e-SMOTE: a train set rebalancing algorithm for time series classification

Chuanhang Qiu[1], Matthew Middlehurst[2], Christopher Holder[1], and Anthony Bagnall[1] (✉)

[1] School of Electronics and Computer Science, University of Southampton
[2] School of Computer Science, University of Bradford

Abstract. Class imbalance, where one class has significantly fewer training instances than others, is a well-studied challenge in machine learning. However, research on handling class imbalance in time series classification (TSC) remains limited, and no comprehensive experimental comparison of existing approaches has been conducted. Many standard imbalance-handling techniques rely on similarity measures, but computing similarity between time series is more complex than for tabular data. Elastic distances, which account for temporal misalignment, have proved effective in many time series machine learning tasks. We explore resampling strategies for imbalanced TSC and introduce e-SMOTE, an extension of the widely used SMOTE algorithm that incorporates the move-split-merge elastic distance metric. We construct a benchmark of 76 imbalanced TSC datasets derived from the UCR and time series machine learning (TSML) repositories to evaluate state-of-the-art (SOTA) TSC algorithms under class imbalance. Our results show that e-SMOTE enhances the performance of TSC classifiers that typically struggle with imbalance and outperforms both generic and time series-specific rebalancing strategies when tested on our new imbalanced dataset archive.

Keywords: Time series · classification · class imbalance

1 Introduction

A classification problem with class imbalance is one where one class, the minority class, occurs much less frequently than the majority class (if we restrict attention to two class problems). The scenario is well studied in the traditional classification literature. Common strategies include generating synthetic examples of the minority class [6], downsampling the majority class [20] or employing cost weighting to favour the minority class internally [1]. Our focus is on resampling the minority class for time series classification (TSC).

TSC is the problem of training a model to predict a discrete target variable using ordered sequences of real valued variables. It is a popular area of research, partly due to the wide range of applications and the availability of easy to use open source implementations of the latest algorithms [23] and an extensive repository of example datasets [7]. Recent TSC research is reviewed and evaluated in [25].

Despite how frequently class imbalance occurs in real world TSC scenarios in, for example, predicting faults in machinery, it has not been well studied in the context of recent TSC research. One reason for this may be the lack of example imbalanced TSC problems in the UCR archive. Despite its popularity, there are known issues with the UCR archive [2]. One often quoted problem is that the datasets are heavily preprocessed, and one aspect of this curation is the creation of artificially balanced data: 33 of 42 binary datasets have been artificially balanced and of the other nine, only one would be considered imbalanced (the train set for wafer has 97 instances of one class and 903 cases of the other class). Unfortunately we cannot recover the original imbalanced problems. Instead, we create an imbalanced version from the time series machine learning (TSML) archive to help assess the performance of TSC algorithms in this circumstance and evaluate resampling schemes to mitigate the problems imbalance creates. We create a new repository of 76 imbalanced binary classification problems that include versions of the 30 new problems introduced to the TSML archive recently [25]. The processing steps and the characteristics of this new archive are described in Section 5. Many of these datasets have small train set sizes. Because of this, our focus is on algorithms that oversample the minority class rather than downsample the majority. It is also a contributory factor in our not concentrating on deep learning algorithms. Instead, our research questions are:

1. How well do the current state of the art TSC algorithms [25] handle class imbalance?
2. Do standard rebalancing techniques improve the performance of time series classifiers?
3. Can we improve these standard algorithms by exploiting time series specific distance functions?

We find that there is variation in how algorithms handle imbalance, and that standard rebalancing provides some improvement to those that handle it worse than others. We also find that using approaches based on recent research on elastic distances and time series clustering [16] (TSCL) in the rebalancing process gives greater improvement than the generic algorithms. We make our imbalanced TSC archive freely available, provide scikit-learn compatible implementations based on the time series machine learning `aeon` toolkit [23][1] and give easy to follow code examples to reproduce our results.

The remainder of this paper is structured as follows. Section 2 highlights the relevant related research on resampling imbalanced problems. Section 3 describes elastic distance algorithms and how they can be applied within generic resampling algorithms. The e-SMOTE resampling algorithm is described in Section 4. Section 5 describes the new imbalanced TSC archive and how we evaluate alternative rebalancing strategies. Section 6 presents the results of an extensive set of experiments designed to answer the research questions above. Finally, we conclude in Section 7 and describe the next stages of this research project.

[1] `https://aeon-toolkit.org/`

2 Background

A time series is a sequence of m ordered real valued values $\mathbf{x} =< x_1, \ldots, x_m >$. If x_i are vectors the series is called multivariate. We assume x_i are scalars, i.e. we are only considering univariate time series. A collection of time series is a set of series denoted $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$. For classification, we have two collections of time series: a train set $\mathbf{X}_{tr}$ of size n and a test set X_{te}. Resampling and model training only happens on the train set and X_{te} is only used in evaluation. For simplicity, we denote the train set $\mathbf{X}$ in algorithmic descriptions. A target variable y is associated with each series. We call the tuple $(\mathbf{x}_i, y_i)$ a case or instance. For imbalanced problems, we assume y_i is binary and that the number of one class is at most 10% of the total number of cases. We call cases of the minority class positive, and denote their set $\mathbf{X}_+$ of size n_+. Elements of the majority class are called negative and belong to set $\mathbf{X}_-$ of size n_-, where $n_+ + n_- = n$ and define an imbalanced problem as one where $n_+ \leq \frac{n}{10}$.

