Ensembles for multivariate time series classification

Alejandro Pasos Ruiz

A thesis submitted for the degree of Doctor of Philosophy

School of Computing Sciences

University of East Anglia

September 2023

Abstract

Time Series Classification (TSC) involves learning predictive models for a discrete target variable from ordered, real-valued, attributes. Over recent years, a new set of TSC algorithms have been developed which have significantly improved the previous state of the art. The main focus has been on univariate TSC, i.e. the problem where each case has a single series and a class label. In reality, it is more common to encounter multivariate TSC (MTSC) problems where multiple series are associated with a single label. Despite this, much less consideration has been given to MTSC than the univariate case. Therefore, this work focuses on MTSC from different perspectives. First, by introducing a set of 33 problems for MTSC in different areas called the UEA MTSC archive. Second, by introducing the state-of-the-art algorithms and comparing on those problems. That experimentation concluded that HIVE-COTE2 (HC2) is the current state of the art. Third, because of that, the remainder of this work focused on two ways to improve HC2: a) By improving one of the components (Shapelet Transform Classifier) and b) by Adding a preprocessing phase for dimension selection in order improve HC2 by removing the dimensions that do not contribute. In the first case, we were able to improve HC2 significantly for MTSC problems, and in the second case, there was no significant improvement in accuracy. Still, there were gains in decreasing the number of dimensions required and hence the run time.

Acknowledgments

First, I would like to thank my supervisor Dr. Anthony Bagnall, who allowed me to get into the Ph.D. program and his invaluable support guiding me in this process. I want to thank all the UEA colleagues from whom I always received the help and support I needed. Also, I want to thank my sponsor PRODEP (Programa para el Desarrollo Profesional Docente).

I want to thank my dad who always encouraged me through this program and without his support I won't be able to finish it. Also, my mom who has been here all my life listening and supporting me. To Alex and Leo who make me to be a better father each day of my life. Finally, but not less importantly to my beautiful wife Sandy for always being there and making me feel the most lucky person for having her at my side.

Contents

List of Figures

List of Tables

1

Introduction

1.1 Introduction

Time series, i.e. real value-ordered observations, are ubiquitous. Recently, there has been a huge increase in the prevalence of devices that can extract and store data. With the increase in storage capability and reduction of the cost now many data sources can be used to solve real-world problems that involve time series classification.

TSC is a form of machine learning that uses real ordered values called time series. This scenario adds a layer of complexity to the classification problems, as important characteristics of the data can be missed by traditional algorithms. Over recent years, a new set of TSC algorithms have been developed which have made significant improvements over the previous state-of-the-art [4].

The focus of recent research has been on univariate TSC, i.e. the problem where each case has a single series and a class label. In reality, it is more common to encounter multivariate TSC (MTSC) problems where multiple series are associated with a single label. Human activity recognition [50], diagnosis based on ECG, EEG and MEG [58] [6] [7] [40] and system monitoring problems are all inherently multivariate. Despite this, much less consideration has been given to MTSC than the univariate case. Therefore, this thesis will focus on MTSC problems.

1.2 Thesis contributions

The contributions described in this thesis can be summarised as follows:

- The evaluation and comparison of current state-of-the-art algorithms for MTSC. [62]
- The evaluation of different quality metrics for a specific type of MTSC algorithm which are based on shapelets.

- Development of novel dimension filtering techniques for high dimensional MTSC problems and evaluate them based on accuracy and performance. [59]

1.3 Thesis structure

In brief, we first define MTSC problems and the current state-of-the-art algorithms in Chapter 2. In Chapter 3 we describe the UCR/UEA multivariate archive version 2018 consisting of 33 problems categorized by type. Chapter 4 presents a set of benchmark experiments to compare the state-of-the-art algorithms on the UCR/UEA archive. After an initial run where no single algorithm was significantly better, another experiment introduced a new algorithm named HIVE-COTE 2 (HC2) [53] which was significantly better than the other algorithms and considered the most accurate in the UCR/UEA archive at the time of writing. The final chapters of the thesis focused on two ways to improve HC2, in Chapter 5, one component of HC2 called Shapelet Transform Classifier (STC) [30] is improved by an ensemble based on different shapelet qualities. In Chapter 6, dimension selection strategies were proposed as a preprocessing step to HC2. Finally, in Chapter 7 a discussion of the main conclusions and future work is presented.

2

Background

2.1 Introduction

Time series classification (TSC) is a subset of supervised machine learning dealing with time-ordered classification problems. In this problems, we have a list of vectors of m observations over n cases. We could use standard machine learning algorithms directly given that we can consider each observation as a feature, but in general cases, this approach does not work because of the time-ordered type of data. There are specific algorithms dealing with this kind of data and detailed in the state of the art.

This thesis focuses on Multivariate Time Series Classification (MTSC) problems For MTSC problems, a dataset is a list of vectors of m observations, over d channels (dimensions) and n cases. We define the scalar $x_{i,j,k}$ as the i example (n), the j observation (m) that belongs to dimension k. This 3-dimensional representation differs from the 2 dimensions used in time series and traditional machine learning datasets where the first dimension denotes the examples (rows) and the second represents the features (columns).

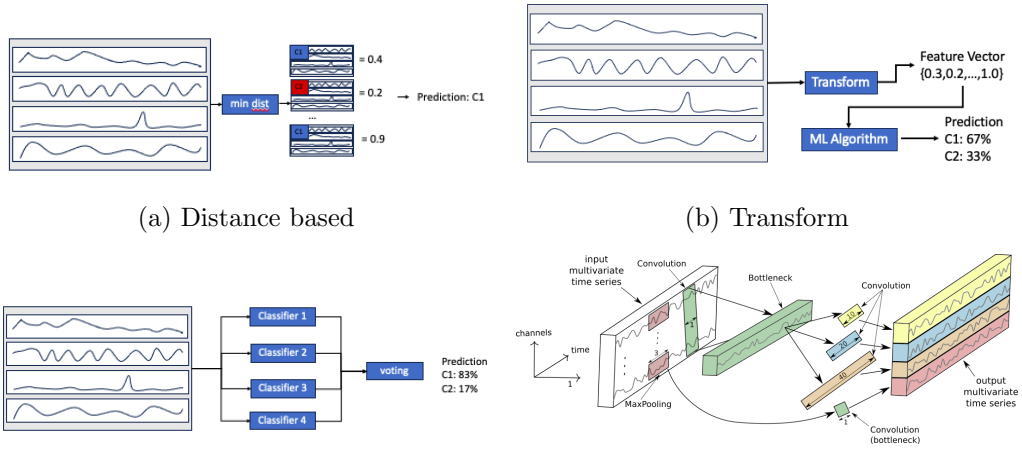
Because of that, MTSC requires specific approaches to handle this kind of data. These approaches are specific to MTSC and others are adapted from univariate time series problems (problems with 1 dimension) by adding specific handlers to the dimensions. This chapter focuses on researching the main algorithms for MTSC in the state of the art. The objective is to be a guide for the decision of selecting algorithms in the following chapters.

On a general basis, an MTSC algorithm can be defined by how we treat different dimensions. On one side, if we can assume that there are no important correlations between dimensions then they can be handled separately. We call this an independent-based approach [66]. This makes the classification process easier because we can transform it into d different univariate problems and then use an ensemble to give the final decision.

However, if there is a correlation between dimensions we must consider these

relations important information about the classification problem otherwise these relations will be missed and this may affect the accuracy of the algorithm. We call this scenario dependent. The problem is that we cannot handle the dependent case as the independent because we have (2^d) different combinations of dimensions which makes it infeasible for high dimensional problems.

Figure 2.1.1: Main approaches for MTSC



(a) Distance based

(b) Transform

(c) Ensemble

(d) Deep learning

In this chapter, we present different algorithmic approaches to implement MTSC classifiers regarding whether they are independent, dependent, or can be implemented in both ways. In the next section, we will outline each of these approaches and next, we will explain them in detail. A summary of the approaches can be seen in figure 2.1.1

2.2 Distance-based

These approaches are based on calculating the distance between two time-series. Usually, this is implemented using the function dynamic time warping (DTW). In MTSC, DTW can be implemented as independent or dependent which will be detailed in this section.

One of the most popular approaches for TSC is to use a 1-nearest neighborhood classifier in conjunction with a bespoke distance function that compensates for

possible confounding offset by allowing some realignment of the series. Dynamic time warping (DTW) is the most popular distance function for this purpose. In DTW, the distance between series $\mathbf{a} = (a_1, a_2, ..., a_m)$ and $\mathbf{b} = (b_1, b_2, ..., b_m)$ is calculated following steps:

1. M is a $m \times m$ matrix where $M_{i,j} = (a_i - b_j)^2$
2. A warping path $P = ((e_1, f_1), (e_2, f_2), ..., (e_s, f_s))$ is a contiguous set of matrix indexes from M, subject to the following constraints
   - $(e_1, f_1) = (1, 1)$
   - $(e_s, f_s) = (m, m)$
   - $0 \leq e_{i+1} - e_i \leq 1$ for all $i < m$
   - $0 \leq f_{i+1} - f_i \leq 1$ for all $i < m$
3. Let $p_i = M_{e_i,f_i}$, be the distance for a path is $D_p = \sum_{i=1}^{m} p_i$
4. There are many warping paths but we are interested in the one that minimizes the accumulative distance $P^* = \min_{p \in P} D_p(a, b)$
5. The optimal distance is obtained by solving the following recurrence relation

$$DTW(i,j) = M_{i,j} + min \begin{cases} DTW(i-1,j). \\ DTW(i,j-1). \\ DTW(i-1,j-1). \end{cases} \tag{2.2.1}$$

   and the final distance is $DTW(m, m)$.

There are several improvements to DTW to make it faster, such as adding a parameter r that limits deviation from the diagonal. Our interest lies primarily in how best to use DTW for MTSC. There are two obvious strategies for using DTW for multivariate problems, defined as the independent and dependent approaches.

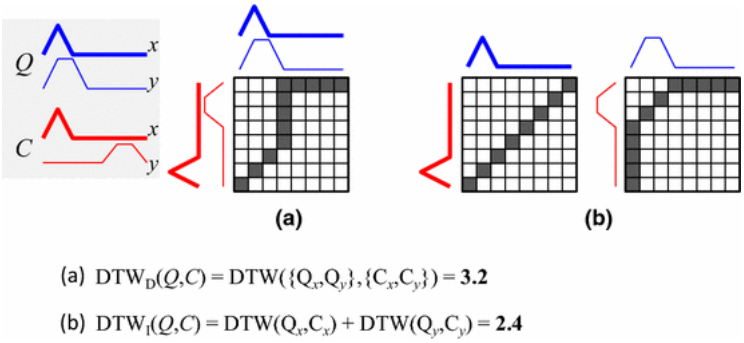


Figure 2.2.1: Difference between DTW_I and DTW_D taken from [66]

2.2.1 Independent DTW (DTW_I)

The independent strategy treats each dimension independently, has a different pointwise distance matrix M for each dimension, and then sums the resulting DTW distances.

$$DTW_I(\mathbf{a}, \mathbf{b}) = \sum_{k=1}^{d} DTW(\boldsymbol{a_k}, \boldsymbol{b_k}) \tag{2.2.2}$$

2.2.2 Dependent DTW (DTW_D)

Respectively, this assumes some relation among the series on the multivariate time series. For handling this case, the matrix $M_{i,j}$ is redefined not as the distance between 2 points on a single series but as the Euclidean distance between the 2 vectors that represent all the series.

$$M_{i,j} = \sum_{k=1}^{d} (a_{i,k} - b_{j,k})^2 \tag{2.2.3}$$

Then, the DTW distance is calculated which leads to distance time wrapping dependent(DTW_D).

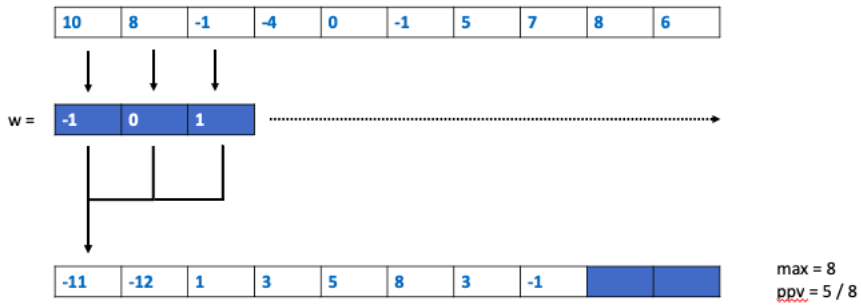


Figure 2.3.1: Example of a convolution/kernel applied to a time series and extracting the features used in ROCKET

2.2.3 Adaptive (DTW_A)

Shokoohi-Yekta et al. [66] discussed when a problem is independent or dependent. This discussion led to the creation of the adaptive case which uses a combination of the previous two and tries to define when it should be used. This method uses the dependent or independent distance depending on a threshold. The threshold is calculated in the training phase using cross-validation on training data and verifying which method works better.

2.3 Transformation based

One way to adapt traditional supervised learning algorithms such as neural networks, random forests, or linear classifiers to MTSC, is to convert from the 3-dimensional input data to a feature vector and then adapt any supervised learning algorithm to classify. The most common transformation-based algorithms for MTSC are: Shapelet based which will be introduced in sections 2.3.3 and 2.3.3, convolution kernels to transform data (ROCKET) 2.3.1 and dictionary approaches (WEASEL-MUSE) 2.4.2

2.3.1 The Random Convolutional Kernel Transform (ROCKET)

The Random Convolutional Kernel Transform, ROCKET [14] is a transformation-based approach that uses the concept of convolution/kernel as used in the deep learning field. A convolution/kernel can be described as a subseries used to derive discriminatory features. Each kernel is convolved with a time series through a sliding dot product. This process can be seen in figure 2.3.1.

ROCKET is a two-step process that consists of first generating a large number of random convolution kernels (10,000 by default) to be used as feature vectors. The second step consists of using a linear classifier of the resulting feature vector.

For each of the kernels generated, the following parameters are selected from the following spaces: The length, l, is selected such that, $l \in \{7, 9, 11\}$; the value of each weight, w_i, in the kernel is selected such that, $w_i \sim \mathcal{N}(\mu, \sigma^2)$, where $\mu = 0$ and $\sigma^2 = 1$; dilation, d, is sampled from an exponential scale up to input length and the binary decision to pad the series is chosen with equal probability, if true the series is zero-padded at the start and end equally such that middle element of the kernel is applied to every point in the input series.

The convolution of an instance and kernel can be interpreted as the dot product between two vectors. The resulting feature map is then used to generate two features that will be used in the next stage: 1) the max value and 2) the proportion of positive values(PPV). The PPV summarises the proportion of the series correlated to the kernel. It was found to significantly improve classification accuracy. Each series is subsequently transformed into a 20,000 attribute instance after all convolutions. This transformed dataset is then used to train the ridge regression classifier.

For MTSC problems, kernels are randomly assigned to several dimensions making this approach independent. Weights are then generated for each dimension. Convolution in this case can be interpreted as the dot product

Figure 2.3.2: Example of a shapelet (red area) extracted from a time series. In this example, it is assumed that the shape of the subsequence can be a good discriminator between classes.

between two matrices as the kernel convolves horizontally across the series.

2.3.2 Arsenal

ROCKET is a very fast classifier that has state-of-the-art accuracy, and it is probably the most important advance in recent development in the field. It represents a different class of approach, but it has a drawback. It does not generate a prediction probability but just indicates the class assigned. This makes it difficult to be combined with other approaches such as in the case of ensembles.

To minimize this problem, a homogeneous ensemble of ROCKET classifiers was proposed [53]. In this case, the probability of belonging to a class is the number of ROCKET classifiers that are assigned to that class. This ensemble is named Arsenal. Arsenal is slower to build than ROCKET, but its probabilities make it a better approach when it is required to get the probability of belonging to a class.

2.3.3 Shapelet based classifiers

This group of classifiers are transformation approaches and also can be implemented as dependent or independent. They are based on the concept of a shapelet that will is introduced in the following section.

Shapelet

A shapelet is a sub-sequence of time series data [78]. Figure 2.3.2 shows an example of a shapelet. In addition to the sub-sequence data, the shapelet contains other parameters which are defined in table 2.1. The shapelet can be used in several ways such as a criteria for a tree split (Generalized random shapelet forest) or as a feature vector (Shapelet transform classifier).

Parameter	Description
class	Class of the series that extracted the shapelet
series-index	Index of the time series where the shapelet was extracted
start	Position of the start of the shapelet within the series
start	Class of the series that extracted the shapelet
length	Size of the shapelet
dimension-index	the dimension where the shapelet was extracted
data	Depending of the nature is a vector or matrix
quality	Numeric value that measure shapelet utility

Table 2.1: Parameters of a shapelet.

Generalized Random Shapelet Forest (gRFS)

In gRFS shapelets are used as split criteria for building decision trees. First, it generates a group of p trees (forest), and based on those it uses an ensemble to do the classification. To reduce the complexity, p trees are generated by sampling the data as described in algorithm 1. Each tree is created by sampling randomly several shapelets and selecting the ones that are better at splitting the data at a decision tree node. This is shown on algorithm 2.

Shapelet transform classifier (STC)

STC is a three-step process which is described on algorithm 3. The first step is the shapelet search. The result of this search is a group of quality shapelets. The

Algorithm 1 Random Shapelet Forest($\boldsymbol{Z}, p, l, u, r$)

Parameters: The training set, $\boldsymbol{Z}$, the number of trees, p, the lower and upper shapelet length, l & u, the number of shapelets, r.
Return: An ensemble of generalized shapelet trees, $\boldsymbol{R} = \boldsymbol{ST}_1 \ldots \boldsymbol{ST}_p$

```
for i ← 1 to p do
    I_i ← sample(Z)
    ST_i ← randomShapeletTree(Z_Ii, l, u, r)
    R ← R ∪ ST_i
return R
```

Algorithm 2 Random Shapelet Tree($\boldsymbol{Z}, l, u, r$)

Parameters: The training set, $\boldsymbol{Z}$, the lower and upper shapelet length, l & u, the number of shapelets, r.
Return: A random shapelet tree, $\boldsymbol{ST}$

```
if isTerminal(Z) then
    return makeLeaf(Z)
for i ← 1 to r do
    S ← S ∪ sampleShapelet(Z, l, u, rand(l, u))
[t, S, k] ← bestSplit(Z, y, S)
ST_L ← randomShapeletTree(Z_L, l, u, r)
ST_R ← randomShapeletTree(Z_R, l, u, r)
return [[t, S, k, ST_L], [t, S, k, ST_L]]
```

shapelets are ranked based on quality criteria which in STC is an information gain factor. The second step consists of extracting features from the shapelets. The value used as a feature vector is the minimum distance between the shapelet and the time series. The third step consists of applying a supervised learning algorithm to the feature vector dataset. In STC, the resulting feature vector dataset is classified with a rotation forest [61].

Algorithm 3 $STC_{TRAIN}(\mathbf{X}, y, params)$

```
shapelets ← ShapeletSearch(X, y, params)
X_T ← transform(shapelets, X, y)
model ← classifier.train(X_T, y)
return model
```

Shapelet search

The shapelet search has several parameters to tune the search process. These parameters are described in table 2.2.

Parameter	Description	Default value
k	Maximum number of shapelets	$min(1000, 10 * trainSize)$
min	Minimum size for a shapelet	3
max	Maximum size for a shapelet	series length
max iterations	Number of maximum iterations	1 million
contract time	Maximum time for shapelet search	24 hours
no improvement	Maximum number of consecutive iterations before stop	1000

Table 2.2: List of STC parameters.

Algorithm 4 ShapeletSearch($\mathbf{X}$,y,params)

```
shapelets ← {}
repeat
  shapelet ← generateShapelet(X, y)
  shapelet.quality ← shapeletQuality(shapelet, X, y)
  shapelets ← shapelets ∪ shapelet
  if iteration%1000 = 0 then
    shapelets ← sortByQuality(shapelets)
    shapelets ← removeShapelets(params.k, shapelets)
until params.stopCriteria
return Shapelets
```

The shapelet search is described on algorithm 4. The *generateShapelet* function guides the search by indicating how to get the next shapelet to consider, then the shapelet quality is calculated and after some iterations, the shapelets are sorted and kept to a maximum distance k.

In the original implementation of STC [45], the *generateShapelet* is processed instance by instance by generating all possible shapelets on each instance and retaining the best ones. This is infeasible on most problems based on the high number of possible shapelets that need to be considered. To minimize this problem some improvements were proposed in [9]. For example, add a contract time and perform a random search of shapelets instead of searching for all options.

Random shapelet generation

Algorithm 5 generateShapelet(**X**,y,params)

$index \leftarrow random(1, |X|)$
$dimension \leftarrow random(1, dimension)$
$start \leftarrow random(1, |x_{index}|)$
$size \leftarrow random(params.min, params.max)$
$shapelet \leftarrow \{index, dimension, start, size, y_{index}\}$
$shapelet.data \leftarrow x_{index}[start, size]$
$shapeletDataset \leftarrow \emptyset$
for all $x_i \in X$ **do**
  $d_i \leftarrow sDist(s.data, x_i)$
  $c_i \leftarrow \begin{cases} 1, & y_i = s.class \\ 0, & o.w \end{cases}$ {Binary}
  $shapeletDataset = shapeletDataset \cup (d_i, c_i)$
$shapelet.quality \leftarrow InformationGain(D)$
return *shapelet*

To improve STC a completely random search process is used, on each iteration a random time series, start, size, and dimension is selected, This process is described on algorithm 5

The generate shapelet method is one of the key procedures that define the accuracy of the Random Shapelet Transform Classifier (RSTC). First, a shapelet is generated randomly (algorithm 5). Then, a new train set *shapeletDataset* is created. This set contains one feature and the class. The feature is the minimum distance between the shapelet and the current time series. In the pseudo-code, the function is named *sDist*. The other parameter is the class of the time series. For multi-class problems, this means that we have c different values. However, in [8] it was shown that using binary values increases accuracy, so the value of the class is 1 if the shapelet and the train data have the same class or 0 otherwise.

