The Canonical Interval Forest (CIF) Classifier for Time Series Classification

Matthew Middlehurst
School of Computing Sciences
University of East Anglia
Norwich, United Kingdom

James Large
School of Computing Sciences
University of East Anglia
Norwich, United Kingdom

Anthony Bagnall
School of Computing Sciences
University of East Anglia
Norwich, United Kingdom

Abstract—Time series classification (TSC) is home to a number of algorithm groups that utilise different kinds of discriminatory patterns. One of these groups describes classifiers that predict using phase dependant intervals. The time series forest (TSF) classifier is one of the most well known interval methods, and has demonstrated strong performance as well as relative speed in training and predictions. However, recent advances in other approaches have left TSF behind. TSF originally summarises intervals using three simple summary statistics. The 'catch22' feature set of 22 time series features was recently proposed to aid time series analysis through a concise set of diverse and informative descriptive characteristics. We propose combining TSF and catch22 to form a new classifier, the Canonical Interval Forest (CIF). We outline additional enhancements to the training procedure, and extend the classifier to include multivariate classification capabilities. We demonstrate a large and significant improvement in accuracy over both TSF and catch22, and show it to be on par with top performers from other algorithmic classes. By upgrading the interval-based component from TSF to CIF, we also demonstrate a significant improvement in the hierarchical vote collective of transformation-based ensembles (HIVE-COTE) that combines different time series representations. HIVE-COTE using CIF is significantly more accurate on the UCR archive than any other classifier we are aware of and represents a new state of the art for TSC.

Index Terms—Time series, Classification, Ensembles, HIVE-COTE, Multivariate

I. Introduction

Time series classification (TSC) has experienced a rapid algorithmic advancement in predictive performance over the last decade. Prior to this, the one nearest neighbour with dynamic time warping distance classifier, with warping window set through cross validation (DTWCV), was considered the gold standard benchmark and difficult to beat. However, a wide scale survey and evaluation of TSC literature [1] found a number of classifiers able to significantly improve on DTWCV, and formed a taxonomy of similar approaches based on representations of discriminatory features. This taxonomy grouped algorithms into categories based on the representations: distance-based; shapelet-based; dictionary-based; frequency-based; and interval-based (details provided in Section II).

Three algorithms proposed subsequent to this evaluation have significantly higher accuracy than all of those evaluated in [1] and can claim to be among the state of the art for general-purpose TSC. These algorithms also represent the best of three alternative approaches to leveraging different representations for TSC. The hierarchical vote collective of transformation-based ensembles (HIVE-COTE) [2] encapsulates representations within modules containing algorithm(s) based on a single representation, then ensembles over these distinct modules. The time series combination of heterogeneous and integrated embeddings forest (TS-CHIEF) [3] embeds multiple representations within the nodes of a decision tree, then ensembles many trees in a way similar to random forest. Thirdly, a bespoke representation for the dataset can be learned through deep learning approaches which have gained in popularity recently within TSC research. InceptionTime [4], an ensemble of randomly initialised ResNet-style networks with Inception modules, is currently the strongest of these.

Our aim is to advance state of the art in TSC by improving one component of HIVE-COTE. HIVE-COTE consists of five modules, each of which represents the best in class (at the time of publication in 2017) for the five different representations proposed in [1]. Our focus is on interval-based classifiers. This class of algorithms derive features of series from randomly selected, but constant over series, intervals. Through taking multiple intervals and randomisation, interval-based classifiers are able to mitigate against regions of series that may confound other algorithms due to, for example, high noise or constancy. These classifiers perform best when the particular location in time of discriminatory features is important for class membership. An example of this is shown in the EthanolLevel problem presented in Figure 1. This describes the visible-near infrared spectra of suspect spirits, where in this case the aim is to predict the concentration of ethanol in the sample. Based on the resonances of ethanol around these ranges, a human expert would know to look at a particular interval of wavelengths. While the noise in the center would confound whole series methods such as DTWCV, interval methods can better extract and summarise the discriminatory subseries. If we were to label the samples according to other factors, such as the presence of artificial caramel colouring, a different interval may be appropriate to use.

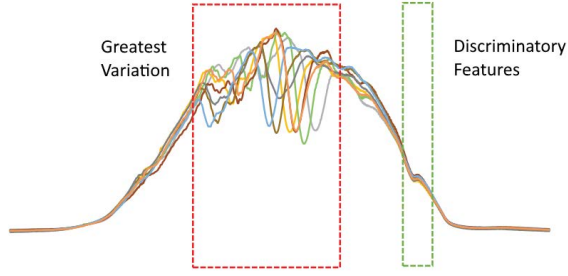Time series forest (TSF) [5] was an early example of an

Fig. 1. Time series from the EthanolLevel dataset. While the middle portion (visible light wavelengths) contains the greatest variation, a particular interval (into the near infrared) contains the most discriminatory information for ethanol concentration. An interval-based classifier may be able to compensate for the confounding effect of the visible spectrum.

interval-based forest of trees ensemble for TSC. TSF has been shown to be significantly better than DTWCV [1], and represents the interval-based group of algorithms in HIVE-COTE. It has a number of secondary desirable properties it holds common with random forests: TSF is very fast to train; it is robust to parameter settings, with parameter tuning having no significant overall effect; and the trained forest can be post-processed to extract the importance of different intervals towards classification [5] - a useful tool for interpreting the model. However, TSF is, on average, the weakest classifier in the HIVE-COTE ensemble in terms of predictive power.

