The Temporal Dictionary Ensemble (TDE) Classifier for Time Series Classification

Matthew Middlehurst, James Large, Gavin Cawley, and Anthony Bagnall

School of Computing Sciences, University of East Anglia, Norwich, UK
M.Middlehurst@uea.ac.uk

Abstract. Using bag of words representations of time series is a popular approach to time series classification. These algorithms involve approximating and discretising windows over a series to form words, then forming a count of words over a given dictionary. Classifiers are constructed on the resulting histograms of word counts. A 2017 evaluation of a range of time series classifiers found the bag of Symbolic-Fourier Approximation symbols (BOSS) ensemble the best of the dictionary based classifiers. It forms one of the components of hierarchical vote collective of transformation-based ensembles (HIVE-COTE), which represents the current state of the art. Since then, several new dictionary based algorithms have been proposed that are more accurate or more scalable (or both) than BOSS. We propose a further extension of these dictionary based classifiers that combines the best elements of the others combined with a novel approach to constructing ensemble members based on an adaptive Gaussian process model of the parameter space. We demonstrate that the Temporal Dictionary Ensemble (TDE) is more accurate than other dictionary based approaches. Furthermore, unlike the other classifiers, if we replace BOSS in HIVE-COTE with TDE, HIVE-COTE is significantly more accurate. We also show this new version of HIVE-COTE is significantly more accurate than the current best deep learning approach, a recently proposed hybrid tree ensemble and a recently introduced competitive classifier making use of highly randomised convolutional kernels. This advance represents a new state of the art for time series classification.

Keywords: Time series · Classification · Bag of Words

1 Introduction

Dictionary based approaches adapt the bag of words model commonly used in signal processing, computer vision and audio processing for time series classification (TSC). A comparison of TSC algorithms, commonly known as the bake off, formed a taxonomy of approaches based on representations of discriminatory features, with dictionary approaches being one of these. From the bake off the bag of Symbolic-Fourier-Approximation symbols (BOSS) [12] ensemble was found to be the most accurate dictionary classifier by a significant amount. BOSS

was found to be the third most accurate algorithm out of the 20 compared. This highlights the utility of dictionary methods for TSC.

This performance lead to BOSS being incorporated into the hierarchical vote collective of transformation-based ensembles (HIVE-COTE) [10], a heterogeneous ensemble encompassing multiple time series representations. The inclusion of BOSS and the subsequent significant improvement in accuracy places HIVE-COTE in the state of the art for TSC among three other algorithms proposed more recently. These are the time series combination of heterogeneous and integrated embeddings forest (TS-CHIEF) [15], which also a hybrid of multiple representations including BOSS, the random convolutional kernel transform (ROCKET) [4], and the deep learning approach InceptionTime [6].

Since the bake off a number of dictionary algorithms have been published, focusing on improving accuracy [14, 8], prediction time efficiency [14], train time and memory efficiency [11]. These algorithms are mostly extensions of BOSS, making alterations to different parts of the original algorithm. Word extraction for time series classification (WEASEL) [14] abandons the ensemble structure in favour of feature selection and changes the method of word discretisation. Spatial BOSS (S-BOSS) [8] introduces temporal information and additional features using spatial pyramids. Contractable BOSS (cBOSS) [11] changes the method used by BOSS to form its ensemble to improve efficiency and allow for a number of usability improvements.

Each of these methods constitutes an improvement to the dictionary representation from BOSS. Our contribution is to combine design features of these four classifiers (BOSS, WEASEL, S-BOSS and cBOSS) to make a new algorithm, the Temporal Dictionary Ensemble (TDE). Like BOSS, TDE is a homogeneous ensemble of nearest neighbour classifiers that use distance between histograms of word counts and injects diversity through parameter variation. TDE takes the ensemble structure from cBOSS, which is more robust and scaleable. The use of spatial pyramids is adapted from S-BOSS. From WEASEL, TDE uses bi-grams and an alternative method of finding word breakpoints.

We found the simplest way of combining these components did not result in significant improvement. We speculate that the massive increase in the parameter space made the randomised diversity mechanism result in too many poor learners in the ensemble. We propose a novel mechanism of base classifier model selection based on an adaptive form of Gaussian process (GP) modelling of the parameter space. Through extensive evaluation with the UCR time series classification repository [3], we show that TDE is significantly more accurate than WEASEL and S-BOSS while retaining the usability and scalability of cBOSS. We further show that if TDE replaces BOSS in HIVE-COTE, the resulting classifier is significantly more accurate than HIVE-COTE with BOSS and all three competing state of the art classifiers.

The rest of this paper is structured as follows. Section 2 provides background information for the four dictionary based algorithms relevant to TDE. Section 3 describes the TDE algorithm, including the GP based parameter search. Sec-

tion 4 presents the performance evaluation of TDE. Conclusions are drawn in Section 5 and future work is discussed.

2 Dictionary Based Classifiers

Dictionary based classifiers have the same broad structure. A sliding window of length w is run across a series. For each window, the real valued series of length w is converted through approximation and discretisation processes into a symbolic string of length l, which consists of α possible letters. The occurrence in a series of each 'word' from the dictionary defined by l and α is counted, and once the sliding window has completed the series is transformed into a histogram. Classification is based on the histograms of the words extracted from the series, rather than the raw data.