2.1 Resampling Algorithms

Resampling the training data is a popular approach to dealing with class imbalance. Perhaps the best known technique is the Synthetic Minority Over-sampling TEchnique (SMOTE) [6]. The SMOTE algorithm is summarised in Algorithm 1. It creates synthetic cases of the minority class based on its nearest neighbours. We assume for simplicity in line 1 that we require an equal number of synthetic cases from each positive example. It creates synthetic examples from the k nearest neighbours of the positive class.

Algorithm 1 Synthetic Minority Over-sampling Technique: SMOTE ($\mathbf{X}_+, p, k$)

Parameters: Positive cases $\mathbf{X}_+$, desired number of synthetic cases p, number of nearest neighbours to consider, k

Return: Collection $\mathbf{X}_{++}$ containing $\mathbf{X}_+$ and p synthetic cases.

1: $r \leftarrow \frac{p}{|\mathbf{X}_+|}$
2: $\mathbf{X}_{++} \leftarrow \mathbf{X}_+$
3: **for** each case $\mathbf{x} \in \mathbf{X}_+$ **do**
4: Let $\mathbf{K}$ be the k-nearest neighbours of $\mathbf{x}$ in $\mathbf{X}_+$
5: **for** $j = 1$ to r **do**
6: Randomly select neighbour $\mathbf{y}$ from $\mathbf{K}$
7: Generate a random number λ in range [0,1]
8: Compute synthetic sample $\mathbf{x}_{new} \leftarrow \mathbf{x} + \lambda \cdot (\mathbf{x} - \mathbf{y})$
9: $\mathbf{X}_{++} \leftarrow \mathbf{X}_{++} \bigcup \mathbf{x}_{\text{new}}$
10: **return** X_{++}

The Adaptive Synthetic Sampling (ADASYN) [15] algorithm is an extension of SMOTE. It considers neighbours of both the positive and negative classes and biases the sampling of neighbours towards positive cases that have more

negative neighbours. A range of alternatives and extensions have been proposed, including SMOTENC, SMOTEN [6], BorderlineSMOTE [14], SVMSMOTE [26], KMeansSMOTE [11], H-SMOTE [19], HS-SMOTE [31] and EB-SMOTE [29]. SMOTE/ADASYN are designed for tabular data and based on similarity that assumes ordering of variables is unimportant. There have been some techniques proposed for time series specific resampling. The enhanced structure preserving oversampling (ESPO) [5] algorithm oversamples the minority class based on the positive case covariance structure combined with an interpolation method. A support vector machine (SVM) with a radial basis function was used for classification. ESPO was compared to SMOTE and ADASYN on seven UCR datasets converted into two class problems.

Distance space oversampling [18] is an implicit approach to augment minority class. First, a suitable distance function is used to map time series from their original feature space to distance space, resulting in a distance matrix. More cases (called *ghost points*) are simulated for minority class by expanding the distance matrix only (rather than the train set). The augmented distance matrix is converted into a kernel matrix for SVM training and prediction. The effectiveness of ghost points was demonstrated on 17 UCR datasets and the MPEG-7 dataset.

The oversampling method to resolve the High-dimensional Imbalanced Time series classification (OHIT) [33] involves clustering, covariance modelling and resampling. It is evaluated on 12 UCR datasets and compared to SVM using a range of alternative resampling algorithms.

T-SMOTE [32] uses sub-series of minority cases to generate candidates near the class border. It uses seven UCR univariate datasets and compares performance against alternative resampling techniques in conjunction with an LSTM classifier. The majority of this related research is evaluated with either a nearest neighbour or SVM classifier. Recent research has shown that the state of the art time series classifiers, described in the next section, are on average over 10% more accurate than these approaches.

2.2 Time Series Classification

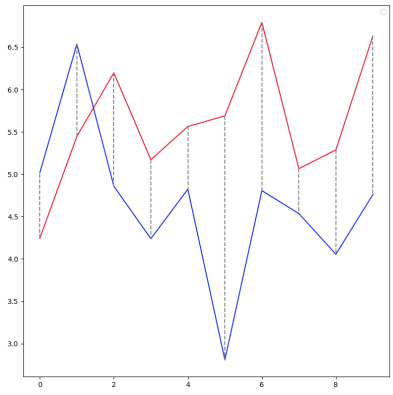
The field of TSC has advanced significantly in the last decade, and yet the research on imbalanced TSC has not reflected this. A recent bake off [25] found that two algorithms, HIVE-COTEv2 (HC2) [24] and MultiRocket-Hydra (MRHydra) [8] performed significantly better than other algorithms on the UCR datasets. However, there is no dominant approach, and different classifiers will be more appropriate for different problems. Performance may vary by problem type, but also may differ by data characteristics. One of our objectives is to compare algorithms in the presence of class imbalance. There have been hundreds of algorithms proposed for TSC. They have been categorised by their basic representation [25]. We take the best performing classifier of each category as our benchmark classifiers:

Distance Based classification is based on some time series specific distance measure between whole series. The best algorithm based on distances was Proximity Forest (PF) [21]. We describe elastic distance functions in more detail in