TSF works on only three basic summary stats for each interval: the mean; standard deviation; and slope. Recently, a set of 22 features, the canonical time-series characteristics (catch22) [6], have been proposed as good primitive summary measures for time series. These features cover a diverse range of concepts and are relatively quick to process. However, by themselves they are not particularly useful for classification when applied as a transform to the whole series. The resulting classifier with a random forest of 500 trees built on the transform is not significantly better than DTWCV.

Our primary contribution is to describe an algorithm for embedding catch22 features in an adaptation of TSF. This new classifier, the Canonical Interval Forest (CIF) is significantly more accurate than TSF; it is not significantly worse than the best recently proposed algorithms based on a single representation; and replacing TSF with CIF in HIVE-COTE results in significant improvement. We additionally expand CIF for multivariate classification, and demonstrate CIF's strong performance in this relatively immature domain.

Our secondary contribution, which facilitates the first, is to provide an evaluation and comparison of eight TSC algorithms described in Section II-A, using the recently expanded UCR archive [7]. These algorithms have been proposed since the bake off [1]. All the code to do this is open source and all the code and results are available on the accompanying website.

The rest of this paper is structured as follows. Section II provides a technical background to TSC and related algo-

rithms. Section III explains the changes made to TSF in order to include catch22 features and form CIF, and Section IV describes our experimental methodology. Section V describes the results of investigations into the improvements made both by CIF over TSF, and to HIVE-COTE with CIF's inclusion over TSF. In Section VI we compare the time and space complexity of a range of classifiers and describe how CIF can be used to aid interpretability and practical usage. Finally, in Section VII we conclude and discuss future work.

II. TIME SERIES CLASSIFICATION

A time series is an ordered vector of continuous values that is typically taken over fixed intervals of time, but may be representative of other ordered data, e.g. an image outline. An instance $(\mathbf{X}, y)$ is a collection of d time series x with measurements at m time points $(x_1, \ldots, x_m)$ and an associated class label y. A dataset is a list of n instances, $\mathbf{T} = \{(\mathbf{X}_1, y_1), \ldots, (\mathbf{X}_n, y_n)\}$. Given a labelled dataset the objective of a TSC classifier is to predict the class of new input time series using models derived during training. We consider the problem of univariate, equal length series in our main results. To demonstrate the robustness of CIF, however, we show results for a range of multivariate datasets.

A. Current State of the Art

There have been significant recent advances in time series classification. Space restrictions mean we can give only the briefest of summaries of this work.

Distance-based classifiers use time series specific distance functions as the basis of classification. Proximity forest (PF) [8] is a classifier that constructs an ensemble of trees built on a randomised selection of time series distance functions. It is currently the most accurate distance based approach, beating the previous best, the Elastic Ensemble (EE) [9].

Dictionary-based classifiers use sliding windows and discretisation to find words in a time series, then classify based on the distribution of these words. WEASEL [10] and S-BOSS [11] are dictionary based algorithms that have been shown to be state of the art for this kind of approach [12]. Both are significantly more accurate than the previous best, the Bags of Symbolic-Fourier-Approximation Symbols (BOSS) [13].

Shapelet-based algorithms use phase independent subseries as the basis for classification. Shapelet transform-based algorithms have historically used full enumeration of the shapelets in the train data. This is unnecessary, and often leads to over fitting. The shapelet transform classifier (STC) [14] is a shapelet transform based classifier that operates under a maximum search time. In one hour of searching per dataset, it finds a transform that is not significantly worse than full enumeration. STC includes other improvements that make it significantly more accurate than the original transform [15].

Spectral classifiers use features in the frequency domain. The most successful of these to date is RISE [16], a tree ensemble that extracts spectral features from intervals.

Interval-based classifiers in the time domain are our focus of interest. The time series forest (TSF) [5] is described in

detail in Section II-B. Two later interval-based algorithms that were more complex successors to TSF, the Time Series Bag of Features [17] and the LSF [18] have been shown to be no more accurate than TSF on average, and considerably slower [1].

The most successful approaches combine these types of features. HIVE-COTE [2] is a meta ensemble of five classifiers, one for each representation: EE, BOSS, STC, RISE and TSF. TS-CHIEF [3] is a tree ensemble that uses a combination of distance, dictionary and spectral features. Unsurprisingly, deep learning approaches to time series classification have become very popular. However, many deep learning algorithms do not appear to perform that well [19], at least on the UCR archive which contains a number of datasets that may be too small for effective deep learning training. The counter-example to this and the best current deep learning classifier is InceptionTime [4].

B. Time Series Forest (TSF)

TSF [5] aims to capture basic summary features from intervals of a time series. For any given time series of length m there are $m(m-1)/2$ possible intervals that can be extracted. TSF takes a random forest-like approach to sampling these intervals. For each tree, k intervals are randomly selected, each with a random start position and length. Each interval is summarised by the mean, standard deviation and slope, and the summaries of each interval are concatenated into a single feature vector of length $3k$ for each time series. A decision tree is built on this concatenated feature vector. New cases are classified using a majority vote of all trees in the forest.

The version of TSF used in the bake off [1] employed the random tree used by random forest. However, the time series tree described in [5] has some minor differences to the random tree. It makes no difference in terms of accuracy, but the time series tree has advantages in terms of interpretability. The computational complexity of TSF is $O(nlog(n) \cdot m \cdot r)$ [5], where r is the number of trees in the forest.