The bag of Symbolic-Fourier-Approximation symbols (BOSS) [12] was found to be the most accurate dictionary based classifier in a 2017 study [1]. Hence, it forms our benchmark for new dictionary based approaches. BOSS is described in detail in Section 2.1. A number of extensions and alternatives to BOSS have been proposed.

- One of the problems with BOSS is that it can be memory and time inefficient, especially on data where many transforms are accepted into the final ensemble. cBOSS (Section 2.2) addresses the scalability issues of BOSS [11] by altering the ensemble structure.
- BOSS ignores the temporal location of patterns. Rectifying this led to an extension of BOSS based on spatial pyramids, called S-BOSS [8], described in Section 2.3.
- WEASEL [14] is a dictionary based classifier by the same team that produced BOSS. It is based on feature selection from histograms for a linear model (see Section 2.4).

We propose a dictionary classifier that merges these extensions and improvements to the core concept of BOSS, called the Temporal Dictionary Ensemble (TDE). It lends from the sped-up ensemble structure of cBOSS, the spatial pyramid structure of S-BOSS, and the word and histogram forming improvements of WEASEL. TDE is fully described in Section 3.

2.1 Bag of Symbolic-Fourier-Approximation Symbols (BOSS) [12]

Algorithm 1 gives a formal description of the bag forming process of an individual BOSS classifier. Words are created using symbolic Fourer approximation (SFA) [13]. SFA first finds the Fourier transform of the window (line 8), then discretises the first l Fourier terms into α symbols to form a word, using a bespoke supervised discretisation algorithm called multiple coefficient binning (MCB) (line 13). It has an option to normalise each window or not by dropping the first Fourier term (lines 6-7). Lines 14-16 encapsulates the process of not counting trivially self similar words: if two consecutive windows produce the same

word, the second occurrence is ignored. This is to avoid a slow-changing pattern relative to the window size being over-represented in the resulting histogram.

BOSS uses a non-symmetric distance function in conjunction with a nearest neighbour classifier. Only the words contained in the test instance's histogram (i.e. the word's count is above zero) are used in the distance calculation, but it is otherwise the Euclidean distance.

Algorithm 1 baseBOSS(A list of n time series of length m, $\mathbf{T} = (\mathbf{X}, \mathbf{y})$)

Parameters: the word length l, the alphabet size α, the window length w, normalisation parameter p

Let $\mathbf{H}$ be a list of n histograms $(\mathbf{h}_1, \ldots, \mathbf{h}_n)$
Let $\mathbf{B}$ be a matrix of l by α breakpoints found by MCB
for $i \leftarrow 1$ to n **do**
  for $j \leftarrow 1$ to $m - w + 1$ **do**
    $\mathbf{s} \leftarrow x_{i,j} \ldots x_{i,j+w-1}$
    if p **then**
      $s \leftarrow$normalise(s)
    $\mathbf{q} \leftarrow$ DFT$(\mathbf{s}, l, \alpha, p)$ { $\mathbf{q}$ *is a vector of the complex DFT coefficients*}
    if p **then**
      $\mathbf{q}' \leftarrow (q_2 \ldots q_{l/2+1})$
    else
      $\mathbf{q}' \leftarrow (q_1 \ldots q_{l/2})$
    $\mathbf{r} \leftarrow$ SFAlookup$(\mathbf{q}', \mathbf{B})$
    if $\mathbf{r} \neq \mathbf{p}$ **then**
      $pos \leftarrow$index$(\mathbf{r})$
      $h_{i,pos} \leftarrow h_{i,pos} + 1$
    $\mathbf{p} \leftarrow \mathbf{r}$

The final classifier is an ensemble of individual BOSS classifiers (parameterised transform plus nearest neighbour classifier) found through first fitting and evaluating a large number of individual classifiers, then retaining only those within 92% accuracy of the best classifier. The BOSS ensemble (also referred to as just BOSS), evaluates and retains the best of all transforms parameterised in the range $w \in \{10 \ldots m\}$ with $m/4$ values where m is the length of the series, $l \in \{16, 14, 12, 10, 8\}$ and $p \in \{true, false\}$. α stays at the default value of 4.

2.2 Contractable BOSS (cBOSS) [11]

Due to its grid-search and method of retaining ensemble members BOSS is unpredictable in its time and memory resource usage, and is impractical for larger problems. cBOSS significantly speeds up BOSS while retaining accuracy by improving how the transform parameter space is evaluated and the ensemble is formed. The main change from BOSS to cBOSS is that it utilises a filtered random selection of parameters to find its ensemble members. cBOSS allows the user to control the build through a time contract, defined as the maximum amount

of time spent constructing the classification model. Algorithm 2 describes the decision procedure for search and maintaining individual BOSS classifiers for cBOSS.

A new parameter k (default 250) for the number of parameter combinations samples is introduced (line 7), of which the top s with the highest accuracy are kept for the final ensemble (lines 13-19). The k parameter is replaceable with a time limit t through contracting. Each ensemble member is built on a subsample of the train data, (line 10) using random sampling without replacement of 70% of the whole training data. An exponential weighting scheme for the predictions of the base classifiers is introduced, to produce a tilted distribution (line 18).

cBOSS was shown to be an order of magnitude faster than BOSS on both small and large datasets from the UCR archive while showing no significant difference in accuracy [11].

