such as finance, biology, medicine, and engineering. The diversity of such data is easily apparent when visiting the University of California, Riverside/University of East Anglia (UCR/UEA) time series classification repository[1] (Bagnall et al. 2016). The UCR/UEA datasets consist of 85 varied and freely available problems that are used throughout the TSC literature.

Given the ubiquitous nature and easy availability of data, many researchers have proposed algorithms for solving TSC problems. The greatest research emphasis has been focused on classifying problems in the time domain through using the raw series, typically defining new elastic distance measures to couple with nearest neighbour classifiers (Jeong et al. 2011;Marteau2009;Stefan et al . 2013). Other approaches have also been proposed however, including dictionary and interval-based techniques (Deng et al. 2013;Lin et al . 2012;Schäfer2015;Ye and Keogh2011), ensemble algorithms (Bagnall et al. 2015;Lines and Bagnall2015), and transformation-based approaches (Bagnall et al . 2012;Górecki and Łuczak2014;Hills et al.2014;Kate2016).

The wealth of solutions that one could choose from when addressing a new TSC problem raises the question of which technique(s) should be considered? A recent empirical evaluation was carried out by (Bagnall et al. 2016) where 19 published TSC algorithms were implemented and tested in a common framework[2]. Experimentation was extensive; each algorithm was tested over 100 resamples of the 85 UCR/UEA datasets. The results showed that, while there are many competitive TSC algorithms with their own merits, one approach significantly outperformed all others in terms of average classification accuracy. This approach, the Collective of Transformation-based Ensembles (COTE) (Bagnall et al. 2015), combines classifiers built on four alternate representations of TSC problems, where the most effective ensembling strategy was found to combine all classifiers into a fiat hierarchy (Flat-COTE).

While this study evaluated the leading TSC algorithms that had previously been evaluated on the UCR/UEA datasets, it did not include any deep learning methods. Deep learning approaches have seen a recent surge in popularity in other fields, with convolutional neural networks (CNN) in particular garnering state-of-the-art results across tasks such as image processing, natural language processing, and speech recognition (Graves et al. 2013;Kalchbrenner et al . 2014;Krizhevsky et al . 2012). It is natural to ask whether CNNs could have such an impact on the field of TSC.

Our work seeks address two questions: first, is Flat-COTE more accurate than deep learning approaches for TSC? We implement a CNN using a common framework and conduct experiments on 85 datasets, and we also consider a recently published TSC-specific CNN implementation (Cui et al. 2016) with results over 44 datasets. We demonstrate that Flat-COTE is significantly more accurate than both deep learning approaches. However, despite its impressive performance, Flat-COTE has certain deficiencies. This leads to our second question: can we improve on Flat-COTE and define a better collective? We answer this by defining a new collective through a number of steps. First, we formally define a heterogeneous ensemble of standard classification algorithms (HESCA) that can be applied to both standard classification problems and TSC problems that have been transformed into alternative feature spaces. We demonstrate the merits of HESCA and justify its adoption through experimentation on 72 standard classification problems, subsequently applying HESCA to shapelet-transformed data in our new collective. Second, we significantly improve upon the current spectral approaches in Flat-COTE and replace them with a new random interval spectral ensemble (RISE). Third, we assimilate classifiers from two further data representations into the

[1]UCR/UEA TSC Repository:www .timeseriesclassification.com
[2]UEA TSC Code Base:bitbucket .org/TonyBagnall/time-series-classification

collective, and include the original whole series approach from Flat-COTE. Finally, we introduce a modular hierarchical structure for the collective with a probabilistic voting scheme that allows us to encapsulate classifiers built on each domain. The resulting meta-ensemble, the Hierarchical Vote Collective of Transformation-based Ensembles (HIVE-COTE), contains modules that capture similarity with whole series measures (Elastic Ensemble), phase-independent subseries (Shapelet Transform with HESCA), an interval-based ensemble (Time Series Forest), a dictionary ensemble (Bag-of-SFA-Symbols), and our new spectral ensemble, RISE. HIVE-COTE captures more sources of possible discriminatory features in time series and has a more modular, intuitive structure. More importantly, we show that HIVE-COTE is significantly more accurate than Flat-COTE over 100 resamples of 85 datasets and it represents a new state of the art for TSC. We compliment these results with a detailed analysis of HIVE-COTE and its individual modules over five types of simulated data and three new case studies. All of our code and data is available from a public code repository and accompanying website [3].

2 TIME SERIES CLASSIFICATION BACKGROUND

There have been many algorithms proposed in the literature for solving TSC problems. Due to the rich and diverse nature of these solutions, we believe that the best way to understand them is to group them by the type of discriminatory features that they use. Figure 1 includes generated series the exhibit typical discriminatory characteristics that we would expect for each type of similarity.

2.1 Whole series

Whole series techniques compare two series either as a vector (as with traditional classification) or by a distance measure that uses all data points. In the latter case, measures are typically combined with one-nearest neighbour (1-NN) classifiers and the simplest variant is to compare series using Euclidean Distance. However, this baseline is easily beaten in practice, and most research effort has been directed toward finding techniques that can compensate for small misalignments between series using **elastic** distance measures. The almost universal benchmark for whole series measures is Dynamic Time Warping (DTW) but numerous alternatives have been proposed. These involve alternative warping criteria (Jeong et al. 2011), using versions of edit distance (Marteau2009;Stefan et al. 2013) and transforming to use first order differences (Batista et al. 2014;Górecki and Łuczak 2014). The most accurate whole series approach in the recent experimental evaluation by (Bagnall et al. 2016) was the Elastic Ensemble (EE). (Lines and Bagnall2015) created an ensemble of 1-NN classifiers using various elastic measures, combining them through a proportional voting scheme. The example series in Figure 1b demonstrates a simple example of a whole series problem where class membership is defined by the presence of a global base shape, where the shape starts at a random location in the series and stretches between 10% and 100% of the full length. Measures such as DTW detect this type of similarity by mitigating against this phase-shift/warping.

2.2 Intervals

Rather than use the whole series, the interval class of algorithm select one or more phase-dependent intervals of the series. At its simplest, this involves a feature selection of a contiguous subset of attributes. However, the three most effective techniques generate multiple intervals, each of which is processed and forms the basis of a member of an ensemble classifier (Baydogan and Runger2016; Baydogan et al. 2013;Deng et al . 2013). Figure 1c demonstrates a simulated example where there are three discriminatory intervals within the problem, and class membership is defined by features extracted within each interval.

[3] www.timeseriesclassification.com/acm2017.php

2.3 Shapelets

Shapelet approaches are a family of algorithms that focus on finding short patterns that define a class and can appear anywhere in the series. A class is distinguished by the presence or absence of one or more shapelets somewhere in the whole series. Shapelets were first introduced by (Ye and Keogh2011). The two leading ways of finding shapelets are through enumerating the candidate shapelets in the training set (Hills et al. 2014;Lines et al . 2012) or searching the space of all possible shapelets with a form of gradient descent (Grabocka et al. 2014). The simulated example in Figure1d shows a problem where class membership is defined by an embedded shapelet. Unlike the previous interval example, the discriminatory feature may appear at any position within the series rather than being fixed to a constrained interval.

2.4 Dictionary-based

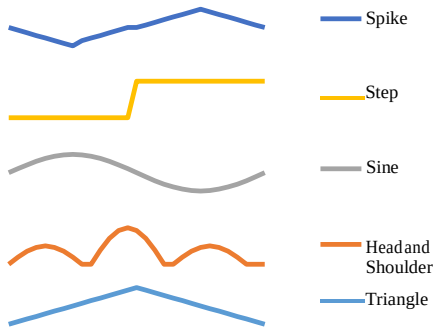
Shapelet algorithms look for subseries patterns that identify a class through presence or absence. However, if a class is defined by the frequency of a pattern, shapelet approaches will be poor. Dictionary approaches address this by forming frequency counts of repeated patterns. They approximate and reduce the dimensionality of series by transforming into representative words, then compute similarity by comparing the distribution of words (Lin et al. 2012;Schäfer2015). The simulated series in Figure1eis an example of a two class problem where instances of each class contain repetitions of the same two base shapes, but each class favours repetitions of one shape over the other. This type of data should confound shapelet classifiers as the same two shapes will appear in all series at least once.

2.5 Spectral

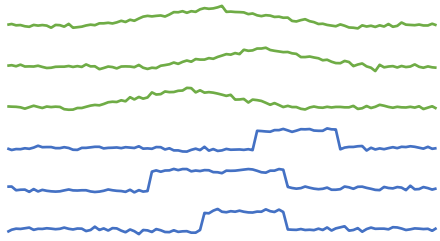
The frequency domain will often contain discriminatory information that is hard to detect in the time domain. Methods include constructing an autoregressive model (Bagnall and Janacek 2014;Corduas and Piccolo2008) or combinations of autocorrelation, partial autocorrelation and autoregressive features (Bagnall et al. 2015). To demonstrate such data, Figure1fprovides an example generated using two autoregressive models, one per class, embedded in noise. It is unlikely that the other groups of classifiers would be able to approximate this type of similarity as it is not clearly apparent in the time domain.

2.6 Combinations of the Previous and Ensemble Classifiers

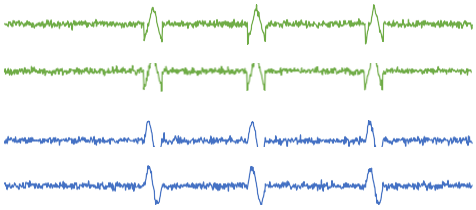
Two or more of the above approaches can be combined into a single classifier. For example, concatenating different feature spaces (Kate2016), forward selection of features for a linear classifier (Fulcher and Jones2014), and transformation into a feature space that represents each group and ensembling classifiers together (Bagnall et al. 2015). More generally, ensemble classifiers contain a set of base classifiers where individual predictions are combined through some process of fusion to classify new cases. A key design principle is to inject diversity amongst constituent classifiers; this is typically achieved by building duplicates of the same base learner and exposing each to different training conditions. For example, this could be achieved by training each on a different subset of attributes, building each with different subsets of the training data, modifying each through instance reweighting or internal randomisation, or a combination of these approaches. Examples in the literature include bagging (Breiman1996), which engenders diversity by boostrap sampling training data with replacement for each constituent classifier. Adaptive boosting (AdaBoost) (F and Schapire1996) iteratively reweights the sampling distribution of the training data based on training accuracies of learners at each iteration. Multiboost (Webb2000) uses a combination of ideas from bagging and boosting. Two highly-cited classification algorithms are themselves ensembles of



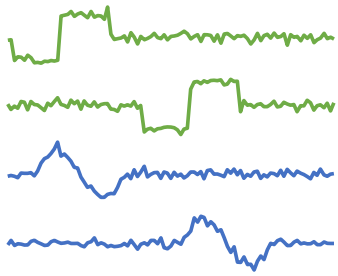
(a) Base Shapes



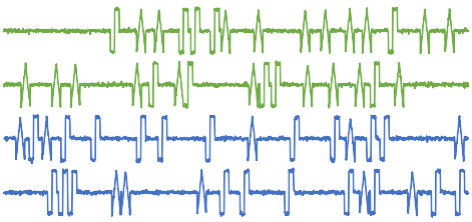
(b) Whole Series/Elastic



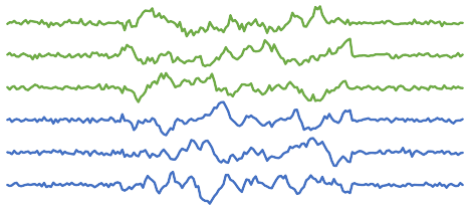
(c) Interval



(d) Shapelets



(e) Dictionary



(f) Spectral

Fig. 1. Simulated example series to represent each class of similarity. Each is generated with the base shapes in (a), and the process used to generate the series is described in more detail later in Section8.3. For clarity, we present these series with low noise added to aid interpretability; we add standard noise in Section8.3to increase the diKiculty of the problems when using them in experiments.

decision tree classifiers; Random Forest (Breiman2001) uses a combination of bagging and random attribute sampling to inject diversity; Rotation Forest (Rodriguez et al. 2006) uses all training data for each tree but partitions the attribute space and transforms the data using principle component analysis, We believe a straight-forward and effective technique for introducing diversity into an ensemble is to simply start with a pool of different classification algorithms as constituents. However, such heterogeneous ensembles are far less common in the literature.

2.7 Experimental Comparison of TSC Classifiers

The results from a recent experimental evaluation of the leading TSC algorithms from each of the previous groups by (Bagnall et al. 2016) are summarised in the critical difference (CD) diagram in Figure2. CD diagrams were introduced by (Demšar2006). They show the average ranks of multiple classifiers over multiple datasets and summarise a significance test between the ranks. The horizontal black bars are *cliques*; if two classifiers are in the same clique, their ranks are not significantly different (found using a Nemenyi post-hoc test). If they are not in the same clique, they are significantly different. The main conclusion of (Bagnall et al. 2016) is that Flat-COTE is significantly more accurate than all the other classifiers evaluated, representing the current state of the art for TSC.