C. Canonical Time Series Characteristics (catch22)

catch22 [6] is a set of 22 descriptive features for use in time series analysis. The motivation for catch22 is to form a concise and informative subset of the time series features from the 7658 contained in the *hctsa* toolbox [20]. These features could be used in any mining context. However, the process of selection of features and the experimentation presented in [6] is primarily based on classification with the UCR archive.

From the original 7658 hctsa features, 766 features sensitive to mean and variance were removed due to the fact that the majority of the data in the UCR archive have been normalised. This was further pruned down to 4791 candidates by removing features which cannot be calculated on over 80% of datasets. This failure is caused by characteristics of the data such as repeating values and negative values. A three step process was used to further reduce the number of features. For each feature, a stratified cross-validation was performed over each dataset in the UCR archive using a decision tree classifier. Features which performed significantly better than random

chance (according to the class-balanced accuracy metric) were retained. These significant features were then sorted by their balanced accuracy, and those below a threshold were removed. A hierarchical clustering was performed on the correlation matrix of the remaining features to remove redundancy. These clusters were sorted by balanced accuracy and a single feature was selected from the 22 clusters formed, taking into account balanced accuracy results, computational efficiency and inter-pretability.

The catch22 features cover a wide range of concepts such as basic statistics of time series values, linear correlations, and entropy. The computational complexity for computing the catch22 features is $O(n \cdot m^{1.16})$ [6], with the exponent on the series length found through computational experiments [6]. For classification, the obvious way to use catch22 is as a transform prior to building a classifier. The reported results [6] are found using an decision tree classifier, although the default implementation in code uses a random forest.

III. CANONICAL INTERVAL FOREST CLASSIFIER (CIF)

catch22 summarises time series features very concisely, but classifier's built on catch22 transformation over the whole series are significantly worse than alternative feature based approaches. TSF is very fast and provides diversity that improves HIVE-COTE, but as a stand alone classifier it is significantly worse than two HIVE-COTE components, BOSS and STC. Our research question was whether by replacing the three simple summary features used in TSF with the more descriptive catch22 feature sets we could find a significantly better interval classifier which in turn could lead to an improvement in HIVE-COTE. Our initial approach on the way to forming CIF is to simply replace the feature extraction operations with no further alterations. We call this feature-swapped version 'hybrid'. The feature set changes from $f(\cdot) = \{mean, stdev, slope\}$ to $f(\cdot) = \{c22f_1, c22f_2, \ldots, c22f_{22}\}$, where $c22f_i$ indicates the ith canonical feature.

Experimentation presented in Section V shows hybrid is significantly more accurate than TSF, but this comes at a time overhead. catch22 features are near-linear to compute, but clearly still more expensive overall than calculating just the mean, standard deviation and slope. There are also char-acteristics of the catch22 features that cause a large time overhead for some data. Two features in particular, the positive and negative *'DN_OutlierInclude'* features, took a very long time to compute on certain intervals of unnormalised datasets, which exhibit large absolute value extremes. Because of this, CIF uses normalised intervals for these two catch22 features. The rest of the features are calculated on the unnormalised intervals, as the absolute values within particular intervals can hold important discriminatory information for those features that do leverage it acceptably.

We include the three TSF features along with the catch22 features in CIF, since evidently for some problems these are sufficient by themselves and very cheap to process. We leverage the larger total feature space and inject additional diversity into the ensemble by randomly sampling the 25

features for each tree. This has the added benefit of improving time efficiency. By default we set the number of features subsampled for each a tree, a, to 8. We found this value to be the smallest well-performing value on average during our exploratory experiments. Finally, we employ the time series tree [5] originally used by TSF rather than the random tree used in the open source Java implementation [1].

TSF was originally designed for univariate time series. We can expand the use of CIF to multivariate problems by expanding the random interval search space, defining an interval as coming from a random dimension, in addition to having random positions and lengths. Consequently, we augment the number of intervals selected per tree to $k = \sqrt{d} \cdot \sqrt{m}$.

The full training procedure for CIF is described in Algorithm 1.

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

Parameters: the number of trees, r, the number of intervals per tree, k, and the number of attributes subsampled per tree, a (default $r = 500$, $k = \sqrt{d} \cdot \sqrt{m}$, and $a = 8$)

1: Let $\mathbf{F} = (\mathbf{F_1} \dots \mathbf{F_r})$ be the trees in the forest
2: **for** $i \leftarrow 1$ to r **do**
3:     Let $\mathbf{S}$ be a list of n cases $(s_1 \dots s_n)$ with $a \cdot k$ attributes
4:     Let $\mathbf{U}$ be a list of a randomly selected attribute indices $(u_1 \dots u_a)$
5:     **for** $j \leftarrow 1$ to k **do**
6:         $b = rand(1, m - 3)$
7:         $l = rand(b + 3, m)$
8:         $o = rand(1, d)$
9:         **for** $t \leftarrow 1$ to n **do**
10:           **for** $c \leftarrow 1$ to a **do**
11:             **if** $u_c <= 22$ **then**
12:               $s_{t,a(j-1)+c} = c22Feature(\mathbf{u_c}, \mathbf{X_{t,o}}, \mathbf{b}, \mathbf{l})$
13:             **else**
14:               $s_{t,a(j-1)+c} = tsfFeature(\mathbf{u_c}, \mathbf{X_{t,o}}, \mathbf{b}, \mathbf{l})$
15:     $F_i.buildTimeSeriesTree([S, y])$