Algorithm 2 cBOSS(A list of n cases length m, $\mathbf{T} = (\mathbf{X}, \mathbf{y})$)

Parameters: the number of parameter samples k, the max ensemble size s

1: Let w be window length, l be word length, p be normalise/not normalise and α be alphabet size.
2: Let $\mathbf{C}$ be a list of s BOSS classifiers $(\mathbf{c}_1, \ldots, \mathbf{c}_s)$
3: Let $\mathbf{E}$ be a list of s classifier weights $(\mathbf{e}_1, \ldots, \mathbf{e}_s)$
4: Let $\mathbf{R}$ be a set of possible BOSS parameter combinations
5: $i \leftarrow 0$
6: $lowest\_acc \leftarrow \infty, lowest\_acc\_idx \leftarrow \infty$
7: **while** $i < k$ AND $|\mathbf{R}| > 0$ **do**
8: $[l, a, w, p] \leftarrow random\_sample(\mathbf{R})$
9: $\mathbf{R} = \mathbf{R} \setminus \{[l, a, w, p]\}$
10: $\mathbf{T}' \leftarrow$ subsample_data($\mathbf{T}$)
11: $cls \leftarrow$ baseBOSS($\mathbf{T}', l, a, w, p$)
12: $acc \leftarrow$ LOOCV(cls) { *train data accuracy*}
13: **if** $i < s$ **then**
14: **if** $acc < lowest\_acc$ **then**
15: $lowest\_acc \leftarrow acc$, $lowest\_acc\_idx \leftarrow i$
16: $c_i \leftarrow cls$, $e_i \leftarrow acc^4$
17: **else if** $acc > lowest\_acc$ **then**
18: $c_{lowest\_acc\_idx} \leftarrow cls$, $e_{lowest\_acc\_idx} \leftarrow acc^4$
19: $[lowest\_acc, lowest\_acc\_idx] \leftarrow$ find_new_lowest_acc($\mathbf{C}$)
20: $i \leftarrow i + 1$

2.3 BOSS with Spatial Pyramids (S-BOSS) [8]

BOSS intentionally ignores the locations of words in series, classifying based on the frequency of patterns rather than their location. For some datasets we know that the locations of certain discriminatory subsequences are important, however. Some patterns may gain importance only when in a particular location,

or a mutually occurring word may be indicative of different classes depending on when it occurs. Spatial pyramids [9] bring some temporal information back into the bag-of-words paradigm.

S-BOSS, described in Algorithm 3 and illustrated in figure 1, incorporates the spatial pyramids technique into the BOSS algorithm. S-BOSS creates a standard BOSS transform at the global level (line 6), which constitutes the first level of the pyramid. An additional degree of optimisation is then performed to find the best pyramid height $h \in \{1, 2, 3\}$ (lines 11-16). Height defines the importance of localisation for this transform. Creating the next pyramid level involves creating additional histograms each sub-region of the series at the next scale. Histograms are weighted to give more importance to similarities in the same locations than global similarity, and are concatenated to form an elongated feature vector per instance. The histogram intersection distance measure, more commonly used for approaches using histograms, replaces the BOSS distance for the nearest neighbour classifiers. S-BOSS retains the BOSS ensemble strategy (line 17), such that each S-BOSS ensemble member is a BOSS transform with its own spatial pyramid optimisation plus nearest neighbour classifier.

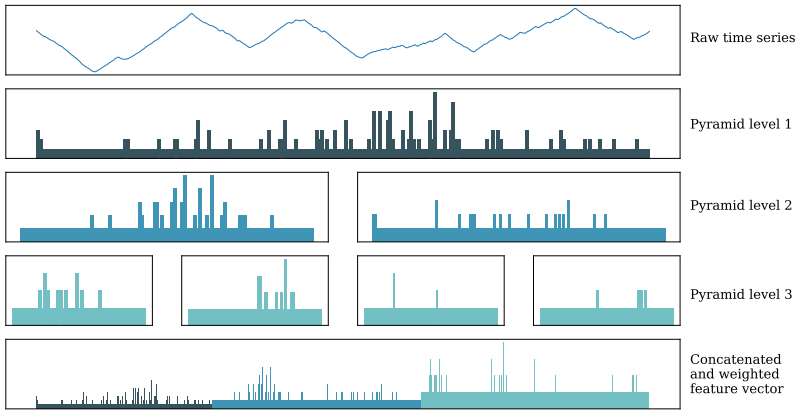

Fig. 1. An example transformation of an OSULeaf instance to demonstrate the additional steps to form S-BOSS from BOSS. Note that each histogram is represented in a sparse manner; the set of words along the x-axis of each histogram at higher pyramid levels may not be equal.

2.4 Word Extraction for Time Series Classification (WEASEL) [14]

Like BOSS, WEASEL performs a Fourier transform on each window, creates words by discretisation, and forms histograms of words counts. It also does this for a range of window sizes and word lengths. However, there are important

Algorithm 3 S-BOSS(A list of n cases length m, $\mathbf{T} = (\mathbf{X}, \mathbf{y})$)

Parameters: the set of possible $[a, w, p]$ parameter combinations $\mathbf{R}$, the set of possible $[l]$ parameter values $\mathbf{L}$, the maximum pyramid height H

