The Contract Random Interval Spectral Ensemble (c-RISE): the effect of contracting a classifier on accuracy

Michael Flynn, James Large, and Tony Bagnall

University of East Anglia

Abstract. The Random Interval Spectral Ensemble (RISE) is a recently introduced tree based ensemble time series classification algorithm, where each tree is built on a distinct set of Fourier, autocorrelation and partial autocorrelation features. It is a component in the meta ensemble HIVE-COTE [9]. RISE has run time complexity of $O(nm^2)$, where m is the series length and n the number of train cases. This is prohibitively slow when considering long series, which are common in problems such as audio classification, where spectral approaches are likely to perform better than classifiers built in the time domain. We propose an enhancement of RISE that allows the user to specify how long the algorithm can have to run. The contract RISE (c-RISE) allows for check-pointing and adaptively estimates the time taken to build each tree in the ensemble through learning the constant terms in the run time complexity function. We show how the dynamic approach to contracting is more effective than the static approach of estimating the complexity before executing, and investigate the effect of contracting on accuracy for a range of large problems.

Keywords: time series classification · spectral features · contract classifier.

1 Introduction

Data sets of increasing size are now common within machine learning. Big data undeniably has its benefits. However, as advancements in processing capabilities begin to slow and the complexity of algorithms increase, we are often faced with more data than we are capable of processing. Even with the rise in popularity of cloud computing platforms and high performance computer facilities, it often becomes infeasible to construct a full learning model on all of the available data. A particularly common area in which the problem arises is the spectral/audio domain. This is typically down to the sample rate used to record data. Consider that the standard audio sample rate is 44.1kHz. Creating models from audio data requires either extensive bespoke preprocessing or adaptations of the learning algorithms to compensate for the volume. We do not look to challenge or reaffirm the traditional, volume of data/increase in accuracy paradigm. Instead, we aim to investigate the relationship between reduced train time and accuracy, assuming a fixed volume of data.

All experimental processes strive to make complete use of the training set and in ideal conditions this will always be preferable. However, in our work with large datasets we have experienced first hand the problems of extreme train and test times of approaches. In working through these problems it has become apparent that very little research has been undertaken in understanding how reduced training time affects accuracy. Homogeneous ensembles typically require a large number of trees to be effective. The most basic way of managing train times is simply to build base models until the time has expired. However, for very large problems, this may result in very small ensembles. The Random Interval Spectral Ensemble (RISE) is a Time Series Classification (TSC) algorithm that uses spectral features. It selects a different random interval of the series for each base classifier, then calculates spectral coefficients to be used as features. For large problems, if we happen to select intervals close to the full series length, we can use all available computation on very few models. To compensate for this, we investigate whether we can predict the run time, then use this prediction to guide the interval sampling, ensuring a minimum size ensemble.

Our aims are twofold; firstly, we aim to make RISE more useful by making it a contract classifier, i.e. a classifier where you can specify approximately the amount of computational time allowed to build the model. Secondly, we aim to compare the basic approach and our adaptive, dynamic approach with respect to their effect on accuracy and ability to adhere to the time contract.

In section 2 we describe RISE in detail and further motivate the need for contracting. In section 3 we introduce our contract version, c-RISE, and describe both timing models. In section 4 we outline the experimental procedure and present the results and in section 6 we conclude.

2 The RISE algorithm

RISE draws on ideas from tree-based ensembles such as random forest [4] and the TSC interval feature classifier time series forest (TSF) [6]. Like TSF, we build trees on random intervals from the data to construct a random forest classifier. A key difference 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. Once we have derived the spectral features, we build a decision tree using the random tree algorithm used by random forest. The base RISE algorithm is described in Algorithm 1. The first tree in RISE is a special case that uses the whole series. We include this step for continuity with the previous spectral classifiers used in The Collective of Transformation-Based Ensembles [3] (COTE) classifier which only used the whole series. The procedure for building RISE is outlined in Algorithm 1.