The last element of this process is the measurement used to verify the quality of the shapelet. It is possible to use several metrics such as correlation and chi-squared but by default, information gain is used [8]. This measure has been used on decision tree algorithms for many years to measure the quality of a feature.

Random Dilated Shapelet Transform (RDST)

RDST [27] is a shapelet-based algorithm that adopts many of the techniques of convolution used in ROCKET. RDST randomly selects a large number of shapelets from the train data and then trains a linear RIDGE classifier on features derived from these shapelets.

RDST employs dilation with shapelets. Dilation is a form of down sampling, in that it defines spaces between time points. Hence, a shapelet with dilation d is compared to time points d steps apart when calculating the distance. RDST also uses two features in addition to $sDist()$: it encodes the position of the minimum distance and records a measure of the frequency of occurrences of the shapelet based on a threshold. Hence the transformed data has 3 features for each shapelet. RDST was not included in the experiments described in this thesis as they were performed before it was created.

2.3.4 The Multiple Representation Sequence Learner (MrSEQL)

The Multiple Representation Sequence Learner, MrSEQL [56] extends previous adaptations of the SEQL classifier [55] in two ways. Firstly, via the introduction of ensembling and secondly, via the addition of integrating the SFA [64] transform. In the resulting approach, shown in Figure 2.3.3, the data is transformed via either Symbolic Aggregate Approximation (SAX) [43] or SFA before being used to train a SEQL classifier. The window length, l, is adjusted before each addition to the ensemble. During testing each instance is transformed accordingly before being classified by the appropriate model. The output probability distribution is then the per-class mean over all models.

The SEQL learner was developed for the classification of biological sequences such as DNA and employs a tree-based approach coupled with a pruning strategy to explore the feature space. As a result, the SFA and SAX approaches are particularly well suited as tools for transformation into the symbolic space. The

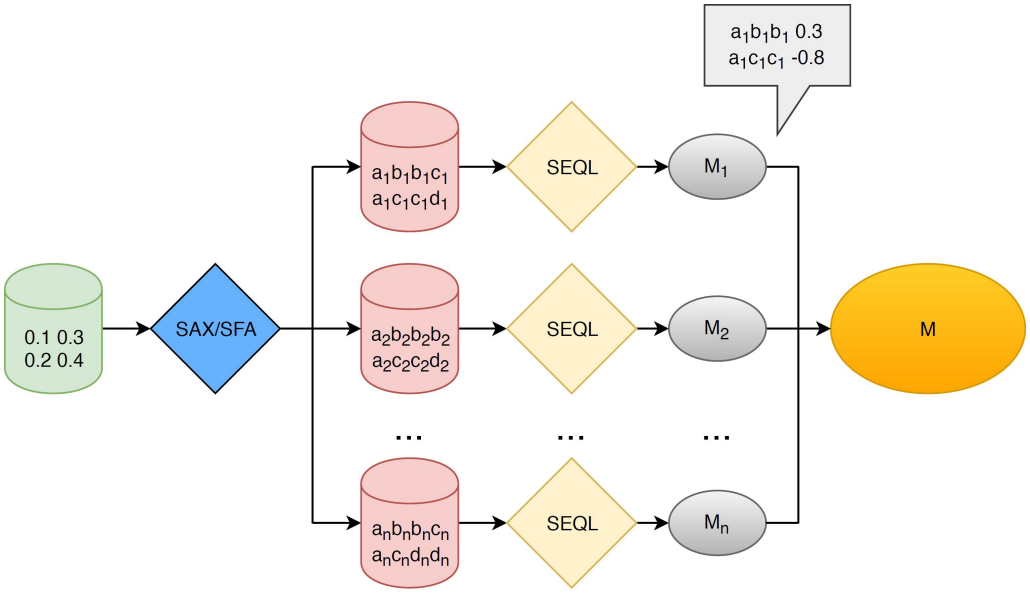


Figure 2.3.3: A depiction of the MrSEQL classifier taken from [55].

SAX approach achieves this conversion by:

1. producing a piece-wise aggregated series;
2. creating a look-up table from the new series, in which the domain is divided by alphabet length a; and
3. deriving the symbolic word, by looking up each aggregated value.

The process of aggregation and the creation of the look-up table is undertaken before sliding a window of length l across the series. At each step, a word of length w is derived and added to the symbolic representation. The SFA approach achieves this conversion by:

1. Performing a discrete Fourier transform (DFT) on each window of the instance;
2. creating an $a \times w$ look-up table in which the alphabet boundaries are distinct for each letter index; and
3. deriving the symbolic word, by looking up each aggregated value.

The process of deriving the lookup table is undertaken after the DFT. The alphabet boundaries are then calculated per word position index. As a result, there are effective w alphabets of size a. MrSEQL can handle multivariate data using the following strategy: During the prepossessing of data, dimensions are processed sequentially and appended to one another creating n instances, each one of size $m \times c$.

MrSQM

MrSQM [57] is an extension of MrSEQL with added features. This includes 1) The use of SFA and SAX to discretize time series subseries into words. 2) Uses a trie to store and rank frequent substrings, 3) Applies either (a) a supervised chi-squared test to identify discriminative words or (b) an unsupervised random substring sampling method to prevent overestimating highly correlated substrings that are likely to be redundant. As in the case of RDST, MrSQM was not included in the experimentation.

2.4 Dictionary approaches

2.4.1 CBOSS

The Bag-of-SFA-Symbols (BOSS) [63] BOSS is an algorithm that compares algorithms based on histograms of discretized words. The steps of BOSS are:

- Extracts substructures (patterns) from a time series. For multivariate data, CBOSS extracts patterns from the different dimensions.
- Applies low pass filtering and quantization to the substructures, which reduces noise and allows for string-matching algorithms to be applied
- Two time series are then compared based on the differences in the set of noise-reduced patterns

CBOSS is an extension of BOSS that adds a contract time to limit the time for extracting patterns.

2.4.2 WEASEL+MUSE

Originally a univariate time series classifier, Word Extraction for Time Series Classification, WEASEL [65] was extended to include the Multivariate Unsupervised Symbols and Derivatives, MUSE stage for MTSC. Words in the form of unigrams and bigrams are extracted for all series and dimensions using a sliding window for a range of window lengths. These words are extracted using the Symbolic Fourier Approximation, SFA [64] with equi-depth or equi-frequency binning. Words for the derivatives (differences between neighboring points in the series) of each dimension are also taken and treated as additional dimensions. The words for each dimension and window length are concatenated into a single bag of words histogram for a series. Because the histogram is specific to a series, we consider this an independent method. As this process produces a lot of words with a presumed amount of redundancy and to filter out unproductive dimensions, a χ^2 test is used for feature selection. The remaining words are used to build a logistic regression classifier.

A 10-fold cross-validation is performed to select parameters for the final WEASEL+MUSE model. These are the word length l, the binning method b, and whether to normalize each window p. The WEASEL+MUSE build process is displayed in Algorithm 6.

2.4.3 Temporal Dictionary Ensemble (TDE)

Temporal Dictionary Ensemble (TDE) [52] is a dictionary classifier that includes several novel features. This approach commonly adapts the bag-of-words model used in other domains such as signal processing, computer vision, and audio processing for time series data

Algorithm 6 WEASEL+MUSE(A list of n cases of length m with dimension d, $\mathbf{T} = (\mathbf{X}, \mathbf{y})$)

Parameters: the word length l, the alphabet size α, the maximal window length w_{max}, mean normalisation parameter p, equi-depth or equi-frequency binning b

Let $\mathbf{H}$ be a collection of n histograms $\mathbf{h}$
Let $\mathbf{B}$ be a matrix of l by α breakpoints found using b
$\mathbf{X}' \leftarrow$ addDerivativesAsDimensions($\mathbf{X}$)
for $i \leftarrow 1$ to n **do**
  for $k \leftarrow 1$ to $2d$ **do**
    for $w \leftarrow 2$ to w_{max} **do**
      for $j \leftarrow 1$ to $m - w + 1$ **do**
        $\mathbf{o} \leftarrow x'_{i,j,k} \dots x'_{i,j+w-1,k}$
        $\mathbf{q} \leftarrow \text{DFT}(o, w, p)$ { $\mathbf{q}$ *is a vector of the complex DFT coefficients*}
        $\mathbf{r} \leftarrow$ SFAlookup($\mathbf{q}, \mathbf{B}$)
        $pos \leftarrow$index($\mathbf{k}, \mathbf{w}, \mathbf{r}$)
        $h_{i,pos} \leftarrow h_{i,pos} + 1$
$h \leftarrow \chi^2(h, y)$ { *feature selection using the chi-squared test* }
fitLogistic(h, y)

TDE is an ensemble of 1-NN classifiers that transforms each series into a histogram of word counts. A sliding window of length w is run along each series, and the subseries is discretized into a word of length l from an alphabet of size α. TDE transforms the window using the Symbolic Fourier Approximation (SFA). Distance between histograms is found using histogram intersection. In addition to word frequencies, TDE also captures the frequencies of bigrams found from non-overlapping windows. Thus a transformed instance includes a histogram of word counts and bigram counts for a given trio of parameters (w, l, α). TDE also includes some spatial information by the utilization of spatial pyramids (Lazebnik et al., 2006). This involves splitting a series into h levels each with $2v$ disjoint subseries, where v is the current pyramid level. Word counts are found for each subseries independently, then the resulting histograms are concatenated. The distance to histograms of deeper levels with smaller spatial areas in the series is weighted higher than global similarity. Bigrams are only recorded for the first level consisting of the full series. The SFA transform requires a set of breakpoints when creating words. The method of generating these breakpoints b is selected between Multiple Coefficient Binning (MCB) and Information Gain Binning (IGB). Windows can

optionally be normalized during the transform with the p parameter. The TDE ensemble is filtered into s total classifiers from k candidates. The accuracy of each candidate is estimated using leave-one-out cross-validation (LOOCV), with the highest s being retained. Diversity is achieved through altering the parameters (w, l, h, b, p) for each new classifier and a 70% sampling of the train data. The first 50 classifiers use randomly selected parameters, while those after are selected using a Gaussian processes regressor. For unseen parameter sets, a prediction of accuracy is made using the parameters of previously built classifiers, with the highest predicted accuracy being chosen for the next classifier build.

2.5 Deep learning

The 3-dimensional data is a natural fit as input for convolutional deep learning approaches. Depending on how the convolutions are processed we can consider these methods either independent or dependent. If the convolution extracts features from all the dimensions we could consider it as a dependent case, if only considering one dimension it can be considered as an independent method.

Deep learning methods have been one of the most successful methods in the last years and it is a natural option for MTSC. Despite their strength and popularity in handling 2D image data, a result of AlexNet's performance on the ImageNet dataset [38], deep learning approaches have only more recently been heavily studied in the 1D time series domain. Knowledge gained from the former can be utilised on the latter, and can now similarly be quickly transferred to the multivariate time series case.

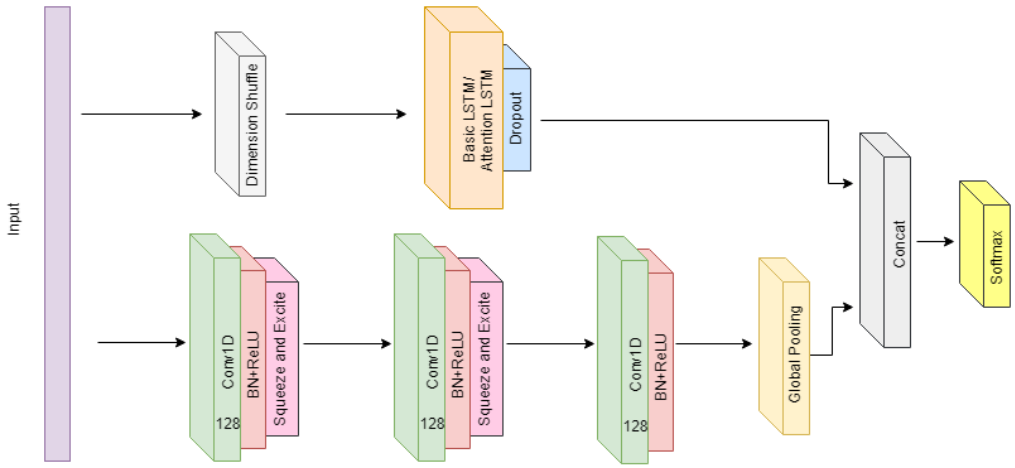


Figure 2.5.1: MLSTM-FCN architecture, figure from [32].

2.5.1 The Multivariate Long Short Term Memory Fully Convolutional Network (MLCN).

This method combines two well-known deep learning architectures: long short-term memory (LSTM) and convolutional networks (CN). LSTM are recurrent networks, which work better on temporal data such as time series. CN is an extension of the classical 3-layer feed-forward neural network but with more layers, each learning some of the patterns. The learning algorithm back propagates the error to each of the layers. Finally, both predictions are combined. This is described in image 2.5.1. To adapt to MTSC a squeeze and excitation block is added [33].

2.5.2 Residual Network (ResNet)

ResNet was first applied to time series classification in [71]. It is a network of three consecutive blocks, each comprised of three convolutional layers, which are connected by residual 'shortcut' connections that add the input of each block to its output. Residual connections allow the flow of gradient directly through the network, combating the vanishing gradient effect [29]. The residual blocks are followed by global average pooling and softmax layers to form features and

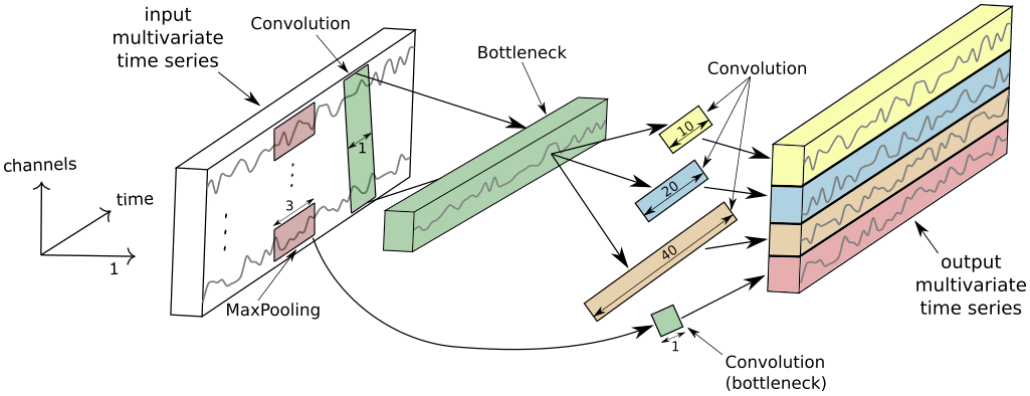


Figure 2.5.2: An Inception module with example parameters, figure from [22]. Three of these are concatenated to form a block in InceptionTime.

subsequent predictions. We maintain all hyperparameter settings and optimizer settings from the [21] evaluation.

2.5.3 InceptionTime

InceptionTime achieves high accuracy through a combination of building on ResNet to incorporate Inception modules [67] and ensembling over five multiple random-initial-weight instantiations of the network for greater stability [22]. A single network out of the ensemble is composed of two blocks of three Inception modules each, as opposed to the three blocks of three traditional convolutional layers in ResNet. These blocks maintain residual connections and are followed by global average pooling and softmax layers as before.

An Inception module is summarised in Figure 2.5.2. It takes an input multivariate series of length m, dimensionality d, and first uses a bottleneck layer with length and stride 1 to reduce the dimensionality to $d' < d$ while maintaining output length m. This greatly reduces the number of parameters to later learn. Convolutions of different lengths are applied to the output of the bottleneck layer to find patterns of different sizes. The outputs of these convolutions are combined with an additional source of diversity, a Max Pooling followed by bottleneck (with the same value of d') applied to the original time series, and all stacked to form the dimensions of the output multivariate time

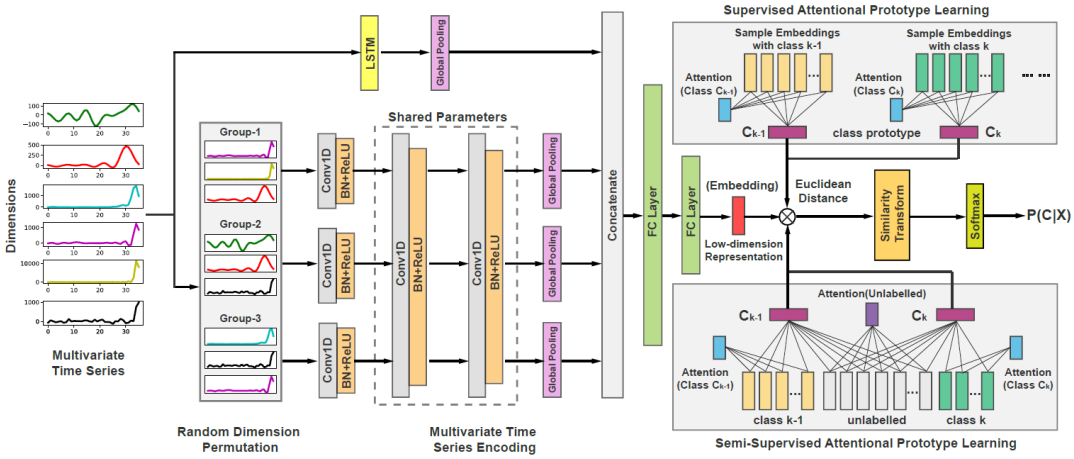


Figure 2.5.3: TapNet architecture, figure from [79].

series to be fed into the next layer.

2.5.4 Time Series Attentional Prototype Network (TapNet)

A novel approach aimed at tackling problems in the multivariate domain, the TapNet architecture draws on the strengths of both traditional and deep learning approaches. [79] notes that deep learning approaches excel at learning low dimensional features without the need for embedded domain knowledge whereas traditional approaches such as 1NN-DTW work well on comparatively small datasets. TapNet combines these advantages to produce a network architecture that can be broken down into three distinct modules: Random Dimension Permutation, Multivariate Time Series Encoding, and Attentional Prototype Learning.

Random Dimension Permutation is used to produce g groups of randomly selected dimensions to increase the likelihood of learning how combinations of dimension values affect class value. The group size is defined as $\varphi = \lfloor \frac{m \cdot \alpha}{g} \rfloor$, where α is the scale factor, controlling the number of dimensions used over m, where m is the number of dimensions. This process is illustrated in Figure 2.5.3 where the six input dimensions are reorganized into three groups of three. Experimentation exploring the effect of this module found that in 22 out of 33 datasets in the UEA

multivariate archive, the accuracy was increased. However, it is unclear whether it has a significant effect or whether the effect on accuracy is a function of dataset characteristics.

Encoding in the TapNet architecture is undertaken in $g + 1$ stages before the output features are concatenated and passed through two fully connected layers. Each group produced in the dimension permutation module is passed through three sets of one-dimensional convolutional layers followed by batch normalization, Leaky Rectified Linear Units, and finally a global pooling layer. For the first of these three sets the weights and bias are distinct for each group. In addition to the group encoding process, the raw data is passed through an LSTM and global pooling layer. The output from each of the global pooling layers is then concatenated before being passed through two fully connected layers. This process results in a low-dimensional feature representation of the original series. The default filter values for the convolution layers are set as 256, 256, and 128 whilst the default kernel values are five, eight, and three. The default value for the LSTM layer is 128. It is intended that interaction between dimensions can be learned more effectively by the Random Dimension Permutation process before the encoding is then combined, producing features aligned with a dataset dimensions. Furthermore, the inclusion of the LSTM layer is intended to learn longitudinal features.

Finally, for each class, a prototype candidate is produced. Although the architecture does allow for unlabelled test data to be included in the prototype derivation via Semi-supervised Attentional Prototype Learning. This feature was not utilized. As a result, the class prototypes are defined solely by the training data. The objective of the candidate production is to minimize the distance to all members of the class which the prototype is produced for whilst maximizing the distance between the prototypes. The probability of class membership is then assigned to test instances as a function of their proximity to each class prototype. In this case, the similarity is measured by way of Euclidean distance.

2.6 Interval based

Interval-based approaches differ from previous approaches by extracting parts of the time series that could be used to discriminate between classes in combination with other techniques. This section focuses on Random Interval Spectral Ensemble (RISE) 2.6.1, Canonical Interval Forest 2.6.2 and Diverse Representation of Canonical Interval Forest 2.6.3

2.6.1 The Random Interval Spectral Ensemble (RISE)

RISE [23] is a tree-based ensemble time series classification algorithm, where each tree is built on a distinct set of Fourier, autocorrelation, and partial autocorrelation features.

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.

2.6.2 Canonical Interval Forest (CIF)

The Canonical Interval Forest, CIF [51] is an ensemble of time series tree [16] classifiers built using the Canonical Time-Series Characteristics, Catch22 [49] features and simple summary statistics extracted from phase dependant intervals. The time series tree uses a simplistic tree structure, comparing all attributes at each node and performing no pruning. However, the tree introduces a novel tie-breaking measure in the form of entrance gain. Catch22 is a set of 22 highly discriminative and low redundancy features extracted from the 7000+ time series features available in the Highly Comparative Time Series Analysis (hctsa) toolbox [24].

To create a diverse ensemble, a summary features of the 25 available are randomly subsampled and k intervals of random length and start point are selected to build

each tree. CIF was extended for MTSC by randomly selecting the dimension each interval extracted from. The build process for the CIF ensemble is described in Algorithm 7.

Algorithm 7 Canonical Interval Forest(A list of n cases of length m with d dimensions, $\mathbf{T} = (\mathbf{X}, \mathbf{y})$)

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

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

2.6.3 Diverse Representation Canonical Interval Forest (DrCIF)

The Diverse Representation Canonical Interval Forest (DrCIF) is an interval-based ensemble and an extension of its prototype version, the Canonical Interval Forest (CIF). Interval-based classifiers extract phase-dependent subseries, aiming to find discriminatory features over different intervals. For time series of length m there are $\frac{m(m-1)}{2}$ possible intervals that can be extracted.