Let $\mathbf{C}$ be a list of s BOSS classifiers $(\mathbf{c}_1, \ldots, \mathbf{c}_s)$
for $i \leftarrow 1$ to $|\mathbf{L}|$ **do**
  $bestAcc \leftarrow 0, bestCls \leftarrow \varnothing$
  for $j \leftarrow 1$ to $|\mathbf{R}|$ **do**
    $[a, w, p] \leftarrow \mathbf{R_j}$
    $cls \leftarrow$ baseBOSS$(\mathbf{T}, L_i, a, w, p)$
    $acc \leftarrow$ LOOCV(cls) {*train data accuracy*}
    if $acc > bestAcc$ **then**
      $bestAcc \leftarrow acc, bestCls \leftarrow cls$
  $cls \leftarrow bestCls$
  for $h \leftarrow 1$ to H **do**
    $cls \leftarrow divideAndConcatenateBags(cls)$
    $acc \leftarrow$ LOOCV(cls) {*train data accuracy*}
    if $acc > bestAcc$ **then**
      $bestAcc \leftarrow acc, bestCls \leftarrow cls$
  $\mathbf{C}_i \leftarrow bestCls$
$keepWithinBest(\mathbf{C}, 0.92)$ {*keep those cls with train accuracy within 0.92 of the best*}

differences. WEASEL is not an ensemble nearest neighbour classifiers. Instead, WEASEL constructs a single feature space from concatenated histograms for different parameter values, then uses logistic regression and feature selection. Histograms of individual words and bigrams of the previous non-overlapping window for each word are used. Fourier terms are selected for retention by the application of an F-test. The retained values are then discretised into words using information gain binning (IGB), similar to the MCB step in BOSS. The number of features is further reduced using a chi-squared test after the histograms for each instance are created, removing any words which score below a threshold. It performs a parameter search for p (whether to normalise or not) and over a reduced range of l, using a 10-fold cross-validation to determine the performance of each set. The alphabet size α is fixed to 4 and the *chi* parameter is fixed to 2. Algorithm 4 gives an overview of WEASEL, although the formation and addition of bigrams is omitted for clarity.

3 Temporal Dictionary Ensemble (TDE)

The easiest algorithms to combine are cBOSS and S-BOSS. cBOSS speeds up BOSS through subsampling training cases and random parameter selection. The number of levels parameter introduced by S-BOSS can be included in the random parameter selection used by cBOSS. For comparisons we call this naive hybrid of algorithms cS-BOSS. We use this as a baseline to justify the extra complexity we introduce in TDE. Algorithm 5 provides an overview of the ensemble build process for TDE, which follows the general structure and weighting

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

Parameters: the word length l, the alphabet size α, the maximal window length w_{max}, mean normalisation parameter p

Let $\mathbf{H}$ be the histogram $\mathbf{h}$
Let $\mathbf{B}$ be a matrix of l by α breakpoints found by MCB using information gain binning
for $i \leftarrow 1$ to n **do**
  for $w \leftarrow 2$ to w_{max} **do**
    for $j \leftarrow 1$ to $m - w + 1$ **do**
      $\mathbf{o} \leftarrow x_{i,j} \ldots x_{i,j+w-1}$
      $\mathbf{q} \leftarrow$ DFT(o, w, p) { $\mathbf{q}$ *is a vector of the complex DFT coefficients*}
      $\mathbf{q}' \leftarrow$ ANOVA-F(q, l, y) { *use only the* $\mathbf{l}$ *most discriminative ones*}
      $\mathbf{r} \leftarrow$ SFAlookup$(\mathbf{q}', \mathbf{B})$
      $pos \leftarrow$index$(\mathbf{w}, \mathbf{r})$
      $h_{i,pos} \leftarrow h_{i,pos} + 1$
$h \leftarrow \chi^2(h, y)$ { *feature selection using the chi-squared test* }
fitLogistic(h, y)

scheme of cBOSS. The classifier returned by improvedBaseBOSS includes spatial pyramids and also includes the following enhancements taken from both S-BOSS and WEASEL. Like S-BOSS, it uses the histogram intersection distance measure which has been shown to be more accurate than BOSS distance [8]. It uses bigram frequencies in the same way as WEASEL. Base classifiers can use either IGB from WEASEL or MCB from BOSS in the discretisation. TDE samples parameters from the range given in Table 1 using a method sampleParameters.

Table 1. Parameter ranges for TDE base classifier selection.

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

3.1 Gaussian process parameter selection

The increase in the parameter search space caused by the inclusion of pyramids and IGB parameters makes the random parameter selection used by cBOSS less effective. Instead, TDE uses a guided parameter selection for ensemble members inspired by Bayesian optimisation [16]. A Gaussian process model is built over the regressor parameter space $\mathbf{R}$ for parameters $[l, a, w, p, h, b]$ to predict the accuracy, using the previously observed (parameter, accuracy) pairs $\mathbf{G}$.