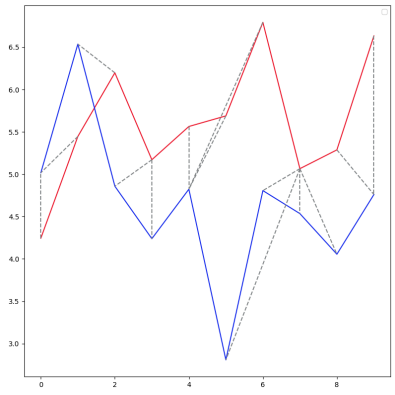
Section 3
Feature Based pipelines transform the time series into summary features (e.g. series mean and variance) then apply a standard classifier. The most effective approach was the feature pipeline called the FreshPRINCE [22].
Interval Based algorithms find summary features over multiple phase dependent intervals of series. Quant [9] classifier is a pipeline that finds quantiles over intervals structured in a hierarchy of partitions then applies a linear classifier.
Shapelet Based algorithms transform time series using shapelets. Shapelets are subseries from the training data that are independent of the phase and can be used to discriminate between classes of time series based on their presence or absence. The best in class was the Random Dilated Shapelet Transform (RDST) [13].
Convolution Based approaches transform the series using randomly generated kernels/convolutions and a pooling operator in a classifier pipeline. The MultiROCKET [30] algorithm used in cunjunction with the **HYbrid Dictionary–ROCKET Architecture (Hydra)** [8] approach was found to be the most effective. We call this algorithm MultiRocket Hydra or MRHydra.
Dictionary Based approaches use histograms of counts of repeating patterns as the features for a classifier. WEASEL v2.0 (W2) [27] was the best in this category.
Deep Learning based algorithms perform neural network based classification. H-InceptionTime [17] won this class.
Hybrid classifiers combine two or more classifiers from the categories above. The second version of the Hierarchical Vote Collective of Transformation Ensemble HIVE-COTEv2 (HC2) [24] combines shapelet, convolution, feature and dictionary classifiers in a meta-ensemble. It was noted in the bake off [25] that HC2 *"does worse than MR-Hydra on imbalanced data"*. We explore this in more detail and evaluate ways to improve performance.

3 Elastic SMOTE

Measuring the distance between time series is a primitive operation that can be used for a range of tasks such as classification, clustering, extrinsic regression, anomaly detection and retrieval. The simplest way to calculate the distance between two time series is to use the L_p distance. SMOTE is based on using the Euclidean distance (L_2) to find neighbours. However, L_p distances take no account of possible misalignments: small offsets can create large distances. A family of algorithms called elastic distances compensate for possible offset by applying a realignment algorithm. The best known of these is dynamic time warping (DTW) which uses dynamic programming to find the optimal path through a cost matrix. Elastic distance measures produce an alignment path, demonstrated in Figure 1. In Figure 1(a), each time point in the red time series aligns exactly with the corresponding time point in the blue time series. In contrast, Figure 1(b) illustrates a more flexible alignment, where, for example, the fourth time point in the red time series is mapped to both the third and fourth time points in the blue time series.



(a) A visualisation of the L_2 alignment between the red and blue time series.



(b) A visualisation of the DTW alignment between the red and blue time series.

Fig. 1: Example of alignment between two time series when using a L_2 distance (a) and the elastic distance DTW (b). The dashed grey line shows which time points are being compared to compute the final distance measure.

3.1 Move-Split-Merge

$$msm\_cost(x, y, z, c) = \begin{cases} c & if\ (y \leq x \leq z) \\ c & if\ (y \geq x \geq z) \\ c + \min \begin{cases} |x - y| \\ |x - z| \end{cases} & otherwise \end{cases} \tag{1}$$

$$CM_{msm}(i, j) = \min \begin{cases} CM_{msm}(i-1, j-1) + |a_i - b_j| \\ CM_{msm}(i-1, j) + msm\_cost(a_i, a_{i-1}, b_j, c) \\ CM_{msm}(i, j-1) + msm\_cost(b_j, a_i, b_{j-1}, c) \end{cases} \tag{2}$$

An alternative family of distance functions are based on the concept of edit distance. An edit distance, such as MSM, considers a diagonal move as a match, a vertical move as an insertion and an horizontal move as a deletion. The move/match operation in MSM uses the absolute pointwise difference rather than the squared Euclidean distance used by DTW. The cost of the split and merge operations are given by cost function msm_cost (Equation 1). The cost of splitting and merging values depends on the value itself and adjacent values. The definition of MSM and the resulting distance are shown in Equations 1 and 2 MSM satisfies triangular inequality and is a metric (see [28] for a proof). We use a constant value of $c = 1$ in all our experiments.

Algorithm 2 Elastic Synthetic Minority Over-sampling Technique
e-SMOTE ($\mathbf{X}_+, p, k, d, f$)

Parameters: Positive cases $\mathbf{X}_+$, desired number of synthetic cases p, number of nearest neighbours to consider k, distance function d, alignment function f
Return: Collection $\mathbf{X}_{++}$ containing $\mathbf{X}_+$ and p synthetic cases.
1: $r \leftarrow \frac{p}{|\mathbf{X}_+|}$
2: $\mathbf{X}_{++} \leftarrow \mathbf{X}_+$
3: **for** each case $\mathbf{x} \in \mathbf{X}_+$ **do**
4: Let $\mathbf{K}$ be the k-nearest neighbours of $\mathbf{x}$ in $\mathbf{X}_+$ using d (e.g., MSM)
5: **for** $j = 1$ to r **do**
6: Randomly select neighbour $\mathbf{y}$ from $\mathbf{K}$
7: Find alignment path $P = \{(p_1, q_1), (p_1, q_2), (p_2, q_2), \dots\}$ between $\mathbf{x}$ and $\mathbf{y}$ using f
8: Initialize $\boldsymbol{\delta}$ as a zero vector of the same length as $\mathbf{x}$
9: **for** $k = 1$ to case length l **do**
10: Randomly select an index $q*$ from the tuples in P where $p = p_k$
11: Compute alignment-based perturbation:

$$\delta_k = \mathbf{x}_{p_k} - \mathbf{y}_{q^*}$$

12: Generate a random number λ in range [0,1]
13: Compute synthetic sample:

$$\mathbf{x}_{\text{new}} \leftarrow \mathbf{x} + \lambda \cdot \boldsymbol{\delta}$$