RISE uses several forms of spectral features: the power spectrum, the autocorrelation function, the partial autocorrelation and the autoregressive model. New classes are classified using a simple majority vote. Further details can be found in [9]. The run time for transforming a series is quadratic in the interval length.

Algorithm 1 BuildRISE(Training data *train*, number of classifiers r, minimum interval length *minLen*)

```
1: Let $\mathbf{F} \leftarrow < F_1 \ldots F_r >$ be the trees in the forest.
2: Let $m$ be the length of series in $train$
3: $wholeSeriesFeatures \leftarrow$ getSpectralFeatures($train$)
4: buildRandomTreeClassifier($\mathbf{F}_1$,$wholeSeriesFeatures$)
5: for $i \leftarrow 2$ to $r$ do
6:     $startPos \leftarrow$ randBetween($1, m - minLen$)
7:     $endPos \leftarrow$ randBetween($startPos + minLen, m$)
8:     $train \leftarrow$ removeAttributesOutsideOfRange($train$,$startPos$,$endPos$)
9:     $intervalFeatures \leftarrow$ getSpectralFeatures($train$)
10:    buildRandomTreeClassifier($\mathbf{F}_i$,$intervalFeatures$)
```

3 The Contracted RISE algorithm

In many areas it may be advantageous or even necessary to constrict the run time of a classification algorithm. Generally, it is not well understood how long classification algorithms take to run for a given problem. It is of practical importance when considering which algorithm to use or how much preprocessing to perform. This is of particular difficulty when using cloud services where the computation is charged for per time period, or situations in which there is a hard deadline or where there is a limit on how long a process is allowed to run. Two solutions to these problems are check-pointing, periodically saving a partial version of the classification model to disk, and contracting, limiting the the amount of computational time an algorithm is allowed. Used together, they make a classifier more flexible and useful to the practitioner. Check-pointing RISE is simple, especially with a Java implementation; we can simply serialise the constructed trees at certain points and adapt RISE to allow the loading from file. Contracting is also simple: we can keep building trees until we run out of time or reach the maximum number. However, this simple contracting approach can result in very small ensembles if the series are very long. We propose an adaptive scheme that avoids this problem by dynamically estimating the build time for each particular tree.

3.1 Algorithmic Improvements

A number of small but influential changes are implemented in c-RISE, with the goal of not significantly decreasing accuracy whilst drastically improving runtime. These changes are outlined below and a more thorough description provided in Algorithm 3.

RISE uses power spectrum (PS), autocorellation function (ACF), partial autocorellation function (PACF) and autoregressive model (AR) features over each interval. These features are clearly related, but it was found that combining them created a more accurate classifier than just using one set [9]. However, the drawback is that although the PS can be found in $O(nlog(n))$ time if the series length is a power of 2, there is no simple way to do this for the PACF and AR terms. Hence c-RISE does not derive PACF or AR features, and only selects intervals that are a power of 2. An interval is still selected randomly, but now it is rounded to the nearest power of two. To correct for intervals exceeding the series length, the interval is then divided by 2, ensuring a valid interval and favouring shorter intervals.

3.2 Timing models

Non-adaptive model. The simplest way to limit the train time of tree based ensemble is to simply set a timer and keep adding trees until the contract is met, or a maximum number of trees have been built. This is described in Algorithm 2.

Algorithm 2 Build c-RISE_Non-adaptive(Training data *train*, number of classifiers *r*, minimum interval length *min*)

```
1: Let $\mathbf{F} \leftarrow < F_1 \ldots F_{500} >$ be the trees in the forest.
2: Let $m$ be the length of series in $train$
3: startForestTimer()
4: for $i \leftarrow 1$ to 500 AND queryForestTimer() do
5:     $validLengths \leftarrow$ getValidPowersOf2($min$, $instanceLength$)
6:     $randomLength \leftarrow$ randBetween(maxValue($validLengths$)/2)
7:     $r \leftarrow$ findClosest($validLengths, randomLength$)
8:     $startPos \leftarrow$ randBetween$(1, m - r)$
9:     $interval \leftarrow$ removeAttributesOutsideOfRange($train, startPos, r$)
10:    $intervalFeatures \leftarrow$ getSpectralFeatures($interval$)
11:    buildRandomTreeClassifier($\mathbf{F}_i$,$intervalFeatures$)
12:    updateTimer()
```