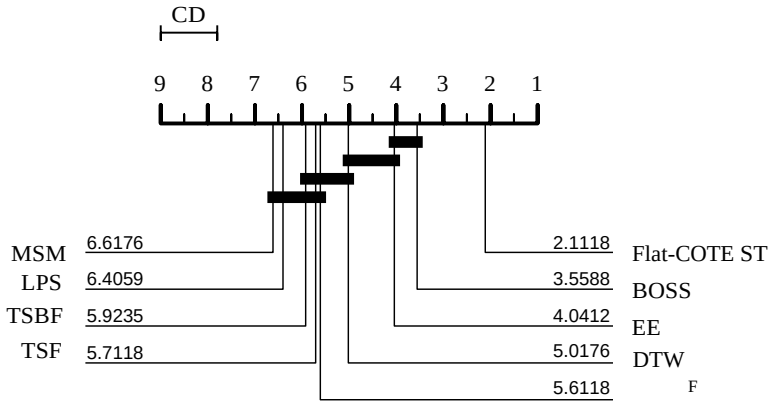


Fig. 2. The critical diKerence diagram of the top 9 classifiers taken from (Bagnall et al. 2016): Flat-COTE, ST (Shapelet Transform Ensemble), BOSS (Bag-of-SFA-Symbols), EE (Elastic Ensemble), DTWF (DTW Features), TSF (Time Series Forest), TSBF (Time Series Bag-of-features), LPS (Learned PaNern Similarity), and MSM (Move-Split-Merge).

3 FLAT-COTE

Figure3shows the overall structure of Flat-COTE and how it combines 35 classifiers into a single ensemble. It contains the constituent classifiers from the whole series TSC ensemble, EE (11 whole series classifiers), 8 classifiers built on shapelet-transformed problems, and 16 spectral classifiers (8 built on autocorrelation features, 8 using the power spectrum). Each classifier is built independently and produces separate training accuracies. Given a test instance, each individual classifier outputs a single class prediction. The prediction of each classifier is weighted by its training accuracy, and weighted test predictions are pooled. The class value with the highest combined vote is output as the prediction by Flat-COTE. This generic proportional ensemble scheme is outlined in Algorithm1.

Algorithm 1 ProportionalEnsemble(*classifiers*, *train*, *test*)

```
trainAccs = ∅;
for i ← 1 to |classifiers| do
    trainAccs_i = findTrainingAcc(train, classifiers[i])
    classifiers_i.buildClassifier(train)
testPreds = ∅
for i ← 1 to |test| do
    votes = ∅
    bsfWeight = 1;
    bsfClass = 1;
    for j ← 1 to |classifiers| do
        p = classifiers_j.classify(test_i)
        votes_p = votes_p + trainAccs_j;
        if votes_p > bsfWeight then
            bsfWeight = votes_p
            bsfClass = p
        testPreds_i = bsfClass
return testPreds
```

The results of the experimental evaluation by (Bagnall et al. 2016) demonstrated that Flat-COTE is significantly more accurate than any of the other algorithms that were evaluated. However, the conclusions give rise to several questions:

(1) Could a classifier from a different area of machine learning do better? The evaluation by (Bagnall et al. 2016) considered many such approaches, but an obvious omission were deep learning algorithms.

(2) Does the fiat structure cause a lack of robustness? For example, Flat-COTE contains 11 whole series classifiers, while the shapelet, ACF, and PS representations only have 8 each. This gives the classifiers taken from EE a higher weight in Flat-COTE. Also, EE contains full DTW and windowed DTW; in cases where the optimal window is 100%, these classifiers will be identical but have two votes out of the 35. Conversely, if we included a tree-based ensemble (such as TSF) with 500 classifiers, do we give it one compound vote, or 500 individual votes? Either is undesirable.

(3) Can we improve upon the current spectral methods used in Flat-COTE? The ACF and PS classifiers make up almost 50% of Flat-COTE, yet the features are deterministically linked and derived from the whole series.

(4) Can we create a better structure for the collective and include classifiers built on further domains? Flat-COTE only contains classifiers from three of the five groups in Section2, and the current structure within Flat-COTE makes it di@cult to fairly add new approaches.

We address these questions over the remainder of this paper through firstly comparing Flat-COTE to two deep learning approaches. After demonstrating that Flat-COTE is more accurate, we define a new collective: HIVE-COTE.

4 TSC DATASETS

4.1 UCR/UEA Time Series Dataset Repository

We use all of the 85 datasets currently in the UCR/UEA repository for the main experimental evaluation. These problems have been commonly adopted by TSC researchers and the datasets are split into pre-defined train/test partitions to allow reproducible research. However, always using the same splits risks overfitting on a single sample. As we are focused on the relative performance of classifiers, we adopt the same methodology as (Bagnall et al. 2016): we resample each dataset 100 times and report the average accuracies over 100 folds for a dataset. We seed the resample so

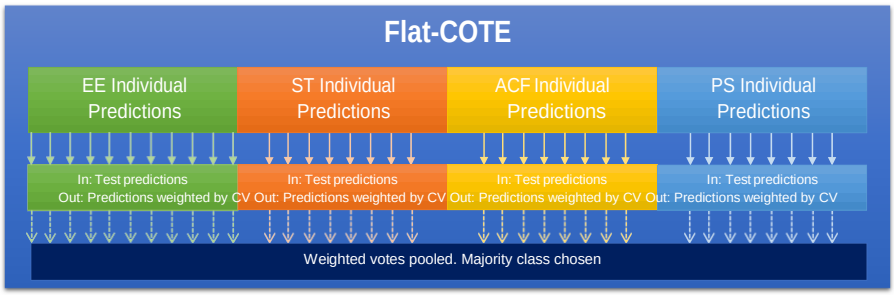


Fig. 3. A graphical representation of Flat-COTE. The classifiers from the four domains (time/whole series, shapelet, autocorrelation, power spectrum) are combined in a simple, flat structure. Each classifier is a module, and each module has a single weighted vote (35 classifiers/modules in total). Weighted votes are pooled together to select the class with the highest weight, which is subsequently used as the predicted class value by Flat-COTE.

that experiments can be reproduced exactly. We also introduce three new case studies and make the data freely available to other researchers.

4.2 Non-intrusive Detection of Ethanol Level in Alcohol

Up to 25% of licensed premises in some parts of the UK have been found to have counterfeit alcohol for sale. Brown-Forman, the company that makes Jack DanielâĂŹs, estimates that around 30% of all alcohol in China is fake. This is a health risk to the consumer as illegally produced spirits may contain contaminants such as methanol, and an economic risk due to the avoidance of taxes. Forgeries can sometimes be detected through external appearance such as poor labelling, but currently there is no way to conclusively tell whether spirits are forged without opening the bottle. However, the alcohol level of genuine spirits is tightly controlled and must equal the level stated on the bottle, but forgeries generally do not have this level of quality control. This means one way of detecting forgeries is by measuring the level of alcohol. Currently, this can only be done by taking a sample which is not feasible for widespread screening. We are investigating non-intrusive ways of testing the alcohol level using spectroscopy. We have conducted experiments using 20 different bottle types and four levels of alcohol: 35%, 38%, 40%, and 45%. Each series is a spectrograph of 1751 observations from bottles from 28 different bottle brands of whisky. Four resamples were taken from each bottle. To avoid experimental bias, we evaluate classifiers using a leave-one-bottle-out cross-validation.

4.3 Detecting the Occurrence of an Epileptic Fit through Motion Detection

(Villar et al. 2016) investigate whether an epileptic fit can be identified using accelerometer data attached to the wrist. They used six participants to complete ten repetitions of four different tasks: walking; running; sawing; and mimicking epileptic seizures. The mimicked seizures were trained and controlled following a protocol defined by a medical expert. Each patient wore a tri-axial accelerometer on the dominant wrist. Three acceleration components were recorded at 16 Hz. The duration of the experiments varied from 13 seconds to just over 4 minutes. We have formatted their data into an equal length time series classification problem. We sample a random segment of 13 seconds from the x-dimension of each series (length 208). We evaluate classifiers using a

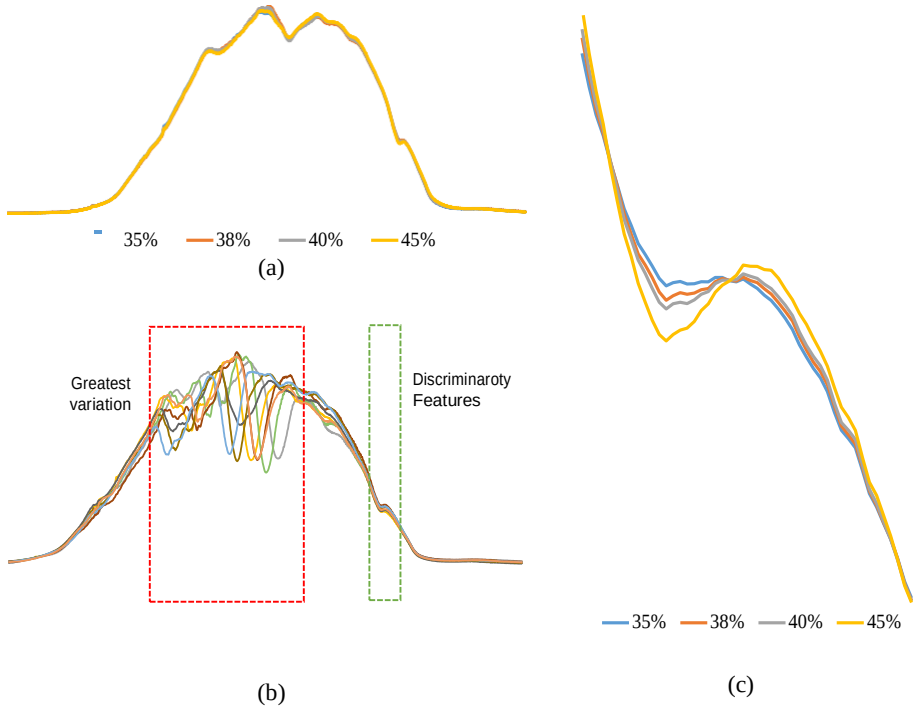


Fig. 4. Examples from the Ethanol Level problem. Part (a) shows there is liNle diKerence between the average profiles of each class, and (b) demonstrates the greatest variation between example series of diKerent classes seems to be in the middle of the series. Expert knowledge suggests that discriminartoy features will be in a band towards the end of the series however. Part (c) expands the interval from (a) and shows there is a clear diKerence beween the average class profiles when focusing only on this specific part of the series.

leave-one-person-out cross-validation, so that a repetition from a single person cannot be in both training and testing sets.

4.4 Vowel Classification from Raw Audio

The GRID audiovisual sentence corpus (Cooke et al. 2006) is a large multi-speaker collection of high-quality audio and video recordings of sentences by 34 speakers. Each sentence consists of six words, where the term at each position is taken from predefined groups: *command*, *colour*, *preposition*, *letter*, *digit*, and *adverb* (as shown in Table1). For example, a valid sentence could be 'place green at J 4 soon'. The groups labelled with an asterisk in Table1are keywords; each speaker recorded every possible combination of colours, letters, and digits. The remaining positions included random selections from the other word groups to create variation between speakers using the same set of keywords. This resulted in in 1,000 sentences per speaker (*w* was not recorded in the GRID corpus as it is the only multisyllabic English alphabetic character).

We create a TSC problem to identify which vowel was spoken from raw speech data. We used a single speaker (speaker 10 from GRID) and crop sentences to a window of 280 milliseconds (equivalent to 7 frames) centred on the 4^{th} term of the sentence, retaining cases only where the spoken letter was a vowel (5 class values). The audio is sampled at 50kHz and the constraint in GRID that all combinations of keywords must be recorded by each speaker guarantees that there will

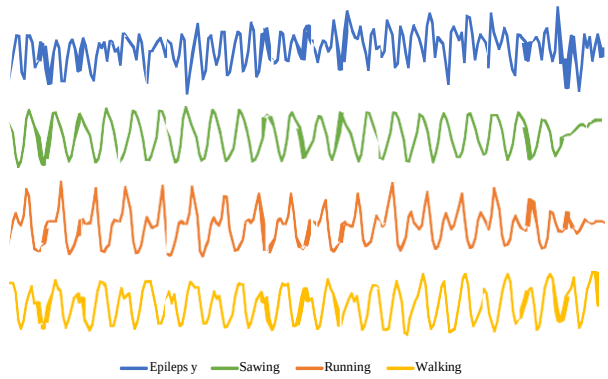


Fig. 5. Example series of each class from a single participant in the Epilepsy problem.

Table 1. The sentence structure in the GRID corpus (adapted from (Cooke et al. 2006)). Keywords are in italic font and identified by (*).

command	*colour(*)*	preposition	*letter(*)*	*digit(*)*	adverb
bin	blue	at	A-Z	0 (zero), 1-9	again
lay	green	by	(excluding w)		now
place	red	in			please
set	white	with			soon

be 40 utterances of each vowel by speaker 10 (as there are 4 colours and 10 digits). We randomise and stratify the data, splitting 50% into training data and 50% into a test set. Unlike Epilepsy and Ethanol, we are free to perform stratified resamples of the train and test partitions of the vowel problem without introducing bias because we use a single speaker.

We use a fixed window when creating the problem primarily to ensure all series are the same length (as per the UCR/UEA datasets), but this also allows us to introduce a degree of complexity into the problem. Utterances of the same vowel by the same speaker will include natural variation in length, as will the pauses before and after an utterance. Using a fixed window means that a case may or may not include the end of the leading preposition at the start of a series, or the start of the following digit at the end (or both). Additionally, using the full raw sound sampled at 50kHz over 280ms would create series with 14,000 attributes. Whilst this length is not infeasible, it is undesirable for the purpose of determining the relative performance of multiple classifiers on the problem, especially if we perform repeated resamples. Therefore, we downsample the data from 50kHz to 5kHz by retaining every 10^{th} reading. This introduces additional complexity as we lose information provided by the higher frequencies.