IV. Experimental Methods

When evaluating the predictive performance of algorithms, we make use of the univariate TSC datasets in the UCR archive [7]. Experiments are run on 112 of the 128 datasets in the archive; we have removed data that have unequal length series or contain missing values, since most algorithm implementations are unable to handle these scenarios. We also remove the dataset Fungi as it only provides a single train case for each class, making parameter optimisations difficult. The UCR archive provides a default split into train and test sets. We resample each dataset 30 times in a random stratified manner. When presenting the performance of a classifier on a dataset, we use the average score across 30 resamples. However, fold 0 is always the original provided split for ease of comparison to other results. The resampling and all classifier parameters are seeded by the dataset fold index, and as such all data, results and analysis are reproducible. For our multivariate experiments

we use the 26 equal length datasets from the UEA multivariate TSC archive [21], presenting results on the default train and test split only. For TSC, data with thousands of instances or attributes can be considered 'big data' due the high time and/or memory complexities required to extract meaningful features. This is especially the case for multivariate due to the increase in dimensionality.

We use two open source software tools that contain implementations provided by the original algorithm designers. `tsml`[1] is a Weka compatible time series machine learning toolkit that contains implementations of the majority of the existing algorithms we have evaluated. `sktime`[2] is scikit-learn compatible toolkit for time series with a deep learning variant called `sktime-dl`[3]. The code to reproduce all experimental results is available on the accompanying website[4].

When comparing two classifiers over multiple data sets, we use a pairwise Wilcoxon signed-rank test on the scores averaged over resamples. We present results of multiple classifiers over multiple data sets using an adaptation of critical difference diagrams [22], with the change that all classifiers are compared with pairwise Wilcoxon signed-rank tests, and cliques formed using the Holm correction [23] rather than the post-hoc Nemenyi test originally used by [22]. We assess classifier performance with accuracy. We were constrained to seven days computation for a single experiment (i.e. a single resample of a data set with a particular classifier).

V. Results

A. Recent TSC Algorithms

In Section II we described eight recently proposed algorithms for TSC. We present the critical difference diagram for the 97 problems all algorithms could complete for accuracy (Figure 2). The full accuracy results for all classifiers shown are provided on the accompanying website, as well as results for AUROC, balanced accuracy and F1 score. Most algorithms finished at least 109 of the 112. HIVE-COTE only completed 97. This was caused by the Elastic Ensemble (EE) component. EE performs a ten fold cross validation to find the HIVE-COTE weight, which slows it down considerably. We include DTWCV for reference and TSF as it is relevant to later results. The number adjacent to each algorithm indicates the average rank over all problems (lower is better). Solid bars represent cliques within which there is no significant difference. Experiments were performed with the parameters used in the paper of origin of each classifier, with the exception of STC which uses a max search time of 1 hour instead of full enumeration. These settings are described on the accompanying website.

Our first observation is that DTWCV is not competitive with modern TSC algorithms. On average, S-BOSS is 7% more accurate per problem, and HIVE-COTE is 10.5% better. There are obviously still scenarios where DTW will be useful for

---

[1]https://github.com/uea-machine-learning/tsml
[2]https://github.com/alan-turing-institute/sktime
[3]https://github.com/sktime/sktime-dl
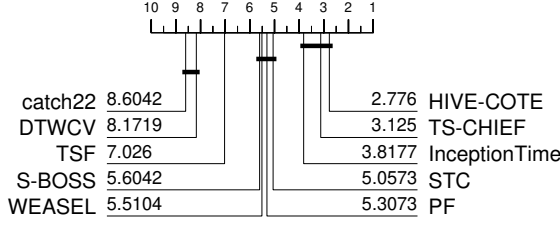[4]http://timeseriesclassification.com/CIF.php

Fig. 2. Critical difference diagram for rankings by classification accuracy of eight recently proposed TSC algorithms, TSF and DTWCV.

classification and it has much broader applications. However, we think that it is no longer hard to beat and, in isolation, is not a valid benchmark for comparisons based on classification accuracy. There are three clear cliques evident in the accuracy ranks shown in Figure 2. catch22 is the only algorithm not significantly better than DTWCV. Like DTW, catch22 has applications beyond classification. However, using it as a whole series transformation prior to classification is not very effective. TSF is significantly better than both, but worse than the first clique of modern classifiers: S-BOSS, WEASEL, PF and STC. Each of these algorithms rely on a single representation/transformation. The top clique, InceptionTime, TS-CHIEF and HIVE-COTE, contains algorithms that use combinations of representations. Although the hybrid approaches are the most accurate, it is, in our opinion, worthwhile researching new algorithms based on single representations. Some data will be best approached by one representation based on the domain specific problem definition. The four single representation algorithms evaluated all represent a significant improvement on the previous best of class that are currently used in HIVE-COTE. Our objective is to produce a new interval-based algorithm that represents a significant improvement over TSF.