Adaptive model c-RISE performs two transformations: A discrete Fourier transform (DFT) to find the power spectrum and construction of the autocorrelation function (ACF). With the simplest implementation, each of these is $O(r^2)$, where r is the interval width. We can improve the efficiency of the Fourier transform to $O(rlog(r))$ by using the fast Fourier Transform (FFT). To gain the full benefit we restrict c-RISE to intervals of length the power of 2. However, the best average case complexity for the ACF is $O(r^2)$. Hence, the transformations will dominate the runtime in relation to the decision tree, so we can reasonably model the runtime t for a single member of the ensemble of interval length r as

$$t = a \cdot r^2 + b \cdot r + c.$$

If we fix the contract time t and knew the constant factors a, b and c we could find the positive root of the quadratic and use that as the maximum allowable interval for the tree. The quadratic terms will of course be both problem and hardware dependent. Hence, we use an adaptive algorithm to learn these parameters. For each tree we build, we record the selected interval and the observed run time. Using is data, we refit a least squares linear regression model. For clarity, we let $x_1 = r_1^2$ and $x_2 = r_1$, our dependent variable matrix is then,

$$X = \begin{bmatrix} 1, x_{11}, x_{12} \\ 1, x_{21}, x_{22} \\ \ldots \\ 1, x_{k1}, x_{k2} \end{bmatrix}$$

the estimates of the parameters are $B = (\hat{a}, \hat{b}, \hat{c})^T$ and our response variable is $Y = (y_1, y_2, \ldots, y_k)^T$. The least squares estimates are then,

$$B = (X^T X)^{-1} X^T Y.$$

Since $(X^T X)$ is based on sums of squares, we do not have to recalculate if from scratch each time. It is also possible to update $(X^T X)^{-1}$ online with the Sherman-Morrison formula, but we leave that for future work. After the construction of each tree, we update the remaining contracted time t, re-estimate the coefficients $t = \hat{a} \cdot r^2 + \hat{b} \cdot r + \hat{c}$, and calculate a new maximum allowable interval r. This is used as the maximum for the next iteration.

Algorithm 3 Build c-RISE_Adaptive(Training data *train*, number of classifiers r, minimum interval length *min*)

```
 1: Let $\mathbf{F} \leftarrow < F_1 \ldots F_{500} >$ be the trees in the forest.
 2: Let $m$ be the length of series in $train$
 3: startForestTimer()
 4: for $i \leftarrow 1$ to 500 AND queryForestTimer() do
 5:     startTreeTimer()
 6:     buildAdaptiveModel()
 7:     $max \leftarrow$ findMaxIntervalLength()
 8:     $validLengths \leftarrow$ getValidPowersOf2($min$, $max$)
 9:     $randomLength \leftarrow$ randBetween(maxValue($validLengths$)/2)
10:     $r \leftarrow$ findClosest($validLengths, randomLength$)
11:     $startPos \leftarrow$ randBetween$(1, m - r)$
12:     $interval \leftarrow$ removeAttributesOutsideOfRange($train, startPos, r$)
13:     $intervalFeatures \leftarrow$ getSpectralFeatures($interval$)
14:     buildRandomTreeClassifier($\mathbf{F}_i$,$intervalFeatures$)
15:     $y \leftarrow$ queryTreeTimer()
16:     updateAdaptiveModel($r, y$)
```