14: $\mathbf{X}_{++} \leftarrow \mathbf{X}_{++} \bigcup \mathbf{x}_{\text{new}}$
15: **return** $\mathbf{X}_{++}$

4 e-SMOTE Resampling Algorithm

The elastic SMOTE (e-SMOTE) algorithm employs elastic distances to make more discriminatory synthetic cases. There are two stages of SMOTE (Algorithm 1) where elastic distances may improve the algorithm: finding the neighbours (line 4) and generating new synthetic examples (line 8). We implement these changes in the e-SMOTE algorithm, described in Algorithm 2. Firstly, we generalise finding neighbours to use any distance, set to MSM in all our experiments. Secondly, we use the alignment path between the series to create a new case. The alignment between two series can be found directly from the distance calculation, but for clarity we separate this into two functions, d to find the distance and f to find the path. The path P is a series of pairs of indexes specifying the alignment that gives the specific distance. e-SMOTE randomly chooses an aligned value for each point (line 10), then perturbs the instance based on distance from the aligned pair. e-SMOTE is designed to make the minimal changes to SMOTE so that we can assess the benefit of using an elastic distance.

5 Methodology

To evaluate algorithms in the face of imbalance we need to specify what we mean by imbalance, find imbalanced datasets for evaluation and define what metrics and tests of difference we use in the assessment.

We restrict our attention to binary classification problems, and define imbalance as the situation where the least commonly occuring class label, referred to as the minority class, represents 10% or less of the cases in the training data. Real world data is often much more imbalanced than this. The decision to use 10% is taken through necessity: it allows us to use a reasonable number of datasets and derive meaningful performance metrics to perform tests of significant differences.

To compare algorithms for imbalanced TSC scenarios, we reformulated the UCR Time Series Classification Archive [7] and the subsequent enhancement TSML archive [25] into binary classification tasks. We assigned the class with the fewest samples as the minority class. In situations where multiple classes had equal sample sizes, we arbitrarily selected the last class as the minority class, assigning the remaining classes to the majority class. To introduce the desired class imbalance, we randomly removed samples from minority class set of the binary-class datasets, ensuring a majority-to-minority ratio of 9:1.

We use the default train/test splits to retain a link to general TSC research. However, many of the UCR datasets have small train set sizes. Making these problems imbalanced makes the train sets even smaller, and often results in a very small number of positive cases. Because of this, we restrict our attention to problems where rebalancing leaves us with at least 10 positive cases, i.e. a train set size of at least 100 cases. The details of our imbalanced data is shown on the accompanying website, where they can be downloaded in standard format [2].

Our primary performance metrics are classification accuracy to measure general performance and balanced accuracy, to assess performance in the presence of imbalance. Accuracy of over 90% and balanced accuracy over 50% implies performance better than predicting the majority class. The quality of the probability estimates is measured with the log loss. The ability to rank predictions is estimated by the area under the receiver operator characteristic curve (AUROC). We use sensitivity and specificity to describe performance and the minority and majority class, and the F1-score is used to measure a model's balance between false positives and false negatives.

For reproducibility, each classifier with random number generation is seeded to 0. We compare multiple classifiers over multiple datasets using an adaptation of the critical difference diagram [10]. The post-hoc Nemenyi test is replaced with a comparison of all classifiers using pairwise Wilcoxon signed-rank tests, and cliques formed using the Holm correction as recommended by [4].

The classifiers we use in experiments are configured with their default parameters, described on the website and in the appendix.

[2] `https://github.com/LinGinQiu/AALTD2025imbalance`

6 Results

In all experiments we use the following abreviations: 1-NN with DTW is **DTW**; Rotation Forest is**RotF**; Proximity Forest is **PF**, Weasel 2.0 is **W2**, H-InceptionTime is **H-IT**; HIVE-COTEv2.0 is **HC2**; and is MultiRocket-Hydra is **MRHydra**.

6.1 TSC Performance on Imbalanced Data

Our first experiment is to investigate the relative performance of SOTA algorithms described on the new imbalanced TSC archive. Figure 2 shows the relative ranked performance of ten classifiers on 76 imbalanced datasets. Rotation forest (RotF) is not a time series classifier, but is included for context since it is known to perform well on this type of problem [3].

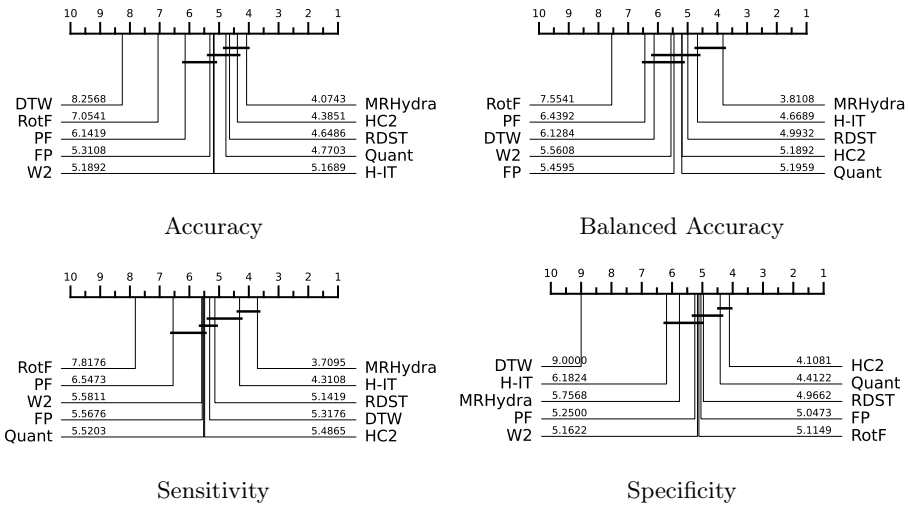


Fig. 2: Relative ranked performance of eight classifiers on the imbalanced classification problems.