Algorithm 5 TDE(A list of n cases length m, $\mathbf{T} = (\mathbf{X}, \mathbf{y})$)

Parameters: the number of parameter samples k, the max ensemble size s

Let w be window length, l be word length, p be normalise/not normalise, α be alphabet size, h be number of pyramid levels and b be MCB or IGB discretisation.
Let $\mathbf{C}$ be a list of s BOSS classifiers $(\mathbf{c}_1, \ldots, \mathbf{c}_s)$
Let $\mathbf{E}$ be a list of s classifier weights $(\mathbf{e}_1, \ldots, \mathbf{e}_s)$
Let $\mathbf{G}$ be a list of k BOSS parameter and accuracy pairs $(\mathbf{g}_1, \ldots, \mathbf{g}_k)$
Let $\mathbf{R}$ be a set of possible BOSS parameter combinations
$i \leftarrow 0$
$lowest\_acc \leftarrow \infty, lowest\_acc\_idx \leftarrow \infty$
while $i < k$ AND $|\mathbf{R}| > 0$ **do**
  $[l, a, w, p, h, b] \leftarrow$ chooseParameters($\mathbf{R}$,$\mathbf{G}$,i)
  $\mathbf{R} = \mathbf{R} \setminus \{[l, a, w, p, h, b]\}$
  $\mathbf{T}' \leftarrow$ subsampleData($\mathbf{T}$)
  $cls \leftarrow$ improvedBaseBOSS($\mathbf{T}', l, a, w, p, h, b$)
  $acc \leftarrow$ LOOCV(cls) { *train data accuracy*}
  if $i < s$ **then**
    if $acc < lowest\_acc$ **then**
      $lowest\_acc \leftarrow acc$, $lowest\_acc\_idx \leftarrow i$
    $c_i \leftarrow cls$, $e_i \leftarrow acc^4$
  else if $acc > lowest\_acc$ **then**
    $c_{lowest\_acc\_idx} \leftarrow cls$, $e_{lowest\_acc\_idx} \leftarrow acc^4$
    $[lowest\_acc, lowest\_acc\_idx] \leftarrow$ findNewLowestAcc($\mathbf{C}$)
  $g_i \leftarrow \{[l, a, w, p, h, b], acc\}$
  $i \leftarrow i + 1$

A Gaussian Process [17] describes a distribution over functions, $f(\boldsymbol{x}) \sim \mathcal{GP}(m(\boldsymbol{x}, k(\boldsymbol{x}, \boldsymbol{x}')))$, characterised by a mean function, $m(\boldsymbol{x})$, and a covariance function, $k(\boldsymbol{x}, \boldsymbol{x}')$, such that

$$m(\boldsymbol{x}) = \mathbb{E}\left[f(\boldsymbol{x})\right],$$
$$k(\boldsymbol{x}, \boldsymbol{x}') = \mathbb{E}\left[(f(\boldsymbol{x}) - m(\boldsymbol{x}))(f(\boldsymbol{x}') - m(\boldsymbol{x}'))\right],$$

where any finite collection of values has a joint Gaussian distribution. Commonly the mean function is constant, $m(\boldsymbol{x}) = \gamma$, or even zero, $m(\boldsymbol{x}) = 0$. The covariance function $k(\boldsymbol{x}, \boldsymbol{x}')$ encodes the expected similarity of the function evaluated at pairs of input-space vectors, $\boldsymbol{x}$ and $\boldsymbol{x}'$. For example, the squared exponential covariance function,

$$k(\boldsymbol{x}, \boldsymbol{x}') = \sigma_f^2 \exp\left\{-\frac{(\boldsymbol{x} - \boldsymbol{x}')^2}{2\ell^2}\right\},$$

encodes a preference for smooth functions, where ℓ is a hyper-parameter that specifies the characteristic length-scale of the covariance functions (large values yield smoother functions) and σ_f governs the magnitude of the variance.

Typically in a regression setting the response variables of the training samples, $\mathcal{D} = \{(\boldsymbol{x}_i, y_i) \mid i = 1, 2, \ldots, n\}$, are assumed to be realisations of a deter-

ministic function that have been corrupted by additive Gaussian noise, i.e.

$$y_i = f(\boldsymbol{x}_i) + \varepsilon_i, \qquad \text{where} \qquad \varepsilon_i \sim \mathcal{N}\left(0, \sigma_n^2\right).$$

In that case, the joint distribution of the training sample, and a single test point, $\boldsymbol{x}_*$, is given by,

$$\begin{bmatrix} \boldsymbol{y} \\ f_* \end{bmatrix} \sim \mathcal{N}\left(\boldsymbol{0}, \begin{bmatrix} \boldsymbol{K} + \sigma_n^2 \boldsymbol{I} & \boldsymbol{k}_* \\ \boldsymbol{k}_*^T & k(\boldsymbol{x}_*, \boldsymbol{x}_*) \end{bmatrix}\right),$$

where $\boldsymbol{K}$ is the matrix of pairwise evaluation of the covariance function for all points belonging to the training sample and $\boldsymbol{k}_*$ is a column vector of the evaluation of the covariance function for the test point and each of the training points. The Gaussian predictive distribution for the test point is then specified by

$$\bar{f}_* = \boldsymbol{k}_*^T \left(\boldsymbol{K} + \sigma_n^2 \boldsymbol{I}\right)^{-1} \boldsymbol{y},$$
$$\mathbb{V}\left[f_*\right] = k(\boldsymbol{x}_*, \boldsymbol{x}_*) - \boldsymbol{k}_*^T \left(\boldsymbol{K} + \sigma_n^2 \boldsymbol{I}\right)^{-1} \boldsymbol{k}_*.$$

The hyper-parameters of the Gaussian process can be handled by tuning them, often via maximisation of the marginal likelihood, or by full Bayesian marginalisation, using an appropriate hyper-prior distribution. For further details, see Williams and Rasmussen [17]. We use a basic form of GP and treat all the regressors (TDE parameters) as continuous. The bestPredictedParameters operation in line 5 of Algorithm 6 is limited to the same parameter ranges used for random search given in Table 1.