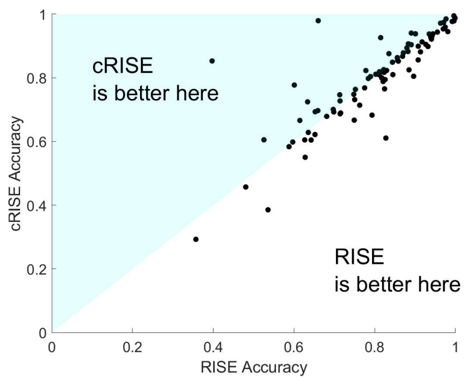
4 Results

In order to ensure the following results are comparable, all approaches were evaluated on 10 stratified random re-samples, of the same 85 datasets from the

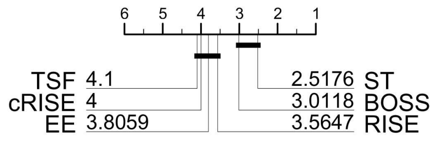
UCR archive. All code is available from the UEA TSC repository[1], whereas, raw results and analysis can can be downloaded from this repository[2]. The experimental pipeline and algorithms used were all implemented in Java and the experiments were carried out on the HPC system at the University of East Anglia.

4.1 RISE vs c-RISE

RISE has been shown to be significantly better than other spectral based approaches on the TSC archive data and on simulated data [2] and therefore was selected as the spectral component for HIVE-COTE. However, RISE is computationally expensive, since each transformed series is based on an O(r^2)operation (finding the PACF), where r is the series length. This is further impacted by the derivation of auto regressive and spectral features. In a summary of experiments over 82 datasets from the UCR archive it was concluded that these features do not significantly effect accuracy, as shown in figure 1. However computation of these features represent significant complexity in the algorithm and as such they have a detrimental effect on runtime. Table 1 presents both the mean, median train time for RISE and cRISE over 85 datasets from the UCR archive.



(a)



(b)

Fig. 1. (a) A pairwise scatter diagram showing average train time over 10 random re-samples of 85 UCR datasets for both RISE and cRISE. (b) A pairwise critical difference diagram showing the ranks of TSF [6], cRISE, EE [7], RISE [9], BOSS [10] and ST [8] over the same 10 random resamples of 85 UCR datasets.

However, on investigation of the impact each of these features has on accuracy and runtime, it became apparent that they did not contribute equally.

[1] https://github.com/TonyBagnall/uea-tsc
[2] https://tinyurl.com/y3robhbk

Fundamentally cRISE behaves in the same manner as RISE when not under contract. This allows us to attribute any changes in runtime or accuracy to the removed transformations. The impact on runtime when deriving the PACF and AR features is unsurprising. Figure 1 (b) is a critical difference diagram displaying various state of the art classifiers, including both RISE and cRISE. The impact of removing PACF and AR features has on accuracy was unexpected. The outcome of these experiments was the removal of PACF and AR derivations.

Table 1. A table showing mean and median train times of RISE and cRISE over all UCR datasets, as well as average speed up.

	cRISE	RISE	Difference
Mean (seconds)	144	3554	3410
Median (seconds)	145	3794	3649

Moving forward all experimental results are achieved with the updated architecute of cRISE.

4.2 Naive vs Adaptive Timing models

In this section we evaluate how the accuracy of cRISE changes as a function of total training time for both approaches. We also asses how well each approach adheres to the contract itself.

In order to achieve this, nine pairs of experiments were carried out. For both approaches a contract is set representing 10% - 90% total training time per dataset in 10% increments. This allowed us to examine changes in accuracy at 10 evenly spaced points in time as well test each approaches ability to stay within the contract.

Figure 2 (a) shows how the actual train time changes over different contracts. Each point represents the mean train time over 85 datasets over 10 folds for each contract. The contracts themselves are defined as a percentage of full train time per dataset. This represents 8,500 experiments per approach over 10 contract percentages.

Figure 2 (a) also shows that the Adaptive approach displays much more predictable behaviour in the context of adhering to contract time. Adhering to relatively small contract times presents more of a challenge to both approaches. This can be explained by the limit imposed on the minimum interval size. Small contracts are better fulfilled by small intervals as each iteration represents a smaller proportion of the contract, allowing for finer control over the total time taken.