5 THE HETEROGENEOUS ENSEMBLE OF STANDARD CLASSIFICATION ALGORITHMS (HESCA)

Flat-COTE is an ensemble approach containing constituent classifiers built on representations of a problem in the time, shapelet, and spectral domains. This is the key principle of Flat-COTE; by determining the domain(s) where discriminatory features are more easily detected *a priori*

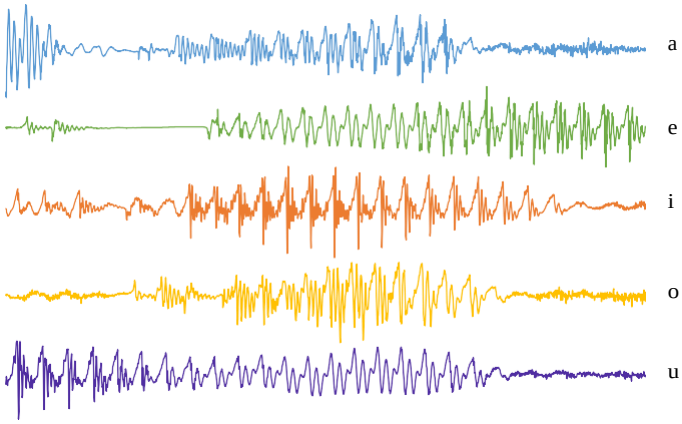


Fig. 6. An example series of each vowel afler downsampling the raw data by a factor of 10

in training, Flat-COTE can weight the contribution of its internal learners when producing test predictions. However, simply building models on various representations and favouring those in the correct domain is not enough if the initial classifiers are poor. Additionally, a further consideration is the quantity of classifiers built in each domain; Flat-COTE contains 11 time domain, 8 shapelet domain, and 16 spectral domain classifiers. While the empirical results of Flat-COTE demonstrate that the current configuration is effective, there is a clear imbalance between representations within the collective. It is unclear how Flat-COTE would be affected by including or removing constituent classifiers in any of the four domains, or by adding classifiers built in additional domains.

To mitigate these issues, it would be desirable to design a new collective using a meta-ensemble structure to encapsulate classifiers built on different representations into *modules*. Rather than contributing d individually weighted votes for a representation with d individual classifiers (as in Flat-COTE), the new collective would combine votes at a *module level* and include a single weighted vote for each module/representation. The rationale of why this is desirable is two-fold. First, as discussed in Section2.6, the effectiveness of ensemble approaches for classification is well documented in the literature. Second, by encapsulating classifiers for each domain into modules, any number of classifiers can be included within a module/representation without concern of causing an imbalance in the collective.

We can simply reuse the Elastic Ensemble (EE) (Lines and Bagnall2015) to encapsulate classifiers for whole series, but the solution is unclear for generated feature spaces (for example, shapelets). In addition to maximising accuracy, we wish to also obtain accurate estimates of test accuracy for modules to allow correct weighting within the collective. Subsequently, our choice of classifier is infiuenced by three factors: maximising accuracy, minimising error when estimating test accuracy, and minimising the variance of test estimates over multiple datasets. A tremendous amount of research has focused on ensemble design and techniques for diversifying identical base classifiers to inject diversity while maintaining accuracy. We take advantage of some of these advances, but an alternative means of increasing diversity is to use a pool of different base learners rather than diversifying a single base classifier. Transforming TSC problems into generated feature spaces creates standard classification problems, which allows us to leverage the plethora of classification algorithms that have been proposed in the literature. Given the abundance of algorithms at our disposal, using different algorithms within an ensemble seems to be the simplest way to inject

diversity but the use of heterogeneous ensembles is often ignored. Therefore we propose the Heterogeneous Ensemble of Standard Classification Algorithms (HESCA).

5.1 HESCA Constituents

HESCA includes eight constituent classifiers, two of which themselves are ensembles: *k* Nearest Neighbour; Naive Bayes; C4.5 decision tree; Support Vector Machines with linear and quadratic basis function kernels; Random Forest (with 500 trees); Rotation Forest (with 50 trees); and a Bayesian network. These classifiers are chosen to give us a balance between probabilistic, instance-based, and tree-based classifiers. HESCA is not specific to TSC and can be employed for any classification task.

A simple majority voting scheme is inappropriate for HESCA because it would not capture the relative performance of classifiers on any given dataset. Instead, each classifier is assigned a weight according to the same proportional voting scheme used in EE (as described in Algorithm1). An estimate of accuracy is obtained for each constituent by carrying out a 10-fold cross-validation experiment on the **training data only**, and these estimates are then used to weight internal test set predictions according to the procedure outlined in Algorithm1. All classifiers in HESCA are the standard WEKA implementations and we do not perform any explicit parameter optimisation.

HESCA is designed to achieve three goals: maximise test accuracy, minimise error between training CV estimates and test accuracies, and minimise variance in the error between training CV estimates and test results over multiple datasets. To test these three objectives we use 72 classification problems from the University of California, Irvine (UCI) Machine Learning Repository[4]. Specifically, we use the version of the data provided by (Fernández-Delgado et al. 2014) who originally converted 125 UCI datasets into real-valued problems and carried out one of the largest machine learning experimental studies to compare a range of classifiers. We have selected the 72 problems that have at least 10 attributes because we wish to only focus on problems with a high dimensional feature space. To match their experimental procedure, for each dataset we performed 100 jackknife samples with 30% of cases used for training and 70% for testing and summarise our findings in the following section by averaging accuracies across the 100 samples. Full results and code to recreate all experiments can be found on the accompanying website for this paper.

HESCA Accuracies on Test Data

We evaluate the test performance of HESCA by comparing results against those of each individual constituent classifier. We present the results of multiple classifiers over multiple datasets using CD diagrams, as previously introduced in Section2. However, we make a slight alteration to the method of forming cliques. Following recommendations in (Benavoli et al. 2016) and (García and Herrera2008), we have abandoned the Nemenyi post-hoc test originally used by (Demšar2006) to form cliques. Instead, we compare all classifiers with pairwise Wilcoxon signed rank tests, and form cliques using the Holm correction, which adjusts family-wise error less conservatively than a Bonferonni adjustment. It is worthwhile noting that cliques formed this way do not necessarily refiect the rank order. For example, if we have three classifiers (A, B, C) with average ranks $(A > B > C)$, it is possible for A to be significantly worse than B but not significantly worse than C in pairwise tests. This relationship cannot easily be displayed on a CD diagram. Happily, we did not encounter this phenomena with any of the results in this paper and are able to use this new method for calculating CD diagrams with the original presentational format throughout the remainder of this paper.

[4]The UCI Machine Learning Repository:http://archive .ics.uci.edu/ml/

NB	7.3247	2.3312	HESCA
C45	6.7662	2.6234	RotF
BN	6.6234	3.3182	RandF
NN	5.8312	5.026	SVMQ
		5.1558	SVML

Fig. 7. Average ranks over 72 datasets of 8 standard classifiers and HESCA, a weighted ensemble of the other 8.

The results in Figure7determine that we can reject the hypothesis that there is no difference between the classifiers. HESCA is significantly more accurate than all constituents over the 72 datasets, with the exception of rotation forest (RotF). These results make a case for using either HESCA or rotation forest in our new collective, but it raises the question of why would we use HESCA over just using a rotation forest? Our hypothesis is that the diversity of the different base learners within HESCA will reduce the error between estimates made from the training data and actual test accuracies, and also reduce the variance of this error over multiple datasets. Both are crucial factors when weighting modules correctly in the new collective and further investigation is necessary.

5.2 Variation Between Training Estimates and TestAccuracy

The experiments in the previous section produced 100 test set accuracies for each classifier over the 72 datasets. Reusing this experimental setup allows us to also carry out a 10-fold cross-validation (CV) on the training data for each classifier and sample to obtain an estimate of the expected test accuracy a priori and compare to the actual test accuracy. Table2summarises the differences training CV and test accuracies for each classifier 9averaged over the 100 resamples of the 72 datasets).

A positive difference in Table2is caused by the test accuracy of a classifier being higher on average than the estimate formed on the train data. The first observation is that each of the classifiers has a positive mean difference that is significantly different to zero (when tested by both a sign test and sign rank test). This means that each classifier is underestimating test set accuracy from the training data, which we believe is due to using a relatively small 30% training split for each resample.

The second point of note is the relationship between test accuracy and the error between training CV and test accuracy. The weaker classifiers in terms of accuracy (NN, C45 and SVMQ) have lower average train/test differences than most of the more accurate classifiers (RandF and RotF). The exception is HESCA, which is both accurate and has a relatively small train/test difference. More concretely, there was no significant difference in test accuracy between HESCA and rotation forest found in the previous experiment, but there is a significant difference in the mean train CV/test difference between HESCA and rotation forest (using F-Tests at 5%). This indicates that HESCA is more consistent in its estimation of test accuracy, and suggests that within HESCA the weaker

Table 2. A summary of the diKerences between training and test accuracy for 9 classifiers on 100 resamples of 72 datasets. A positive value indicates that on average over all folds and datasets, the test accuracy was higher than the train accuracy.

	Mean Diff. (%)	Median Diff. (%)	Std Error	Train < Test
NN	0.4997	0.1691	0.2012	52
HESCA	0.6106	0.1354	0.2680	54
SVMQ	0.6554	0.1704	0.2901	53
C45	0.6967	0.2388	0.2185	54
RotF	0.8940	0.1578	0.3208	58
NB	1.1262	0.1055	0.5345	49
RandF	1.1312	0.2075	0.3717	60
BN	1.2387	0.1197	0.4971	49
SVML	1.4758	0.1556	0.6696	48

classifiers are mitigating some of the errors made by rotation forest alone. The diversification within HESCA is is not enough to significantly increase accuracy on these problems, but it does significantly improve estimates made on training data and also reduces the variance in these estimates. Both of these are key priorities for us and justify using HESCA over rotation forest in our new collective. However, we do not wish to overstate these results as the beneficial effects are small, and there are many unanswered questions. For example, we could include additional algorithms, optimise parameters in training, etc. This is beyond the scope of this work however; our objective is to develop a robust classification scheme by balancing accurate predictions with accurate training estimates. We observe that in relation to its best components, HESCA is not less accurate, has lower average difference between train CV and test accuracies, and the variation of this difference is significantly lower.

5.3 HESCA for TSC

We have demonstrated the utility of HESCA using the UCI datasets, rather than the UCR/UEA datasets, as each constituent is designed for use with standard classification problems. However, our intention is to use HESCA within our new collective for TSC problems that have been transformed from the time domain into generated feature spaces. To demonstrate that the strengths of HESCA hold for transformed TSC problems, Figure8shows the critical difference diagram between HESCA and each of its constituents over 100 resamples of the 85 UCR datasets that have been transformed into the shapelet domain.

The results of HESCA with the shapelet-transformed problems (ST-HESCA) reinforces our previous findings; using shapelet-transformed data, HESCA is at least as accurate as any of its constituent parts. Further, with shapelet-transformed data, ST-HESCA is significantly more accurate than all of its component parts on the UCR/UEA datasets. We omit a detailed discussion between training CV and test accuracy here as it is redundant in the case where ST-HESCA significantly outperforms all of the alternatives, but we note that same relationship between accuracies was observed as with the UCI datasets (full results are available online from the website accompanying this paper).

These results give a clear indication that HESCA should be used in our new collective for shapelet-transformed problems. We could also use HESCA with problems transformed into the spectral domain, but we believe that there is a more effective approach than using the whole series

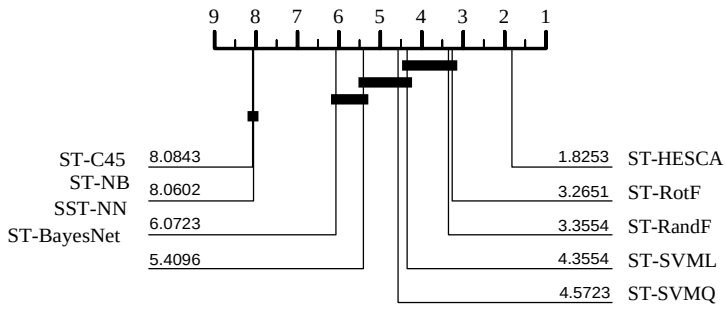


Fig. 8. Average ranks of HESCA and its constituent classifiers over 100 resamples of the 85 UCR datasets transformed into the shapelet domain.

to represent the data for this module. We discuss our novel spectral approach in the following section.

6 THE RANDOM INTERVAL SPECTRAL ENSEMBLE (RISE)

In many problem domains such as speech processing, discriminatory features are found in the frequency domain rather than the time domain. Common approaches for solving problems such as this involve using power spectrum or autocorrelation-based features (for example, (Bagnall and Janacek2014;Caiado et al . 2006)). This is represented within Flat-COTE by building 8 standard classifiers on truncated spectral features from the power spectrum (PS) and a further 8 classifiers on a combination of autocorrelation, partial autocorrelation, and autoregressive features (ACF). The results in (Bagnall et al. 2015) demonstrated that transforming data from the time domain to PS and ACF features worked well within Flat-COTE, but there are two limitations with this method.

First, the true autocorrelation function is the inverse of the power spectrum, so in many ways the two transforms are measuring the same thing (albeit at different resolutions). This means that almost half of the constituents in Flat-COTE are spectral-based and may be causing an imbalance within the collective. Second, the Flat-COTE approach for spectral classifiers uses the whole series for PS and ACF transformations. This may cause the obfuscation of embedded discriminatory features, especially for long series where spectral features may change over time. For example, speech processing approaches often use a 20 millisecond sliding window to generate a two dimensional spectrograph of frequency magnitude over time. This this approach is hard to generalise to TSC problems however as windowing massively increases the feature space and also introduces an additional parameter, which will require a further level of cross-validation to optimise. To overcome the increased feature space and the problem of setting the window size, we propose a new classifier: the Random Interval Spectral Ensemble (RISE).