There is a top clique of four classifiers for accuracy, with MRHydra and HC2 performing best. This reflects the ordering of algorithms on the standard archive. Table 1 shows the average accuracy, balanced accuracy, area under the ROC curve (AUROC), log loss, sensitivity and specificity for all classifiers. Fig 2 and Table 1 show that HC2 has the highest average accuracy, AUROC, LogLoss and Specificity, whereas MRHydra has the highest Balanced Accuracy and Sensitivity.

We quantify the impact of making the data imbalanced by comparing balanced accuracy performance of HC2 and MRHydra on the 14 two class problems from the original archive that we have rebalanced. Table 2 shows the balanced

Table 1: Comparison of classifiers based on multiple performance metrics.

	DTW	**FP**	**H-IT**	**HC2**	**MRH**	**PF**	**Quant**	**RDST**	**RotF**	**W2**
Accuracy	0.9033	0.9371	0.9227	**0.9411**	0.9403	0.9313	0.9385	0.9395	0.9304	0.9366
Bal. Acc.	0.7366	0.7563	0.7675	0.7554	**0.7909**	0.7275	0.7532	0.7621	0.6837	0.7486
AUROC	0.7366	0.9025	0.8909	**0.9066**	0.7909	0.8776	0.9055	0.7621	0.8671	0.7486
LogLoss	3.4848	0.1864	0.5448	**0.1682**	2.1534	0.2760	0.1896	2.1806	0.2290	2.2843
Sensitivity	0.5292	0.5315	0.5739	0.5243	**0.6050**	0.4741	0.5226	0.5415	0.3767	0.5146
Specificity	0.9441	0.9811	0.9610	**0.9864**	0.9767	0.9809	0.9838	0.9827	**0.9907**	0.9825

accuracy of both HC2 and MRHydra on these data before and after removing training cases. This is useful because it provides an upper bound on performance: it is very unlikely a classifier will perform better on rebalanced data than the original. On average, the balanced accuracy is about 20% less on the imbalanced data for both algorithms. Table 2 highlights two other characteristics of the data. Firstly, both classifiers perform worse than 50% for FordB and SemgHandGenderCh2. HC2 on FordB does not predict a single positive case correctly. Other classifiers have similar difficulty with these data. This suggests imbalancing has made these problems impossible. Secondly and conversely, Wafer seems to be a trivial problem both before and after imbalancing. These characteristics are worth considering in detailed analysis of results, but to avoid seemingly cherry picking, we continue experiments with the same datasets. These results set the

Table 2: Balanced accuracy before and after making data imbalanced for problems that were originally two class problems.

Dataset	HC2	imb	diff	MRHydra	imb	diff
Computers	76.00%	60.74%	-15.26%	76.80%	63.78%	-13.02%
DistalPhalanxOutlineCorrect	75.03%	61.45%	-13.58%	77.45%	58.20%	-19.25%
FordA	95.61%	79.19%	-16.42%	95.69%	85.85%	-9.84%
FordB	83.71%	31.78%	-51.93%	83.45%	27.02%	-56.43%
HandOutlines	92.72%	84.40%	-8.32%	94.27%	88.04%	-6.23%
MiddlePhalanxOutlineCorrect	83.98%	58.33%	-25.65%	83.78%	63.59%	-20.19%
PhalangesOutlinesCorrect	80.34%	59.39%	-20.95%	81.02%	64.56%	-16.46%
PowerCons	98.33%	75.00%	-23.33%	98.33%	80.00%	-18.33%
ProximalPhalanxOutlineCorrect	85.12%	79.55%	-5.57%	88.13%	81.57%	-6.56%
SemgHandGenderCh2	95.62%	40.90%	-54.72%	94.74%	33.21%	-61.53%
Strawberry	97.98%	95.94%	-2.04%	97.22%	95.31%	-1.91%
Wafer	100.00%	100.00%	0.00%	99.85%	99.84%	-0.01%
WormsTwoClass	80.68%	62.50%	-18.18%	77.65%	75.00%	-2.65%
Yoga	92.62%	57.55%	-35.07%	92.78%	65.20%	-27.58%
Average	88.41%	67.62%	-20.79%	88.65%	70.08%	-18.57%

Table 3: Performance comparison of HC2 and MRHydra methods before and after rebalancing.

basis for assessing whether rebalancing the train set by creating synthetic samples of the minority class improves performance.

6.2 Do standard rebalancing techniques improve performance?

Ideally, we would hope that rebalancing allows the classifier to achieve more correct positive test cases without any loss of accuracy in negative cases. This would lead to an improvement in both overall accuracy and balanced accuracy. However, given the small number of positive test cases, a significant improvement in accuracy is unlikely. Instead, we primarily aim for a substantial increase in balanced accuracy. This can also be framed as an increase in sensitivity at the cost of specificity.

We ran both SMOTE and ADASYN with the same configuration, using $k = 3$ neighbours due to the small training set sizes. The newly transformed training files are available on the associated website[3]. Table 4 shows the relative performance of six TSC algorithms when trained on the imbalanced data and on data rebalanced with SMOTE. The differences are averaged over all datasets. The pattern is similar for all classifiers. They are less accurate overall, but have better balanced accuracy. This is reflected in increased sensitivity at the cost of reduced specificity. The level of this effect varies significantly. HC2 has the biggest drop in accuracy and the biggest rise in balanced accuracy. Conversely, the accuracy and balanced accuracy of MRHydra change much less. A similar pattern is observed when rebalancing with ADASYN.