Initially this appears a major flaw in both approaches. However, this problem is largely exacerbated by the existence of many small problems in the UCR 85 database. Problems that without intervention take between 1 and 4 hours to complete.

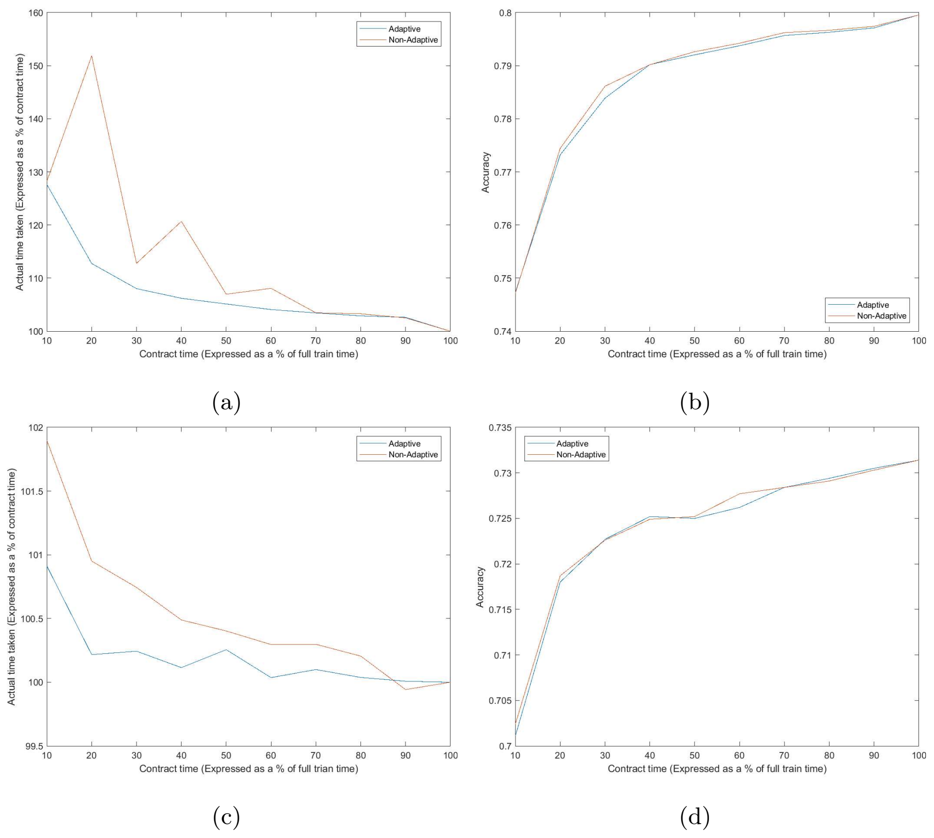


Fig. 2. The graphs on the left show the performance of the Adaptive and Non-adaptive approaches in respect to their ability to adhere to contract time. The graphs on the right show the predictive accuracy of the Adaptive and Non-adaptive approaches over 10 contracts. The top row displays results averaged over all datasets from the UCR database. The bottom row displays results averaged over all problems in the UCR datasets with at least 700 attributes.

In order to remove the bias introduced by smaller datasets the same experiments were repeated with all datasets containing data over 700 attributes.

Figure 2 (c) shows how the actual time taken changes over different contract times for problems from the UCR archive with 700 or more attributes. Figure 2 (c) shows that both approaches were confounded by smaller problems. It also illustrates the Adaptive approaches superior ability to adhere more closely to the contract than the Non-adaptive approach.

Interestingly, these changes in contract accuracy have very little to no effect on accuracy. Figure 2 (b) and figure 2 (d) show how accuracy changes as a function of contract time for all UCR datasets and datasets over 700 attributes respectively. This is important as it confirms that the superior ability of adhering to contract time comes at no cost to accuracy for the Adaptive approach.