RISE draws on ideas from tree-based ensembles such as random forest and the TSC interval feature classifier time series forest (TSF) (Deng et al. 2013). Like TSF, we build trees on random intervals from the data to construct a random forest classifier. A key difference however is that TSF uses time domain features by calculating the mean, variance, and slope of each interval, but RISE extracts spectral features over each random interval instead. We start by selecting 500 random intervals and calculate spectral features for each interval independently. We train a separate decision tree classifier on each set of features, then combine trees into a forest. The resulting ensemble classifier contains 500 base learners that are diversified through interval selection. Additionally,

the first tree in RISE is a special case that uses the whole series. The procedure for building RISE is outlined in Algorithm2.

Algorithm 2 BuildRISE(Training data *train*, number of classifiers *r*, minimum interval length *minLen*)

```
Let F =< F_1 . . . F_r >be the trees in the forest.
Let m be the length of series in train
wholeSeriesFeatures = getSpectralFeatures(train)
F_1.buildClassifier(wholeSeriesFeatures)
for i ← 2 to r do
    startPos = randBetween(1, m-minLen)
    endPos = randBetween(startPos + minLen, m)
    interval = train.removeAttributesOutsideOfRange(startPos, endPos)
    intervalFeatures = getSpectralFeatures(interval)
    F_i.buildClassifier(intervalFeatures)
```

There are many options for the spectral features that we could use within RISE. We could use PS features, ACF features, or a combination of the two (more details of alternative schemes are available in (Bagnall and Janacek2014)). We hypothesise that the best classifier will be produced through combining PS and ACF features for a single classifier (for context, the initial iteration of this work in (Lines et al. 2016) used only ACF features). We test this hypothesis using the 85 UCR/UEA datasets in the following experiments , but first demonstrate the intuitive strengths of RISE over current spectral methods in a motivational example using simulated data.

6.1 RISE vs. Current Spectral Methods

We run two sets of experiments to demonstrate that RISE produces a significantly more accurate classifier than the full series spectral methods, used either independently or in conjunction. We represent the current approaches by converting TSC problems into ACF and PS features and train a classifier on each; our findings in Section5naturally lead us to select HESCA. We design our experiments to test three hypotheses:

(1) RISE is not significantly less accurate than current spectral approaches when discriminatory features are located in the spectral domain over the whole series;
(2) RISE is significantly more accurate than the current spectral approaches when discriminatory features are located in the spectral domain over specific intervals;
(3) RISE is significantly more accurate than current spectral approaches on the UCR/UEA datasets.

We generate two synthetic problems to test hypotheses 1 and 2. First, we create a spectral problem using two autoregressive models of the type described in (Bagnall and Janacek2014) to represent classes in a binary problem where discriminatory features span the whole series. Second, we repeat this process, but embed the spectral features within a problem-dependent random interval surrounded by white noise. The intuition is that all spectral approaches will perform well on the first problem, but the white noise in the second problem will confound the whole series approaches and RISE will outperform competing techniques. We create both problems with 80 series of length 100 and partition 50% of the data for training. Similarly to Section5we report classification results for each classifier averaged over 100 runs, although we now generate independent data sets rather than resample the same dataset. We experiment with a range of spectral classifiers used in conjunction with HESCA: PS-HESCA, ACF-HESCA, and PSACF-HESCA. We report results over 100 runs, and also include a rotation forest and DTW 1-NN as benchmarks for comparison. The full results are available on the supporting website and are summarised in the boxplots in Figure9.

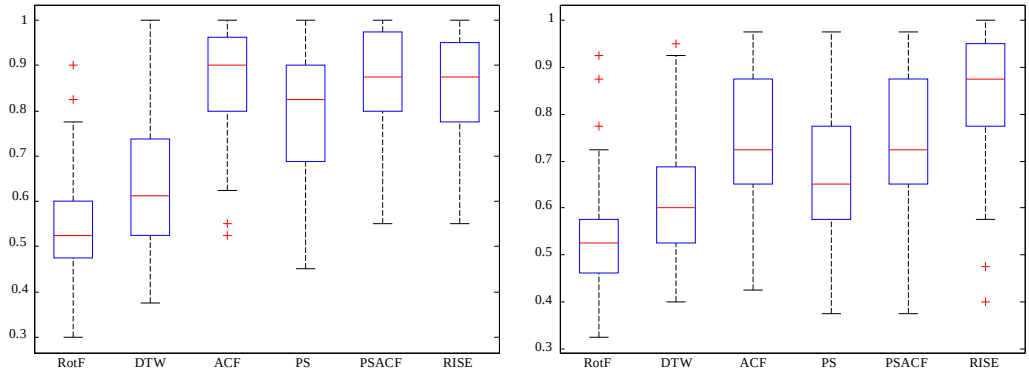


Fig. 9. Box plot of accuracies of six classifiers for 100 AR simulation experiments (a) and 100 AR simulation experiments embedded in noise (b).

The results support our first two hypotheses for RISE. First, there was no significant difference between RISE and any of the spectral-HESCA implementations on the whole series problem. The boxplot in Figure9ademonstrates that there was little difference in performance between RISE, ACF-HESCA and PSACF-HESCA, though PS-HESCA clearly did not perform as well as the other three spectral methods. This observation informally supports our intuition that including both PS and ACF features will produce the best spectral classifier. Second, RISE was significantly more accurate than all of the spectral-HESCA approaches when discriminatory information isembedded in a random interval surrounded by noise (confirmed with rank-based pairwise tests). In fact, the boxplot in Figure9bshows that embedding the problem in noise had very little, if any, effect on the ability of RISE to detect discriminatory features as the classifier reported almost idential performance.

The results for RISE are promising. Using simulated problems, it appears that RISE fulfils our desired criteria and is robust when spectral features span either the whole problem or a specific interval. However, it is easy to provide experimental support for an algorithm using only problems that are designed to play to its strengths. These initial experiments confirm our intuitions about RISE under ideal circumstances, but we also use the 100 seeded resamples of the 85 UCR/UEA datasets to run experiments with RISE and the three variants of HESCA (ACF, PS, and PSACF). The full results are available on the website and are summarised in Figure10.

The results in Figure10demonstrate two points. First, RISE is significantly more accurate than HESCA with any combination of spectral features on the UCR/UEA data. This both reinforces the results in the previous experiment and confirms our third RISE hypothesis. Secondly, PSACF-HESCA significantly outperforms the other HESCA implementations. These results informally support our hypothesis that the best solution for spectral problems is to combine ACF and PS features into a single classifier, rather than just using one or the other. However, to confirm this finding is consistent within RISE, we implement variants of RISE using only ACF or PS features and also run them over the 100 resamples of the UCR/UEA datasets. We compare against the implementation of RISE from Figure10that included combined ACF and PS features. The final results are summarised in the pairwise critical difference diagram in Figure11. The results confirm that combining PS and ACF features leads to a significantly more accurate classifier than using only ACF or PS features.

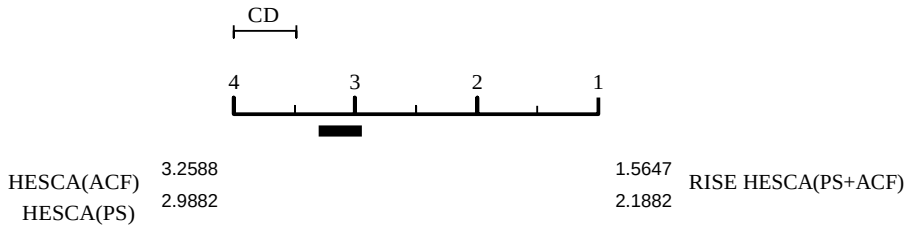


Fig. 10. A pairwise critical diKerence diagram of RISE and the full series spectral approaches over 100 resamples of the 85 UCR/UEA datasets

Further, RISE is significantly more accurate than our initial spectral approach in (Lines et al. 2016) which is equivalent to using RISE with ACF features only.

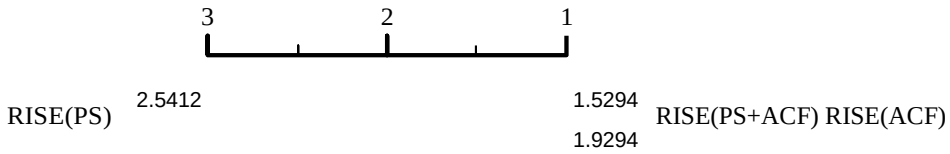


Fig. 11. A pairwise critical diKerence diagram to show the diKerence between implementations of RISE using all available spectral features and RISE using either ACF or PS features only.

7 A NEW COLLECTIVE: HIVE-COTE

We introduce a new version of COTE that we call the Hierarchical Vote Collective of Transformation-based Ensembles (HIVE-COTE). HIVE-COTE is an improved version of Flat-COTE that uses a modular hierarchical meta-ensemble structure. Given a problem where the classifiers across all four internal domains of Flat-COTE achieve similar training accuracies, the collective will be biased due to an uneven number of classifiers built in each domain. HIVE-COTE overcomes this potential design bias by modularising the elements of each group of classifiers. It allows only a single probabilistic prediction from each domain (whole series; interval; shapelet; dictionary; and spectral). The components of a module (ensemble of classifiers on a certain type) then becomes an encapsulated design decision. From the top level, it does not matter if a module contains one classifier or five hundred. The overseer simply defines how to combine module predictions into a single overall estimate.

7.1 Hierarchical Voting Structure

More formally, suppose we have g modules for a problem with C classes, where $|\Omega| = c$. Each module produces an estimate of the probability of the class variable y, $p_j(y = i)$ for $j = 1 \ldots g$ and $i = 1 \ldots c$. Each module has a weight w_j, which is an estimate of the accuracy on unseen data

formed from the train data (found through cross-validation). The collective probability is simply the normalised weighted sum over modules,

$$p(y = i) = \frac{\sum_{j=1}^{g} w_j \cdot p_j(y = i)}{\sum_{k=1}^{c} \sum_{j=1}^{g} w_j \cdot p_j(y = k)}.$$

The class prediction $\hat{y}$ is then just

$$\hat{y} = \arg\max_{i=1...g} p(y = i).$$

Flat-COTE treats each classifier as a single module, whereas HIVE-COTE uses each constituent ensemble as a module. This structure creates a more balanced and intuitive collective as each module corresponds to a different base ensemble, and encapsulating domain predictions into separate modules means that the number of classifiers built in each domain will no longer be a source of bias in HIVE-COTE.

7.2 HIVE-COTE Modules

HIVE-COTE contains five modules: the Elastic Ensemble for whole series similarity, HESCA trained with shapelet-transformed problems, two modules from other published research for interval and dictionary-based similarity, and our new spectral ensemble, RISE.

7.2.1 Elastic Ensemble (EE). EE, proposed by (Lines and Bagnall 2015), combines 1-nearest neighbour (1-NN) classifiers using various whole-series measures. The majority of research emphasis in TSC has been placed on defining similarity measures to couple with 1-NN classifiers. Given the wide choice in measures that could be used, a preliminary experiment by (Lines and Bagnall 2015) showed that there was no similarity measure that significantly outperformed all others when coupled with 1-NN classifiers. The results did however demonstrate that the classifiers made predictions in significantly different ways. Using this finding, EE was created to utilise the diversity between alternative elastic measures by building different 1-NN classifiers to combine into a proportional ensemble (using the same scheme outlined earlier in Algorithm 1), which was significantly more accurate than any of its constituent parts. All parameter settings for the measures are set through cross-validation on the training data.

7.2.2 Shapelet Transformed data with HESCA (ST-HESCA). The shapelet transform (ST) described by (Hills et al. 2014) separates shapelet discovery from the classifier by finding the top k shapelets from a single run, rather than through recursively searching to build a decision tree classifier as in the original shapelet implementation (Ye and Keogh 2011). Data are transformed with ST by using extracted shapelets, where attributes in a new instance are the distances from an input series to each shapelet. We use the most recent version of ST (Bostrom and Bagnall 2015) that balances the number of shapelets per class and evaluates each shapelet on how well it discriminates a single class. It is possible to use any standard classification algorithm with problems after being processed by ST; given our findings in Section 5, we build HESCA with shapelet-transformed data (ST-HESCA).

7.2.3 Bag-of-SFA-Symbols (BOSS) Ensemble. The core process for dictionary methods involves forming words by passing a sliding window of length w over each series, approximating each window to produce l values, then discretising these values by assigning each a symbol from an alphabet of size α. BOSS, introduced by (Schäfer 2015), uses a truncated discrete Fourier transform to compress each window, then discretises through multiple coe@cient binning. The resulting distribution of words forms the basis for 1-NN classification and uses a bespoke non-symmetrical

distance function. BOSS also includes a parameter that determines whether the subseries are normalised or not. During the parameter search of window sizes, the BOSS ensemble retains all classifiers with training accuracy within 92% of the best. New instances are classified by a majority vote.

7.2.4 Time Series Forest (TSF). (Deng et al. 2013) proposed TSF, which overcomes the problem of the huge interval feature space by employing a random forest approach with summary statistics of each interval as features. Training a single tree involves selecting $\sqrt{m}$ random intervals, generating the mean, standard deviation, and slope of the random intervals for every series. Trees are trained on the resulting $3\sqrt{m}$ features and classification is by majority vote.

7.2.5 Random Interval Spectral Ensemble (RISE). The final module we include in HIVE-COTE is our new spectral ensemble, RISE. We use the combination of ACF and PS features as described in Section 6.