Algorithm 6 chooseParameters($\mathbf{R}$,$\mathbf{G}$,i)

1: **if** $i < 50$ **then**
2: $[l, a, w, p, h, b] \leftarrow$ randomSample($\mathbf{R}$)
3: **else**
4: $gp \leftarrow$ buildGaussianProcesses($\mathbf{G}$)
5: $[l, a, w, p, h, b] \leftarrow$ bestPredictedParameters($\mathbf{R}, gp$)
6: **return** $[l, a, w, p, h, b]$

4 Results

Our experiments are run on 112 datasets from the recently expanded UCR/UEA archive [3], removing any datasets that are unequal length or contain missing values. We also remove the dataset Fungi as it only provides a single train case for each class. For each classifier-dataset combination we run 30 stratified resamples, with the first sample being the original train test split. For reproducability each dataset resample and classifier is seeded to its resample number. All experiments were run single threaded on a high performance computing cluster with

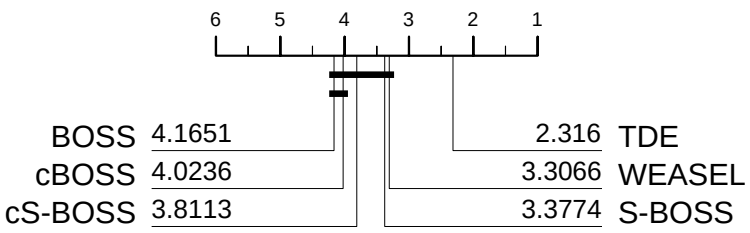

Fig. 2. Critical difference diagram for six dictionary based classifiers on 107 UCR time series classification problems. Full results are available on the accompanying website.

a run time limit of 7 days. We used an open source Weka compatible code base called tsml for experimentation[1]. Implementations of BOSS, cBOSS, S-BOSS, WEASEL, HIVE-COTE and TS-CHIEF provided by the algorithm inventors are all available in tsml. InceptionTime and ROCKET experiments were run using the Python based package sktime and a deep learning extension thereof[2].

Guidance on how to recreate the resamples and code to reproduce the results is available on the accompanying website[3]. We also provide results for all classifiers used in experimentation.

Our experiments are designed to test whether TDE is better in terms of predictive performance and run time than other dictionary based classifiers, and whether it improves HIVE-COTE when it replaces BOSS in the meta ensemble HIVE-COTE.

4.1 TDE vs other dictionary classifiers

For the dictionary classifiers, we were only able to obtain complete results for 107 of the 112 datasets. This was due to the long run time of S-BOSS and WEASEL. For consistency, in this Section we only present results with these datasets. The missing problems are: ElectricDevices; FordA; FordB; HandOutlines; and NonInvasiveFetalECGThorax2.

Figure 2 shows a a critical difference diagram [5] for the six dictionary based classifiers considered. The number on each line is the average rank of an algorithm over 107 UCR datasets (lower is better). The solid bars are cliques. There is no detectable significant difference between classifiers in the same clique. Comparison of the performance of classifiers is done using pairwise Wilcoxon signed rank tests and cliques are formed using the Holm correction, following recommendations from [2] and [7].

TDE is significantly more accurate than all other classifiers. There are then two cliques: BOSS and cBOSS are significantly worse than S-BOSS, cS-BOSS

[1] https://github.com/uea-machine-learning/tsml
[2] https://github.com/sktime
[3] https://sites.google.com/view/ecmlpkdd-tde/home

and WEASEL. This also confirms that the simple hybrid cS-BOSS is no better than S-BOSS in terms of accuracy.

Table 2 summarises the run time of these classifiers. All the classifiers can complete on most of the UCR problems in minutes. S-BOSS is the slowest algorithm. Four problems are not included in this study because either S-BOSS or WEASEL did not to finish within seven days. For WEASEL this was caused by the requirement to cross validate to estimate the accuracy for HIVE-COTE. S-BOSS is just relatively slow. We omit these four problems from all analysis for completeness. All experiments are conducted sequentially on a single processor. TDE completes all the problems in under a day. It is faster than WEASEL and considerably faster than S-BOSS.

Table 2. Summary of run time for six dictionary based classifiers over 107 UCR problems. The median time over 30 resamples is used for each dataset.

Classifier	Max run time (hrs)	Total run time (hrs)
BOSS	11.33	52.10
cBOSS	0.63	3.74
S-BOSS	34.11	148.82
cS-BOSS	2.91	12.62
WEASEL	4.86	28.50
TDE	5.67	21.73

The max run timings for BOSS and S-BOSS demonstrate the problem with the traditional BOSS algorithm addressed by cBOSS and cS-BOSS: the ensemble design means they may require a long runtime, and it is not very predictable when this will happen. Figure 3 shows the scatter plot of runtime for BOSS vs TDE and demonstrates that TDE scales much better than BOSS.

4.2 TDE with HIVE-COTE

TDE is significantly more accurate than all other dictionary based time series classification algorithms, and faster than the best of the rest, S-BOSS and WEASEL. We believe there is merit in finding the best single representation classifier because there will be occasions when domain knowledge would recommend a single approach. However, with no domain knowledge, the state of the art in time series classification involves hybrids built on multiple representations, or deep learning to fit a bespoke representation. HIVE-COTE is a meta ensemble of classifiers built using different representations.