The base classifier for DrCIF is a simple information gain-based tree used in TSF. Features from the tree are derived from multiple intervals taken from the base series, the first-order difference series and the periodograms of the whole series. Intervals from each are randomly selected. Seven basic summary statistics are part of a pool of possible features extracted from an interval of any one of the three representations. These are: the mean; standard-deviation; slope; median;

inter-quartile range; min; and max. DrCIF adds the catch-22 features to this selection of summary statistics to form a candidate pool of 29 features. out of the 29 features available are randomly selected for each tree. For each of the 3 representations, k phase-dependent intervals with randomly selected positions and lengths are extracted. The selected features are then calculated from each interval. These features are concatenated into a $3 \cdot k \cdot a$ length vector for each series, and the new dataset is used to build the tree. Diversity is achieved by providing each base classifier with different intervals and a different subset of the 29 features.

Generally, we select k as a function of the representation series length rm. Each representation will differ in its length, with the periodogram being half the size of the base series and the differences having one less value. As such it is likely the number of intervals selected for each representation will differ. For multivariate data, DrCIF randomly selects the dimension used for each interval.

2.7 Heterogeneous ensembles

An ensemble uses several classifier predictions and makes a final prediction based on the specific predictions on each component. The ensembles can be homogeneous (versions of the same classifier) or heterogeneous (using different classifiers). The ensembles are a natural way to extend the functionality of current MTSC approaches by combining several classifiers into one. The final decision can be based, for example, on a majority vote, or a weighted decision based on component importance. Some of the algorithms presented in the previous section are homogeneous. This section focuses on the most important heterogeneous ensembles: HIVE-COTE version 1 2.7.1, version 2 2.7.2 and independent 2.7.3.

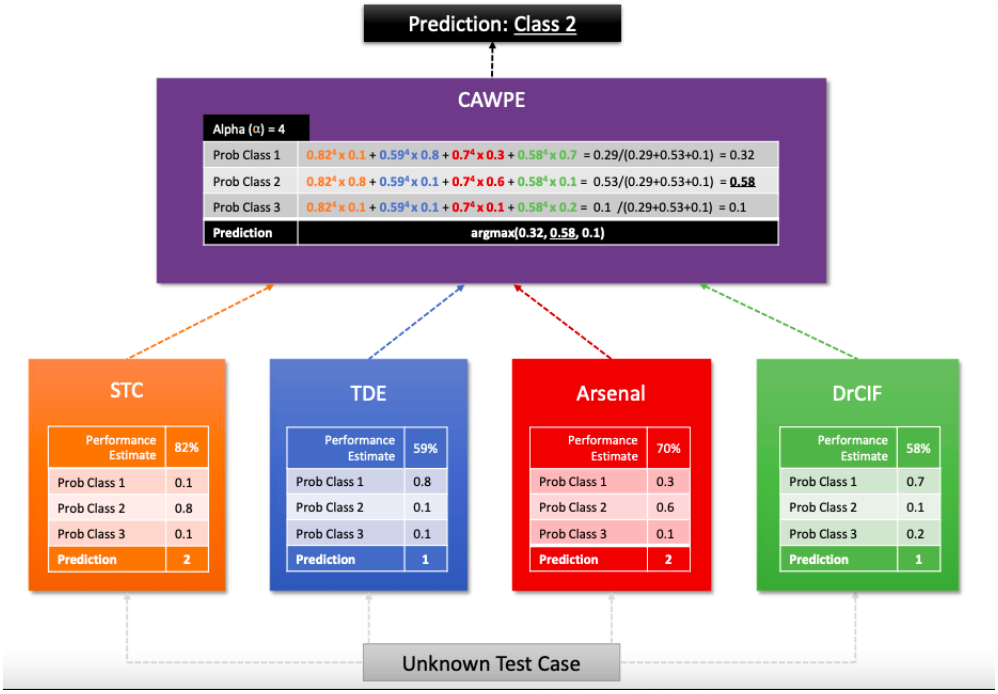


Figure 2.7.1: An example of a HIVE-COTE 2 prediction taken from [53].

2.7.1 HIVE-COTE 1

HIVE-COTE version 1 [3] is a heterogeneous ensemble of four classifiers that combine to improve the probability of an example belonging to a class. On all HIVE-COTE variants, the probability is obtained using Cross-validation Accuracy Weighted Probabilistic Ensemble (CAWPE) [42]. This version includes as classifiers Shapelet Transform Classifier (STC) [30]; Time Series Forest (TSF) [16]; Contracted Bag of Symbolic-Fourier Approximation Symbols (CBOSS) [63] and Random Interval Spectral Ensemble (RISE)[46].

2.7.2 HIVE-COTE 2

The second version of HIVE-COTE modified three of four classifiers were replaced and only STC remained, The classifiers added are Temporal Dictionary Ensemble (TDE) 2.4.3, Diverse Representation Canonical Interval Forest (DrCIF) 2.6.3 and Arsenal 2.3.2.

2.7.3 HIVE-COTE Independent

As mentioned earlier, one of the most straightforward techniques to adapt TSC algorithms to multivariate is to consider independence over dimensions and ignore relations among them. This approach can be a good baseline for assessing bespoke MTSC.

In this approach, each component builds a separate classifier on every dimension, then combines the predictions from each dimension to produce a single probability distribution for each component using CAWPE [42].

3

The UCR/UEA MTSC Archive

Contributing publications

- Alejandro Pasos Ruiz, Michael Flynn, and Anthony J. Bagnall. Benchmarking multivariate time series classification algorithms. ArXiv e-prints, ArXiv:2007.13156, 2020 [60]

3.1 Introduction

There has been an increase in data caused by the facility to obtain it from many sensors such as mobile phone accelerometers, medical devices, sound waves, etc. These are natural sources of data for MTSC given they are temporally ordered data and usually, there are several different sensors used on the classification problem. Therefore, it is important to have an archive that can include many different types of problems and ease the process of testing and comparing the newest algorithms.

There have been several efforts to generate an archive of problems. For example, [5] presented a list of univariate and multivariate problems. The most important is the UEA/UCR [12] archive for univariate problems. It contains data for several problem types and includes over 100 problems from several domains.

In 2018, the researchers at UEA started a repository of problems to be used as standard benchmarking MTSC algorithms. In this chapter, we will introduce a list of 33 problems in this repository categorized by sensor type (accelerometer, medical scan, handwriting, gesture recognition, sound, and others). This list of problems is presented in table 6.2. Details can be found on the associated website [1].

This chapter and the contributing paper collaborate with the effort of creating a reliable archive for MTSC problems to help researchers have comparable results to improve the quality of the algorithms. Specifically, some of the datasets had

[1] https://www.timeseriesclassification.com

Table 3.1: Summary of the 33 datasets in the 2018 version used in experimentation.

	Name	Train size	Test size	Num Series	Series length	Classes
AWR	ArticularyWordRecognition	275	300	9	144	25
AF	AtrialFibrillation	15	15	2	640	3
AOC	AsphaltObstaclesCoordinates	390	391	3	Variable	4
APTC	AsphaltPavementTypeCoordinates	1054	1055	3	Variable	3
ARC	AsphaltRegularityCoordinates	751	751	3	Variable	2
BM	BasicMotions	40	40	6	100	4
CT	CharacterTrajectories	1422	1436	3	Variable	20
CR	Cricket	108	72	6	1197	12
DDG	DuckDuckGeese	50	50	1345	270	5
EW	EigenWorms	128	131	6	17984	5
EP	Epilepsy	137	138	3	206	4
EC	EthanolConcentration	261	263	3	1751	4
ER	ERing	30	270	4	65	6
FD	FaceDetection	5890	3524	144	62	2
FM	FingerMovements	316	100	28	50	2
HMD	HandMovementDirection	160	74	10	400	4
HW	Handwriting	150	850	3	152	26
HB	Heartbeat	204	205	61	405	2
IW	InsectWingbeat	30000	20000	200	Variable	10
JV	JapaneseVowels	270	370	12	Variable	9
LIB	Libras	180	180	2	45	15
LSST	LSST	2459	2466	6	36	14
MI	MotorImagery	278	100	64	3000	2
NATO	NATOPS	180	180	24	51	6
PD	PenDigits	7494	3498	2	8	10
PEMS	PEMS-SF	267	173	963	144	7
PS	PhonemeSpectra	3315	3353	11	217	39
RS	RacketSports	151	152	6	30	4
SRS1	SelfRegulationSCP1	268	293	6	896	2
SRS2	SelfRegulationSCP2	200	180	7	1152	2
SAD	SpokenArabicDigits	6599	2199	13	Variable	10
SWJ	StandWalkJump	12	15	4	2500	3
UW	UWaveGestureLibrary	120	320	3	315	8

The UEA MTSC archive [1] released in 2018 contains multivariate datasets, of which seven are not all equal in length.

inconsistencies and small errors that made them fail on some algorithms. Some of the fixes involved removing outliers and incorrect values. In some cases, adjust the data to have an equal length which helps some algorithms, Finally, convert the data into different file formats while preserving the equality of the information.

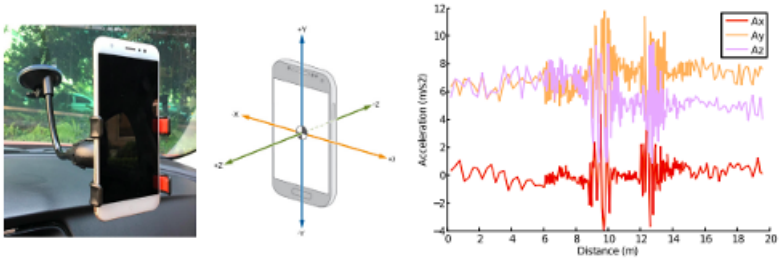


Figure 3.2.1: Multivariate time series example for asphalt problems extracted from [13]

In addition, a new problem grouping based on the type of problem was added. The purpose of this is to discover if some algorithms perform better in specific types of problems. In the rest of the Chapter, we introduce this grouping and a brief description of each problem.

3.2 Accelerometer data

The fast adoption of mobile devices that integrate accelerometers and gyroscopes has made it easy to use this sensor on a variety of MTSC problems. Generally, these problems gather the variation of movement on 3 axes (x, y, z).

3.2.1 Asphalt

This is a series of 3 problems [13] related to asphalt classification which can be seen in figure 3.2.1. In this project, data from cellular phone's accelerometers were used to determine certain properties on asphalt which were:

- Regularity: Two class problem that determine if a pavement is regular or deteriorated
- Pavement type: The idea is to detect if the street is made of pavement, cobblestone, or dirty road
- Obstacles: In this case, the goal is to detect if the street contains a speed

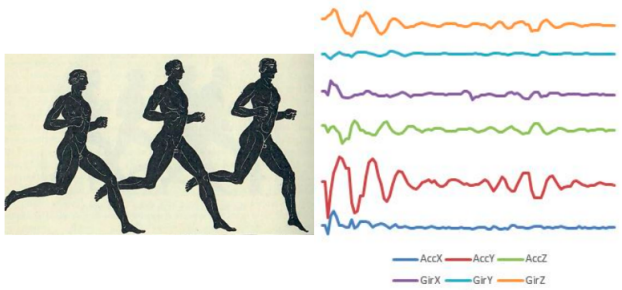


Figure 3.2.2: Multivariate time series example for basic motions extracted from [1]

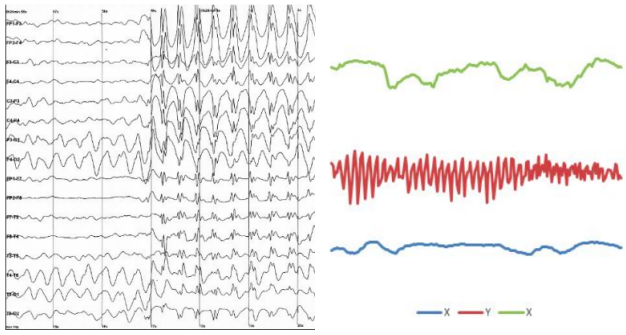


Figure 3.2.3: Multivariate time series example for epilepsy extracted from [1]

bump, vertical patch, pavement markers, or crosswalk.

This project used 1-NN combined with several distance-based functions such as dynamic time wrapping, derivative time wrapping, derivative transform function, and longest common sub-sequence. To get the amplitude invariant the same, a complexity invariant correction factor was added to the distance function.

3.2.2 Basic Motions

This student project is shown in figure 3.2.2 which has the goal of predicting if a person is walking, resting, running, or playing badminton using the accelerometer sensors of a mobile device.

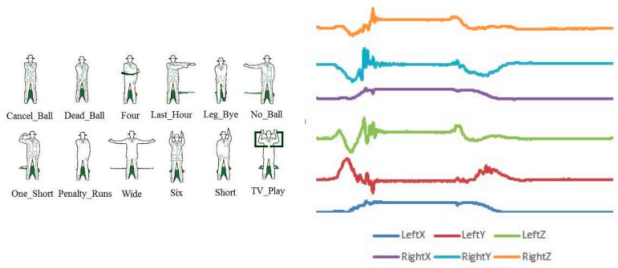


Figure 3.2.4: Multivariate time series example for cricket extracted from [1]

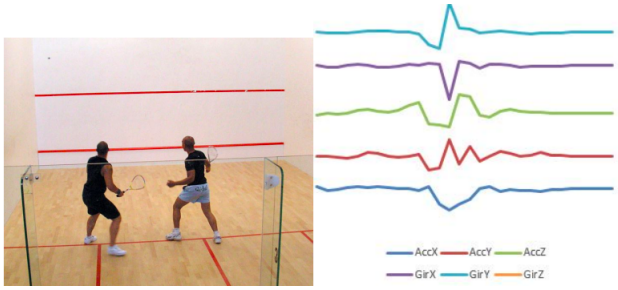


Figure 3.2.5: Multivariate time series example for racket sports extracted from [1]

3.2.3 Epilepsy

The main goal of this problem is to detect if a person is having an epilepsy attack compared to performing another activity such as walking, sawing, etc.[69]. In figure 3.2.3 there are some signal examples extracted from this problem.

3.2.4 Cricket

The problem looks to automatically detect which signal is sent by a cricket umpire. In figure 3.2.4 we can see some examples of these signals. The data was collected by placing accelerometers on each wrist of cricket umpires [37].

3.2.5 Racket sports

This student problem aims to identify which stroke the players are making. An example of this movement can be seen in figure 3.2.5. The data is obtained from an accelerometer and gyroscope of a smartwatch.

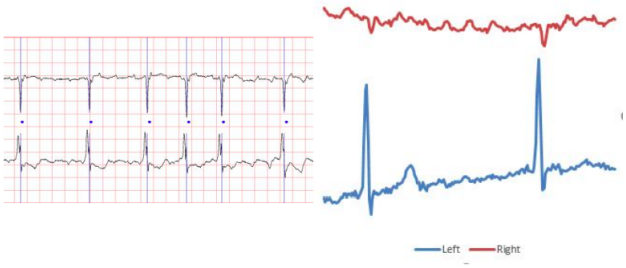


Figure 3.3.1: Multivariate time series example for atrial fibrillation extracted from [1]

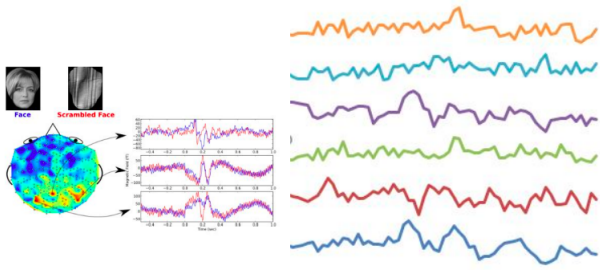


Figure 3.3.2: Multivariate time series example for face detection extracted from [1]

3.3 Medical scan data

Many specialized medical devices generate data that can be considered as a multivariate time series to help detect several health problems.

3.3.1 Atrial fibrillation

This problem looks to predict spontaneous termination of atrial fibrillation (AF) [26] using electrocardiogram (ECG) signals. In figure 3.3.1 there is an example of an ECG signal.

3.3.2 Face detection

This problem uses brain scans as shown in figure 3.3.2 to detect if it belongs to a normal brain or a scrambled using Magnetoencephalography (MEG) device [58].

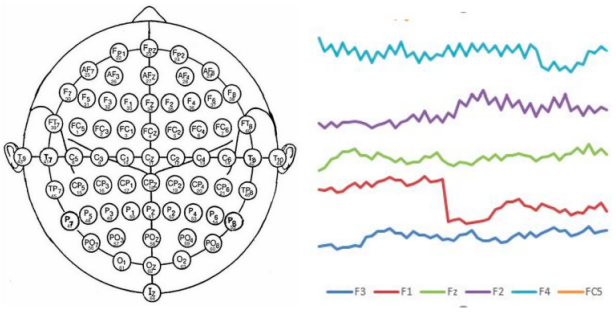


Figure 3.3.3: Multivariate time series example for finger movements extracted from [1]

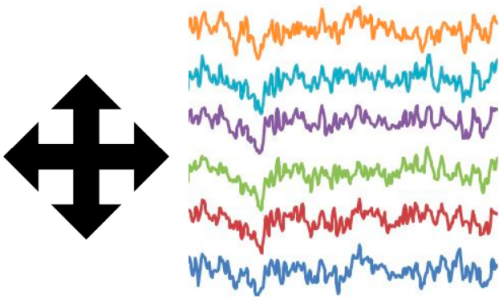


Figure 3.3.4: Multivariate time series example for hand movement direction extracted from [1]

3.3.3 Finger movements

This problem also uses scans to predict the finger movements of a person using its index and little finger [6]. A series of different sensors are put in a person's head to scan the activity. This can be seen in figure 3.3.3.

3.3.4 Hand movement direction

This problem looks to predict which direction a hand is moved but in this case, the sensors are obtained using a Magnetoencephalography (MEG) device [7].

3.3.5 Heart beat

This problem looks to classify a normal heartbeat from an abnormal [47]. The data is obtained by recording the heartbeat from different parts of the body.

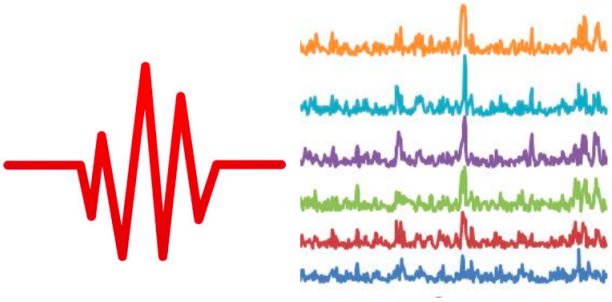


Figure 3.3.5: Multivariate time series example for heartbeat extracted from [1]

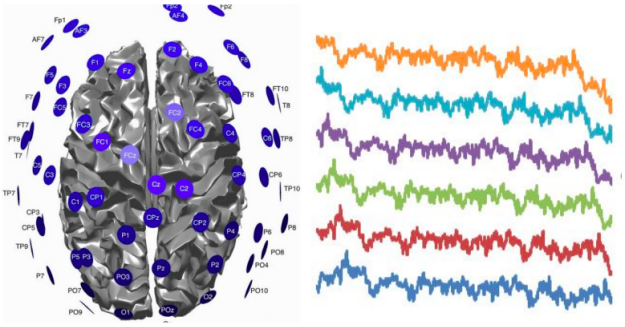


Figure 3.3.6: Multivariate time series example for motor imagery extracted from [1]

3.3.6 Motor imagery

In this problem, the goal is to detect different states concerning the motivation of a person [40]. The data is obtained using Electrocorticography (ECoG) scanning and this can be seen in figure 3.3.6.

3.3.7 Stand walk jump

This problem looks to detect if a person is standing, walking, or a jump. Instead of using an accelerometer, this problem uses electrocardiogram (ECG) scans. The sensors are aligned to the hearth as can be seen on figure 3.3.7

3.3.8 Self regulations 1 and 2

In this problem, a person was asked to move a cursor up and down on a computer screen, while his cortical potentials were taken. During the recording, the subject

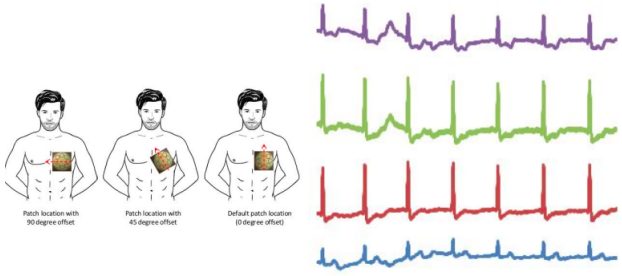


Figure 3.3.7: Multivariate time series example for stand walk jump extracted from [1]

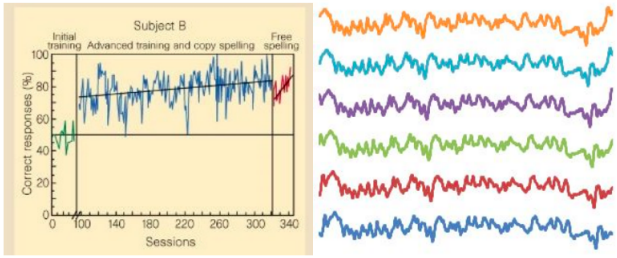


Figure 3.3.8: Multivariate time series example for self-regulations 1 extracted from [1]

received visual feedback on his slow cortical potentials (Cz-Mastoids). Cortical positivity leads to a downward movement of the cursor on the screen. Cortical negativity leads to an upward movement of the cursor [54].

3.4 Handwriting problems

Handwritten character detection has been one of the most studied problems in machine learning. One way to solve this problem is to consider the way the character is generated as a time series.