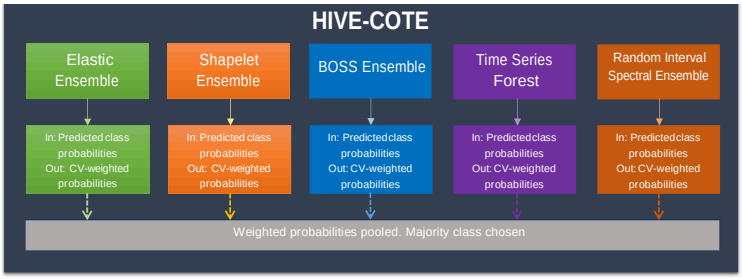


Fig. 12. A graphical representation of HIVE-COTE. The classifiers from the five constituent ensembles are combined using a probabilistic hierarchy. The key diKerence to Flat-COTE is that classifiers in each domain form a single module in HIVE-COTE, encapsulating predictions over each domain to create a more balanced structure.

7.3 Default Parameters

To remain consistent with the literature, we use the parameter setting configurations described in (Bagnall et al. 2016) for the modules in HIVE-COTE, with the exception of RISE which is new to this work. For simplicity, we set RISE to use the same number of intervals/trees as TSF. The list of modules and associated parameters are summarised in Table 3.

Table 3. The default parameters used to build HIVE-COTE modules. EE and BOSS use leave-one-out cross-validation (LOOCV), ST-HESCA uses 10-fold CV, and RISE and TSF do not require any parameters to be set in training.

Classifier	Parameters	Training Folds
EE	100 options for each constituent (full list available in (Lines and Bagnall 2015))	LOOCV
ST-HESCA	ST: l = from 3 to m; HESCA: default params as in Section 5	10-fold
RISE	r = **500**, random interval lengths from 16 to m	N/A
TSF	r = 500	N/A
BOSS	a = **4**, w from 10 to m with min(200, $\sqrt{m}$), $l \in$ **8, 10, 12, 14, 16**	LOOCV

7.4 Time Complexity

The time complexity of HIVE-COTE is no different to Flat-COTE. The complexity of HIVE-COTE is bounded by ST-HESCA due to the shapelet extraction procedure, and Flat-COTE also contains classifiers built on shapelet-transformed data. Therefore both collectives have the same time complexity: $O(n^2m^4)$.

Table 4. The training time complexities for the constituent modules in HIVE-COTE.

Classifier	Train Time	Parameters
EE	$O(n^2m^2)$	
ST-HESCA	$O(n^2m^4)$	
RISE	$O(knm^2)$	k : number of intervals
TSF	$O(rmn \log n)$	r : number of trees
BOSS	$O(nm(n-w))$	w : window length

There are two main criteria for comparing classification algorithms: test accuracy and time complexity. In this work we focus specifically on comparing classifiers based on test accuracy. It is certainly of interest to reduce the time complexity without reducing the accuracy of strong TSC algorithms, but there is little merit in accelerating poor models.

The objective of HIVE-COTE is to be significantly more accurate than Flat-COTE. It is trivial to make an algorithm faster if it is significantly less accurate (e.g. random guessing), but it is not trivial to make a significantly more accurate algorithm regardless of runtime. While runtime is an important (and sometimes crucial) factor for certain applications, this is not the motivation behind HIVE-COTE and is of secondary interest to us in this work. Our hypothesis is that HIVE-COTE will fulfil the first criteria and be significantly more accurate than Flat-COTE, and there will also be no significant difference between runtimes. Additionally, we have found through internal investigations that we can speed up both HESCA and the shapelet transform by at least an order of magnitude with no significant difference in accuracy through using simple heuristics. Therefore direct time comparisons between algorithms would also be misleading at this point. We defer the discussion of speedups to HIVE-COTE for future work, and focus specifically on test classification accuracy in the following experiments.

8 RESULTS

We report our experiments in four parts. First, we investigate how a benchmark CNN and a recently proposed TSC-specific CNN compare with the current state of the art for TSC, Flat-COTE. Second, we compare our new probabilistic hierarchical meta-ensemble, HIVE-COTE, to Flat-COTE and the other leading TSC algorithms. Third, we analyse the performance of HIVE-COTE and its constituent parts under various conditions using simulated data. Finally, we include in-depth results of three new case study problems using HIVE, RISE, and HESCA.

8.1 Flat-COTE and Deep Learning

The CNN and MCNN used by (Cui et al. 2016) were evaluated on 44 of the 85 UCR/UEA datasets. However, the CNN was used only as a comparison to MCNN and the actual results of the CNN were not published, while the published MCNN results were only recorded on roughly half of the UCR/UEA datasets. We wish to run our own standard CNN over the 85 problems as a benchmark to understand how the standard approach compares to other algorithms before comparing MCNN to the state of the art. We create CNNs in the Theano framework (Theano Development Team 2016) using stochastic gradient descent with momentum and one convolutional layer, followed by a max-pooling layer and three fully connected layers. Each convolutional/fully connected layer

contains 256 filters/units. The hyper-parameters (and the range of values we consider) that must be set are: the learning rate (0.1, 0.01, 0.001), filter size (0.05, 0.1, 0.2), pooling size (2, 3, 5), and the number of training epochs (50, 100, 200). We select parameters through minimising error in training, favouring smaller epochs in the event of ties to avoid overfitting. Figure13summarises the results of the CNN over the 85 datasets to the current state of the art, Flat-COTE, and the common benchmark of DTW 1-NN with warping set in training.

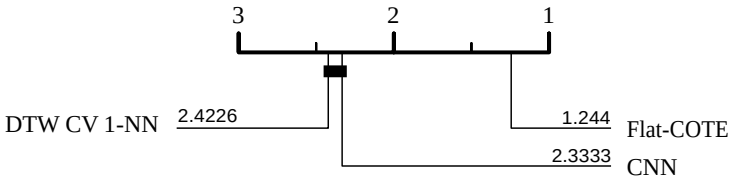


Fig. 13. Flat-COTE compared to DTW 1-NN and CNN on 85 UCR/UEA datasets.

Flat-COTE significantly outperforms the CNN on the 85 UCR/UEA datasets, while the average rank of CNN is not significantly different to DTW. This demonstrates that the standard CNN is at least competitive with the DTW benchmark, but cannot match Flat-COTE. However, the results reported by (Cui et al. 2016) stated that their MCNN significantly outperformed a standard CNN over 44 UCR/UEA datasets. We compare the published MCNN results to our CNN and also find a significant difference, so it is worthwhile also comparing MCNN to Flat-COTE. Over the 44 datasets that they report result for, Flat-COTE wins on 28 datasets, MCNN on 14, and they tie on 2. The difference is significant according to both a binomial test and a Wilcoxon signed-rank test.

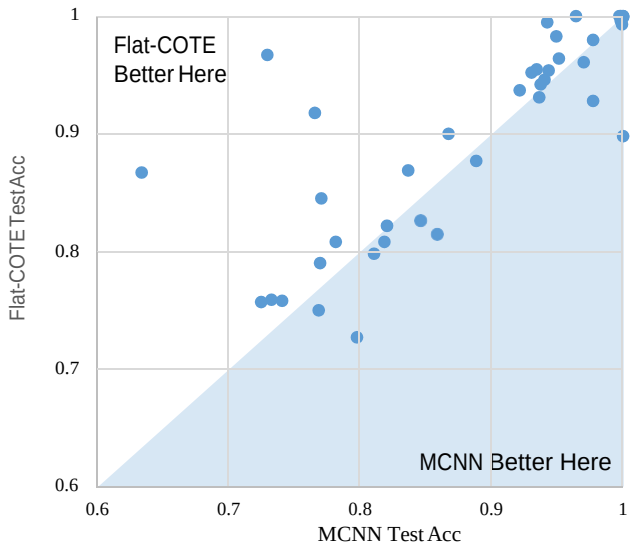


Fig. 14. A scaNer plot of the test accuracies over the 44 UCR datasets used by in (Cui et al. 2016) (two Flat-COTE wins omiNed for clarity). Flat-COTE wins on 28, MCNN wins on 14, and they tie on 2. The diKerence is significant. The average test accuracy of Flat-COTE over these problems is 88.6%, and MCNN is 86.5% (+2.1% in favour of Flat-COTE).

Though Flat-COTE is significantly more accurate than MCNN, comparing MCNN to DTW 1-NN over the 44 datasets finds a significant difference in favour of MCNN. This is still an impressive feat, as only a handful of algorithms evaluated by (Bagnall et al. 2016) actually outperformed DTW. It demonstrates promise, and warrants further investigation of deep learning applications to TSC. However, the current state of the art is confirmed to be Flat-COTE and our next objective is to evaluate whether HIVE-COTE is a significant improvement.

8.2 HIVE-COTE on UCR/UEA Datasets

Section2included a CD diagram summarising the relative performance of the top classifiers in the experimental evaluation by (Bagnall et al. 2016). These results were generated using 100 resamples of the 85 UCR/UEA datasets, where resamples were seeded to allow researchers to recreate the experimental procedure exactly. This allows us to run HIVE-COTE under the same conditions and update the evaluation to include our new collective. Table5and Table6report the 100-fold average classification accuracies for HIVE-COTE and the classifiers in (Bagnall et al. 2016) that significantly outperformed Rotation Forest and DTW 1-NN.

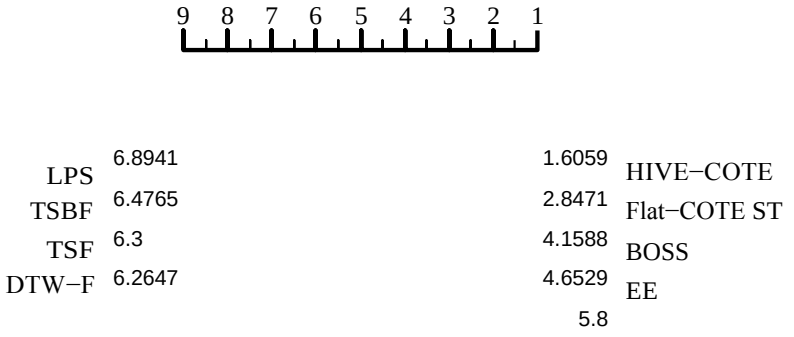


Fig. 15. The top TSC algorithms from (Bagnall et al. 2016) compared to HIVE-COTE using a pairwise CD diagram.

HIVE-COTE is significantly more accurate than all alternatives, including Flat-COTE. Against all algorithms, HIVE-COTE wins on 45 out of the 85 datasets and is ranked within the top 3 classifiers on 83 problems. This underlines the utility of transformation-based ensembles in general, but also demonstrates the effectiveness of the new hierarchical structure given the relative performance against Flat-COTE. We could not include MCNN in the full comparison as results were only published for a single train/test split for 44 of the 85 datasets. However, we can compare MCNN to HIVE-COTE on the default train/test splits for the 44 datasets. The result of this pairwise comparison is that HIVE-COTE wins on 30, MCNN on 11, and they tie on 3. The difference is significant and shown in further detail in Figure16.

8.3 HIVE-COTE on Simulated Datasets

In Section2we assigned TSC algorithms to five groups: whole series/elastic; interval; shapelet; dictionary; and spectral. We have designed HIVE-COTE to include a module to represent each group, and we posit that each will be optimal for data with different discriminatory characteristics.

Table 5. Test Results Part I: The average accuracy of the best classifiers from (Bagnall et al. 2016) and HIVE-COTE over 100 resamples of the UCR datasets (continued in Table6). The classifiers included are: ST-HESCA (Shapelet Transform data with HESCA), BOSS (Bag of SFA Features), DTWF (DTW Features), TSF (Time Series Forest), TSBF (Time Series Bag-of-features), LPS (Learned PaNern Similarity), EE (Elastic Ensemble), Flat-COTE, and HIVE-COTE. The best result for each dataset is highlighted in bold. HIVE-COTE wins on 45/85 datasets overall, and beats Flat-COTE on 69/85 problems in a head-to-head comparison.