5 Eigenworms Case Study

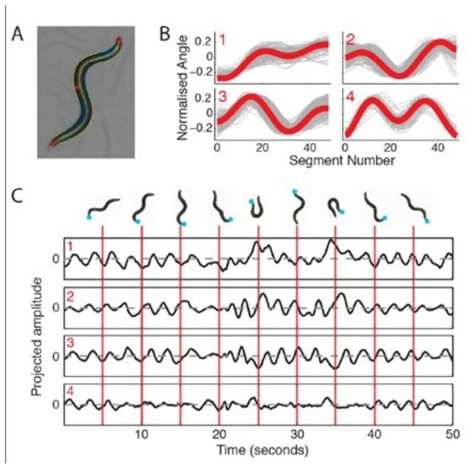


Fig. 3. An image showing an example of Caenorhabditis elegans, the 4 Eigenworm shapes and an illustration of a corresponding 4 dimensional time series with positional images [5]

5.1 The dataset

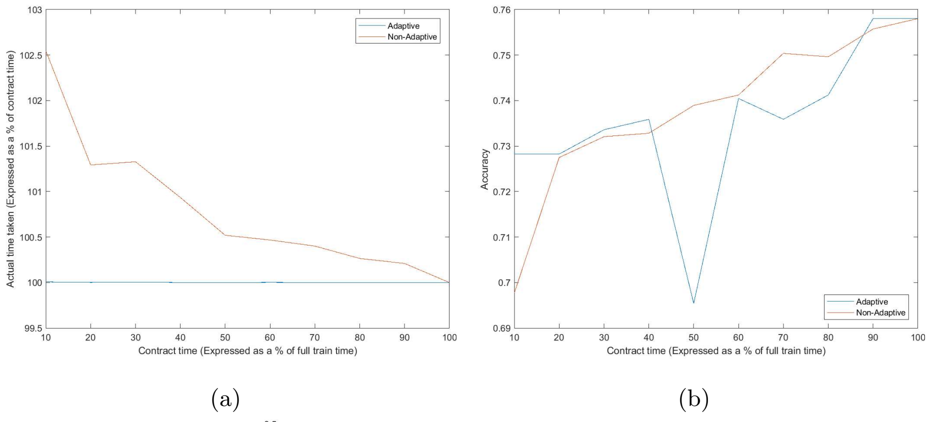
Caenorhabditis elegans is commonly used in genetic studies as a model organism. Their movements have proven to be a robust indicator in determining behavioural traits. [5] outline a number of human-defined features [11] as well a procedure to derive them. It has been shown that all positional variants that the organism adopts can be expressed as combinations of 4 base shapes, known as eigenworms, shown in figure 3(B). The raw worm outline is captured and similarity value assigned to each of the 6 base shapes at each frame of motion.

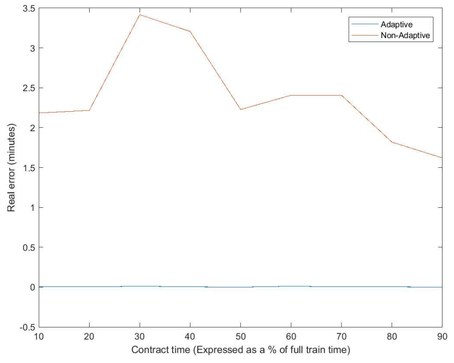
Using the data collected in [5] worms can be classified as either wild type, or 1 of 4 mutant types: goa-1, unc-1, unc-38, unc-63. The dataset is split into a 131 instance train set and a 128 instance test set. The original classification dataset, available at [1], consists of 6 dimensions of 17,984 attributes. However, these were concatenated to create instance lengths of 107,903 for this problem.

5.2 Results

Figure 4 (a) shows how the actual train time changes over different contract times. Each point represents the mean train time over the Eigenworms dataset over 10 folds for each contract. Interestingly, the Non-adaptive approach appears to follow a similar downward trend to that displayed in 2 (c), this is in contrast to the Adaptive model which has shown improvement as series length increases.