B. CIF as a standalone classifier

Our first experiment compares the hybrid classifier that uses all of the catch22 features instead of the three TSF features. This serves as a basic test of concept. Figure 3 presents a pairwise scatter diagram comparing accuracies of TSF and hybrid. The hybrid is significantly better than TSF. If we examine the resamples on each dataset and perform a paired sample t-test, we find that the hybrid is significantly better on 76 datasets and significantly worse on just one. There is improvement from using the catch22 features with TSF. However, the simple usage of catch22 presents some problems. The hybrid is drastically slower than TSF on some problems. There are only results 109 problems in Figure 3 because hybrid failed to complete three problems within 7 days. CIF is designed to improve efficiency (see Section VI) but also inject further diversity. Figure 4 shows the rank data for CIF, catch22, TSF and the hybrid. It clearly demonstrates that CIF improves the accuracy of the simple catch22-TSF hybrid. The actual accuracy improvements over TSF and catch22 are not small. The average improvement of CIF over all data sets is
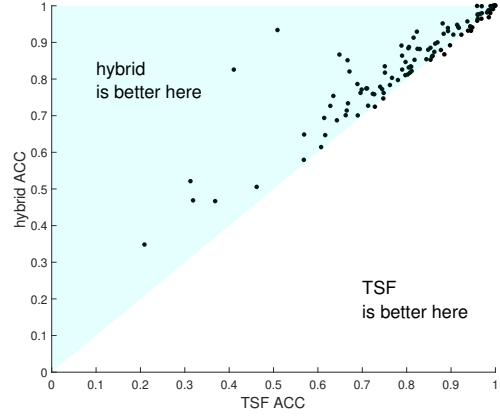

Fig. 3. Scatter plot of TSF vs hybrid on 109 problems. Hybrid wins on 87 problems, ties on 3, and loses on 19 problems.
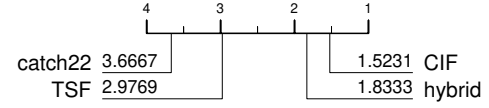

Fig. 4. Accuracy rank comparison for three interval classifiers and catch22 on 109 UCR problems.

4.56% against TSF and 6.34% vs catch22. CIF has higher accuracy on 99 of the 109 problems.

CIF is the best in class for the interval-based classifiers. Figure 5 compares CIF to the latest single representation classifiers: PF; STC; WEASEL; and S-BOSS. There is no significant difference between these five classifiers. We conclude it achieves our goal of an interval-based classifier that is at least as good as other single representation algorithms.

C. CIF on multivariate data

We have demonstrated CIF's performance on univariate data. Strong performance here does not necessarily transfer to the multivariate case, however. To demonstrate this we compare to against a range of multivariate classifiers on the UEA multivariate archive. DTW is still a reasonable benchmark in the multivariate case, using the different strategies for generalisation described in [24]. DTW_I assumes independence between dimensions, DTW_D assumes dependence, while DTW_A adaptively selects between the two on an instance by instance basis. The generalised random shapelet forest (gRSF) [25] constructs an ensemble of shapelet trees,
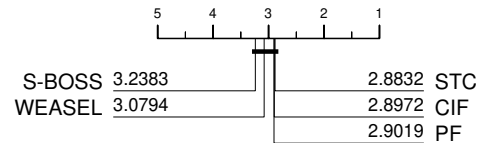

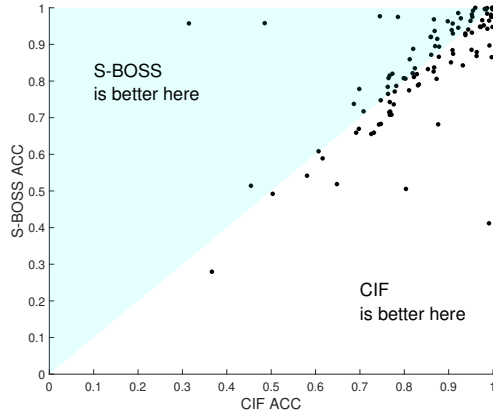Fig. 5. Accuracy rank comparison for CIF and latest single representation classifiers.

Fig. 6. Scatter plot of S-BOSS and CIF. CIF wins on 59 problems, ties on 1, and loses on 47 problems.
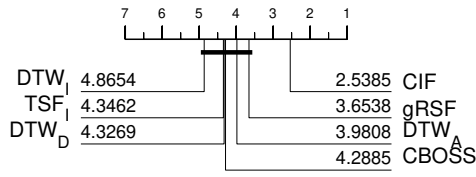

Fig. 7. Accuracy rank comparison for CIF and multivariate classifiers on 26 equal length UEA datasets.
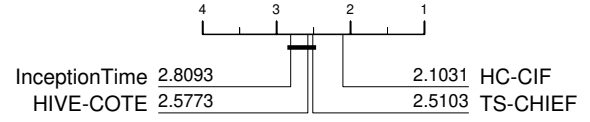

Fig. 8. Accuracy rank of three state of the art classifiers and HIVE-COTE with CIF replacing TSF (HC-CIF) on 30 resamples of 97 UCR classification problems.
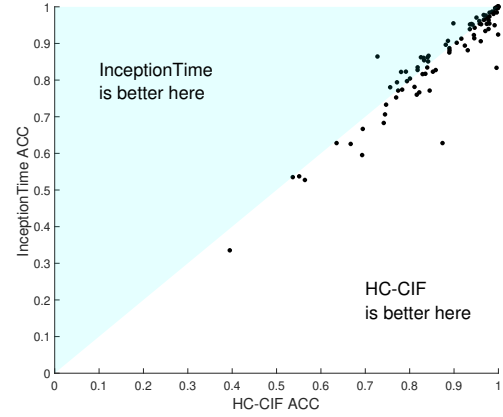