Dataset	ST-HESCA	BOSS	DTW_F	TSF	TSBF	LPS	EE	Flat-COTE	Hive-COTE
Adiac	76.84	74.94	60.50	70.72	72.68	76.50	66.50	80.98	**81.54**
ArrowHead	85.11	87.52	77.58	78.94	80.09	80.63	85.97	87.68	**88.77**
Beef	73.57	61.50	54.60	64.77	55.43	51.97	53.20	**76.4**	72.27
BeetleFly	87.45	94.85	85.25	84.25	79.85	89.25	82.25	92.10	**95.9**
BirdChicken	92.70	**98.4**	86.50	83.85	90.20	85.40	84.80	94.10	95.05
Car	90.18	85.50	85.13	75.83	79.55	83.58	79.95	89.90	**92.55**
CBF	98.56	99.81	97.87	95.77	97.73	98.45	99.30	99.80	**99.94**
ChlorineConcentration	68.21	65.96	65.76	71.88	68.34	64.16	65.86	**73.56**	72.51
CinCECGtorso	91.83	90.04	71.42	97.37	71.64	74.28	94.59	98.27	**98.8**
Coffee	99.50	98.86	97.29	98.86	98.18	95.00	98.93	**99.96**	99.82
Computers	78.46	80.23	65.90	76.81	76.54	72.60	73.16	76.97	**81.94**
CricketX	77.71	76.36	76.92	69.14	73.06	69.60	80.11	81.40	**83.04**
CricketY	76.22	74.93	75.63	68.79	72.83	70.62	79.43	81.49	**83.7**
CricketZ	79.78	77.57	78.52	70.67	73.76	71.36	80.36	82.69	**84.77**
DiatomSizeReduction	91.12	93.94	94.22	94.13	89.03	91.45	**94.6**	92.48	94.19
D.PhalanxAgeGroup	81.94	81.41	75.96	81.29	81.17	74.15	76.80	82.12	**82.63**
D.PhalanxCorrect	**82.93**	81.46	79.62	80.95	81.56	76.74	76.83	80.46	82.49
D.PhalanxTW	69.04	67.30	65.83	68.57	69.04	61.83	65.41	69.32	**69.82**
Earthquakes	73.73	74.59	**74.75**	74.67	74.65	66.78	73.50	74.68	74.70
ECG200	84.02	**89.05**	81.85	86.82	84.68	80.75	88.10	87.29	88.19
ECG5000	94.34	94.05	93.96	94.39	93.78	91.73	93.86	94.61	**94.73**
ECGFiveDays	95.50	98.33	90.71	92.16	84.88	83.95	84.72	98.62	**98.95**
ElectricDevices	**89.54**	79.95	87.43	80.42	80.85	85.32	83.07	88.28	88.97
FaceAll	96.76	97.42	96.27	94.93	94.18	96.15	97.57	99.00	**99.63**
FaceFour	79.40	**99.56**	90.92	89.06	86.18	88.89	87.93	85.03	94.95
FacesUCR	90.93	95.06	88.91	89.68	84.93	91.00	94.83	96.66	**98.36**
FiftyWords	71.30	70.22	74.84	72.80	74.44	77.62	**82.07**	80.12	80.68
Fish	97.42	96.87	93.11	80.72	91.26	91.17	91.34	96.22	**97.62**
FordA	**96.54**	91.95	88.41	81.56	83.10	86.95	75.06	95.46	95.97
FordB	91.51	91.10	84.30	79.01	75.06	85.23	75.69	**92.86**	92.72
GunPoint	**99.87**	99.41	96.36	96.17	96.45	97.19	97.42	99.19	99.67
Ham	80.84	83.60	79.46	79.50	71.07	68.50	76.34	80.52	**84.08**
HandOutlines	**92.39**	90.28	91.53	90.86	87.95	86.75	88.01	89.39	91.20
Haptics	51.19	45.90	46.36	46.68	46.31	41.52	45.06	51.69	**53.03**
Herring	**65.34**	60.53	60.92	60.63	59.05	54.86	56.63	63.19	63.39
InlineSkate	39.30	50.27	38.22	37.85	37.71	44.87	47.64	**52.65**	52.59
InsectWingbeatSound	61.65	51.03	60.21	61.30	61.60	51.91	58.10	**63.89**	63.54
ItalyPowerDemand	95.31	86.60	94.79	95.76	92.62	91.40	95.08	**97.03**	96.78
LargeKitchenApps	**93.25**	83.66	82.30	64.44	55.06	67.97	81.58	90.00	92.27
Lightning2	65.89	81.00	71.02	75.67	75.98	75.66	**83.52**	78.48	79.70
Lightning7	72.44	66.56	67.12	72.26	68.03	63.08	76.34	79.95	**81.11**
Mallat	97.23	94.86	92.88	93.65	95.11	90.82	96.07	97.42	**97.55**
Meat	96.57	98.03	98.33	97.83	98.25	96.80	97.85	98.08	**98.72**
Average Accuracy	83.80	83.35	80.54	79.58	79.87	79.51	81.19	85.79	**86.92**
Average Rank	4.16	4.66	6.26	6.30	6.48	6.89	5.80	2.84	**1.61**

Furthermore, we hypothesise that HIVE-COTE will be as accurate, or at least not significantly worse, than the best approach for each data type. This is di@cult to evaluate with real data as there are many possible confounding factors and biases towards the type of data that can be readily collected. With this in mind we have created data simulators for each type of problem to test these hypotheses. Briefiy, each simulator places one or more of the shapes shown in Figure1ainto standard normal noise. The location, size, frequency, and/or type of shape defines the simulator. Examples generated by each simulator with very low noise were given as examples series in Section2. Examples of

Table 6. Test Results Part II (continued from Table5)

Dataset	ST-HESCA	BOSS	DTW_F	TSF	TSBF	LPS	EE	Flat-COTE	HIVE-COTE
MedicalImages	69.11	71.46	70.11	75.74	70.07	70.99	76.05	78.50	**81.54**
M.PhalanxAgeGroup	69.39	66.60	58.09	67.65	67.25	59.67	60.88	**72.23**	70.49
M.PhalanxCorrect	**81.51**	80.82	79.75	79.40	80.04	77.02	78.18	80.14	80.89
M.PhalanxTW	57.93	53.74	51.94	57.70	56.82	50.32	52.51	**58.69**	57.12
MoteStrain	88.23	84.60	89.08	87.44	88.64	91.65	87.51	90.16	**94.68**
NonInvasiveThorax1	**94.68**	84.09	87.67	88.05	84.24	80.69	84.87	92.92	93.17
NonInvasiveThorax2	**95.39**	90.36	89.79	91.44	86.21	82.59	91.39	94.61	95.23
OliveOil	88.07	87.00	86.37	88.30	86.43	89.17	87.90	**90.13**	89.77
OSULeaf	93.41	96.74	80.90	63.66	67.81	76.35	81.16	94.91	**97.05**
PhalangesCorrect	79.45	82.08	79.29	80.43	**82.54**	78.96	78.05	78.28	82.12
Phoneme	32.91	25.62	21.97	21.13	27.79	24.49	29.92	36.20	**38.55**
Plane	99.96	99.79	99.58	99.41	99.32	99.95	**100**	**100**	**100**
P.PhalanxAgeGroup	84.09	81.90	82.37	84.56	84.15	79.97	80.53	84.84	**84.85**
P.PhalanxCorrect	**88.09**	86.74	82.86	84.74	86.12	85.05	83.89	87.09	87.63
ProximalPhalanxTW	80.28	77.28	77.42	80.84	79.82	72.22	75.91	**81.47**	81.16
RefrigerationDevices	76.08	78.46	65.63	61.54	63.84	67.55	67.59	74.23	**80.1**
ScreenType	67.61	58.60	49.90	57.28	53.79	50.59	55.43	65.13	**71.1**
ShapeletSim	93.36	**100**	88.78	50.99	91.27	87.42	82.73	96.38	99.13
ShapesAll	85.42	90.88	79.59	80.01	85.35	88.45	88.59	91.07	**92.59**
SmallKitchenApps	80.25	75.02	75.29	81.26	67.37	72.37	70.33	78.78	**83.71**
SonyAIBORobot1	88.76	89.74	88.38	84.49	83.93	84.18	79.42	**89.91**	88.68
SonyAIBORobot2	92.44	88.77	85.86	85.60	82.54	85.10	87.01	**95.98**	94.54
StarlightCurves	97.74	97.76	95.96	96.87	97.77	96.83	94.15	97.96	**98.15**
Strawberry	96.84	**97.03**	96.98	96.27	96.81	96.35	95.88	96.31	96.98
SwedishLeaf	93.85	91.77	88.55	89.20	90.78	92.56	91.59	96.67	**96.86**
Symbols	86.16	96.12	93.00	88.76	94.43	95.98	95.74	95.27	**96.58**
SyntheticControl	98.69	96.79	98.58	99.03	98.65	97.16	99.40	99.92	**99.96**
ToeSegmentation1	95.40	92.88	92.20	66.10	85.82	84.12	78.76	93.37	**95.5**
ToeSegmentation2	94.72	95.97	90.38	78.24	88.58	92.64	90.70	95.15	**96.64**
Trace	99.99	99.99	99.74	99.80	98.14	96.65	99.60	99.99	**100**
TwoLeadECG	98.44	98.45	95.75	84.22	90.96	92.79	95.85	98.25	**99.35**
TwoPatterns	95.17	99.12	99.95	99.05	97.42	96.69	**100**	99.98	99.99
UWaveGestureAll	94.21	94.45	96.29	96.19	94.37	96.77	96.83	96.48	**96.98**
UWaveGestureX	80.59	75.32	80.63	80.64	83.44	81.87	80.47	83.05	**83.84**
UWaveGestureY	73.70	66.12	71.66	72.74	74.57	75.27	73.05	76.56	**77.55**
UWaveGestureZ	74.68	69.52	73.65	74.08	77.60	76.65	72.63	75.95	**77.83**
Wafer	**99.98**	99.90	99.62	99.65	99.61	99.51	99.69	99.94	99.97
Wine	**92.61**	91.17	89.20	88.06	87.93	88.43	88.67	90.35	91.20
WordSynonyms	58.24	65.88	67.42	64.35	66.90	72.81	**77.84**	74.81	74.80
Worms	71.95	**73.49**	67.34	62.79	66.84	64.16	64.44	72.51	73.40
WormsTwoClass	77.87	**80.97**	73.00	68.51	75.49	74.26	71.74	78.52	78.39
Yoga	82.25	90.99	86.35	86.70	83.45	87.37	88.54	89.78	**91.7**
Average Accuracy	83.80	83.35	80.54	79.58	79.87	79.51	81.19	85.79	**86.92**
Average Rank	4.16	4.66	6.26	6.30	6.48	6.89	5.80	2.84	**1.61**

the actual series that we have used in experiments are shown in Figure17alongside low-noise comparisons.

To simulate whole series elastic data, a single shape defines the class, and the shape is stretched between 10% and 100% of the total length for each series. For interval data, a single shape also defines the class, but we place several much shorter versions at fixed locations so that there is a high ratio of noise to shape. For shapelet data, a single short fixed-length shape defines the class, and the shapelet is placed at a random location for every series of each class. For dictionary data, we generate series using many repetitions of two shapes in random locations, and the discriminatory feature defining class membership is the frequency that each shape appears. Finally, for spectral problems we simulate data as described in Section6when motivating RISE. We create 200 two-class problems with each simulator and evaluate 10 TSC algorithms over the 1000 datasets. We include a brief summary of the simulation results, and a more detailed description of the simulators, parameter settings, and results is available in (Bagnall et al.2016).

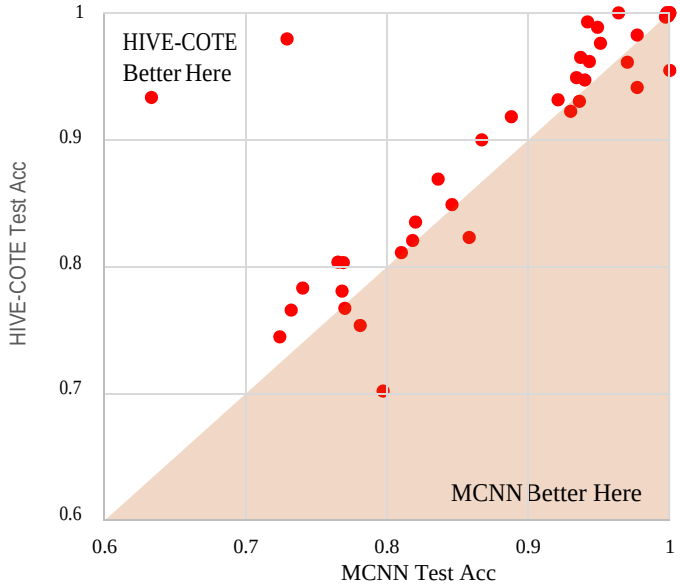


Fig. 16. A comparison of the test accuracies between HIVE-COTE and MCNN over the 44 UCR datasets used by (Cui et al. 2016). HIVE-COTE wins on 30, MCNN wins on 11, and they tie on 3. The diKerence is significant.

The classifiers we use are selected to be representative of each group of algorithm outlined in Section2. Five classifiers are the individual modules from HIVE-COTE, and the remaining five are included based on the experimental results in (Bagnall et al. 2016). First, we use two standard benchmarks: Rotation Forest (Rodriguez et al. 2006) with 50 trees (RotF) and Dynamic Time Warping (Ratanamahatana and Keogh2005) with window size set through cross-validation (DTW). Next, to represent whole series techniques we use EE and HESCA. For interval-based similarity we use TSF, and shapelet-based classification we use ST-HESCA. BOSS is used for dictionary-based similarity, and our new ensemble RISE is used for classification in the spectral domain. Finally, Flat-COTE and HIVE-COTE are both included to represent combinations of techniques. Our prior belief was that a classifier from a given group would perform significantly better than all other algorithms on data that was designed for that group. Specifically, we expected EE to be best on whole series/elastic simulations, TSF on interval simulations, ST-HESCA on shapelet simulations, BOSS on dictionary simulations, and RISE on spectral simulations. The goal of HIVE-COTE and Flat-COTE is to be able to dynamically adjust to each type of discriminatory feature and perform well across all simulations. The objective of these simulations is to answer three questions:

(1) Can the two COTE classifiers perform at least as well as the best in class classifier for a given problem type?
(2) Is HIVE-COTE significantly more accurate than other approaches when we do not know the origin of the data?
(3) Is HIVE-COTE significantly more accurate than Flat-COTE on these simulated TSC problems?