3.4.1 Character Trajectories

This dataset was obtained from the UCI repository and is used as a character detection problem. The data obtained is the coordinates (x,y,z) trajectory of handwriting characters. An example of this trajectory is shown in figure 3.4.1

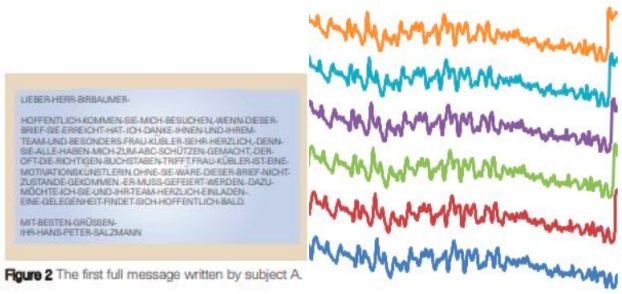


Figure 3.3.9: Multivariate time series example for self-regulations 2 extracted from [1]

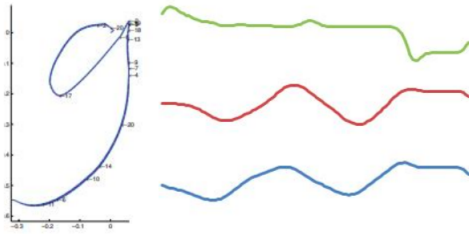


Figure 3.4.1: Multivariate time series example for character trajectories 1 extracted from [1]

In other work, this data has been used to solve the problem of character generation [73] [74] [75] using Hidden Markov Models but that problem is not considered here. This dataset is used to determine which character is being written based on its trajectory.

3.4.2 Handwriting

A data set of motion taken from a smartwatch whilst the subject writes the 26 letters of the alphabet created at UCR [66]. An example of this data can be seen in figure 3.4.2

3.4.3 Pen digits

Another handwritten problem but in this case are the digits that need to be classified using the (x,y) position of the pen movement on the plane. [20]. Some examples are shown in figure 3.4.3

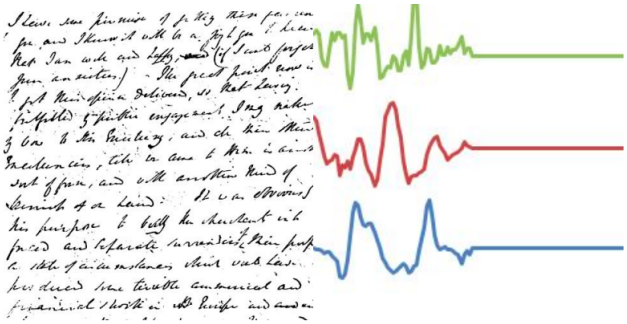


Figure 3.4.2: Multivariate time series example for handwriting extracted from [1]

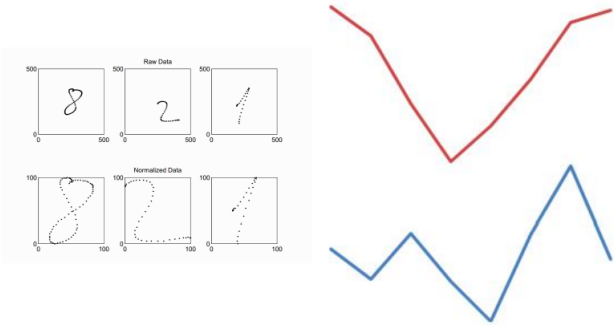


Figure 3.4.3: Multivariate time series example for pen digits extracted from [1]

3.5 Gesture recognition

These types of problems are focused on detecting a type of movement made by a person using a part of their body mostly by the hands.

3.5.1 Ering

This problem is about detecting hand and finger gestures [72]. The data is obtained using electric field sensing on the hand movement. Examples of these gestures are shown in figure 3.5.1

3.5.2 NATOPS

These problems classify distinct gestures [25] using the (x,y,z) trajectories of a person on different parts of their body. Some of these gestures are shown in figure

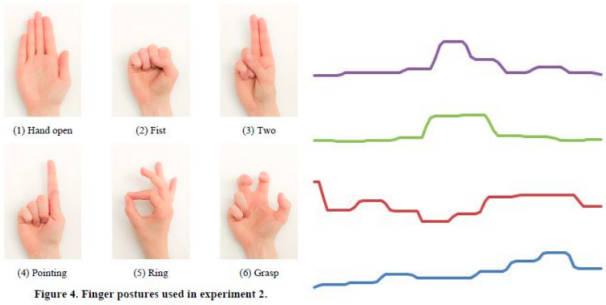


Figure 3.5.1: Multivariate time series example for e-ring extracted from [1]

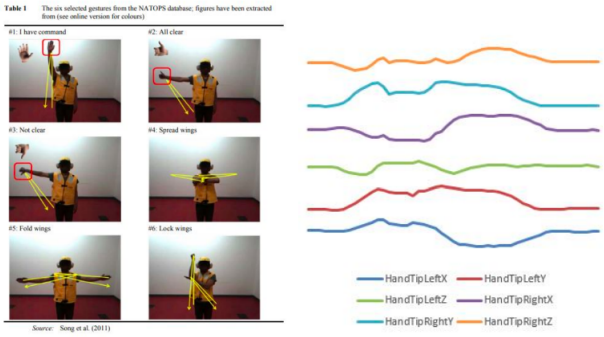


Figure 3.5.2: Multivariate time series example for NATOPS extracted from [1]

3.5.2

3.5.3 UWave gesture library

In this problem, 8 gestures taken from mobile devices are required to be learned [48]. Examples of these gestures can be seen in figure 3.5.3. This project uses the DTW algorithm to classify the gestures and discretize the data to reduce float calculations.

3.5.4 Libras

Again, this is another sign-detecting language problem that focuses on the Portuguese sign language [11]. Information is processed using video of the recording of the signals.

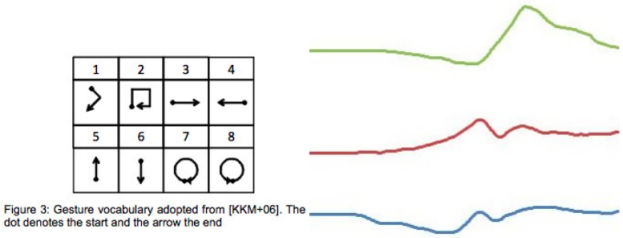


Figure 3.5.3: Multivariate time series example for uwave gesture library extracted from [1]

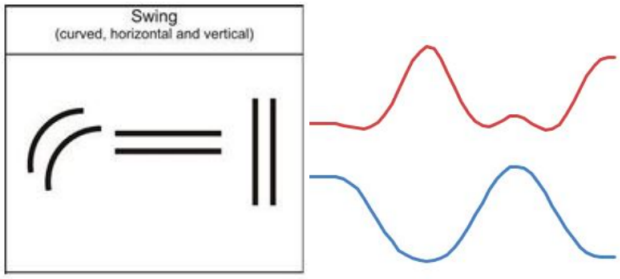


Figure 3.5.4: Multivariate time series example for libras extracted from [1]

3.6 Sound data

Recorded sound waves are considered time series problems. When there are several channels of recording the problem becomes multivariate.

3.6.1 Duck duck gees

This problem looks to predict a specific wild bird based on their sound [2]. The data is obtained from recording the sounds of birds and is processed to convert the sound wave as multivariate time series data.

3.6.2 Japanese vowels

This problem looks to classify which vowel is written by a person [39]. The data is obtained by processing the sound made.

[2]https://xeno-canto.org/

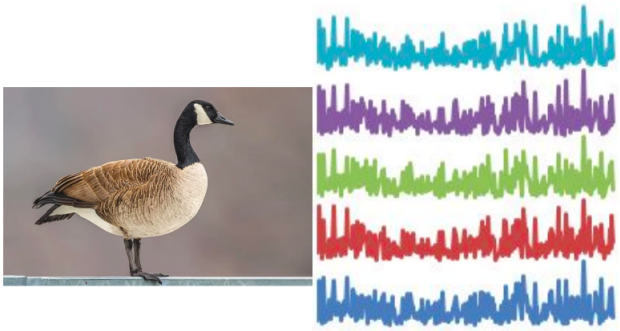


Figure 3.6.1: Multivariate time series example for duck duck geese extracted from [1]

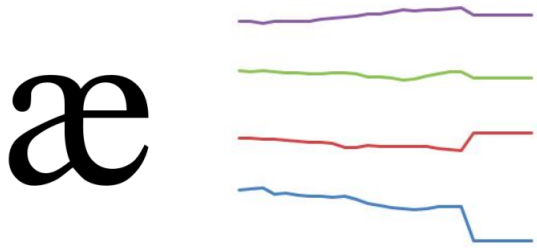


Figure 3.6.2: Multivariate time series example for Japanese vowels extracted from [1]

3.6.3 Insect wing beat

This problem looks to classify an insect type based on the sound recorded by passing through different sensors [76].

3.6.4 Phoneme spectra

Based on different recorded sounds, this problem looks to classify different phonemes [31].

3.6.5 Spoken Arabic digits

This is a sound recognition problem that looks to detect Arabic spoken digits [28]. The data is obtained by processing the sound waves from different speakers.

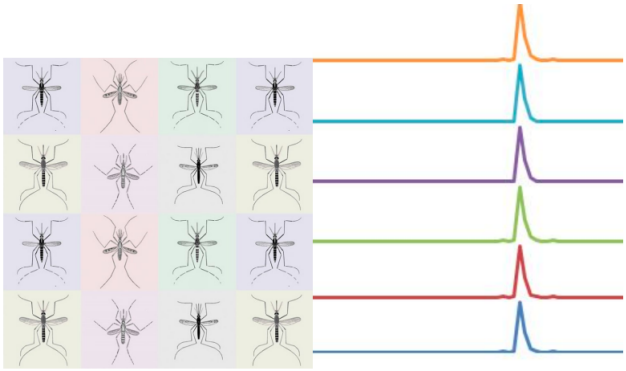


Figure 3.6.3: Multivariate time series example for insect wing beat extracted from [1]

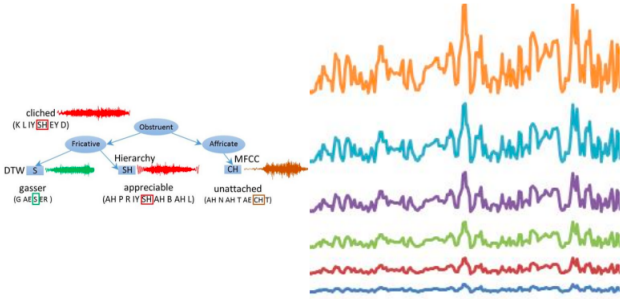


Figure 3.6.4: Multivariate time series example for phoneme spectra extracted from [1]

3.7 Other sensors

In addition to the mentioned categories, some problems use problem-specific sensors that cannot be categorized. This section gives a general overview of these problems and the sensors used.

3.7.1 Articulary word recognition

This problem consists of predicting a word based on the movement made by the tongue and lips when a person pronounces the word. The data was collected on sensors put in the tongue and lips of some participants [70] which can be seen in figure 3.7.1. In that work, a supported vector machine algorithm was used and the data was transformed using symbolic aggregation approximation to reduce the dimension of the problem.

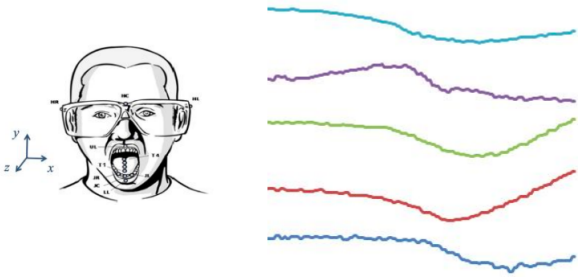


Figure 3.7.1: Multivariate time series example for articulary word recognition extracted from [1]

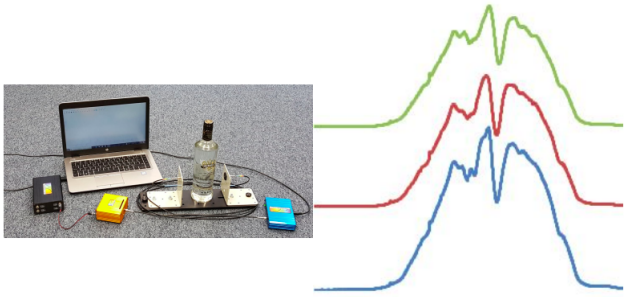


Figure 3.7.2: Multivariate time series example for ethanol concentration extracted from [1]

3.7.2 Ethanol concentration

The goal of this problem is to determine the alcohol concentration of a sample contained within an arbitrary bottle [41]. The data is obtained using sensors that detect the ethanol level of a specific bottle. These sensors can be seen in figure 3.7.2

3.7.3 Eigen worms

The goal of this problem is to detect the type of a worm based on information from its motion [18]. The data is obtained by recording the movements of the worms.

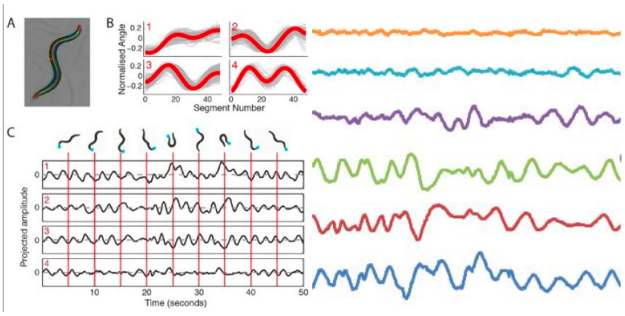


Figure 3.7.3: Multivariate time series example for eigen worms extracted from [1]

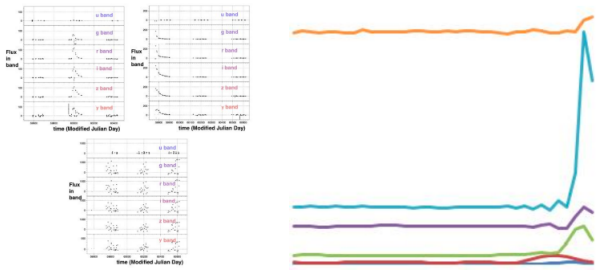


Figure 3.7.4: Multivariate time series example for LSST extracted from [1]

3.7.4 LSST

The project looks to classify astronomical simulated data to prepare when the data from the Large Synoptic Survey Telescope (LSST).

3.7.5 PEMS-SF

This problem is classified as the correct day of the week based on the information on the car occupancy lane rate in the San Francisco area [10]. An example of the data can be seen in figure 3.7.5

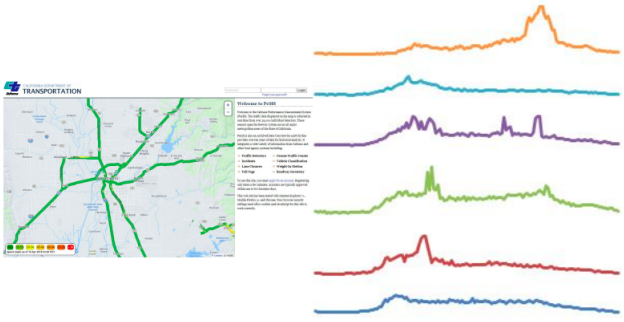


Figure 3.7.5: Multivariate time series example for PEMS-SF extracted from [1]

4

MTSC bake-off

Contributing publications

- Alejandro Pasos Ruiz, Michael Flynn, James Large, Matthew Middlehurst, and Anthony J. Bagnall. The great multivariate time series classification bake off: a review and experimental evaluation of recent algorithmic advances. Data Mining and Knowledge Discovery, 35:401 – 449, 2021. [62]

4.1 Introduction

In the previous chapter, a list of MTSC problems was introduced to have a standard way of testing MTSC algorithms. This chapter presents a benchmark experiment conducted in 2018 that includes the most relevant algorithms in the state of the art on MTSC.

The first contribution of this paper was to search for the main algorithms for MTSC. The second contribution is the experiments developed on this group of problems-algorithms. At the moment of the writing of the thesis, the univariate bakeoff was the only similar experiment but applied to a different archive of problems. The third contribution is to present a methodology that aims to allow to replicate of the results of the experiments followed by an analysis by a comparison of the results and an analysis by problem.

4.2 Methodology

This experiment uses 26 datasets with fixed series lengths. Each dataset is provided with a default split of train and test datasets. Table 4.1 shows the number of train and test instances for each of the datasets used.

Each experiment is limited to a maximum seven-day execution time and 500 GB of memory. Each dataset was resampled 30 seeded by sample index. Two

Table 4.1: Summary of the 26 datasets used in the benchmark.

	Name	Train size	Test size	Num Series	Series length	Classes
AWR	ArticularyWordRecognition	275	300	9	144	25
AF	AtrialFibrillation	15	15	2	640	3
BM	BasicMotions	40	40	6	100	4
CR	Cricket	108	72	6	1197	12
DDG	DuckDuckGeese	50	50	1345	270	5
EW	EigenWorms	128	131	6	17984	5
EP	Epilepsy	137	138	3	206	4
EC	EthanolConcentration	261	263	3	1751	4
ER	ERing	30	270	4	65	6
FD	FaceDetection	5890	3524	144	62	2
FM	FingerMovements	316	100	28	50	2
HMD	HandMovementDirection	160	74	10	400	4
HW	Handwriting	150	850	3	152	26
HB	Heartbeat	204	205	61	405	2
LIB	Libras	180	180	2	45	15
LSST	LSST	2459	2466	6	36	14
MI	MotorImagery	278	100	64	3000	2
NATO	NATOPS	180	180	24	51	6
PD	PenDigits	7494	3498	2	8	10
PEMS	PEMS-SF	267	173	963	144	7
PS	PhonemeSpectra	3315	3353	11	217	39
RS	RacketSports	151	152	6	30	4
SRS1	SelfRegulationSCP1	268	293	6	896	2
SRS2	SelfRegulationSCP2	200	180	7	1152	2
SWJ	StandWalkJump	12	15	4	2500	3
UW	UWaveGestureLibrary	120	320	3	315	8

toolkits were used: tsml [1] based on Java an aeon [2] based on Python. The list of which algorithm is implemented in a toolkit is presented on the table‘4.2. All algorithms presented in the background chapter were included. The default parameters were used on each algorithm. The list of the algorithms and their parameters are presented in table A.1. Some algorithms include as a parameter the option of normalizing or not the data. On that consideration, early results suggested that there is no improvement over normalizing data, therefore, the data were not normalized in any experiment. Some algorithms internally normalize the data. If this is the case we consider the normalization as part of the training

[1] https://github.com/time-series-machine-learning/tsml-java
[2] https://github.com/aeon-toolkit/aeon

Table 4.2: Classifier availability in the two toolkits tsml and aeon.

Algorithm	`tsml`	`aeon`
DTW_D	X	X
DTW_I	X	X
DTW_A	X	
MUSE	X	
gRFS		X
MrSEQL		X
ROCKET		X
CIF	X	X
TapNet		
ResNet		X
InceptionTime		X
CBOSS	X	X
STC	X	X
RISE	X	X
TSF	X	X
HIVE-COTE	X	X

process.

4.3 Evaluation

For evaluation, the critical difference (CD) diagrams [15] based on different metrics will be used as they are the standard approach to compare multiple problems on multiple algorithms.

CD diagrams consist of two parts: 1) algorithm rank and 2) statistical significance test. For the first step, CD calculates the algorithm performance on a specific problem based on the position based on a metric. These positions are averaged and given a score in the range $(1, m)$ where m is the number of classifiers, the lower the score the better. To verify if two classifiers have statistical significance,

the Wilcoxon-Signed test was used. In the diagram, there is a horizontal line for each group of algorithms that does not have statistical significance.

The metrics to be considered in this work are the most commonly used in machine learning literature:

- Accuracy: The proportion of correctly classified over the total test set
- Balanced accuracy: It is the same as the previous but adds a pondering the proportion of the class over the complete test set
- Area Under the Receiving Operating Characteristic (AUROC): It is a metric that aims to calculate the separation between classes.
- F1: This score aims to balance two other metrics: Precision (measures how many of the positive predictions were correct) and recall (measures how many of the positive class samples present in the dataset were correctly identified).

4.4 Results

We could not obtain results for all algorithms on all datasets within our constraints. DTW_A did not complete Eigenworms within the seven-day limit, and InceptionTime could not complete Eigenworms due to out-of-memory errors on the GPU. MrSEQL failed to finish FaceDetection and PhonemeSpectra in time. TapNet completed 23 datasets, but could not allocate enough memory for PhonemeSpectra, EigenWorms and MotorImagery. The bottleneck for MUSE is memory. It failed to complete six problems: DuckDuckGeese; EigenWorms; FaceDetection; MotorImagery; PEMS-SF; and PhonemeSpectra. We ran gRSF with default parameters on all datasets without problems. However, tuning with the recommended parameter ranges [34] proved infeasible. Only nine of the 26 experiments were completed in seven days.

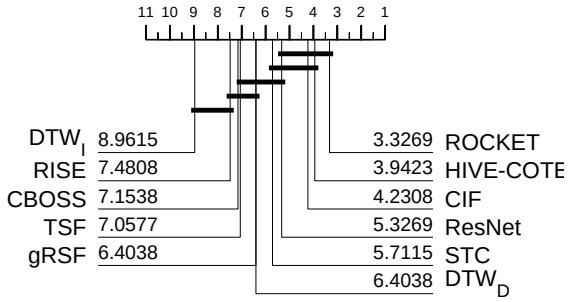
It is possible we could have engineered these algorithms and their parameter spaces to work on the problematic datasets. However, our goal is to test classifiers based on the configuration recommended by the original authors. We do not want to bias our results by optimizing algorithms for particular datasets. All 16 classifiers completed 20 problems, and 11 classifiers completed all 26 problems. We could have given the algorithms more than seven days to run for the missing problems. However, none of these problems are truly large by modern data standards (the biggest train file is 500MB), and a seven-day run with no external tuning seems a reasonable limit.

We selected DTW_D as the baseline classifier for the benchmark given the reliability of dynamic time warping applied to time series and the dependant version to consider the value of all dimensions in MTSC.