Fig. 9. Scatter plot of InceptionTime and HC-CIF. HC-CIF wins on 68 problems, ties on 5, and loses on 36 problems.

randomly selecting the dimension each shapelet is generated from. Also included are TSF and CBOSS [26], a more scalable version of the HIVE-COTE component BOSS [13]. For both classifiers a model is trained for each dimension independently, which are then ensembled with new cases being classified using majority vote. We call these TSF_I and $CBOSS_I$. We attempted to obtain results for two other classifiers, however, we have needed to omit them: the deep learning method Tap-Net [27], for which we could not satisfactorily reproduce the results for using the authors code; and WEASEL+MUSE [28], which required more than 500GB on 4 problems.

Figure 7 shows the critical difference diagram for CIF and the multivariate classifiers presented. DTW remains a good benchmark for multivariate data. TSF and CBOSS are significantly better then univariate DTW, this does not translate to the multivariate case. However, CIF performs significantly better than the other algorithms used in the analysis. Whilst the comparison is not exhaustive, it demonstrates that it is easy and intuitive to generalise CIF effectively for multivariate data.

D. CIF as a HIVE-COTE component

CIF is not better than other single representation classifiers, so why do we need CIF if we already have STC, PF, S-BOSS and WEASEL? It is not just a matter of the efficiency benefits (discussed in Section VI). Despite there being no overall difference between these single representation classifiers, there is still significant diversity over specific problems. Figure 6 shows the accuracy scatter plot for CIF and S-BOSS. Each representation specialises on different data characteristics,

causing large differences in accuracy. The same pattern can be observed with the other classifiers, for example, CIF and STC vary significantly over data. STC beats CIF by over 40% on PigCVP and PigAirwayPressure, whereas the opposite is true with TwoPatterns. There are 12 problems with a difference in accuracy of 10% or more. This diversity is what helps HIVE-COTE perform so well. If CIF can improve HIVE-COTE, then it is further evidence as to the value of improving interval based approaches. To test this we replace TSF with CIF and rerun HIVE-COTE, whilst leaving the other components the same. To differentiate between the old and new variants of HIVE-COTE, we shall call them HIVE-COTE and HC-CIF respectively. Figure 8 gives the accuracy ranks of the state of the art classifiers and HC-CIF. HC-CIF is significantly more accurate than all three. We conclude that HC-CIF is the most accurate classifier to date (to our knowledge) on the UCR time series classification archive, and as such represents a genuine advance in state of the art for this field. We stress we have done nothing to HIVE-COTE except replace TSF with CIF.

Figure 9 show the scatter plots for HC-CIF against InceptionTime, and includes the win/tie/lose statistics for information. There is still wide variation, but the improvement from HC-CIF is visually apparent. HC-CIF has, on average, 1.56% higher accuracy than InceptionTime.

VI. CIF EFFICIENCY AND USABILITY

A. Time and space efficiency

CIF is significantly more accurate than TSF, but at what cost in terms of time and memory? Figure 10 plots the average
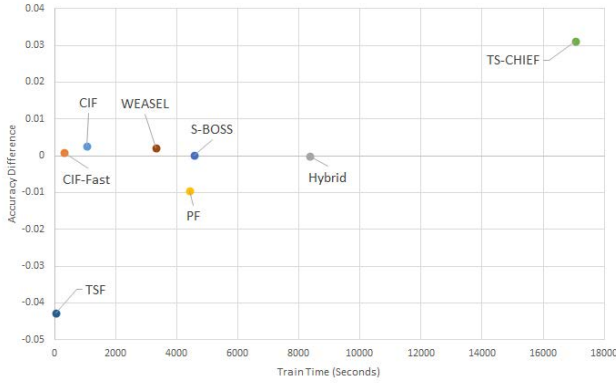
Fig. 10. A comparison of classifiers in terms of accuracy and train time. The y-axis shows the average difference in accuracy from the default hybrid approach. Time and accuracy are averaged over 104 UCR problems.
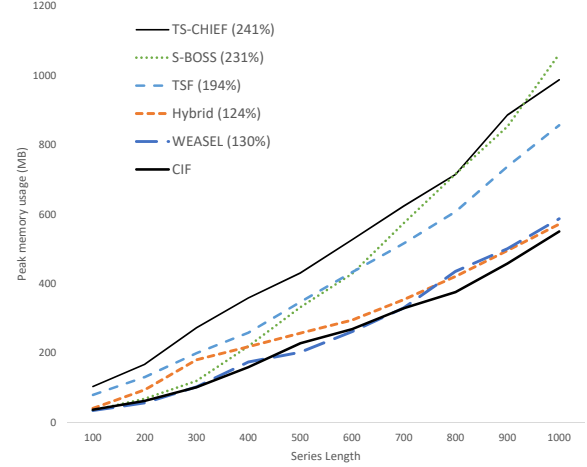