We have run 200 independent trials for each simulator by generating a random data set and performing a random stratified split into train and test data. Any parameter optimisation is performed

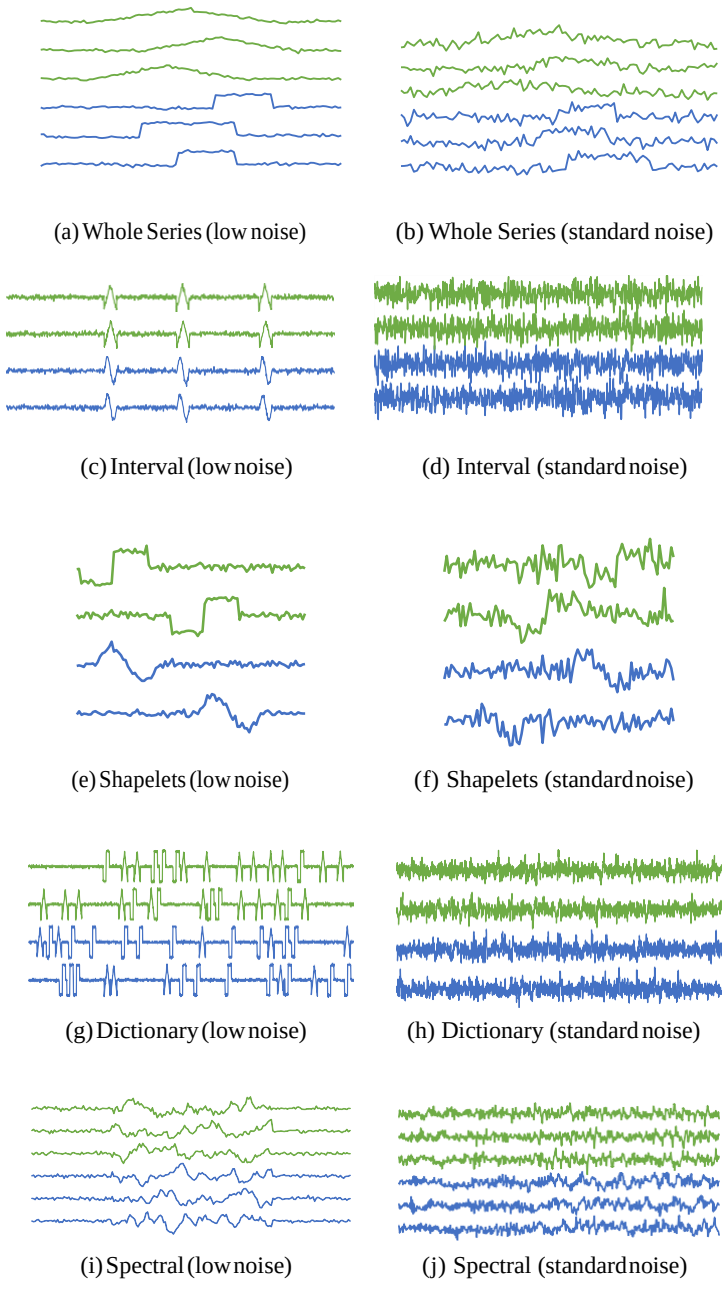


Fig. 17. Example simulated series. The left column includes examples with low noise (repeated from Section 2) and the right includes series generated with standard noise; the simulation experiments use standard noise series to increase the complexity of the problems.

on the training data and we report accuracy results on the test data. The mean ranks, and the relative ordering in brackets, are shown in Table7. The entries in bold are not significantly worse than the best classifier. We use a Wilcoxon sign-rank test with α adjusted to 0.005to compensate for multiple testing. It is worth noting that 200 is a relatively large sample size, and significance results reported with a sign-rank test are identical to those found with a sign test, a paired t-test, and a binomial test.

Table 7. Results of the simulation exeriments for all classifiers. Relative ranking of classifiers on each is shown in brackets. Results in bold font are not significantly worse than the best classifier on that simulated problem type.

	Elastic	Interval	Shapelet	Dictionary	Spectral
HIVE-COTE	**2.49 (1)**	3.06 (2)	**2.26 (2)**	**1.9 (1)**	2.91 (2)
Flat-COTE	3.16 (2)	3.5 (3)	**2.33 (3)**	2.55 (3)	2.99 (4)
ST-HESCA	6.50 (8)	3.67 (4)	**2.13 (1)**	4.96 (5)	2.98 (3)
RISE	4.71 (4)	8.25 (9)	5.7 (5)	4.34 (4)	**1.42 (1)**
BOSS	4.99 (7)	7.59 (8)	4.38 (4)	2.54 (2)	8.04 (8)
TSF	4.86 (5)	6.26 (7)	6.42 (7)	5.09 (6)	6.5 (6)
EE	4.65 (3)	5.52 (5)	6.39 (6)	7.75 (8)	5.78 (5)
DTW	4.87 (6)	9.85 (10)	6.97 (8)	7.21 (7)	6.8 (7)
HESCA	8.90 (9)	**1.65 (1)**	9.1 (9)	9.14 (9)	8.8 (10)
ROTF	9.88 (10)	5.68 (6)	9.33 (10)	9.52 (10)	8.79 (9)

Our first observation is that HIVE-COTE is either the best or not significantly worse than the best on three of simulations (Elastic, Shapelet and Dictionary). The two cases where it is significantly worst than the best are interval and spectral simulators. These are best approached with HESCA and RISE, but HIVE-COTE is the second best for each. This is broadly in line with our expectations, but there are some unexpected features in these results. We summarise these below and address these findings in more detail in (Bagnall et al.2016).

For elastic simulators, there is no significant difference between BOSS, DTW, TSF, RISE, and EE. When we ensemble with Flat-COTE (which includes constituents from ST-HESCA and spectral features) we get a significantly more accurate classifier, but HIVE-COTE (which also includes TSF and BOSS) is significantly more accurate than Flat-COTE. This implies that the accuracies of the constituents is to some degree inversely correlated, and that HIVE-COTE captures this diversity to find a better classifier overall. It contradicts our prior beliefs and was initially surprising. On refiection, we believe that this is caused by two confounding factors in the data: warping of the shape and random noise. DTW and EE can compensate for warping but are confounded by noise. BOSS, RISE and TSF all involve some form of averaging and smoothing and are thus better at coping with random noise. Our conclusion is that is dangerous to put too much credence on prior beliefs as to the best approach for a problem and that HIVE-COTE can to some degree remove the need for this possible source of bias.

The interval simulator results are also surprising. Our expectation that TSF and RISE would be the best was not born out by the results. The best classifier was in fact the ensemble of standard classifiers, HESCA. The poor performance of TSF and RISE is likely caused by using intervals that are too short. The relatively good performance of ST-HESCA is an artefact of the fact that each class only uses a single shapelet over all intervals. This simulator could be better designed for future experiments, as it is essentially just simulating a standard classification problem with a large number of redundant features. However, it does provide supporting evidence for HESCA, which is

significantly more accurate than any of its components. We believe that heterogeneous ensembles are an under-researched area. The interval simulation also reinforces our conclusion from the whole series data, as HIVE-COTE effectively captures the diversity of ST and EE to improve overall.

The shapelet and dictionary results broadly confirmed our prior belief. ST-HESCA and BOSS are the best approaches respectively, but HIVE-COTE is not significantly worse than either. The spectral results were a little unexpected, in that HIVE-COTE is unable to compensate for the confounding predictions of the components other than RISE. This suggests we should investigate alternative fusion strategies in cases where there is a strong bias towards a single module within the collective.

Our second question was whether HIVE-COTE is better when we do not know the origin of the data. Our hypothesis is that HIVE-COTE will significantly outperform the other approaches on data of unknown origin, doing so by steering classification decisions towards the correct domains through its internal modular hierarchical structure. We can design an experiment to test this hypothesis by randomly picking one of the five simulators and measuring performance over multiple samples. However, we can simply achieve this effect by combining the 200 results from the 5 individual simulator experiments into a single set of 1000 resamples. Figure18shows the pairwise critical difference diagram of the 10 classifeirs over the 1000 samples.

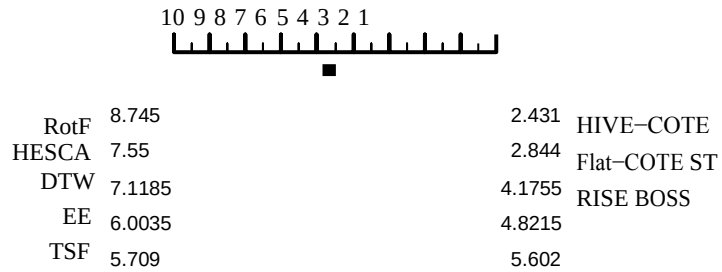


Fig. 18. Average ranks and cliques for ten classifiers over 1000 simulations.

The main claim of this paper is that HIVE-COTE is significantly more accurate than other classifiers, including Flat-COTE, for TSC problems. Figure18shows that both COTE approaches are significantly more accurate when data can be generated from any of the simulators and answers our second question in this section. Crucially however, these results also answer our third question, as HIVE-COTE is significantly more accurate than Flat-COTE on simulated problems when we do not know the source of the data. As is standard practice, we have focused exclusively on ranks for our comparative analysis. However, variation in ranks can often be represented by tiny differences in accuracy; this is not the case with these simulations. We summarise the variation in accuracy for the classifiers in a boxplot of accuracies over all 1000 experiments (Figure19). The plot demonstrates the relative stability of HIVE-COTE compares to the other algorithms.

8.4 Cases Studies

We have shown that HIVE-COTE represents a new state of the art for TSC on the UCR/UEA TSC problems, and we have also demonstrated how the collective and its internal modules perform on

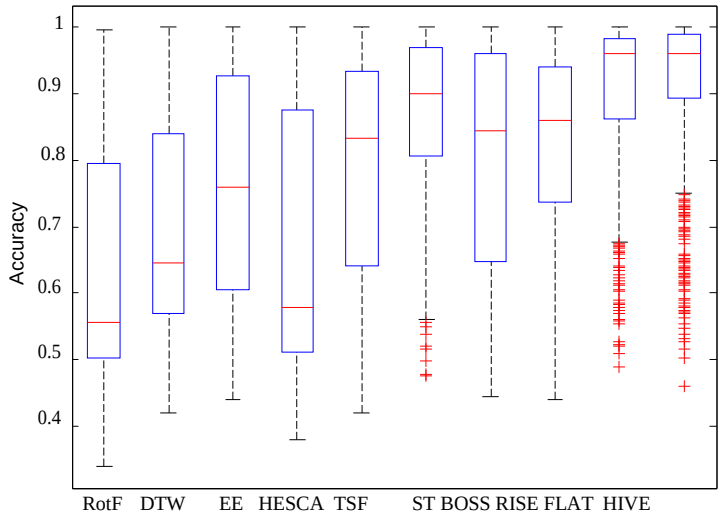


Fig. 19. Box plot of accuracies of ten classifiers over 1000 simulations.

five types of simulated problems. The following case studies demonstrate the versatility of HIVE-COTE on new problems where we have no prior experimental knowledge of the best approaches to use.

8.4.1 Non-intrusive Detection of Ethanol Level in Alcohol. Our first use case is the ethanol level problem described in Section4.2. Our prior beliefs were that whole series methods would be poor at this problem, as expert knowledge suggested that discriminatory information would only be contained within a small window at the end of the series. This would lead us to anticipate that shapelet and interval-based approaches would perform best. We designed a leave-one-bottle-out experiment to avoid bias, and built each of the classifiers used previously in the simulation experiments: HIVE-COTE, Flat-COTE, EE, ST-HESCA, TSF, BOSS, RISE, and we again include HESCA, Rotation Forest, and DTW 1NN (with window set through CV) as baseline comparisons. The average accuracy and standard deviation across all folds for each classifier is reported in Table8.

As anticipated, whole series approaches were poor; EE and DTW were no better than random guessing on this four class problem. The vector-based rotation forest and HESCA both performed very well. While this finding was initially suprising, it is perhaps to be expected given the static nature of data in spectral problems. There is no opportunity for discriminatory features to be shifted or occur in inconsistent locations when considering fixed wavelengths, making this TSC problem share many similarities with the format of standard classification problems. The best approach on this dataset however was ST-HESCA, though it was only marginally better than HIVE-COTE. HIVE-COTE itself was over 2% more accurate than Flat-COTE; this an interesting result as both HIVE-COTE and Flat-COTE contain the classifiers from ST-HESCA, which was suited to this problem, and EE, which was not suited to this problem. This is a good example of how the modular hierarchy in HIVE-COTE is robust when internal classifiers are not optimal for a problem, compensating much better than the fiat structure allows within Flat-COTE. Interestingly, the interval-based TSF did not perform well on this problem. We believe this is due to the simple nature

Table 8. A summary of the results of the leave-one-boNle-out ethanol level problem. Classifiers are presented in descending order of accuracy.

Classifier	Avg	Stdev
ST-HESCA	84.88%	0.096%
HIVE-COTE	84.37%	0.084%
RotF	84.17%	0.114%
HESCA	83.87%	0.112%
Flat-COTE	82.29%	0.078%
RISE	66.84%	0.112%
TSF	64.45%	0.160%
BOSS	51.98%	0.091%
EE	26.59%	0.068%
DTW	26.41%	0.069%

of the summary statistics that are calculated for each extracted interval, meaning the classifier cannot fully capture the discriminatory information in such a short band for this problem.

8.4.2 Detecting the Occurrence of an Epileptic Fit through Motion Detection. Our second case study is the epilepsy problem described in Section4.3. Our prior beliefs for this dataset were that this problem would be more easily approached than the ethanol level problem, as the data is seemingly suited to more approaches. We would expect dictionary, shapelet, and whole series approaches all to be able to discover discriminatory features in this problem. Therefore, we expect each to perform well, and the ensemble approaches to perform best through leveraging from the performance of the constituent parts. We performed a leave-one-person-out experiment and report the results in Table9.

Table 9. A summary of the results of the leave-one-person-out epilepsy problem. Classifiers are presented in descending order of accuracy.