Table 4: Difference in test data performance between imbalanced data and train data rebalanced with SMOTE.

	Accuracy	Balanced Acc	Sensitivity	Specificity
HC2	-0.73%	3.24%	8.18%	-1.70%
MRHydra	-0.20%	1.21%	2.97%	-0.55%
RDST	-0.31%	2.14%	5.18%	-0.91%
H-IT	-0.53%	3.13%	7.72%	-1.45%
Quant	-0.94%	2.69%	7.20%	-1.83%
RotF	-0.22%	2.12%	5.03%	-0.80%

HC2 and MRHydra are the SOTA algorithms and demonstrate the extremes of change when rebalanced with SMOTE. We compare performance on the imbalanced data (IMB) against data rebalanced using SMOTE and ADASYN. We also include data rebalanced through simple random perturbation of the minority cases (RAND). Figure 3 shows that all three forms of rebalancing improve both classifiers. For MRHydra, there is no difference among the three rebalancing techniques. However, with HC2 both ADASYN and SMOTE are significantly better than RAND. There is also no significant difference between ADASYN and SMOTE in either case. We conclude that whilst rebalancing helps, there is

[3] `https://github.com/LinGinQiu/AALTD2025imbalance`

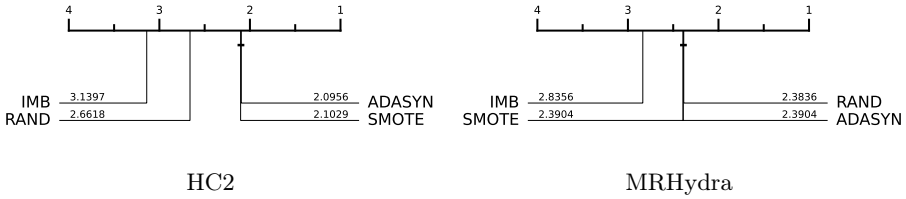


Fig. 3: Relative ranked performance in terms of balanced accuracy with imbalanced data (IMB) and SMOTE, ADASYN and random rebalancing (RAND).

greater room for improvement for HC2 and that focussing on SMOTE is unlikely to make a difference to our overall conclusions.

6.3 e-SMOTE evaluation

We compare e-SMOTE, described in Section 3, based on the performance of the HC2 classifier after different rebalancing operations. We compare to raw imbalanced data (IMB), randomly perturbed data (RAND), SMOTE, T-SMOTE [32] and OHIT [33]. The OHIT implementation is adapted from the Matlab version provided by the papers author [4]. We could not find an implementation of T-SMOTE and the authors did not reply to our enquiries. Hence, we have reimplemented T-SMOTE based on the description in the paper.

Both algorithms are parameterised as described in the literature. OHIT method has several key hyperparameters. Let $n*$ be the number minority samples. k, the number of nearest neighbours in the Shared Nearest Neighbour similarity calculation, is set as: $k = \text{int}(\lceil\sqrt{n*} \times 1.25\rceil)$. κ, the nearest neighbour parameter used for defining the density ratio, is set as: $\kappa = \text{int}(\lceil\sqrt{n*}\rceil)$. drT, the threshold of the density ratio, is set to 0.9. In the T-SMOTE method, we reimplement the spy-based approach to initialize hyperparameters. The spy sample size is set to 0.15, and spy classifier f, which generates the prediction score, is chosen as the `aeon` K-Neighbours Time Series Classifier. As for the fitting time, the time required for rebalancing is only a tiny fraction of the total, and our experiments show that all SMOTE-related methods take less than 1% of the classifier's fitting time.

Figure 4 shows the ranks for accuracy and balanced accuracy for six rebalancing algorithms. e-SMOTE is top ranked for both and is significantly better than all but SMOTE for balanced accuracy. Figure 5 shows the scatter plot of e-SMOTE vs SMOTE for accuracy. e-SMOTE is significantly better than SMOTE with a pairwise test without the multiple correction test used in Figure 4.

Table 5 summarises the performance of HC2 on the imbalanced data and with five resampling techniques. e-SMOTE achieves the best balance between sensitivity and specificity, as demonstrated by the fact it has the highest average rank and value for F1 score.

[4] `https://github.com/zhutuanfei/OHIT`

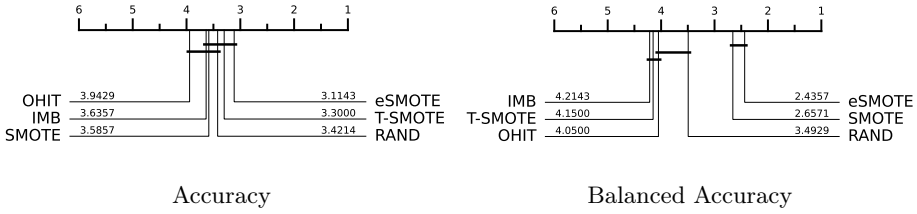


Fig. 4: Relative performance of HC2 classifier using different resampling techniques.

Table 5: Summary performance of HC2 with different resampling techniques.