All HIVE-COTE variants used in our experiments are built with four components: the random interval spectral ensemble (RISE), shapelet transform classifier (STC) and time series forest (TSF) plus one other classifier (see [10] for details). We omit the elastic ensemble because it is infeasible to run it on a large number of problems. The other components are relatively fast and can complete

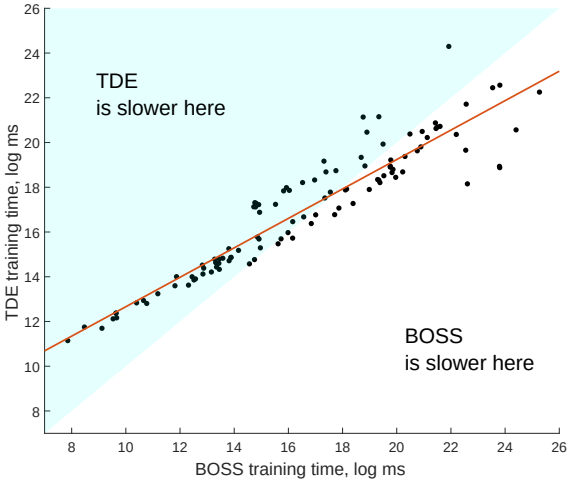

Fig. 3. Pairwise scatter diagram, on log scale, of TDE and BOSS training times. TDE has larger overheads which make it slower on smaller problems, but it scales much better towards larger problems.

a single resample of one problem in under a day for all problems. The latest version of STC on the open source repository tsml does not do a full enumeration of the shapelet space. Instead, it randomly samples shapelets for a fixed time. We set this to 4 hours for all experiments with STC.

With these other settings fixed, we have reconstructed HIVE-COTE using TDE instead of BOSS. We call this HC-TDE for differentiation purposes.

TS-CHIEF [15] is a tree ensemble that embeds dictionary, spectral and distance based representations, and is set to build 500 trees. InceptionTime [6] is a deep learning ensemble, combining 5 homogeneous networks each with random weight initialisations for stability. ROCKET [4] uses a large number, 10,000, of randomly parameterised convolution kernels in conjunction with a linear ridge

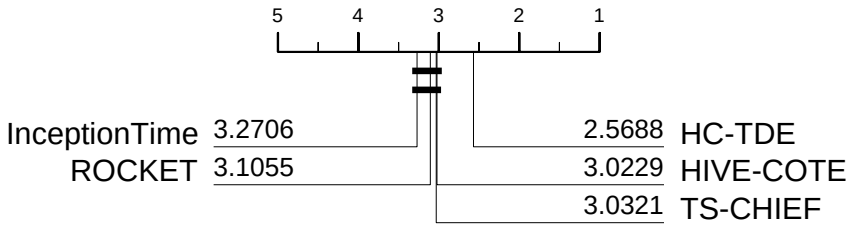

Fig. 4. Critical difference diagram for six dictionary based classifiers on 109 UCR time series classification problems. Full results are available on the accompanying website.

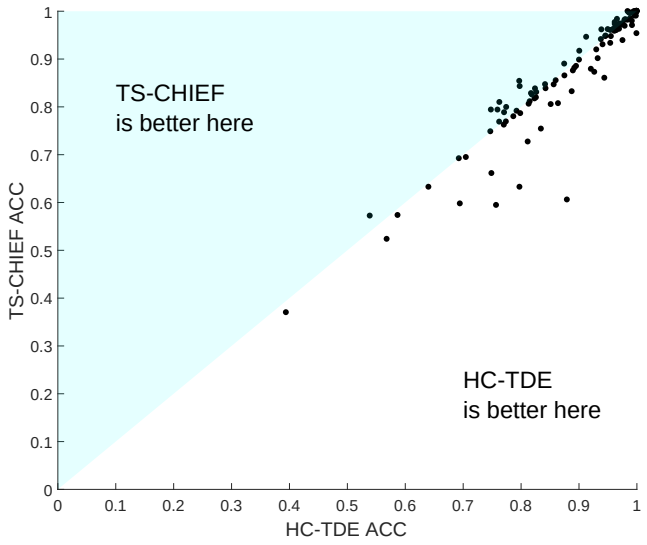

Fig. 5. Scatter plot of TS-CHIEF against HC-TDE. HC-TDE wins on 62, draws on 6 and loses on 41 data sets.

regression classifier. We use the configurations of each classifier described in their respective publications.

Figure 4 shows the ranked performance of HC-TDE against HIVE-COTE, TS-CHIEF, InceptionTime and ROCKET on 109 problems. We are missing three data, HandOutlines, NonInvasiveFetalECGThorax1 and NonInvasiveFetalECGThorax2 because TS-CHIEF could not complete them within the seven day limit.

HC-TDE is significantly better than all four algorithms currently considered state of the art. The actual differences between HIVE-COTE and HC-TDE are understandably small, given their similarities. However, they are consistent: replacing BOSS with TDE improves HIVE-COTE on 69 problems, and makes it worse on just 32 (with 8 ties). HC-TDE does show significant variation to TS-CHIEF (see Figure 5) and is on average over 1% more accurate. HC-TDE is the top performing algorithm using a range of performance measures such as AUROC, F1 and balanced accuracy (see accompanying website). The improvement over InceptionTime is even greater: it is on average 1.5% more accurate.