Classifier	Avg	Stdev
HIVE-COTE	99.27%	0.010
Flat-COTE	99.27%	0.014
RISE	98.55%	0.024
BOSS	97.82%	0.041
ST-HESCA	95.64%	0.029
EE	94.91%	0.057
DTW	93.09%	0.064
RotF	89.45%	0.063
TSF	84.36%	0.143
HESCA	78.91%	0.116

As anticipated, many approaches reported strong results on this problem. HIVE-COTE and Flat-COTE were equally best, getting only 2/275 instances incorrect. A stand-out result however is RISE; the random spectral interval approach worked much better than anticipated on this problem. It is in direct contrast to the time domain-based interval approach, TSF, beating it by over 14% across all data. Though there is no clear discriminatory interval in the data, it would appear that the trees in RISE combine well to capture the spectral features across many intervals.

8.4.3 Vowel Classification from Raw Audio. Our final use case experiments focus on the vowel problem described in4.4. Though DTW originally came from the speech field before being applied to TSC problems, we would not expect it (or EE) to perform best on this problem due to the possibility of the previous/next term being included at the start/end of the series. Shapelet-based classification should therefore have an advantage, and we would also expect RISE to perform well too. It is common to apply spectral transforms to speech data, and this coupled with the fact that RISE creates intervals means that it should be able to extract discriminatory information without being confounded by noise at the start or end of the series. Unlike the previous case studies, the data does not need to be cut specifically to avoid bias as as all utterances were recorded by a single speaker. Therefore we can create a standard train/test split and perform seeded stratified resample experiments. We also plan to include the vowel problem in future releases of the UCR/UEA datasets. Table10reports the results of running 100 resample experiments.

Table 10. A summary of the results of the vowel problem over 100 resamples. Classifiers are presented in descending order of accuracy.

Classifier	Avg	Stdev
RISE	94.93%	0.025
HIVE-COTE	93.26%	0.023
ST-HESCA	90.80%	0.032
Flat-COTE	90.26%	0.033
BOSS	80.11%	0.037
EE	72.14%	0.044
DTW	68.57%	0.044
TSF	33.64%	0.042
HESCA	25.55%	0.038
RotF	24.23%	0.037

RISE was the best performing classifier on this problem. This was expected, but it is perhaps surprising to get almost 95% accuracy with no parameter optimisation or preprocessing, especially given that we only retained 10% of the original data from the 50kHz recordings. To further investigate this result and determine whether it is simply just a problem well suited to spectral approaches, we also ran PS-HESCA and ACF-HESCA for context. These classifiers reported 77.4% and 68.9% respectively. This clearly demonstrates that RISE fills a niche in TSC that current spectral approaches do not address. We believe that this is because the interval aspect of RISE mitigates against potential noise at the start and end of the series, but it could also be the interaction between ACF and PS features within the same classifier.

Finally, Table10shows that HIVE-COTE also achieves high accuracy on this problem; it is second overall and outperforms Flat-COTE by 3%. This is even more impressive when considering that HIVE-COTE contains TSF, which reported only 33.6%. Even with TSF included, the hierarchical structure of HIVE-COTE is able to leverage from the strong performances of RISE and ST-HESCA without being mislead by TSF. The performance of HIVE-COTE across all three of these case study datasets underlines the robustness of the new hierarchical structure and the performance of the new classification modules. HIVE-COTE is at least in the top two classifiers on these use cases, and where it is beaten, it is only marginally outperformed by one of its own modules (RISE or ST-HESCA).

9 CONCLUSIONS AND FUTURE DIRECTION

We set out to address two key questions. First, is the existing state of the art, Flat-COTE, significantly more accurate than current deep learning approaches for TSC? We answer this question by comparing Flat-COTE to two deep learning approaches: a benchmark CNN that we implemented ourselves and a TSC-specific CNN from the literature. Our analysis shows that Flat-COTE outperforms both. Flat-COTE is significantly more accurate than the standard CNN over 85 UCR/UEA datasets and significantly more accurate than the TSC-specific CNN over the 44 datasets reported in (Cui et al. 2016). These findings motivate our second question; can we improve on Flat-COTE and make a significantly more accurate collective? We introduce HIVE-COTE, a new meta-ensemble with updated constituent classifiers and an encapsulated hierarchical structure. HIVE-COTE contains classifiers that can detect five types of discriminatory features: whole series, shapelet, dictionary, interval, and spectral. As part of HIVE-COTE we formally define a heterogeneous ensemble of standard classification algorithms (HESCA) that we pair with shapelet-transformed problems, and a novel random interval spectral ensemble (RISE) that we demonstrate is significantly more accurate than existing spectral approaches. Classifiers built in each of the five domains within HIVE-COTE are encapsulated in modules, and modules are combined through a hierarchical probabilistic voting structure. Through extensive experimentation on 100 resamples of 85 public datasets, 5 types of simulated data, and 3 new case studies, we demonstrate that HIVE-COTE is significantly more accurate than all alternatives for TSC, including Flat-COTE. To the best of our knowledge, HIVE-COTE is the most accurate published TSC algorithm. We release all source code for our algorithms and experiments, and encourage the community to contribute to this ever-expanding resource.

In addition to refinements that we could still make to HIVE-COTE, such as exponential voting schemes or weighting predictions by class, the promising results of RISE and HESCA both suggest that further work could lead to advancement of these algorithms. We have not evaluated which constituents we should include in HESCA, nor have we optimised parameter settings for any of the internal classifiers (with the exception of 1-NN where there is minimal cost to do so). We have also not optimised the number of intervals/base classifiers to include in RISE, or whether we could include different combinations of features other than ACF and PS. It seems likely that we would be able to refine and extend these constituents within HIVE-COTE, which would also contribute to a more effective collective overall. Finally, the complexity of HIVE-COTE may make the approach prohibitive in use cases where real time classification is required on low-powered computing equipment. Therefore a further future research direction is to investigate whether the runtime of HIVE-COTE may be significantly reduced without a significant reduction in the accuracy of the approach. This could be tackled both theoretically through runtime complexity comparisons, or more practically through an evaluation of experimental runtimes between algorithms. This would generate further research questions to ensure any wallclock comparisons were conducted in a fair environment, but results of such a study would likely be of great benefit to practitioners in the field.

CODE AND DATA ACCESS

All code and data used in this work is open source and freely available from [5] and [6]. The source code is implemented in Java to extend the Weka Machine Learning Toolkit (Hall et al. 2009) and an example use case with a common TSC problem is given in the main method of the HIVE-COTE class. An interested researcher can run this classifiers on standard commodity hardware using any TSC problem formatted in Weka's Attribute Relation Format File type.

[5] UCR/UEA TSC Repository:www .timeseriesclassification.com
[6] UEA TSC Code Base:bitbucket .org/TonyBagnall/time-series-classification

ACKNOWLEDGEMENTS

This work is supported by the UK Engineering and Physical Sciences Research Council (EPSRC) [grant numbers EP/M015087/1 and EP/M014053/1]. 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 also gratefully acknowledge the support of the NVIDIA Corporation with the donation of a Titan X GPU that was used for running the CNN experiments in this work. Finally, the authors would also like to thank James Large for recording the Ethanol Level problem used in this work.

REFERENCES

A. Bagnall, A. Bostrom, J. Large, and J.Lines. 2016. *Simulated Data Experiments for Time Series Classification Part 1: Accuracy Comparison with Default Settings*. Technical Report. School of Computing Sciences, University of East Anglia.

A. Bagnall, L. M. Davis, J. Hills, and J. Lines. 2012. Transformation Based Ensembles for Time Series Classification.. In *Proceedings of the 2012 SIAM International Conference on Data Mining*, Vol. 12. 307–318.

A. Bagnall and G. Janacek. 2014. A run length transformation for discriminating between auto regressive time series. *Journal of Classification* 31 (2014), 154–178. Issue 2.

A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh. 2016. The Great Time Series Classification Bake Off: a Review and Experimental Evaluation of Recent Algorithmic Advance. *Data Mining and Knowledge Discovery* (2016), 1–55.

A. Bagnall, J. Lines, J. Hills, and A. Bostrom. 2015. Time-Series Classification with COTE: The Collective of Transformation-Based Ensembles. *IEEE Transactions on Knowledge and Data Engineering* 27 (2015), 2522–2535. Issue 9.

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

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

M. Baydogan, G. Runger, and E. Tuv. 2013. A Bag-of-Features Framework to Classify Time Series. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 25, 11 (2013), 2796–2802.

A. Benavoli, G. Corani, and F. Mangili. 2016. Should We Really Use Post-Hoc Tests Based on Mean-Ranks? *Journal of Machine Learning Research* 17 (2016), 1–10.

A. Bostrom and A. Bagnall. 2015. Binary Shapelet Transform for Multiclass Time Series Classification. In *Proc. 17th International Conference on Big Data Analytics and Knowledge Discovery (DAWAK).*

L. Breiman. 1996. Bagging predictors. *Machine learning* 24, 2 (1996), 123–140.

Leo Breiman. 2001. Random forests. *Machine learning* 45, 1 (2001), 5–32.

J. Caiado, N. Crato, and D. Pena. 2006. A periodogram-based metric for time series classification. *Computational Statistics and Data Analysis* 50 (2006), 2668–2684.

M. Cooke, J. Barker, S. Cunningham, and X. Shao. 2006. An audio-visual corpus for speech perception and automatic speech recognition. *The Journal of the Acoustical Society of America* 120, 5 (2006), 2421–2424.

M. Corduas and D. Piccolo. 2008. Time series clustering and classification by the autoregressive metric. *Computational Statistics and Data Analysis* 52, 4 (2008), 1860–1872.

Z. Cui, W. Chen, and Y. Chen. 2016. Multi-Scale Convolutional Neural Networks for Time Series Classification. *arXiv:1603.06995* (2016).

J. Demšar. 2006. Statistical Comparisons of Classifiers over Multiple Data Sets. *Journal of Machine Learning Research* 7 (2006), 1–30.

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

Y. F and R. Schapire. 1996. Experiments with a new boosting algorithm. In *icml*, Vol. 96. 148–156.

M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim. 2014. Do we Need Hundreds of Classifiers to Solve Real World Classification Problems? *Journal of Machine Learning Research* 15 (2014), 3133–3181.

B. Fulcher and N. Jones. 2014. Highly comparative feature-based time-series classification. *IEEE Transactions on Knowledge and Data Engineering* 26, 12 (2014), 3026–3037.

S. García and F. Herrera. 2008. An Extension on âĂIJStatistical Comparisons of Classifiers over Multiple Data SetsâĂİ for all Pairwise Comparisons. *Journal of Machine Learning Research* 9 (2008), 2677–2694.

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

J. Grabocka, N. Schilling, M. Wistuba, and L. Schmidt-Thieme. 2014. Learning Time-Series Shapelets. In *Proc. 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

A. Graves, A. Mohamed, and G. Hinton. 2013. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE, 6645–6649.

M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. Witten. 2009. The WEKA Data Mining Software: An Update. *SIGKDD Explorations* 11, 1 (2009), 10–18.

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

Y. Jeong, M. Jeong, and O. Omitaomu. 2011. Weighted dynamic time warping for time series classification. *Pattern Recognition* 44 (2011), 2231–2240. Issue 9.

N. Kalchbrenner, E. Grefenstette, and P. Blunsom. 2014. A convolutional neural network for modelling sentences. *arXiv:1404.2188* (2014).

R. Kate. 2016. Using dynamic time warping distances as features for improved time series classification. *Data Mining and Knowledge Discovery* 30, 2 (2016), 283–312.

A. Krizhevsky, I. Sutskever, and G. Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems 25*. Curran Associates, Inc., 1097–1105.

J. Lin, R. Khade, and Y. Li. 2012. Rotation-invariant similarity in time series using bag-of-patterns representation. *Journal of Intelligent Information Systems* 39, 2 (2012), 287–315.

J. Lines and A. Bagnall. 2015. Time Series Classification with Ensembles of Elastic Distance Measures. *Data Mining and Knowledge Discovery* 29 (2015), 565–592. Issue 3.

J. Lines, L. Davis, J. Hills, and A. Bagnall. 2012. A Shapelet Transform for Time Series Classification. In *Proc. the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

J. Lines, S. Taylor, and A. Bagnall. 2016. HIVE-COTE: The Hierarchical Vote Collective of Transformation-based Ensembles for Time Series Classification. In *Proc. IEEE International Conference on Data Mining*.

P. Marteau. 2009. Time Warp Edit Distance with Stiffness Adjustment for Time Series Matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31, 2 (2009), 306–318.

C. Ratanamahatana and E. Keogh. 2005. Three Myths about Dynamic Time Warping Data Mining. In *Proc. 5th SIAM International Conference on Data Mining (SDM)*.

Juan José Rodriguez, Ludmila I Kuncheva, and Carlos J Alonso. 2006. Rotation forest: A new classifier ensemble method. *IEEE transactions on pattern analysis and machine intelligence* 28, 10 (2006), 1619–1630.

P. Schäfer. 2015. The BOSS is concerned with time series classification in the presence of noise. *Data Mining and Knowledge Discovery* 29, 6 (2015), 1505–1530.

A. Stefan, V. Athitsos, and G. Das. 2013. The Move-Split-Merge Metric for Time Series. *IEEE Transactions on Knowledge and Data Engineering* 25, 6 (2013), 1425–1438.

Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints* abs/1605.02688 (2016).

J. Villar, P. Vergara, M. Menéndez, E. de la Cal, V. González, and J. Sedano. 2016. Generalized Models for the classification of abnormal movements in daily life and its applicability to epilepsy convulsions recognition. *International Journal of Neural Systems* (2016).

G. Webb. 2000. Multiboosting: A technique for combining boosting and wagging. *Machine learning* 40, 2 (2000), 159–196.

L. Ye and E. Keogh. 2011. Time series shapelets: a novel technique that allows accurate, interpretable and fast classification. *Data Mining and Knowledge Discovery* 22, 1-2 (2011), 149–182.