	IMB	OHIT	RAND	SMOTE	T-SMOTE	e-SMOTE
Accuracy	94.41%	94.07%	94.27%	94.04%	**94.48%**	94.37%
BalAcc	75.23%	75.94%	76.59%	78.59%	75.47%	**78.63%**
Sens	51.39%	53.40%	54.62%	**59.37%**	51.83%	59.07%
Spec	99.08%	98.48%	98.57%	97.80%	**99.11%**	98.20%
F1	57.37%	58.52%	59.79%	63.04%	58.32%	**63.57%**

7 Conclusions

Our first objective is to establish TSC with imbalance on a sound footing through a reproducible experimental setting. We have created an archive of 76 imbalanced TSC problems derived from the TSML archive. These are all binary classification problems with the minority class making up at most 10% of the data. We evaluate the SOTA classifiers on this archive and demonstrated that the top performing algorithms, HIVE-COTEV2 (HC2) [24] and MultiRocket-Hydra (MRHydra) [8] perform as expected: HC2 performs relative worse on imbalanced problems, but is better if probabilistic estimates are required. We find rebalancing improves the performance of both classifiers. However, random perturbation is as good as SMOTE and ADASYN for MRHydra, and the improvement is more pronounced with HC2. We propose e-SMOTE, an adaptation of SMOTE that uses the elastic distance metric Move-Split-Merge [28]. We compare e-SMOTE to SMOTE and two algorithms proposed in the literature. We show e-SMOTE is significantly better than the published alternatives and gives an overall improvement for HC2 compared to SMOTE.

The improvement is slight and there is room for improvement. There is potential for using a range of elastic distances in e-SMOTE to find matches, and we could use one of the many SMOTE variants, although preliminary results indicate they do offer any improvement. Ultimately, distance based rebalancing may not be the best approach for TSC with class imbalance. In future work we hope to investigate alternative approaches to class imbalance. The topic is closely related to the more general field of generating synthetic samples through deep learning generative models or a time series specific technique (e.g. [12]).

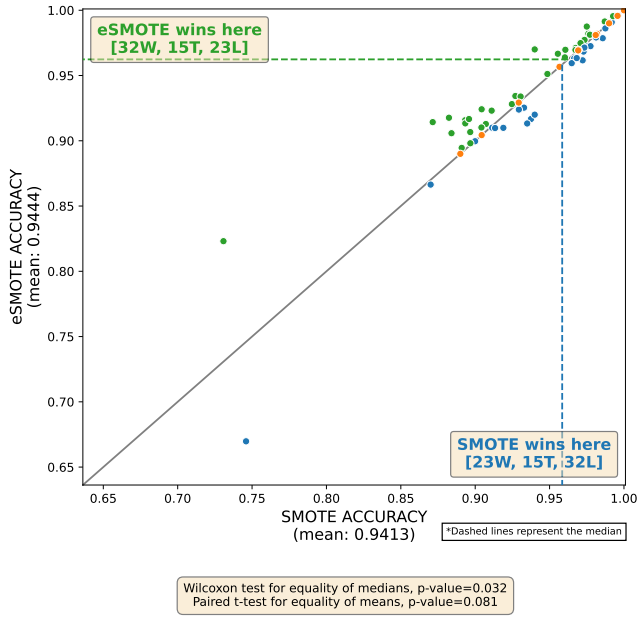


Fig. 5: Accuracy scatter plot for e-SMOTE vs SMOTE.

These algorithms usually address the problem of small train set size rather than data imbalance. However, it is natural to assess their suitability for imbalance.

Another area of enquiry is to see if we can explain the differences in classification performance in terms of algorithm design. HC2 is a hierarchical ensemble, and investigation into the relative performance of components may indicate alternative designs to allow the classifier to automatically internally compensate for imbalance, in particular with regard to the probability weighting mechanism.

Finally, we will expand the archive to encompass multi-class classification tasks, multivariate datasets, and problems with larger data volumes. If the train sets are large, then downsampling the majority class may be a better approach.

Reproducibility We are committed to the FAIR (Findable, Accessible, Interoperable, and Reusable) principles in research. We will contribute our implementations to aeon. In the interim, implementations of the algorithms used and the new imbalanced archive can be found on an anonymous GitHub[5].

Acknowledgements

This work is supported by the UK Engineering and Physical Sciences Research Council (EPSRC) (grant ref.: EP/W030756/2). The authors acknowledge the use of the IRIDIS High Performance Computing Facility, and associated support services at the University of Southampton. We would like to thank all those

[5] `https://github.com/LinGinQiu/AALTD2025imbalance`

contributing to open-source implementations of the algorithms and datasets used in experimentation.