It is worth considering whether replacing BOSS with either S-BOSS or WEASEL would give as much improvement to HIVE-COTE as TDE does. We replaced BOSS with WEASEL (HC-WEASEL) and S-BOSS (HC-S-BOSS). Figure 6 shows the performance of these relative to HC-TDE, InceptionTime and TS-CHIEF. Whilst it is true that HC-S-BOSS is not significantly worse than HC-TDE, it is also not significantly better than the current state of the art. HC-

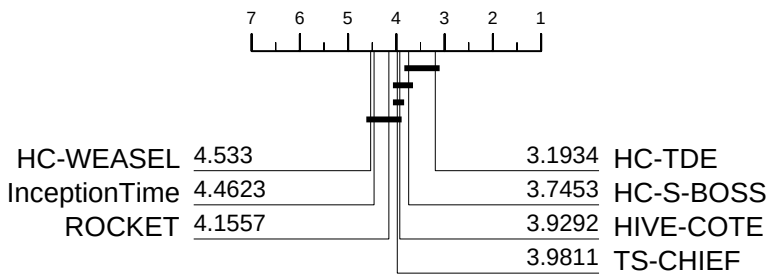

Fig. 6. Critical difference diagram for seven classifiers on 106 UCR time series classification problems. Full results are available on the accompanying website.

WEASEL does not perform well. We speculate that this is because the major differences in WEASEL mean that its improvement is at problems better suited to other representations, and this improvement comes at the cost of worse performance at problems suited to dictionary classifiers.

5 Conclusion

TDE combines the best elements of existing dictionary based classifiers with a novel method of improving the ensemble through a Gaussian process model for parameter selection. TDE is more accurate and scalable than current top performing dictionary algorithms. When we replace BOSS with TDE in HIVE-COTE, the resulting classifier is significantly more accurate than the current state of the art. TDE has some drawbacks. It is memory intensive. It requires about three times more memory than BOSS, and the maximum memory required was 10 GB for ElectricDevices. Like all nearest neighbour classifiers, TDE is relatively slow to classify new cases. If fast predictions are required, WEASEL may be preferable. Future work will focus on making TDE more scalable.

Acknowledgements

This work is supported by the UK Engineering and Physical Sciences Research Council (EPSRC) iCASE award T206188 sponsored by British Telecom. 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.

References

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

2. Benavoli, A., Corani, G., Mangili, F.: Should we really use post-hoc tests based on mean-ranks? The Journal of Machine Learning Research **17**(1), 152–161 (2016)
3. Dau, H.A., Bagnall, A., Kamgar, K., Yeh, C.C.M., Zhu, Y., Gharghabi, S., Ratanamahatana, C.A., Keogh, E.: The ucr time series archive. IEEE/CAA Journal of Automatica Sinica **6**(6), 1293–1305 (2019)
4. Dempster, A., Petitjean, F., Webb, G.I.: Rocket: Exceptionally fast and accurate time series classification using random convolutional kernels. arXiv preprint arXiv:1910.13051 (2019)
5. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. Journal of Machine learning research **7**(Jan), 1–30 (2006)
6. Fawaz, H.I., Lucas, B., Forestier, G., Pelletier, C., Schmidt, D.F., Weber, J., Webb, G.I., Idoumghar, L., Muller, P.A., Petitjean, F.: Inceptiontime: Finding alexnet for time series classification. arXiv preprint arXiv:1909.04939 (2019)
7. Garcia, S., Herrera, F.: An extension on"statistical comparisons of classifiers over multiple data sets"for all pairwise comparisons. Journal of machine learning research **9**(Dec), 2677–2694 (2008)
8. Large, J., Bagnall, A., Malinowski, S., Tavenard, R.: On time series classification with dictionary-based classifiers. Intelligent Data Analysis **23**(5), 1073–1089 (2019)
9. Lazebnik, S., Schmid, C., Ponce, J.: Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In: 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06). vol. 2, pp. 2169–2178. IEEE (2006)
10. Lines, J., Taylor, S., Bagnall, A.: Time series classification with hive-cote: The hierarchical vote collective of transformation-based ensembles. ACM Transactions on Knowledge Discovery from Data (TKDD) **12**(5), 52 (2018)
11. Middlehurst, M., Vickers, W., Bagnall, A.: Scalable dictionary classifiers for time series classification. In: International Conference on Intelligent Data Engineering and Automated Learning. pp. 11–19. Springer (2019)
12. Schäfer, P.: The boss is concerned with time series classification in the presence of noise. Data Mining and Knowledge Discovery **29**(6), 1505–1530 (2015)
13. Schäfer, P., Högqvist, M.: Sfa: a symbolic fourier approximation and index for similarity search in high dimensional datasets. In: Proceedings of the 15th International Conference on Extending Database Technology. pp. 516–527 (2012)
14. Schäfer, P., Leser, U.: Fast and accurate time series classification with weasel. In: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management. pp. 637–646 (2017)
15. Shifaz, A., Pelletier, C., Petitjean, F., Webb, G.I.: Ts-chief: a scalable and accurate forest algorithm for time series classification. Data Mining and Knowledge Discovery pp. 1–34 (2020)
16. Snoek, J., Larochelle, H., Adams, R.P.: Practical bayesian optimization of machine learning algorithms. In: Advances in neural information processing systems. pp. 2951–2959 (2012)
17. Williams, C.K., Rasmussen, C.E.: Gaussian processes for machine learning, vol. 2. MIT press Cambridge, MA (2006)