Figure 4 (c) displays the real error in minutes of the 2 approaches. Although the Non-adaptive approach's error does vary in respect to contract time, it does not display a convincing downward trend. This suggests that the error may be bound to either the series length, number of cases, or both. Conversely, the Adaptive method produces results that indicate that it is robust to increases in series length. Figure 4 (b) shows how accuracy changes as a function of contract time. The Non-adaptive approach displays a more consistent assent in accuracy as the contract time increases. The performance of the Adaptive approach contradicts the results shown in figure 2 (b) & (d) which suggest the accuracy of both models over all contracts do not deviate from one another.





(c)

Fig. 4. (a) shows the performance of the Adaptive and Non-adaptive approaches in respect to their ability to adhere to contract time. (b) shows displays the predictive accuracy of the Adaptive and Non-adaptive approaches over 10 contracts. (c) shows the real error, in minutes, of the Adaptive and Non-adaptive models over the same contracts.

6 Conclusions

In conclusion, we present changes to RISE that consistently result in a significantly faster train time. This result was achieved using an established experimental design on an established Time Series database. These changes remove two costly derivations making cRISE at least twice as fast whilst the cost to test accuracy is shown to be insignificant.

We also present and compare the Adpative and Non-Adaptive timing models in the context of cRISE. On an established Time Series Database of 85 problems the experimental design does not show any significant difference in accuracy between models. Although, one approach did show a superior ability to adhere to contract.

This superior ability to adhere to a contract time was made further evident when considering a problem consisting of significantly longer series. It was shown that the Adaptive model is robust to scaling of series length, whereas the Non-adaptive approach is bound by series length, number of cases or both.

Although relatively incremental, these changes collectively represent a significant improvement in the usability of cRISE and consequently HIVE-COTE. They also serve to address the commonly unanswered question of, how best to contract?

Acknowledgements

This work is supported by the Biotechnology and Biological Sciences Research Council [grant number BB/M011216/1], and the UK Engineering and Physical Sciences Research Council (EPSRC) [grant number EP/M015807/1]. 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 and using a Titan X Pascal donated by the NVIDIA Corporation.

References

1. Bagnall, A., Dau, H., Lines, J., Flynn, M., Large, J., Bostrom, A., Southam, P., Keogh, E.: The UEA multivariate time series classification archive, 2018. ArXiv e-prints **arXiv:1811.00075** (2018), `http://arxiv.org/abs/1809.06705`
2. 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)
3. Bagnall, A., Lines, J., Hills, J., Bostrom, A.: Time-series classification with COTE: The collective of transformation-based ensembles. IEEE Transactions on Knowledge and Data Engineering **27**, 2522–2535 (2015)
4. Breiman, L.: Random forests. Machine Learning **45**(1), 5–32 (2001)
5. Brown, A.E., Yemini, E.I., Grundy, L.J., Jucikas, T., Schafer, W.R.: A dictionary of behavioral motifs reveals clusters of genes affecting caenorhabditis elegans locomotion. Proceedings of the National Academy of Sciences **110**(2), 791–796 (2013)

6. Deng, H., Runger, G., Tuv, E., Vladimir, M.: A time series forest for classification and feature extraction. Information Sciences **239**, 142–153 (2013)
7. Lines, J., Bagnall, A.: Time series classification with ensembles of elastic distance measures. Data Mining and Knowledge Discovery **29**, 565–592 (2015)
8. Lines, J., Davis, L., Hills, J., Bagnall, A.: A shapelet transform for time series classification. In: Proc. the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2012)
9. Lines, J., Taylor, S., Bagnall, A.: Time series classification with HIVE-COTE: The hierarchical vote collective of transformation-based ensembles. ACM Trans. Knowledge Discovery from Data **12**(5) (2018)
10. 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)
11. Yemini, E., Jucikas, T., Grundy, L., Brown, A., Schafer, W.: A database of *caenorhabditis elegans* behavioral phenotypes. Nature Methods **10**, 877–879 (2013)