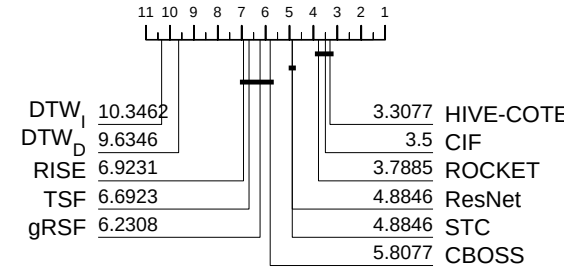
4.4.1 Comparison of Eleven Classifiers on Twenty-Six Datasets

Figure 4.4.1 shows the critical difference diagrams for the 11 classifiers that completed all 26 problems. The top clique is (ROCKET, HC1, CIF, ResNet) and the top three classifiers are all significantly more accurate than the baseline DTW_D. The middle cliques indicate that there is no significant difference between DTW_D and any of the other classifiers except DTW_I, which is significantly worse. Balanced accuracy and F1 give a very similar pattern of results, indicating class imbalance is not a factor. DTW_D and DTW_I cannot be judged by AUROC, since they do not provide probabilities with which to order the instances. AUROC demonstrates that the top three algorithms in terms of accuracy are significantly better than all others at ordering the data. We also compared all classifiers using a paired Student's t-test. For $\alpha = 0.05$, there would only be two different decisions: STC is not significantly worse than ROCKET with a t-test but is with a sign rank test, and CBOSS is significantly worse than STC with a sign rank test, but not with a t-test. Critical difference diagrams can sometimes mask differences between individual classifiers, because

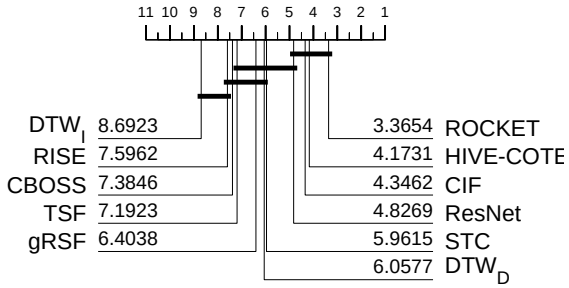
(a) Accuracy



(b) AUROC



(c) Balanced Accuracy



(d) F1

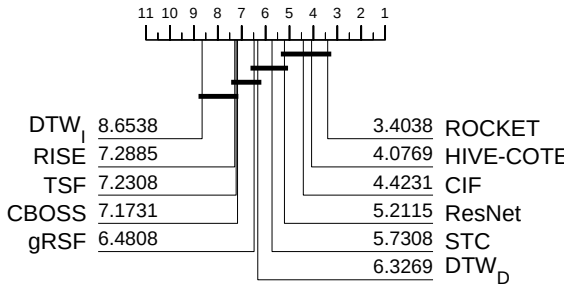


Figure 4.4.1: Critical difference diagrams for 11 classifiers on the 26 equal length UEA datasets using pairwise Wilcoxon test to form cliques.

of the nature of forming cliques. It is worthwhile, therefore, presenting p-values and summarising accuracy distributions. Table 4.3 presents the pairwise p-values for all 11 combinations, with the upper diagonal being sign rank and the lower diagonal the t-test. Note that in this table no adjustments for multiple testing have been made. The top clique using t-test would now include STC, but there are few practical differences.

	RCKT	HC	CIF	ResNet	STC	DTW_D	gRSF	TSF	CBOSS	RISE	DTW_I
RCKT	0.0000	**0.1742**	0.5506	0.0619	**0.0283**	0.0004	0.0004	0.0039	0.0006	0.0024	0.0000
HC	**0.9128**	0.0000	0.5338	0.1285	0.0585	0.0092	0.0023	0.0004	0.0000	0.0003	0.0000
CIF	0.6660	0.6402	0.0000	0.2692	0.0520	0.0043	0.0009	0.0006	0.0017	0.0001	0.0001
ResNet	0.0759	0.1246	0.1184	0.0000	0.9899	0.3282	0.5812	0.5506	0.2277	0.1500	0.0092
STC	**0.4282**	0.0621	0.0580	0.4512	0.0000	0.1068	0.1513	0.0578	**0.0074**	0.0020	0.0004
DTW_D	0.0005	0.0159	0.0166	0.2989	0.1630	0.0000	0.4091	0.8689	0.7509	0.6204	0.0022
gRSF	0.0003	0.0168	0.0041	0.7645	0.1532	0.5509	0.0000	0.5338	0.5338	0.1218	0.0010
TSF	0.0039	0.0044	0.0006	0.6544	0.0837	0.7847	0.7881	0.0000	0.6938	0.5338	0.0230
CBOSS	0.0008	0.0022	0.0037	0.3621	**0.0669**	0.9258	0.5020	0.7520	0.0000	0.3949	0.0054
RISE	0.0024	0.0008	0.0000	0.3258	0.0044	0.6638	0.1534	0.3549	0.4838	0.0000	0.1307
DTW_I	0.0000	0.0001	0.0001	0.0036	0.0009	0.0074	0.0028	0.0096	0.0042	0.1013	0.0000

Table 4.3: P-values for pairwise tests between classifiers. The upper diagonal values are found using the Wilcoxon sign-rank test. The lower diagonal is found using a paired t-test. So, for example, the p-value for STC vs CBOSS is 0.0074 using a sign rank test, but 0.0669 with a paired t-test. Classifiers are ordered by overall rank, so a p-value below the critical value for STC vs CBOSS indicates STC is significantly more accurate on these data.

The differences in accuracy between the complete classifiers and DTW_D are summarised in Figure 4.4.2. Here we can see some of the widespread of relative performances by classifiers over the datasets. STC has the widest distribution of difference in accuracies which explains the fact that STC has the biggest difference in test results between sign rank and t-test shown in Table 4.3.

Figure 4.4.3 demonstrates this further for the top clique of classifiers and shows scatter plots of test accuracies against the DTW_D baseline. ROCKET is better on 22 and worse on 3, with a mean difference of 5.9% and a median difference of 3.3%. HC1 is better on 17 and worse on 9, with a mean difference of 5.8% and a median difference of 3.28%. CIF is better on 19 and worse on 7 with a mean difference of 6.5% and a median difference of 3.18%. ResNet is better on 14 and

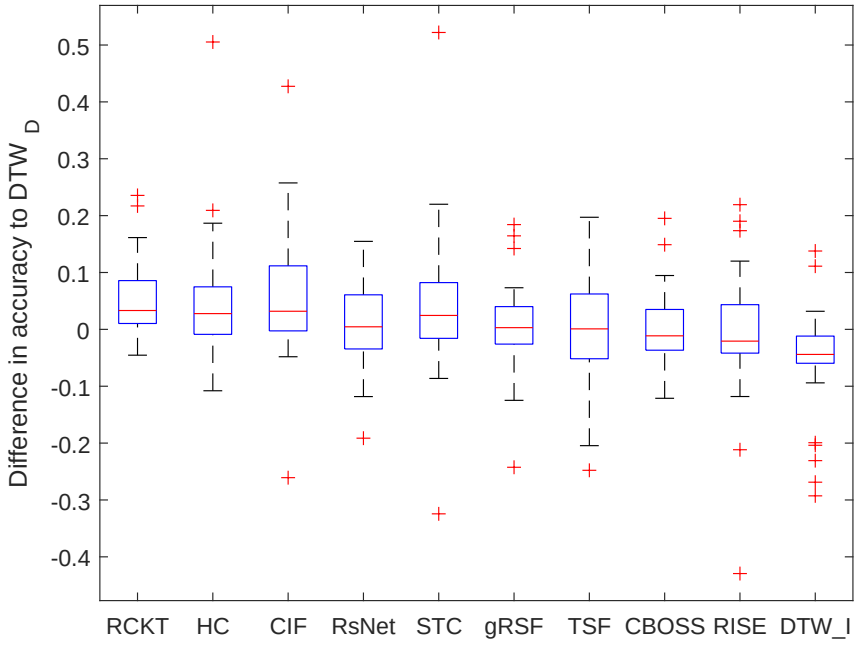


Figure 4.4.2: Box plots of the differences in accuracy relative to DTW_D over datasets.

worse on 12 with a mean difference of 1.7% and a median difference of 0.45%.

Table 4.4 gives the detailed results for the three classifiers significantly better than DTW_D, including the standard error over resamples. This table demonstrates that there will still be problems, such as HandWriting, where DTW_D is the best approach but that, lacking any extra information, the other algorithms will generally give significantly more accurate classifiers. While ROCKET and HC1 lose by a relatively smaller margin when DTW_D does outperform them, the HandWriting case shows that CIF has a much clearer gap in the types of problems it can effectively handle.

Run times are hard to compare because of both software and hardware differences. Nevertheless, to get some idea of the relative performance, we recorded run time for all experiments. Table 4.5 gives the summary run time information, and Figure 4.4.4 plots accuracy against runtime.

ROCKET lives up to its name: it is by far the fastest algorithm and remarkably

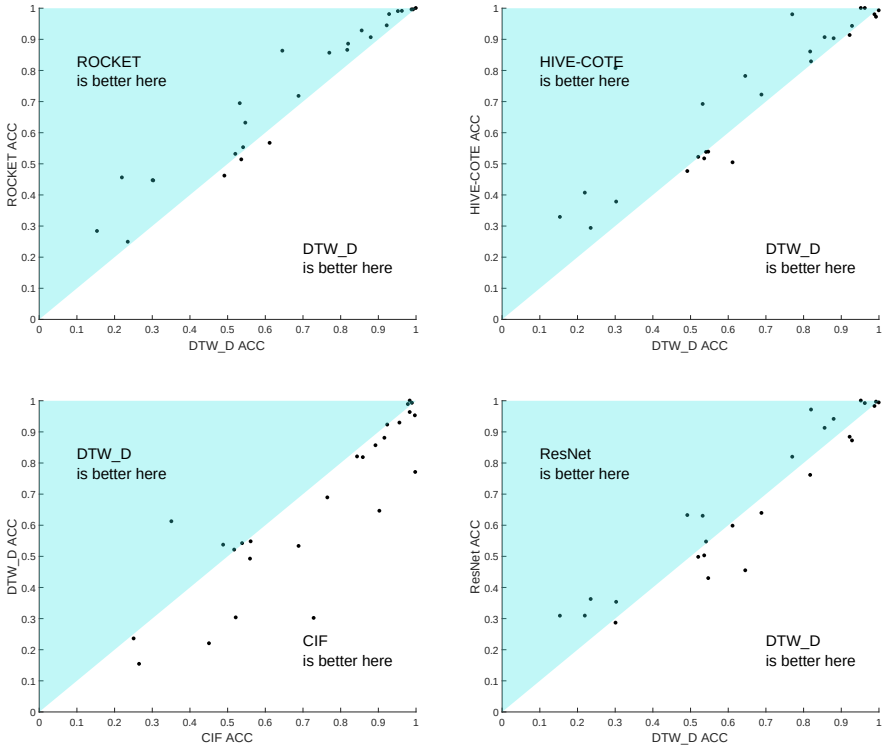


Figure 4.4.3: Scatter plots of the accuracy of ROCKET, HC1, CIF, and ResNet against DTW_D.

can build a model for all 26 data sets in just over an hour. If it is set to be threaded, it completes 30 resamples of the 26 problems in less than two hours. Given its accuracy, this seems strong evidence to support it as a new baseline. CIF is much slower, requiring about 6 days for all the problems, but it averages around 5 hours per problem. HC1 is by far the slowest and if run sequentially would take over a year to complete all the problems. Strictly speaking, it would violate our run time constraints if we ran it in this way. However, we included it here because we did not run it sequentially. We ran each component/dimension combination independently and in parallel. The nature of dimension-independent ensembles makes this much easier to do than with algorithms such as MUSE and TapNet. It is also noteworthy that STC is the slowest component, but that is due to our parameter choice. STC is contracted and defaults to 24 hours of compute time on each dimension. For high-dimensional problems, this would lead to huge run times if completed sequentially. However, it is hardly ever necessary to search for shapelets for 24 hours. Table 4.5 shows that, on average, STC is 4.06% more

accurate than DTW, but overall, it is not significantly better. This demonstrates that there are problems where a specific representation is much better.

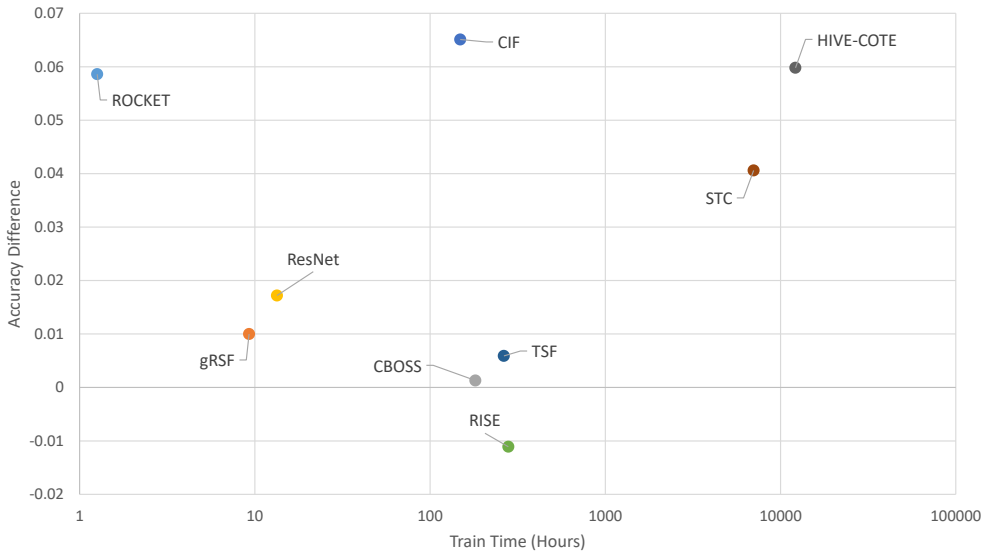


Figure 4.4.4: Average difference in accuracy to DTW_D vs train time for 9 MTSC algorithms.

Memory usage is even harder to determine, since we are concerned with the maximum memory used during a run, not just the final memory footprint of the classifier. We can record the maximum memory usage in `tsml`, but this capability is not yet in `aeon` and its variants. Table 4.6 shows the maximum and total memory usage of eight `tsml` classifiers. HC1 is the most memory-intensive classifier, but even HC1 requires at most 3.5 GB (MotorImagery) and just 21GB for all problems. Memory is not a significant constraint for these classifiers.

To summarise, only three of the ten classifiers able to complete all problems are significantly more accurate than the baseline DTW_D (ROCKET, CIF, and HC1). Figure 4.4.5 shows the relative performance of ROCKET against CIF and HC1. These figures show that ROCKET consistently beats the other two, but that when it fails, it tends to fail badly. This is demonstrated by the fact it is marginally worse on average than both when looking at the mean difference, but better when the median is considered. It is also highlighted with the p-values shown in Table 4.3. The non-parametric sign rank test p-values for ROCKET against CIF and HC1 are much lower than the parametric t-test

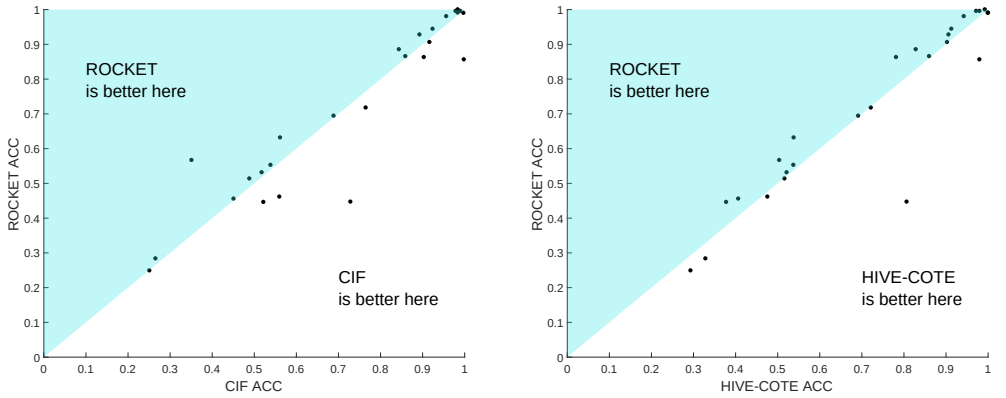


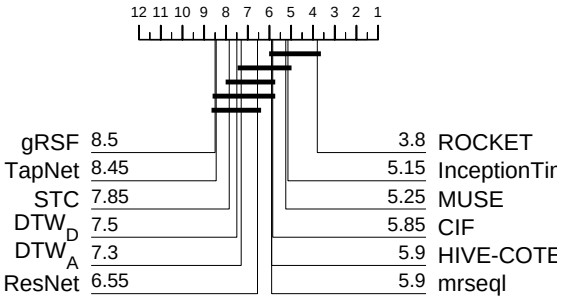
Figure 4.4.5: Scatter plots of accuracy on 26 UEA MTSC problems for ROCKET against CIF and HC1. ROCKET beats CIF on 17 problems, with mean and median differences in accuracy are -0.12% and 0.85%). ROCKET beats HC1 on 17 problems with mean and median differences in accuracy are -0.66% and 0.66%.

p-values. ROCKET performs at least as well as HC1 and CIF is by far the fastest and would be our recommended starting point for an investigation of a new MTSC problem. The evidence of the occasional large failure could help drive future design improvements.

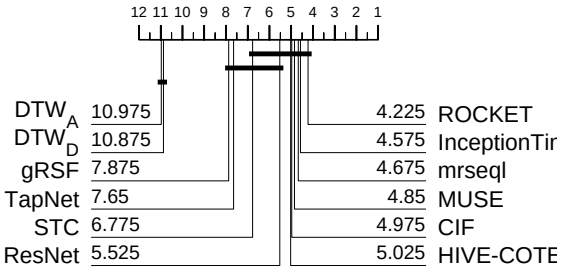
4.4.2 Comparison of Sixteen Classifiers on Twenty Datasets

DTW_A, MUSE, MrSEQL, TapNet, and InceptionTime did not complete all problems. Rather than a lengthy individual analysis, we present the results for the twenty problems that all algorithms completed. For clarity, we remove the four worst-performing classifiers (DTW_I, RISE, TSF and CBOSS). Figure 6.6.2 shows the critical difference diagrams for the top twelve classifiers on the twenty data sets that all algorithms completed within our constraints. MUSE is memory intensive. On these 20 problems, it required an average of 8 GB, compared to just 500 MB for HC1. Fewer datasets make it harder to detect significant differences. The top clique is now (ROCKET, InceptionTime, MUSE, CIF). However, these cliques do not reflect the differences to the baseline. With a critical value of $\alpha = 0.05$, only ROCKET and CIF are significantly better than DTW_D on these 20 problems. With 25 problems,

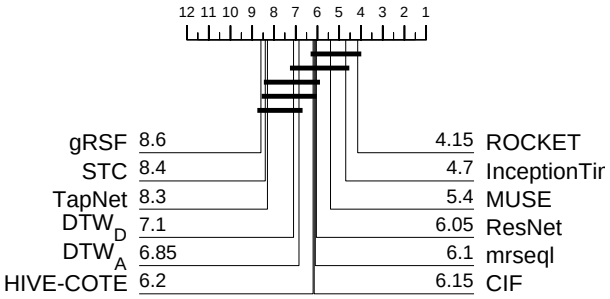
a) Accuracy



b) AUROC



c) Balanced Accuracy



d) F1

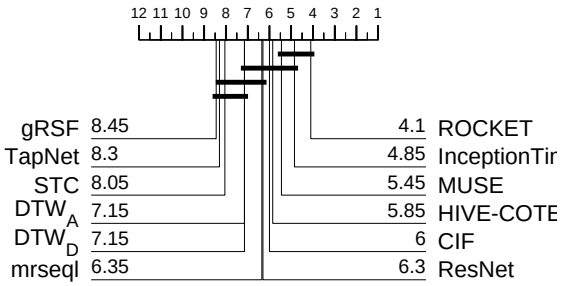


Figure 4.4.6: Critical difference diagrams for the top 12 classifiers on the 20 equal-length UEA datasets all algorithms completed.

InceptionTime is also significantly better than DTW_D, as is HC1 with 26. Table 4.7 gives the p-value for the pairwise test on the datasets completed by each algorithm.

MUSE does well, but is so memory intensive it will be hard to use for many problems. MrSEQL is also promising, although not significantly better than DTW. It is not clear why it failed to complete the two problems. InceptionTime, HC1, and CIF all beat the baseline and have potential. However, the stand-out classifier is still ROCKET. It has the lowest average rank, beats the baseline on most problems and it is incredibly quick, especially when parallelized. We think it is the clear winner of this experimental study.

	DTW_D	ROCKET	CIF	HC1
AWR	98.87%±0.05%	**99.56%±0.13%**	97.89%±0.15%	97.99%±0.10%
AF	23.56%±1.39%	24.89%±1.68%	25.11%±2.18%	**29.33%±1.31%**
BM	95.25%±0.23%	99.00%±0.00%	99.75%±0.14%	**100.0%±0.84%**
CR	**100.0%±0.00%**	**100.0%±0.13%**	98.38%±0.29%	99.26%±0.00%
DDG	49.20%±0.99%	46.13%±1.04%	**56.00%±1.03%**	47.60%±1.20%
EW	64.58%±0.53%	86.28%±1.21%	**90.33%±0.54%**	78.17%±0.62%
EP	96.30%±0.17%	99.08%±0.00%	98.38%±0.27%	**100.0%±0.26%**
EC	30.15%±0.54%	44.68%±0.43%	72.89%±0.56%	**80.68%±0.50%**
ER	92.91%±0.12%	**98.05%±0.49%**	95.65%±0.42%	94.26%±0.40%
FD	53.28%±0.23%	**69.42%±0.30%**	68.89%±0.27%	69.17%±0.14%
FM	54.17%±0.90%	**55.27%±0.84%**	53.90%±0.81%	53.77%±0.93%
HMD	30.32%±1.00%	44.59%±0.87%	**52.21%±1.08%**	37.79%±0.81%
HW	**61.21%±0.42%**	56.67%±0.42%	35.13%±0.40%	50.41%±0.42%
HB	68.88%±0.37%	71.76%±0.02%	**76.52%±0.30%**	72.18%±0.52%
LIB	88.04%±0.44%	90.61%±0.45%	**91.67%±0.49%**	90.28%±0.61%
LSST	54.76%±0.08%	**63.15%±0.16%**	56.17%±0.22%	53.84%±0.14%
MI	52.10%±0.73%	**53.13%±0.78%**	51.80%±1.03%	52.17%±0.74%
NATO	82.04%±0.32%	**88.54%±0.44%**	84.41%±0.32%	82.85%±0.32%
PD	99.28%±0.05%	**99.56%±0.14%**	98.97%±0.08%	97.19%±0.06%
PEMS	77.05%±0.58%	85.63%±0.38%	**99.85%±0.09%**	97.98%±0.59%
PS	15.39%±0.10%	28.35%±0.12%	26.56%±0.13%	**32.87%±0.07%**
RS	85.64%±0.26%	**92.79%±0.45%**	89.30%±0.51%	90.64%±0.37%
SRS1	81.81%±0.35%	**86.55%±0.31%**	85.94%±0.28%	86.02%±0.32%
SRS2	**53.69%±0.49%**	51.35%±0.59%	48.87%±0.56%	51.67%±0.67%
SWJ	22.00%±1.87%	**45.56%±2.72%**	45.11%±2.65%	40.67%±1.54%
UW	92.28%±0.21%	**94.43%±0.35%**	92.42%±0.32%	91.31%±0.23%

Table 4.4: Average accuracies with standard error over resamples for DTW_D and the three classifiers are significantly more accurate than DTW_D.