Fig. 11. Memory usage of six classifiers for varying series length. Figures in brackets are the overall percentage of memory used in comparison to CIF.
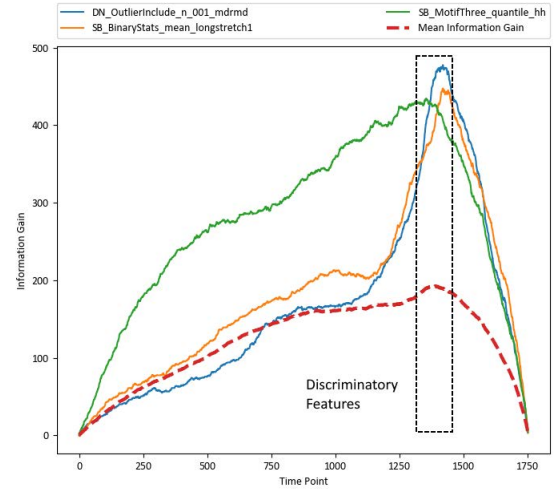

Fig. 12. Temporal importance curves for the EthanolLevel problem. Higher values indicate greater importance for classification. The dotted box indicate the most important region as specified by a domain expert (see Figure 1).

difference in accuracy from a baseline (hybrid) against the average run time on the UCR data for CIF and 6 algorithms used in our previous comparisons. All experiments were conducted in a single thread, even though some such as PF and TS-CHIEF can employ multiple threads. This is to make sure comparisons are fair.

At the extremes, we see TSF is by far the fastest, but the least accurate; and that TS-CHIEF is the most accurate, but the slowest. CIF is approximately eight times faster than the hybrid, and is the fastest of the single representation classifiers displayed in Figure 5, between which there is no significant difference in accuracy. CIF-Fast is a reduced form of CIF with fewer trees (250) and intervals per tree ($\sqrt{m}^{0.85}$). We include it to demonstrate that, if the problem is very large, CIF can be configured to be faster with little loss of accuracy. To further improve usability, we provide a contracted version of CIF in line with other HIVE-COTE components [14], [16], [26] that adds trees to the ensemble incrementally. The ability to set a fixed time to train until allows a degree of certainty for time sensitive training requirements, such as clusters with a maximum job time limit.

Figure 11 characterises the memory footprint of six algorithms using simulated data. We fix the number of cases and increase series length. TS-CHIEF and S-BOSS require the most memory, (241% and 231% of that required by CIF respectively). Both employ dictionary-based nearest neighbour classifiers, which require storing representations of the whole dataset. TSF memory usage was a surprise. We can explain the relatively high memory usage by reference to the implementation. The Weka random tree it employs makes a local copy of the data for every tree. This data is not stored after the build phase, but it seems the Java garbage collection is slow in deallocating the memory. The hybrid uses slightly more memory than CIF due to the increased size of its feature space.

B. Model Visualisation

One benefit of TSF is that it can be used to evaluate the relevance of regions of the time series. Temporal importance

curves [5] are a visualisation method that displays the importance of time series intervals to classification. To achieve this, the information gain of each node's splitting attribute is collected for all trees in the forest. Each attribute the trees are built on corresponds to a summary feature and a time interval. A curve is created for each summary feature, adding the gain from any node split to its curve at the time points covered by its interval. We can adapt this algorithm for CIF, although displaying all 25 features can be too cluttered. We display the top $v = 3$ features by max gain over all time points, as these have effectively been discovered to be the most important, in addition to the mean of all features for each time point.

Figure 12 displays the CIF temporal importance curves for the EthanolLevel problem shown previously in Figure 1. It demonstrates that the most important region for classification

is the same region identified by a domain expert. While by default CIF subsamples features per tree, we have not done this for the above as it can have a large impact on the production of these curves. The temporal importance curves can be generated for model visualisation and are still meaningful while subsampling. However, randomly restricting access to features, and therefore promoting weaker features that would otherwise have lost to more consistently informative ones when splitting, may not give the best representation to perform a higher level analysis of a time series problem. The goals of maximising accuracy, minimising computational cost, and maximising interpretability, often run counter to each other.

VII. Conclusion

Our contribution is to propose a new time series classifier, CIF, that combines the best elements of TSF and catch22 in a novel way. CIF is significantly more accurate than TSF and catch22. It is not significantly worse than the best other classifiers built on a single representation (WEASEL, S-BOSS, PF and STC) and is generally faster and requires less memory than them. When CIF replaces TSF in HIVE-COTE the resulting classifier, HC-CIF, is significantly more accurate than HIVE-COTE, TS-CHIEF and InceptionTime. HC-CIF represents a new state of the art for TSC in terms of classification accuracy on the UCR archive.

There are many other improvements we could attempt with HIVE-COTE. For example, we could have used PF and S-BOSS instead of EE and BOSS. However, that would have introduced more sources of variation and obscured our core contribution: CIF is a new classifier that is best of its class, and by making the single change in HIVE-COTE of replacing TSF with CIF, the result is a classifier with significantly higher accuracy on average than the current state of the art.

Acknowledgements

References

[1] A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh, "The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances," *Data Mining and Knowledge Discovery*, vol. 31, no. 3, pp. 606–660, 2017.

[2] J. Lines, S. Taylor, and A. Bagnall, "Time series classification with hive-cote: The hierarchical vote collective of transformation-based ensembles," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 12, no. 5, p. 52, 2018.

[3] A. Shifaz, C. Pelletier, F. Petitjean, and G. I. Webb, "Ts-chief: a scalable and accurate forest algorithm for time series classification," *Data Mining and Knowledge Discovery*, pp. 1–34, 2020.