References

1. J. Ahmed and R. C. Green II. Cost aware LSTM model for predicting hard disk drive failures based on extremely imbalanced smart sensors data. *Engineering Applications of Artificial Intelligence*, 127:107339, 2024.
2. Y. C. B. Hu and E. Keogh. Time series classification under more realistic assumption. In *proceedings of the 13th SIAM International Conference on Data Mining*, 2013.
3. A. Bagnall, A. Bostrom, G. Cawley, M. Flynn, J. Large, and J. Lines. Is rotation forest the best classifier for problems with continuous features? *ArXiv e-prints*, arXiv:1809.06705, 2018.
4. A. Benavoli, G. Corani, and F. Mangili. Should we really use post-hoc tests based on mean-ranks? *The Journal of Machine Learning Research*, 17(1):152–161, 2016.
5. H. Cao, X.-L. Li, Y.-K. Woon, and S.-K. Ng. Integrated oversampling for imbalanced time series classification. *IEEE Transactions on Knowledge and Data Engineering*, 25(12):2809–2822, 2013.
6. N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. SMOTE: synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16(1):321–357, 2002.
7. H. Dau, A. Bagnall, K. Kamgar, M. Yeh, Y. Zhu, S. Gharghabi, C. Ratanamahatana, A. Chotirat, and E. Keogh. The UCR time series archive. *IEEE/CAA Journal of Automatica Sinica*, 6(6):1293–1305, 2019.
8. A. Dempster, D. F. Schmidt, and G. I. Webb. Hydra: Competing convolutional kernels for fast and accurate time series classification. *Data Mining and Knowledge Discovery*, 37(5):1779–1805, 2023.
9. A. Dempster, D. F. Schmidt, and G. I. Webb. Quant: A minimalist interval method for time series classification. *arXiv preprint arXiv:2308.00928*, 2023.
10. J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.
11. G. Douzas, F. Bacao, and F. Last. Improving imbalanced learning through a heuristic oversampling method based on k-means and smote. *Information sciences*, 465:1–20, 2018.
12. G. Forestier, F. Petitjean, H. A. Dau, G. I. Webb, and E. Keogh. Generating synthetic time series to augment sparse datasets. In *2017 IEEE international conference on data mining (ICDM)*, pages 865–870. IEEE, 2017.
13. A. Guillaume, C. Vrain, and W. Elloumi. Random dilated shapelet transform: A new approach for time series shapelets. In *ICPRAI*, 2022.
14. H. Han, W.-Y. Wang, and B.-H. Mao. Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning. In *International conference on intelligent computing*, pages 878–887. Springer, 2005.
15. H. He, Y. Bai, E. A. Garcia, and S. Li. ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In *Proc. IEEE International Joint Conference on Neural Networks*, pages 1322–1328, 2008.
16. C. Holder, M. Middlehurst, and A. Bagnall. A review and evaluation of elastic distance functions for time series clustering. *Knowledge and Information Systems*, 66:765–809, 2024.

17. A. Ismail-Fawaz, M. Devanne, J. Weber, and G. Forestier. Deep learning for time series classification using new hand-crafted convolution filters. In *2022 IEEE International Conference on Big Data (Big Data)*, pages 972–981. IEEE, 2022.
18. S. Köknar-Tezel and L. J. Latecki. Improving SVM classification on imbalanced time series data sets with ghost points. *Knowledge and information systems*, 28:1–23, 2011.
19. L. C. M. Liaw, S. C. Tan, P. Y. Goh, and C. P. Lim. A histogram smote-based sampling algorithm with incremental learning for imbalanced data classification. *Information Sciences*, 686:121193, 2025.
20. X.-Y. Liu, J. Wu, and Z.-H. Zhou. Exploratory undersampling for class-imbalance learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(2):539–550, 2008.
21. B. Lucas, A. Shifaz, C. Pelletier, L. O'Neill, N. Zaidi, B. Goethals, F. Petitjean, and G. Webb. Proximity forest: an effective and scalable distance-based classifier for time series. *Data Mining and Knowledge Discovery*, 33(3):607–635, 2019.
22. M. Middlehurst and A. Bagnall. The FreshPRINCE: A simple transformation based pipeline time series classifier. In *International Conference on Pattern Recognition and Artificial Intelligence*, pages 150–161. Springer, 2022.
23. M. Middlehurst, A. Ismail-Fawaz, A. Guillaume, C. Holder, D. Guijo-Rubio, G. Bulatova, L. Tsaprounis, L. Mentel, M. Walter, P. Schäfer, and A. Bagnall. aeon: a python toolkit for learning from time series. *Journal of Machine Learning Research*, 25(289):1–10, 2024.
24. M. Middlehurst, J. Large, M. Flynn, J. Lines, A. Bostrom, and A. Bagnall. HIVE-COTE 2.0: a new meta ensemble for time series classification. *Machine Learning*, 110:3211–3243, 2021.
25. M. Middlehurst, P. Schäfer, and A. Bagnall. Bake off redux: a review and experimental evaluation of recent time series classification algorithms. *Data Mining and Knowledge Discovery*, 2024.
26. H. M. Nguyen, E. W. Cooper, and K. Kamei. Borderline over-sampling for imbalanced data classification. *International Journal of Knowledge Engineering and Soft Data Paradigms*, 3(1):4–21, 2011.
27. P. Schäfer and U. Leser. WEASEL 2.0: a random dilated dictionary transform for fast, accurate and memory constrained time series classification. *Machine Learning*, 11:4763—-4788, 2023.
28. A. Stefan, V. Athitsos, and G. Das. The Move-Split-Merge metric for time series. *IEEE Transactions on Knowledge and Data Engineering*, 25(6):1425–1438, 2013.
29. H. Sun, J. Li, and X. Zhu. A novel expandable borderline smote oversampling method for class imbalance problem. *IEEE Transactions on Knowledge and Data Engineering*, 2025.
30. C. W. Tan, A. Dempster, C. Bergmeir, and G. Webb. MultiRocket: multiple pooling operators and transformations for fast and effective time series classification. *Data Mining and Knowledge Discovery*, 36:1623–1646, 2022.
31. S. Wang, Y. Bao, and S. Yang. Hs-smote: Oversampling method for multiple dynamic interpolations based on regular hexagon scoring mechanism. *Expert Systems with Applications*, 265:125855, 2025.
32. P. Zhao, C. Luo, B. Qiao, L. Wang, S. Rajmohan, Q. Lin, and D. Zhang. T-SMOTE: Temporal-oriented synthetic minority oversampling technique for imbalanced time series classification. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*, pages 2406–2412, 2022.
33. T. Zhu, C. Luo, Z. Zhang, J. Li, S. Ren, and Y. Zeng. Minority oversampling for imbalanced time series classification. *Knowledge-Based Systems*, 247:108764, 2022.