Classifier	Total time (hrs)	Difference in accuracy to DTW_D
ROCKET	1.26	5.86%
gRSF	9.27	1.0%
ResNet	13.38	1.72%
CIF	148.55	6.51%
CBOSS	181.60	0.13%
TSF	263.88	0.59%
RISE	279.64	-1.11%
STC	7019.69	4.06%
HC1	12172.44	5.98%

Table 4.5: Total run time for a single resample of all 26 problems and mean difference in accuracy to DTW_D for 9 classifiers.

Classifier	Max memory	Total memory
DTW_I	1883	5587
DTW_D	1845	5952
RISE	2624	10242
TSF	2670	10632
CBOSS	2675	10537
STC	2163	9778
CIF	2954	15900
HC1	3577	21217

Table 4.6: Memory usage (in MB) for eight `tsml` classifiers. Max memory is the maximum memory on any single problem, total memory is the aggregated memory over all twenty-six problems.

Algorithm	Completed data	P-value	W/D/L
MUSE	20	0.1005	13/0/7
TapNet	23	0.9015	10/0/13
MrSEQL	24	0.0593	16/0/8
DTW_A	25	0.6900	10/2/13
InceptionTime	25	0.0149	17/0/8
STC	26	0.1067	15/0/11
HC1	26	0.0043	17/0/9
CIF	26	0.0092	19/0/7
ROCKET	26	0.0004	22/1/3

Table 4.7: Performance was relative to the baseline classifier DTW_D. The P-value is from the Wilcoxon sign rank test.

Table 4.8: Accuracy of twelve algorithms averaged over thirty stratified resample data sets for the UEA MTSC archive. Default accuracy is for predicting the majority class.

Problem	Default	ROCKET	IT	MUSE	CIF	HC	mrseql	ResNet	DTW$_A$	STC	DTW$_D$	TapNet	gRSF	TSF	CBOSS	RISE	DTW$_I$
AWR	4.00%	99.56%	99.10%	98.87%	97.89%	97.99%	98.98%	98.26%	98.94%	97.51%	98.87%	97.13%	98.21%	94.82%	97.56%	95.73%	94.31%
AF	33.3%	24.89%	22.00%	74.00%	25.11%	29.33%	36.89%	36.22%	22.44%	31.78%	23.56%	30.22%	27.56%	29.78%	30.44%	24.44%	34.67%
BM	25.0%	99.00%	100.0%	100.0%	99.75%	100.0%	94.83%	100.0%	99.92%	97.92%	95.25%	99.17%	100.00%	99.83%	98.75%	100.0%	72.17%
CR	8.33%	100.0%	99.44%	99.77%	98.38%	99.26%	99.21%	99.40%	100.0%	98.94%	100.0%	97.50%	97.41%	93.15%	97.55%	97.78%	95.74%
DDG	20.0%	46.13%	63.47%		56.00%	47.60%	39.27%	63.20%	56.67%	43.47%	49.20%	58.27%	44.47%	38.87%	43.07%	50.80%	29.27%
EW	42.0%	86.28%			90.33%	78.17%	72.16%	45.45%		74.68%	64.58%		83.00%	76.62%	62.80%	81.93%	44.20%
EP	26.8%	99.08%	98.65%	99.64%	98.38%	100.0%	99.93%	99.18%	97.37%	98.74%	96.30%	96.09%	96.01%	97.34%	99.83%	99.86%	67.03%
EC	25.1%	44.68%	27.92%	48.64%	72.89%	80.68%	60.18%	28.62%	29.57%	82.36%	30.15%	28.99%	34.06%	45.42%	39.62%	49.16%	30.68%
ER	16.7%	98.05%	92.10%	96.89%	95.65%	94.26%	93.19%	87.19%	92.89%	84.28%	92.91%	89.46%	91.98%	89.84%	84.48%	82.44%	91.42%
FD	50.0%	69.42%	77.24%		68.89%	69.17%		62.97%	53.13%	69.76%	53.28%	52.87%	55.36%	68.95%	52.32%	51.17%	51.53%
FM	49.0%	55.27%	56.13%	54.77%	53.90%	53.77%	55.53%	54.70%	54.93%	53.40%	54.17%	51.33%	54.43%	53.17%	51.03%	52.10%	55.50%
HMD	18.9%	44.59%	42.39%	38.02%	52.21%	37.79%	35.23%	35.32%	30.72%	34.95%	30.32%	32.34%	32.07%	48.51%	28.87%	28.24%	26.67%
HW	3.8%	56.67%	65.74%	51.85%	35.13%	50.41%	54.04%	59.78%	60.55%	28.77%	61.21%	32.95%	36.96%	36.42%	49.09%	18.27%	34.33%
HB	72.2%	71.76%	73.20%	73.59%	76.52%	72.18%	72.52%	63.89%	68.07%	72.15%	68.88%	73.97%	74.89%	72.28%	72.15%	73.22%	63.80%
LIB	6.7%	90.61%	88.72%	90.30%	91.67%	90.28%	86.57%	94.11%	87.85%	84.46%	88.04%	83.63%	75.56%	79.72%	85.26%	81.67%	78.63%
LSST	31.5%	63.15%	33.97%	63.62%	56.17%	53.84%	60.28%	42.94%	56.96%	57.82%	54.76%	46.33%	58.15%	34.31%	43.62%	50.58%	49.57%
MI	50.0%	53.13%	51.17%		51.80%	52.17%	53.00%	49.77%	50.37%	50.83%	52.10%		51.87%	53.80%	52.37%	49.83%	49.63%
NATO	16.7%	88.54%	96.63%	87.13%	84.41%	82.85%	86.43%	97.11%	81.48%	84.35%	82.04%	90.30%	82.37%	77.72%	82.48%	80.59%	76.07%
PD	10.4%	99.56%	99.68%	98.68%	98.97%	97.19%	97.14%	99.64%	99.27%	97.70%	99.28%	93.65%	96.12%	94.11%	95.61%	87.47%	99.22%
PEMS	11.6%	85.63%	82.83%		99.85%	97.98%	97.15%	81.95%	78.73%	98.40%	77.05%	79.21%	91.27%	96.76%	96.57%	98.98%	80.23%
PS	2.6%	28.35%	36.74%		26.56%	32.87%		30.86%	15.39%	30.62%	15.39%		22.71%	14.52%	19.43%	26.78%	10.18%
RS	28.3%	92.79%	91.69%	89.56%	89.30%	90.64%	88.73%	91.23%	85.79%	88.09%	85.64%	85.81%	87.79%	88.29%	88.90%	84.17%	81.69%
SRS1	50.2%	86.55%	84.69%	73.58%	85.94%	86.02%	82.86%	76.11%	81.34%	84.73%	81.81%	95.68%	79.74%	84.73%	81.33%	73.17%	80.63%
SRS2	50.0%	51.35%	52.04%	49.52%	48.87%	51.67%	49.61%	50.24%	52.43%	51.63%	53.69%	53.46%	48.69%	50.65%	50.02%	50.28%	48.48%
SWJ	33.3%	45.56%	42.00%	34.67%	45.11%	40.67%	42.00%	30.89%	25.56%	44.00%	22.00%	35.11%	38.44%	33.33%	36.89%	34.00%	35.78%
UW	12.5%	94.43%	91.23%	90.39%	92.42%	91.31%	91.32%	88.35%	91.51%	87.03%	92.28%	88.39%	89.59%	85.05%	86.13%	71.11%	87.58%

4.5 HIVE-COTE 2 in MTSC

One of the top-tier algorithms from the previous work was HC1. In [53], HC2 was presented and executed the same experiments. In n figure 4.5.1 we show a critical difference diagram presented in that work. [53] proved that HC2 is statistically better than the top-tier algorithms and can be considered the most accurate in the UEA archive. We include this work as it is relevant to the bake-off benchmark but it was not included because the algorithm was not developed at the time of the experiments.

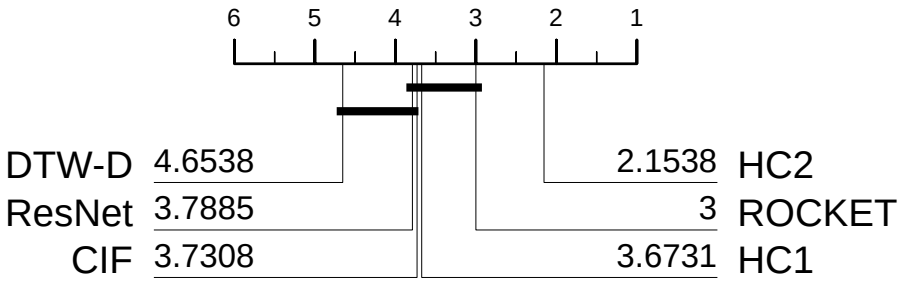


Figure 4.5.1: Critical difference diagrams for 26 equal-length UEA datasets all algorithms compared with Hive Cote 2.

4.6 Conclusion

MTSC is a fast-moving field and new approaches are released in a short time. From the time this thesis is being written, HC2 is the most accurate in the UEA archive. However, the results also show that, for many problems, there is a lot of improvement that can be made. Some of the improvements can be based on improving the performance (train time) or improving the classifiers used in the ensemble. In the remainder of the thesis, we will explore these improvements in

the following way:

- Improving STC: From HC1 to HC2 the only algorithm that remained is the Shapelet Transform Classifier. We are going to present improvement variants for this algorithm.
- Dimension Selection: One way to improve the performance is to reduce the dimensions of the multivariate classifier. We will research current techniques for dimension selection on MTSC and use them as filters for HC2 to verify if there are improvements in performance and accuracy,

5

Multivariate Shapelet Classifiers

5.1 Introduction

The shapelet transform classifier (STC) is one component of HC2 and the one that remains from HC1. In chapter 2.3.3 we introduced the shapelet transform classifier algorithm. In this chapter, we explore novel techniques to improve the shapelet transform classifier algorithm and use this approach to improve HC2.

The idea of using different shapelet quality criteria was first explored on [44] where the Krustal-Wallis and Moods Median nonparametric tests were considered as alternatives to information gain to a subset of univariate time series problems. This chapter presents different quality criteria and is applied to multivariate time series classification on dependent or independent shapelets.

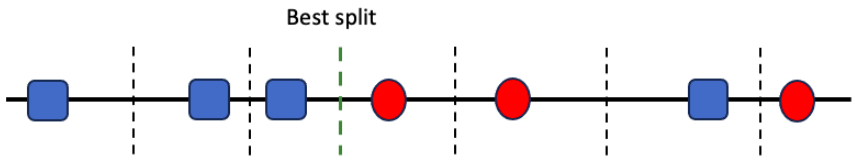
The first contribution of this chapter is to introduce RSTC which is a modification of STC that adds some new features. For example, improved random search of shapelets, and new stop criteria for shapelet search, among others. The second contribution is an experiment RSTC using different quality criteria in combination with dependent/independent shapelets. The third contribution is to show that an ensemble RSTC with different quality criteria STC can improve the original STC algorithm without increasing the train time. The fourth contribution is to show that HC2 improves its performance by replacing STC with the ensemble of RSTC.

5.2 Quality criteria

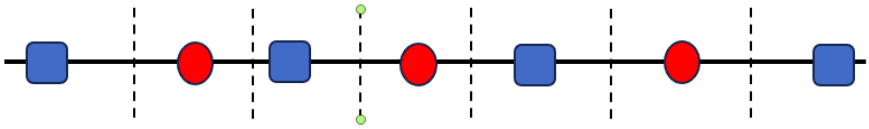
The quality criteria is an important aspect of STC because it ranks the shapelets that are candidates to be used as feature vectors. The process of the quality criteria is as follows:

1. Calculate the distance from the shapelet candidate to every series in the dataset. This generates a list of n values which is called an orderline: a pair

Figure 5.2.1: Orderline example



(a) Example of an orderline produced by a high-quality shapelet



(b) Example of an orderline produced by a low-quality shapelet

(d, c) where d is the distance and c is the class label

2. Sort the list in ascending order based on distance

3. Calculate all the split points on the orderline list using quality criteria. This can be seen in figure 5.2.1

4. Select the best-split point and use this as the final shapelet quality value

An ideal shapelet should produce small distance values when compared to the time series of the same class and large distance values with other classes. In figure 5.2.1a, we can see a good shapelet that will have a good score because there is a split point where most examples from the same class are grouped. On the contrary in figure 5.2.1b we see that the shapelet cannot have a good split point as in all the options there is a mix of elements for all classes

In the next section, we will detail all shapelet quality functions considered.

5.2.1 Information Gain

This is the default quality measure and it has been widely used in several machine learning problems such as in decision trees to evaluate a split point [34]. In the

scenario, when the decision tree has a numerical feature IG is used to select the optimum split point that generates the branches in the sub-tree. IG is based on entropy which is a measure of the uncertainty associated with a random variable. The entropy is calculated by the following formula:

$$H(S) = \sum_{i} -p_i \log_2 p_i \tag{5.2.1}$$

To measure the information gain of a split point, it is required to calculate the entropy of the data and reduce the weighted entropy of the possible values. For all possible split points a we select the minimum to calculate the information gain

$$IG(S) = H(S) – min_a \sum_{v \in a} \frac{S_a(v)}{S} \cdot H(S_a(v)) \tag{5.2.2}$$

5.2.2 Chi-squared (CHI)

This is a statistical hypothesis test to calculate the differences between expected and observed frequencies in random variables. It has also been used on decision trees to calculate the quality of a split point, making it a good candidate to be considered as an alternative quality measure. The formula is:

$$\chi^2 = \sum_{i}^{r} \sum_{j}^{c} \frac{(o_{ij} - e_{ij})^2}{e_{ij}} \tag{5.2.3}$$

Where e_{ij} is the expected number of instances that fall in that category if the split point follows the same distribution of instances as the attribute. Therefore, o_{ij} is the real observed value and this formula will assign higher values if the observed values are too different from the expected values. In this case, the χ^2 is calculated for several split points, and the highest one is chosen.

5.2.3 Pearson correlation (COR)

This is a measure of the linear correlation between two vectors, in this case, two feature vectors are generated. The formula is:

$$PC(x,y) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{5.2.4}$$

5.2.4 OneR (ONER)

This is a naive classifier that has been used as the base for weak ensembles of trees. It creates a simple 1-level decision tree. This is done by classifying the input provided and observing the accuracy of the model generated as a quality measure.

5.2.5 F-Stat

Evaluate the variance of two different sets of data to define if they are drawn from the same statistical distribution. It assumes that the variance of the class values and the distances are similar the shapelet can be a good discriminator.

5.2.6 Symmetrical uncertainty (SYM)

Evaluates the quality of an attribute by measuring the symmetrical uncertainty concerning the class. It is based also on entropy and is considered as a variation of information gain

$$H(Class, Attribute) = 2 \cdot \frac{H(Class) - H(Class|Attribute)}{H(Class) + H(Attribute)} \tag{5.2.5}$$

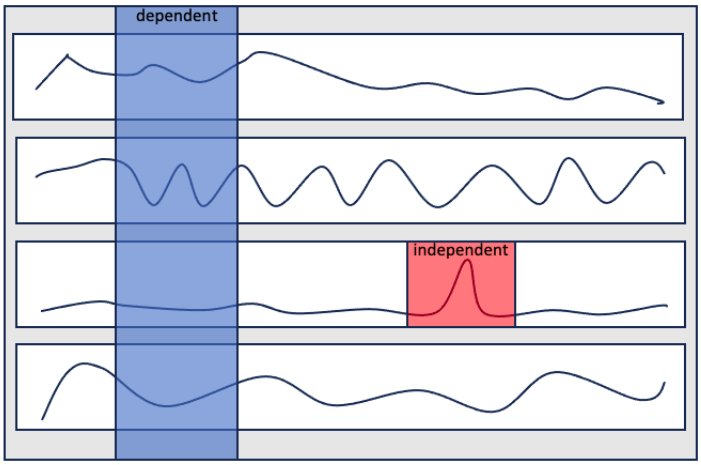


Figure 5.3.1: Example of a multivariate time series of 4 dimensions. A dependent shapelet (blue) covers a subset of all dimensions whereas an independent (red) covers only one of them.

5.3 Independent / Dependent Shapelets

There are two main strategies for using shapelets in MTSC problems: Independent ($MSTC_I$) and Dependent ($MSTC_D$). On independent, we consider a shapelet a 1-dimensional vector as with univariate time series. The difference is that a shapelet is associated with a specific dimension. Therefore, for independent shapelets, it is necessary to add a property to the shapelet to specify which dimension the shapelet belongs to compare it with other examples in the *sDist* function.

On dependent, we consider that a shapelet covers all dimensions. Therefore, the shapelet data is a $n \times m$ data where n is the shapelet size and m is the number of dimensions. In this case, the *sDist* function needs to be updated to compare against all dimensions.

5.4 Experiment settings

To compare different shapelet variants a first experiment was performed. It ran 30 resamples of the 26 multivariate datasets. In the results, the original

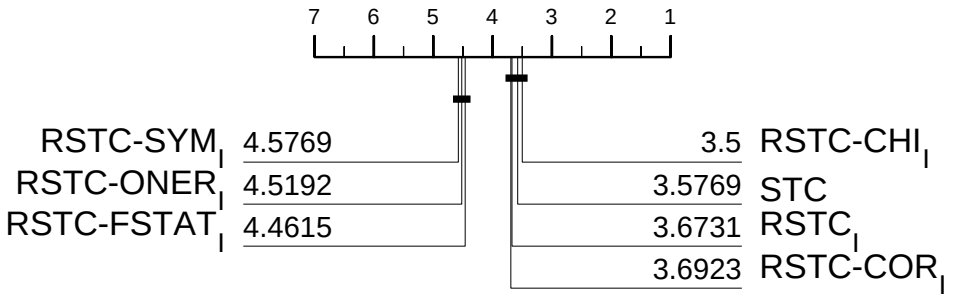


Figure 5.5.1: Comparing variants of shapelet quality and shapelet dependent with STC and RSTC Ensemble on 26 MTSC problems.

algorithm is named STC. Our approaches will be named by the acronym RSTC-X$_y$ where X is the quality metric and y is either Dependent or Independent. For example, RSTC-SYM$_I$ represents the shapelet transform classifier using the quality criteria symmetrical uncertainty and using Independent shapelets. If no quality criteria are specified the original information gain criteria are used. Our goal is to improve the state-of-the-art shapelet transform classifier (STC) with one or any combination of RSTC

5.5 Results

First, it was important to analyze which of the different combinations works better. Second, we explore a strategy to combine these criteria to create a more robust algorithm. Finally, verify if one of these approaches improves STC to plug it as part of HC2.

5.5.1 Shapelet quality variants

In the first experiment, all different quality combinations are compared. In this case, all dependent versions did not perform well. The main reason is that the high number of dimensions makes the comparison of shapelets a time-consuming operation producing a poor performance. To have simpler figures we did not add the dependent results to the results. The results are shown in figure 5.5.1. The

Algorithm	Hours
$RSTC - CHI_I$	345.82
$RSTC_I$	330.50
$RSTC - COR_I$	345.38
STC	7019.68
$RSTC_D$	501.61
$RSTCE_Q$	1528.01

Table 5.1: The number of hours to complete all 30 resamples on 26 MTSC problems.

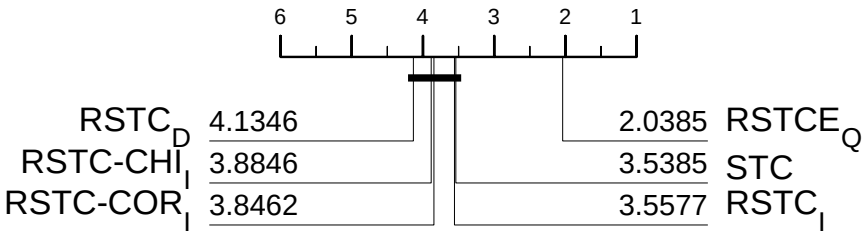


Figure 5.5.2: Comparing variants of shapelet quality and shapelet dependent with STC and RSTC Ensemble on 26 MTSC problems.

quality criteria χ^2, Pearson correlation, and information gain are significantly better than other approaches but they are not better than the state-of-the-art STC. We will consider only these quality measures in the following experiments.

5.5.2 Ensemble shapelets using different quality measures

In this section, we introduce a random shapelet transform classifier ensemble (RSTCE) of shapelet qualities and compare it with. In this version, we consider an ensemble of four elements: The independent versions of the quality measures on the top tier (1. information gain, 2.χ^2, 3. Pearson). The other element of the ensemble and the information gain with dependent shapelets. As we are adding the variations that performed better on the test results from the previous step the ensemble generated can have some bias. In future work, we can consider a variant selection phase on the training phase to remove this bias.

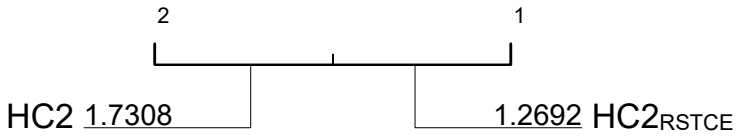


Figure 5.5.3: Comparing HC2 results with STC component against using RSTC Ensemble on 26 MTSC problems.