[4] H. I. Fawaz, B. Lucas, G. Forestier, C. Pelletier, D. F. Schmidt, J. Weber, G. I. Webb, L. Idoumghar, P.-A. Muller, and F. Petitjean, "Inceptiontime: Finding alexnet for time series classification," *arXiv preprint arXiv:1909.04939*, 2019.

[5] H. Deng, G. Runger, E. Tuv, and M. Vladimir, "A time series forest for classification and feature extraction," *Information Sciences*, vol. 239, pp. 142–153, 2013.

[6] C. H. Lubba, S. S. Sethi, P. Knaute, S. R. Schultz, B. D. Fulcher, and N. S. Jones, "catch22: Canonical time-series characteristics," *Data Mining and Knowledge Discovery*, vol. 33, no. 6, pp. 1821–1852, 2019.

[7] H. A. Dau, A. Bagnall, K. Kamgar, C.-C. M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatana, and E. Keogh, "The ucr time series archive," *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 6, pp. 1293–1305, 2019.

[8] B. Lucas, A. Shifaz, C. Pelletier, L. O'Neill, N. Zaidi, B. Goethals, F. Petitjean, and G. I. Webb, "Proximity forest: an effective and scalable distance-based classifier for time series," *Data Mining and Knowledge Discovery*, vol. 33, no. 3, pp. 607–635, 2019.

[9] J. Lines and A. Bagnall, "Time series classification with ensembles of elastic distance measures," *Data Mining and Knowledge Discovery*, vol. 29, no. 3, pp. 565–592, 2015.

[10] P. Schäfer and U. Leser, "Fast and accurate time series classification with weasel," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, 2017, pp. 637–646.

[11] J. Large, A. Bagnall, S. Malinowski, and R. Tavenard, "On time series classification with dictionary-based classifiers," *Intelligent Data Analysis*, vol. 23, no. 5, pp. 1073–1089, 2019.

[12] A. Bagnall, J. Large, and M. Middlehurst, "A tale of two toolkits, report the second: bake off redux. chapter 1. dictionary based classifiers," *ArXiv e-prints*, vol. arXiv:1911.12008, 2019. [Online]. Available: http://arxiv.org/abs/1911.12008

[13] P. Schäfer, "The boss is concerned with time series classification in the presence of noise," *Data Mining and Knowledge Discovery*, vol. 29, no. 6, pp. 1505–1530, 2015.

[14] A. Bostrom and A. Bagnall, "Binary shapelet transform for multiclass time series classification," in *Transactions on Large-Scale Data-and Knowledge-Centered Systems XXXII*. Springer, 2017, pp. 24–46.

[15] J. Lines, L. M. Davis, J. Hills, and A. Bagnall, "A shapelet transform for time series classification," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2012, pp. 289–297.

[16] M. Flynn, J. Large, and A. Bagnall, "The contract random interval spectral ensemble (c-rise): the effect of contracting a classifier on accuracy," in *International Conference on Hybrid Artificial Intelligence Systems*. Springer, 2019, pp. 381–392.

[17] M. G. Baydogan, G. Runger, and E. Tuv, "A bag-of-features framework to classify time series," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 11, pp. 2796–2802, 2013.

[18] M. G. Baydogan and G. Runger, "Time series representation and similarity based on local autopatterns," *Data Mining and Knowledge Discovery*, vol. 30, no. 2, pp. 476–509, 2016.

[19] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Deep learning for time series classification: a review," *Data Mining and Knowledge Discovery*, vol. 33, no. 4, pp. 917–963, 2019.

[20] B. D. Fulcher and N. S. Jones, "hctsa: A computational framework for automated time-series phenotyping using massive feature extraction," *Cell systems*, vol. 5, no. 5, pp. 527–531, 2017.

[21] A. Bagnall, H. A. Dau, J. Lines, M. Flynn, J. Large, A. Bostrom, P. Southam, and E. Keogh, "The uea multivariate time series classification archive, 2018," *arXiv preprint arXiv:1811.00075*, 2018.

[22] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine learning research*, vol. 7, no. Jan, pp. 1–30, 2006.

[23] S. Garcia and F. Herrera, "An extension on"statistical comparisons of classifiers over multiple data sets"for all pairwise comparisons," *Journal of Machine Learning Research*, vol. 9, no. Dec, pp. 2677–2694, 2008.

[24] M. Shokoohi-Yekta, B. Hu, H. Jin, J. Wang, and E. Keogh, "Generalizing dtw to the multi-dimensional case requires an adaptive approach," *Data mining and knowledge discovery*, vol. 31, no. 1, pp. 1–31, 2017.

[25] I. Karlsson, P. Papapetrou, and H. Boström, "Generalized random shapelet forests," *Data mining and knowledge discovery*, vol. 30, no. 5, pp. 1053–1085, 2016.

[26] M. Middlehurst, W. Vickers, and A. Bagnall, "Scalable dictionary classifiers for time series classification," in *International Conference on Intelligent Data Engineering and Automated Learning*, ser. Lecture Notes in Computer Science. Springer, 2019, pp. 11–19.

[27] X. Zhang, Y. Gao, J. Lin, and C.-T. Lu, "TapNet: Multivariate time series classification with attentional prototypical network," in *In proc. 34th AAAI conference on artificial intelligence*, 2020.

[28] P. Schäfer and U. Leser, "Multivariate time series classification with weasel+ muse," *arXiv preprint arXiv:1711.11343*, 2017.