The predictions of the ensemble are done by CAWPE (the same strategy as HC). The results can be seen in figure 5.5.2. We can observe that the ensemble is significantly better than any of its components and the state-of-the-art STC.

In terms of performance, we will use train time as a metric to compare with STC. In table 5.1 we can observe that the time required to complete the ensemble is significantly less than the original STC. One of the reasons for this is the stop criteria used in RSTC. For small problems, usually, the max iterations are reached in a few seconds, and for large problems, the no improvement conditions help to detect when the algorithm stalls and stops improving which also avoids overfitting.

5.5.3 Adding to HC2

As a final experiment we compare the state-of-the-art HC2 but using the RSTCE ensemble instead of STC. The results are shown in figure 5.5.3 and show that the ensemble improves significantly the accuracy of HC2 and also improves the performance and reduces the train time.

5.5.4 Compared with univariate TSC

Given the success of improvement HC2 for MTSC it would be interesting to repeat the procedure for the univariate time series classification problems. We ran 30 resamples of the 112 problems from the univariate UEA archive. Of course,

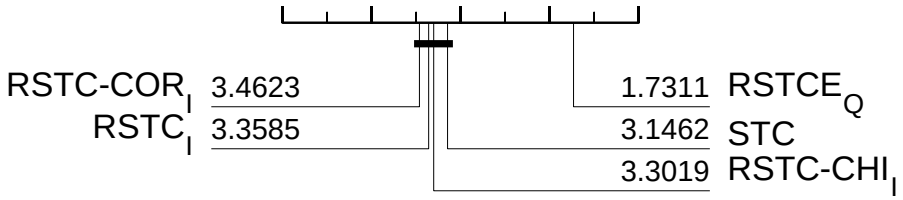


Figure 5.5.4: Comparing variants of shapelet quality and shapelet dependent with STC and RSTC Ensemble on 112 TSC problems.

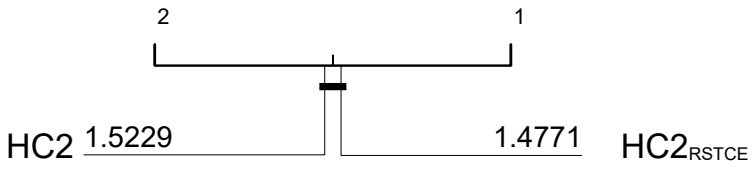


Figure 5.5.5: Comparing HC2 results with STC component against using RSTC Ensemble on 112 TSC problems.

one change must be made to do this, as the previous ensemble contains $RSTC_D$ which is based on using several dimensions and TSC always has one. Therefore, this component was dropped and instead, we added STC as the fourth element of the ensemble. The results in figure 5.5.4 show that the ensemble is significantly better than the individual results showing similar results as in the multivariate case.

Finally, as in multivariate, we added the ensemble as the basis for HC2 instead of STC. The results in figure 5.5.5 show that even though the ensemble is significantly better, it does not improve HC2 like in the multivariate case. This can be explained in several ways but the main reason is that univariate time series problems have less search space to find making the original STC good enough to find the best quality shapelets.

5.6 Conclusions

In this chapter, we developed a variation of the shapelet transform classifier for MTSC by modifying shapelet quality measures for dependent and independent shapelets. Then, we created an ensemble of the best qualities and this strategy proved to be significantly better than STC. Moreover, when used as a component of HC2 improved significantly the results and reduced the train time on multivariate time series classification. This also illustrates the power of using ensembles for machine learning by combining individual strategies that did not perform well and make a significant improvement when combined.

As the individual classifiers could not improve significantly STC, there is still room for improvement on shapelet classifiers based on quality. For example, the shapelet-dependent classifiers always underperform the independent versions. This suggests that not all dimensions are important for MTSC problems. It could be the case that filtering not only important dimensions can improve the accuracy of the results on the dependent algorithms but also the independent ones by reducing the feature space and focusing on the important dimensions. This approach will be explored in more detail in the next chapter.

6

Dimension Selection Strategies

Contributing publications

- Alejandro Pasos Ruiz and Anthony J. Bagnall. Dimension selection strategies for multivariate time series classification with hive-cote v2.0. volume 13812 of Lecture Notes in Computer Science, pages 133–147. Springer, 2022. [59]

6.1 Introduction

The second approach to improve HC2 consists of selecting the most important dimensions as a preprocessing strategy to reduce the training time and possibly improve accuracy. This could be possible in theory given the fact that many algorithms like STC involve a search phase for patterns. Reducing the search space can optimize this process.

Standard approaches include employing a filter to select a subset of attributes or transforming the data into a lower dimensional feature space using, for example, principal component analysis. Our focus is on dimensionality reduction through filtering.

Dimension selection can, on average, either increase, not change, or decrease the accuracy of classification. The first case implies that the higher dimensionality is confounding the classifier's discriminatory power. In the second case, it is often still desirable to filter due to improved training time. In the third case, filtering may still be desirable, depending on the trade-off between performance (e.g. accuracy) and efficiency (e.g. train time): a small reduction in accuracy may be acceptable if build time reduces by an order of magnitude.

In this chapter, we address the problem of how to select a subset of dimensions for high-dimensional problems. The first contribution of this chapter is to propose a method based on dimension ranking by criteria and selecting through the elbow method. We use existing methods like ECP, ECS, and MTSC. The

second contribution is to use the proposed method and the existing approaches as a preprocessing step on HC2. Finally, the third contribution is to select a subset of high-dimensional problems from the UEA archive and introduce four new problems.

6.2 Dimension selection problem

Detecting the best subset of dimensions is not a straightforward problem, since the number of combinations to consider increases exponentially with the number of dimensions. Selection is also made more complex by the fact that the objective function used to assess a set of features may not generalize well to unseen data. Furthermore, since the primary reason for filtering the dimensions is improving the efficiency of the classifier, dimension selection strategies themselves need to be fast. HC2 is not as fast as ROCKET. We investigate whether we can use the speed and competitive accuracy of ROCKET to serve as a dimension filter for HC2. We use a stripped-back version of ROCKET called miniROCKET to assess and select dimensions through cross-validation, then measure the impact this has on HC2 on both the train and the test data. We compare the miniROCKET filter to recently proposed algorithms from the literature. Our contribution is to incrementally improve our understanding of how best to classify high dimensional time series: we introduce four new high dimensional MTSC problems to the UEA archive; we propose a hybrid approach for classifying high dimensional MTSC problems using miniROCKET as a filter; and we compare our approach to a range of alternative algorithms and analyze the results.

6.3 Related work

The first algorithm to consider is the Common principal component loading-based variable subset selection (CleVer) [77]. Clever is based on Principal Component Analysis (PCA) which is a technique used in machine

learning as feature extraction and to find similarities among features. As PCA works on the feature vector representation, it cannot be used directly on time series. The first step is to execute PCA independently on each instance and extract the PCA. Next, all components that belong to the same class are combined to create common principal components through a process called Common Principal Component Analysis (cPCA). A proportion of the common components are used to create a feature space. These features are clustered, and the closest dimension to each centroid is chosen as the selected dimension. Clever requires a separate PCA on each series, which is both time and space-consuming. It is also complex, and we cannot find an open-source implementation. Since we are looking for a lightweight feature selector, we do not evaluate CleVer in this study.

Another approach was based on one nearest neighbor classification in combination with dynamic time warping (1-NN DTW) [35]. This approach used a merit score function (MSTS) to assess the quality of a subset of dimensions. This is a formula that balances the quality of the prediction of each dimension to the class and the similarity of each dimension. The first part is achieved by using 1-NN with DTW get the accuracy of prediction between a dimension with the classes. The similarity between each dimension is estimated using the adjusted mutual information (AMI) which is based on information gain. Once this is done, the merit score function is calculated for each possible subset of variables and selecting the top 5% of subsets. The merit score function is calculated as follows:

$$MS(subset) = \frac{k\overline{DC}}{\sqrt{k + k(k-1)\overline{DD}}} \tag{6.3.1}$$

Where $\overline{DC}$ is the average of dimension-to-class of each dimension in the subset and $\overline{DD}$ is the average of dimension-to-dimension of each pair of dimensions in the subset. The evaluation of all dimension combinations makes MSTS infeasible

Algorithm 8 MSTS($\boldsymbol{X}, y, |X|$)

Parameters: Training data $\boldsymbol{X}$, labels y, the number of dimensions $|X|$

for $i \leftarrow 1$ to $|X|$ **do**
  $\hat{y_i} \leftarrow$ CrossValidate($\boldsymbol{X_i}$, $classifier : DTW$, $folds : 3$)
  $\boldsymbol{DC_i} \leftarrow accuracy(\hat{y_i}, y_i)$
for (i, j) in $pairs(|X|)$ **do**
  $\boldsymbol{DD_{i,j}} \leftarrow AMI(\hat{y_i}, \hat{y_j})$
$bestSubset \leftarrow \emptyset$
$bestScore \leftarrow -\infty$
for each $subset \subseteq |X|$ **do**
  $subsetScore \leftarrow MS(subset)$
  if $subsetScore > bestScore$ **then**
    $bestSubset \leftarrow subset$
    $bestScore \leftarrow subsetScore$
return $bestSubset$

for very high dimensional problems.

Another approach that is more closely aligned to our work is described in [17], where dimensions are selected based on distances between series within classes. A synthetic series that characterizes each dimension/class combination is found by averaging the relevant dimension of the series belonging to that class. A matrix of the pairwise Euclidean distance between all dimension/class centroids is then found. These algorithms use a Distance Matrix (size $d \times c \cdot (c-1)$) which is calculated as follows.

Algorithm 9 CalculateDistanceMatrix($\boldsymbol{X}, y, dimensions, classes$)

Parameters: Training data $\boldsymbol{X}$, labels y, the number of dimensions and classes

for c in $classes$ **do**
  for d in $dimension$ **do**
    Calculate $Centroid(d, c)$
for each $pair(c_a, c_b)$ in $classes$ **do**
  for d in $dimensions$ **do**
    $DistanceMatrix(d, [c_a, c_b]) \leftarrow dist(\text{Centroid(d,c}_a), Centroid(d, c_b))$
return $DistanceMatrix$

After calculating the matrix, three approaches are introduced:

1. The **KMeans** approach applies k-means clustering (with $k = 2$) on the distance matrix to separate the channels. The cluster centroid represents the mean distance of dimensions across all class pairs and the average of the

centroid describes the within-cluster variation of dimensions. The K-means algorithm selects all dimensions in the cluster with the largest average.

2. The **Elbow Class Sum (ECS)** algorithm sums each row of the distance matrix, and then uses the elbow cut method to select dimensions based on the rank order of the sums.

3. The **Elbow Class Pairwise (ECP)** iterates through every class pair, and selects the best set of dimensions for that pair using the same elbow cut method as ECS. Finally, it takes the union of dimensions over all pairs.

6.4 Proposed method

We propose a range of methods for dimension selection, including adaptations of the algorithms described in Section 6.3, to make HC2 more efficient.

Our classifier pipeline involves dimension selection followed by the HC2 classifier. We want to evaluate the effect of changing the dimension selection mechanism whilst keeping everything else the same. Dimension selection is either through scoring and ranking then selection or dimension subset evaluation.

Our first filtering approach is to employ miniROCKET as a mechanism for scoring features from the training data, then using the elbow method to select features. This involves scoring a miniROCKET classifier on each dimension independently, then ranking dimensions. We consider three scores all based on miniROCKET predictions found through three-fold cross-validation:

- Accuracy (A): proportion of cases correctly classified.

- Silhouette (S): As an alternative to using accuracy the silhouette method is used in clustering to determine the optimal numbers of clusters. It is a score that goes from -1 to 1 indicating how good is the clustering based on the distances within a cluster and their differences from the points from other

clusters. To use in dimension selection, the train data is cross-validated with 3 folds, and the predictions are used as clusters. The formula for the silhouette method is:

$$S = \frac{(b - a)}{max(a, b)} \tag{6.4.1}$$

where a is the mean distance between data points in the same cluster and b is the mean distance between all other data points of the next nearest cluster.

- Adjusted Mutual information (M). It is a variation of mutual information that adds an element of chance, usually used in clustering. The formula is:

$$AMI(U, V) = \frac{MI(U, V) - E(MI(U, V))}{avg(H(U), H(V)) - E(MI(U, V))} \tag{6.4.2}$$

We also consider using both ECP and ECS as a filtering algorithm for HC2. As another cluster variant, we propose that instead of calculating the centroid distances as with ECP and ECS, we calculate the distance between each instance and the centroid which is calculated as the mean vector of all instances that belong to that class. This method is called CLUSTER. Finally, we also evaluate using the MSTS subset selection algorithm, although we make two changes to the version described in 6.3.

- We use miniROCKET instead of DTW as classifier on line 2 of Algorithm 9;
- The exhaustive subset selection is done on lines 8-12 of Algorithm 9 is infeasible for some problems because there are 2^d possible subsets of attributes. Instead, a forward selection procedure is used where the best k subsets starting with size two are selected and one dimension is added per step until the merit score function MSTS stops improving.

Table 6.1 summarises the attribute selection methods used in our evaluations.

Table 6.1: Summary of different dimension selection ranking methods with elbow method.

Attribute ranking then selection with elbow method	
Algorithm	Ranking
ECS	Sum of difference between centroid pair distance [17]
ECP	Union of sum of individual centroid pair distances [17]
CLUSTER	Error between centroid and examples
ROCKET$_A$	Accuracy of miniROCKET predictions on each dimension
ROCKET$_S$	Sillouette of miniROCKET predictions on each dimension
ROCKET$_M$	AMI of miniROCKET predictions on each dimension
Attribute subset selection	
MSTS	Subset selection using merit score [35]
KMeans	Cluster distance function [17]

6.5 Evaluation

We use the time series machine learning toolkit Aeon for our experiments. All of the algorithms used have been implemented in aeon format and are available at the GitHub repository associated with this page[1].

6.5.1 Data

The UEA multivariate time series repository contains 30 datasets from a wide range of fields such as EEG classification and human activity recognition [62]. In our experience, filtering does not improve the performance of HC2, so our priority is improving efficiency. Low-dimensional data can mask the performance differences of filtering algorithms, so we restrict our attention to higher dimensional problems, which we define as nine or more dimensions. Ideally, we would set the threshold even higher, but there are just nine equal-length problems in the archive with nine or more dimensions. Two high-dimensional data are unequal length: JapaneseVowels and SpokenArabicDigits. We made these equal lengths by padding to the longest

[1]https://github.com/aeon-toolkit/aeon

Table 6.2: Summary of 15 data sets used in experimentation. (*) indicates a padded series, and bold indicates a data set new to the UEA archive.

Name	Train size	Test size	Dimensions	Length	Classes
ArticularyWordRecognition	275	300	9	144	25
DuckDuckGeese	50	50	1345	270	5
EMOPain	1093	50	30	180	3
FingerMovements	316	100	28	50	2
MotionSenseHAR	217	144	12	200	6
HandMovementDirection	160	74	10	400	4
Heartbeat	204	205	61	405	2
JapaneseVowels(*)	270	370	12	25	9
MindReading	727	653	204	200	5
MotorImagery	278	100	64	3000	2
NATOPS	180	180	24	51	6
PEMS-SF	267	173	963	144	7
PhonemeSpectra	3315	3353	11	217	39
Siemens	700	395	39	180	10
SpokenArabicDigits (*)	6599	2199	13	65	10

length. We also include four new high-dimensional datasets in the archive to help improve the power of our tests of performance. These four datasets are available on the UEA archive website[2].

EMOPain: The goal of the project that generated this data was the automatic detection of pain behaviors [19] and pain levels, based on data collected from people with chronic pain performing movements that are identical to those that make up daily physical functioning. The data consists of 26 sensors calculating angle positions on distinct parts of the body and 4 electromyography sensors that have the objective of measuring the electric signals generated by a muscle when is moved. The sensors are positioned on the upper fibers of the trapezium and the lumbar paraspinal muscles approximately at the 4/5 lumbar vertebra.

[2]www.timeseriesclassification.com

MindReading: The data consists of MEG recordings [36] of a single subject, made during two separate measurement sessions (consecutive days). In each session, the subject was watching five different movie categories without audio. The goal is to predict the category of the movie the subject is watching.

Siemens: This data consists of a group of sensors from four tanks that pump water from a reservoir tank to three small tanks [68]. The goal is to detect the type of failure the tank is experiencing based on the value of the different sensors.

MotionSenseHAR: This dataset includes time series data generated by accelerometer and gyroscope sensors (attitude, gravity, user acceleration, and rotation rate) [50]. A total of 24 participants in a range of genders, ages, weights, and heights performed 6 activities in 15 trials in the same environment and conditions: downstairs, upstairs, walking, jogging, sitting, and standing. With this dataset, we aim to look for personal attribute fingerprints in a series of sensor data, i.e. attribute-specific patterns that can be used to infer the gender or personality of the data subjects in addition to their activities.

6.5.2 Experiments

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. Each classifier was trained on the same 30 train-test resamples of the 15 high-dimensional datasets. Build time was limited to 7 days. Our performance metric is test set accuracy. To compare multiple classifiers on multiple data sets use ranks rather than accuracy, and we use critical difference diagrams [15] to display average ranks and cliques: a clique is a group of classifiers that are labeled as not significantly different from each

other. We find cliques through pairwise comparison at the 5% alpha level with an adjustment for multiple testing commonly called the Holm correction. This adjustment is less severe than a Bonferroni adjustment: we order classifiers by rank and then start with the best-performing classifier as our control. We pairwise test using the Wilcoxon sign-rank test in order, adjusting as to the maximum size of the clique. Thus, if testing 11 classifiers, the maximum number of tests to find the top clique is 10, so we require a p-value of alpha/10 to be considered significantly different. Once we find a classifier that fails the pairwise test, we form a clique of those before it. We then repeat the process with the second-best classifier, with the caveat that if a clique is found that is contained with one found already, we ignore it.

This is the most robust way we have found to form cliques, but it can still lead to anomalies. Given three classifiers, A, B, and C, where A is the highest rank and C is the lowest, A may be significantly better than B but not significantly better than C. However, our approach would put A and C in different cliques. We intend to move towards a more graphical display of pairwise tests and recommend that CD diagrams should only form part of the methodology of presenting results that compare classifiers.

6.6 Results

Our first experiment defines the scope of further experiments by bounding our expectations as to the accuracy of filtering before training HC2. Figure 6.6.1 shows the ranks of full HC2, full ROCKET, HC2 with 20% of dimensions selected randomly, and HC2 with 60% of dimensions selected randomly. Figure 6.6.1 illustrates that HC2 is significantly more accurate than ROCKET. We reran the experiments with the aeon implementation of HC2, so this serves to recreate the results reported in [53]. HC2 is, on average, 2.5% more accurate than ROCKET, winning on 11 problems and losing on 4. By default, HC2 is not configured for speed: it takes several hours to complete one resample of experiments, whereas

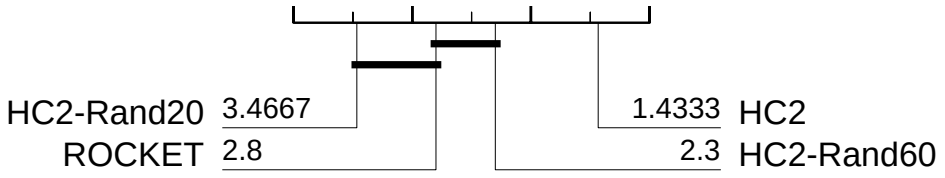


Figure 6.6.1: Critical difference diagram for comparing ROCKET, full HC2 and HC2 with random dimension selection.

ROCKET takes minutes. Figure 6.6.1 also shows that our basic straw man for comparison, randomly selecting 20% or 60% of attributes, results in a significant loss of accuracy in HC2. Our second experiment addresses the question as to whether applying any of the dimension selection algorithms listed in Table 6.1 can speed up HC2 without loss of accuracy. Figure 6.6.2 shows the relative ranked performance of eight different filtering algorithms in addition to the four classifiers shown in Figure 6.6.1. Cliques were formed with the p-values shown in Table 6.3. The average accuracy data for four classifiers is provided in Table 6.4, and full results are available in the associated repository.

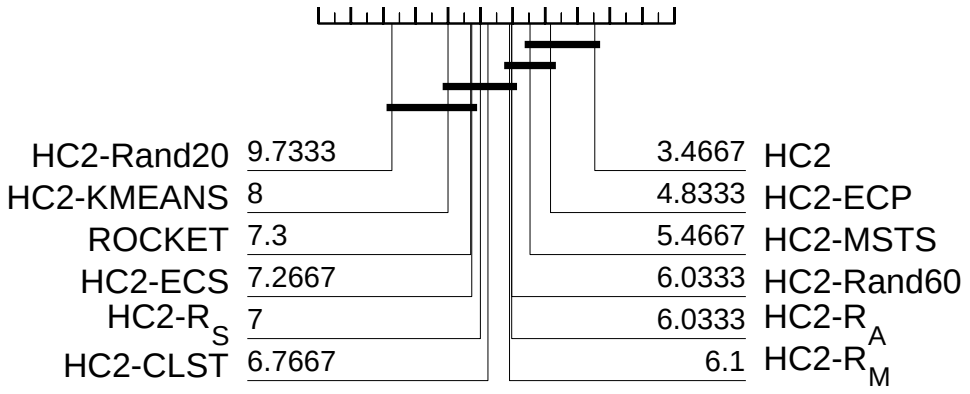


Figure 6.6.2: Critical difference diagram for comparing all dimension selection methods proposed.

Our first conclusion is that our hypothesis that ROCKET could be a good way of filtering dimensions for HC2 is not supported by these results. The three ROCKET variants are significantly worse than HC2, and no better than randomly selecting 60% of dimensions. Table 6.4 shows that using HC2-R$_A$ results on average in an approximate 1% decrease in accuracy. However, the two filters

Table 6.3: P-values for pairwise Wilcoxon rank-sum test on 15 high dimensional MTSC problems.

	ECP	MSTS	R60	R_A	R_M	CLST	R_S	ECS	RCKT	KMNS	R20
HC2	0.753	0.198	0.004	0.011	0.016	0.019	0.004	0.096	0.041	0.048	0.001
ECP		0.256	0.272	0.173	0.246	0.124	0.140	0.011	0.100	0.020	0.015
MSTS			0.394	0.570	0.056	0.125	0.173	0.334	0.334	0.281	0.006
R60				0.551	0.551	0.041	0.272	0.246	0.173	0.233	0.001
R_A					0.975	0.096	0.158	0.433	0.307	0.233	0.004
R_M						0.246	0.397	0.551	0.496	0.496	0.015
CLST							0.331	0.510	1.000	0.691	0.140
R_S								0.925	0.910	0.158	0.002
ECS									0.532	0.683	0.061
RCKT										0.826	0.307
KMNS											0.364

ECP and MSTS both achieve an accuracy rank that is not significantly worse than HC2. Table 6.4 shows that HC2-ECP performs very similarly to full HC2, but that HC2-MSTS may slightly reduce accuracy on average.

Of course, filtering will perfectly recreate HC2 results if it selects all dimensions. Table 6.5 shows the proportion of dimensions selected for three classifiers. On average, HC2-MSTS selects fewer dimensions, and in some cases, massively fewer. For example, with PEMS-SF[3] it selects just 13 out of 963 attributes and achieves an accuracy very close to that of full HC2. Each dimension in PEMS-SF is a single traffic sensor, and the data is measured over time. There will be high correlation between adjacent sensors, and HC2-MSTS is effective at removing a high degree of redundancy in this data.

Similarly, with DuckDuckGeese[4], HC2-MSTS chooses just 33 of the 1345 attributes and gets comparable accuracy to HC2, although HC2-ECP improves on HC2 when selecting about a third of attributes. DuckDuckGeese contains audio spectrograms of different bird species, and each dimension represents a frequency range. Both HC2-MSTS and HC2-R_A filter on the problem MindReading, whereas HC2-ECP correctly selects a larger number of

[3]http://www.timeseriesclassification.com/description.php?Dataset=PEMS-SF
[4]http://www.timeseriesclassification.com/description.php?Dataset=DuckDuckGeese

Table 6.4: Accuracy of four classifiers averaged over 30 resamples of 15 high dimensional datasets.

Name	HC2	HC2-ECP	HC2-MSTS	HC2-R_A
ArticularyWordRecognition	**99.51**	**99.51**	99.37	99.27
DuckDuckGeese	55.07	**57.93**	55.2	54
EMO	88.78	**89.87**	89.05	88.77
FingerMovements	**55.57**	52.53	54.5	55
MotionSenseHAR	**99.71**	99.67	99.66	99.67
HandMovementDirection	41.26	**42.48**	41.53	41.89
Heartbeat	73.59	**74.29**	73.06	73.24
JapaneseVowelsEq	93.15	**94.51**	93.06	91
MotorImagery	53.63	52.77	**53.73**	53.53
MindReading	**68.36**	68.17	60.81	60.44
NATOPS	**89.04**	87.54	85.98	87.17
PEMS-SF	**99.96**	97.23	99.92	**99.96**
PhonemeSpectra	**32.01**	31.72	**32.01**	31.67
Siemens	**100**	**100**	99.92	**100**
SpokenArabicDigitsEq	**99.67**	99.62	99.64	99.63
Average	76.62	76.52	75.83	75.68
Wins	9	7	2	2

dimensions and achieves a similar accuracy to full HC2.

Table 6.6 summarises the average training time for the HC2 and the three main filtering methods, and includes the time taken to filter. There is hardly any difference in time between the three algorithms, and each method takes about 60% of the time of full HC2.

MSTS and R_A both rely on miniROCKET to score dimensions but differ primarily in how attributes are selected. It seems that the subset selection used by MSTS may be marginally better than the elbow method used by R_A, although our tests do not have the power to reject the null hypothesis that

Table 6.5: Percentage of dimensions used.

Name	d	HC2-ECP	HC2-MSTS	HC2-R_A
ArticularyWordRecognition	9	100	**55.17**	57.09
DuckDuckGeese	1345	29.93	**2.48**	21.08
EMOPain	30	55.29	**31.72**	35.52
FingerMovements	28	38.18	**17.49**	50.49
MotionSenseHAR	12	86.78	**30.75**	65.8
HandMovementDirection	10	83.1	**44.14**	56.21
Heartbeat	61	15.38	**13.85**	45.56
JapaneseVowelsEq	12	75.57	97.41	**62.07**
MotorImagery	64	25.11	**6.25**	54.8
MindReading	204	61.56	**12.81**	16.23
NATOPS	24	79.31	**45.98**	63.07
PEMS-SF	963	35.03	**1.38**	13.8
PhonemeSpectra	11	**18.18**	100	59.25
Siemens	39	30.77	**10.88**	55.61
SpokenArabicDigitsEq	13	53.85	**53.85**	54.91
Average		52.54	**34.94**	47.43

there is no difference. ECP is based on distances between predictions and rather than accuracy and uses 1-NN DTW to make predictions. It tends to select more attributes than MSTS, but the extra time the resultant HC2 classifier takes is offset by the more time-consuming components of MSTS.

Overall, these experiments show that, although there is no significant difference between HC2-ECP, HC2-MSTS and HC2-R_A, only ECP and MSTS reduce dimensionality without reducing the accuracy of HC2 significantly. Both can prove useful tools for filtering before using HC2 with high dimensional data, with ECP marginally preferred because it more closely recreates HC2 results.

Table 6.6: Train time in hours, including the time to filter.

Name	HC2	HC2-ECP	HC2-MSTS	HC2-R_A
ArticularyWordRecognition	3.80	4.13	**3.42**	3.47
DuckDuckGeese	8.16	5.34	4.09	**3.87**
EMO	23.99	17.72	12.57	**11.52**
FingerMovements	4.15	3.29	**2.98**	3.70
HAR	21.36	21.70	**12.82**	19.39
HandMovementDirection	3.63	3.55	**3.20**	3.46
Heartbeat	6.71	**3.94**	3.96	4.42
JapaneseVowelsEq	3.04	3.07	3.06	**3.00**
MotorImagery	33.06	15.35	**9.32**	26.44
MindReading	42.38	29.9	**14.06**	15.56
NATOPS	3.06	3.00	**2.70**	2.90
PEMS-SF	32.64	13.54	**5.74**	9.44
PhonemeSpectra	112.49	**68.18**	113.20	85.18
Siemens	14.96	7.70	**4.96**	10.16
SpokenArabicDigitsEq	118.89	79.21	78.59	**76.84**
Sum	432.31	279.63	**274.65**	279.33

6.7 Conclusion

In this chapter, we focused on selecting dimensions as a preprocessing step and then executing HC2. It may be interesting to explore how filtering may affect each component, and indeed other classifiers. It may be more useful to embed the dimension selection within the component classifiers to create different feature subsets for each. we have also only evaluated dimension selection algorithms. Dimension creation algorithms may also be of use in MSTC.

We have donated four new datasets to the archive, but even then we are limited to evaluating 15 datasets. Furthermore, many of these are not genuinely high dimensional. More realistic cut-off points would be 50 or 100 dimensions. MTSC

data is very diverse in origin, and finding algorithms significantly better than other over all problem domains may prove unrealistic. In future work, we will continue to seek out new high-dimensional problems, and we intend to focus more specifically on EEG/MEG datasets, to make our research question more specific to that problem domain.

7

Conclusions / Future work

7.1 Discussion of contributions

This thesis started by defining the MTSC problem and the most important state-of-the-art algorithms. We mention the importance of having a robust archive of problems to test and introduce an archive of 33 problems in chapter 3 that we call the UEA archive. We outline a methodology for a set of experiments to find the most accurate algorithm for the UEA archive. The first set of experiments did not give a clear winner but Hive Cote, Canonical Interval Forest, and Rocket as statistically better than the other algorithms. Given that Rocket is significantly faster than the others. Next, we mentioned Hive Cote 2 (HC2) which proved to be significantly better than the other algorithms. We concluded that HC2 was the most accurate algorithm in the UEA archive.

Therefore, the rest of the thesis focused on two ways to improve HC2: First, by improving the shapelet transform classifier, which is the only component that remained from HC1 to HC2, and second, by giving that HC2 is a complex algorithm that takes a lot of time to train, we consider adding a dimension selection preprocess step to speed up the training time and if possible improve the accuracy.

For improving STC, we introduced a set of shapelet qualities to verify if any shapelet quality criteria is better than the other, which we proved is not the case. Then, we propose an ensemble of STC that uses different quality measures. We proved that the ensemble improved significantly the original results while improving training time. Finally, we ran an experiment of using HC2 with the ensemble we created and compared with the original HC2 results. We use the univariate and multivariate archives. We proved that in the multivariate case, HC2 with the shapelet ensemble is statistically better than HC2 with the original shapelet algorithm. However, in the case of univariate, this was not the case as they were equivalent.

In dimensions selection, it was possible to significantly reduce the number of

dimensions efficiently and without decreasing the performance.

7.2 Reflection and Future work

MTSC is an area of research that is still in its first steps and there is a lot of improvement to make. Compared with univariate problems there are a lot of improvements to be made. We presented some ideas in this thesis but many improvements can be made. In this section, we present some of these ideas.

7.2.1 UCR/UEA Archive chapter

There is a set of 33 problems but we did not include all of them as four problems do not have equal length in their examples. We added this limitation as many algorithms were not prepared to handle unequal time series. This limitation on the algorithms needs to be fixed as many problems in MTSC can have unequal series length as part of the nature of the problem and the algorithms need to fix that.

The univariate archive has more than 100 problems. Considering the 33 plus the 4 introduced in Chapter 6 is still a very low number. It is necessary to improve the research to find, categorize, and prepare the datasets that need to be included to consider at least 50 problems in the archive. More problems can give more accurate results to consider an algorithm as the state of the art and not only the most accurate in the archive.

The idea of grouping the problems by problem type was to find if any algorithm is better for some specific subset of problems. In future work, this can be a line of research to follow.

7.2.2 MTSC Bakeoff chapter

This chapter gives the most detailed work for this MTSC archive. However, as time goes new algorithms are developed in these need to be considered. Some algorithms performed poorly in the experiments and can easily be dropped to reduce the number of algorithms and increase the number of problems.

The deep learning algorithms did not perform well in these problems. It would be interesting to analyze what was the reason such as incorrect architecture, not enough train time, or any specific deep learning issue. Deep learning has proved to be useful in other domains so it is possible that with more research on the area, it can improve its results.

HC2 is the most accurate in the UEA archive. However, Rocket is the fastest algorithm that has good results. We base this on the 7-day constraint. HC2 passes this constraint but there are problem domains that maybe cannot afford to have this amount of time to train available. Another idea would be to define a smaller constraint (for example 1 day) and verify how many algorithms can pass this constraint and which one is the best.

7.2.3 Multivariate shapelet classifiers chapter

The major drawback in this chapter is that for the ensemble we handpicked the components which introduced bias to the model. For future work, a training phase should include a component selection to be included in the ensemble. It should analyze how this affects the train time and also if most algorithms select the same components or if some components are preferred for different problems.

Another element to analyze is why HC2 with the ensemble improved for multivariate problems but not for univariate ones. It can be that the univariate archive is stronger, or the more complex nature of MTSC problems or other causes but it is usually the case that if an algorithm improved univariate, this

improvement also reflects on multivariate like in the case of HC2.

Random dilated shapelet transform is different from than shapelet transform classifier as it does not have the shapelet quality selection part which is very time-consuming. That introduced the question of whether this part is necessary and if the improvement justifies the amount of time it takes.

7.2.4 Dimension selection strategies

This is perhaps one of the newest areas on MTSC as the results did not improve or decrease the accuracy of not filtering. This strategy was used to filter HC2 because HC2 is time-consuming in train time and the most accurate in the UEA archive. We perform filtering first and then start HC2. A different approach would be to filter each component of HC2 and verify if this individual filter strategy is better than the one used.

More research is required to find novel ways to improve the filtering to improve performance on problems. In this case, it would be necessary to answer the question to verify if a problem has redundant or unnecessary dimensions or if all dimensions are needed. For huge dimensional problems (more than 100) the first case is probably the case but some statistical analysis needs to be developed to measure this and include filtering as part of the algorithm.

The previous chapter introduced independent vs dependent shapelets and compared both approaches. An intermediate model called semi-dependent shapelets has been introduced where a subset of random dimensions is chosen. In this case, the filtering approaches here can be used to have a more intelligent way to make the subset selection.

A

Appendix

Table A.1: Classifier Configuration

Algorithm	Configuration
DTW_D	Full warping window
DTW_I	Full warping window
DTW_A	Full warping window
MUSE	χ=2
gRFS	Default (Max depth: none, min sample split: 2, num shapelets: 10
	min size: 0%, max size: 100%, metric: Euclidean distance)
MrSEQL	seql_mode: fs, symrep: ['sax', 'sfa']
ROCKET	Ridge regression classifier, 10,000 kernels
CIF	Default (trees: 500, intervals:$\sqrt{(m)} \times \sqrt{(d)}$, 8 attributes per tree)
TapNet	Default (Epochs: 3000, Learning rate: $1e-5$, weight decay: $1e-3$
	stop threshold: $1e-9$, num filters: [256 256 128], kernels: [8 5 3]
	dilation: 1, dropout: 0%)
	ArticularyWordRecognition (dilation: 10)
	EthanolConcentration (dilation: 200, learning rate: $1e-6$)
	FaceDetection (filters: [64 64 32], learning rate: $5e-5$)
	Heartbeat (dilation: 200, learning rate: $1e-6$, filters: [64 64 32])
	PenDigits (kernels: [4 1 1], learning rate: $1e-3$)
	PhonemeSpectra (filters: [64 64 32], learning rate: $1e-3$)
	SelfRegulationsCP1 (learning rate: $1e-6$)
	SelfRegulationsCP2 (learning rate: $1e-9$)
	SpokenArabicDigits (filters: [128 128 64], learning rate: $1e-4$)
ResNet	Epochs: 1500, batch size: 16, learning rate: $1e-3$ and halved after
	no improvement for 50 epochs.
	Three residual blocks each with three conv layers with kernel sizes [8, 5, 3]
	filters per conv layer for each block [64, 128, 128]
InceptionTime	Epochs: 1500, batch size: 64, learning rate: $1e-3$ and halved after no
	improvement for 50 epochs.
	Two residual blocks each with three Inception modules with kernel sizes
	per module [10, 20, 40]
	plus bottleneck filters for all conv layers 32
CBOSS	Default, see [2]
STC	Default, see [2]
RISE	Default, see [2]
TSF	Default, see [2]
HC1 [2]	

Bibliography

[1] A. Bagnall, H. Dau, J. Lines, M. Flynn, J. Large, A. Bostrom, P. Southam, and E. Keogh. The UEA multivariate time series classification archive, 2018. *ArXiv e-prints*, arXiv:1811.00075, 2018.

[2] A. Bagnall, M. Flynn, J. Large, J. Lines, and M. Middlehurst. On the usage and performance of HIVE-COTE v1.0. In *proceedings of the 5th Workshop on Advances Analytics and Learning on Temporal Data*, volume 12588 of *Lecture Notes in Artificial Intelligence*, 2020.

[3] A. Bagnall, M. Flynn, J. Large, J. Lines, and M. Middlehurst. A tale of two toolkits, report the third: on the usage and performance of HIVE-COTE v1.0, 2020.

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

[5] M. Baydogan and G. Runger. Learning a symbolic representation for multivariate time series classification. *Data Mining and Knowledge Discovery*, 29(2):400–422, 2015.

[6] B. Blankertz, G. Curio, and KR. Müller. Classifying single trial EEG: towards brain computer interfacing. In *proceedings of Advances in Neural Information Processing Systems 15*, pages 157–164, 2002.

[7] B. Blankertz, G. Dornhege, M. Krauledat, K. Müller, and G. Curio. The non-invasive berlin brain-computer interface: Fast acquisition of effective performance in untrained subjects. In *NeuroImage*, volume 37(2), pages 539,550, 2007.

[8] A. Bostrom and A. Bagnall. Binary shapelet transform for multiclass time series classification. *Transactions on Large-Scale Data and Knowledge Centered Systems*, 32:24–46, 2017.

[9] A. Bostrom, A. Bagnall, and J. Lines. Evaluating improvements to the shapelet transform. *Knowledge Discovery and Data Mining, in Workshop on Mining and Learning from Time Series*, 2016.

[10] M. Cuturi. Fast global alignment kernels. In *proceedings of the 28th International Conference on Machine Learning*, pages 929–936, 2011.

[11] S. Peres D. Dias. Algoritmos bio-inspirados aplicados ao reconhecimento de padroes da libras: enfoque no parâmetro movimento. *16 Simpósio Internacional de Iniciaçao Cientifica da Universidade de Sao Paulo*, 2016.

[12] 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.

[13] V. Alves de Souza. Asphalt pavement classification using smartphone accelerometer and complexity invariant distance. *Engineering Applications of Artificial Intelligence*, 74:198–211, 09 2018.

[14] A. Dempster, F. Petitjean, and G. Webb. ROCKET: Exceptionally fast and accurate time series classification using random convolutional kernels. *FData Mining and Knowledge Discovery*, 34:1454–1495, 2020.

[15] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

[16] H. Deng, G. Runger, E. Tuv, and M. Vladimir. A time series forest for

classification and feature extraction. *Information Sciences*, 239:142–153, 2013.

[17] B. Dhariyal, T. Le Nguyen, and G. Ifrim. Fast channel selection for scalable multivariate time series classification. 2021.

[18] L. Grundy L. Tadas E. Yemini, A. Brown and W. Schafer. 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.

[19] J. Egede, A. Olugbade, C. Wang, S. Song, N. Bianchi-Berthouze, M. Valstar, A. Williams, H. Meng, H. Aung, and N. Lane. Emopain challenge 2020: Multimodal pain evaluation from facial and bodily expressions. *2020 15th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2020)*, pages 849–856, 2020.

[20] E. Alpaydin F. Alimoglu, Y. Doc and Y. Denizhan. Combining multiple classifiers for pen-based handwritten digit recognition, 1996.

[21] H. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P. Muller. Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, 33(4):917–963, 2019.

[22] HI. Fawaz, B. Lucas, G. Forestier, C. Pelletier, D. Schmidt, J. Weber, G. Webb, L. Idoumghar, P. Muller, and F. Petitjean. Inceptiontime: Finding alexnet for time series classification. *ArXiv*, 2019.

[23] M. Flynn, J. Large, and A. Bagnall. The contract random interval spectral ensemble (c-RISE): The effect of contracting a classifier on accuracy. In *International Conference on Hybrid Artificial Intelligence Systems*, volume 11734 of *Lecture Notes in Computer Science*, pages 381–392. 2019.

[24] B. Fulcher and N. Jones. HCTSA: A computational framework for automated time-series phenotyping using massive feature extraction. *Cell Systems*, 5(5):527–531, 2017.

[25] N. Ghouaiel, PF. Marteau, and M. Dupont. Continuous pattern detection and recognition in stream-a benchmark for online gesture recognition. *International Journal of Applied Pattern Recognition*, 4(2):146–160, 2017.

[26] A. Goldberger, L. Amaral, L. Glass, J. Hausdorff, P. Ivanov, R. Mark, J. Mietus, G. Moody, CK. Peng, and E. Stanley. PhysioBank, physiotoolkit, and physionet: components of a new research resource for complex physiologic signals. *Circulation*, 101(23):e215–e220, 2000.

[27] A. Guillaume, C. Vrain, and W. Elloumi. Random dilated shapelet transform: A new approach for time series shapelets. In *Pattern Recognition and Artificial Intelligence*, pages 653–664. Springer International Publishing, 2022.

[28] M. Hammami and M. Bedda. Improved tree model for arabic speech recognition. In *Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on*, volume 5, pages 521–526. IEEE, 2010.

[29] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

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

[31] H. Hossein and A. Mueen. Dual-domain hierarchical classification of phonetic time series. In *Data Mining (ICDM), 2014 IEEE International Conference on*, pages 160–169. IEEE, 2014.

[32] F. Karim, S. Majumdar, H. Darabi, and S. Harford. Multivariate LSTM-FCNs for time series classification. *Neural Networks*, 116:237–245, 2019.

[33] F. Karim, S. Majumdar, h. Darabi, and S. Chen. Lstm fully convolutional networks for time series classification. *IEEE access*, 6:1662–1669, 2017.

[34] I. Karlsson, Papapetrou P, and H. Boström. Generalized random shapelet forests. *Data Mining and Knowledge Discovery*, 30(5):1053–1085, 2016.

[35] B. Kathirgamanathan and P. Cunningham. A feature selection method for multi-dimension time-series data. In *Advanced Analytics and Learning on Temporal Data*, pages 220–231, Cham, 2020. Springer International Publishing.

[36] A. Klami. Proceedings of icann/pascal2 challenge: Meg mind reading. Technical report, 2011.

[37] MH. Ko, G. West, S. Venkatesh, and M. Kumar. Online context recognition in multisensor systems using dynamic time warping. In *Proceedings of the International Conference on Intelligent Sensors, Sensor Networks, and Information Processing*, pages 283–288, 2005.

[38] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[39] M. Kudo, J. Toyama, and M. Shimbo. Multidimensional curve classification using passing-through regions. *Pattern Recognition Letters*, 20(11-13):1103–1111, 1999.

[40] T. Lal, T. Hinterberger, G. Widman, M. Schröder, J. Hill, W. Rosenstiel, C. Elger, N. Birbaumer, and B. Schölkopf. Methods towards invasive human brain-computer interfaces. In *Proceedings of Advances in Neural Information Processing Systems 18*, pages 737–744, 2005.

[41] J. Large, K. Kemsley, N. Wellner, I. Goodall, and A. Bagnall. Detecting forged alcohol non-invasively through vibrational spectroscopy and machine learning. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 298–309, 2018.

[42] J. Large, J. Lines, and A. Bagnall. A probabilistic classifier ensemble

weighting scheme based on cross validated accuracy estimates. *Data Mining and Knowledge Discovery*, 33(6):1674—1709, 2019.

[43] J. Lin, E. Keogh, L. Wei, and S. Lonardi. Experiencing SAX: a novel symbolic representation of time series. *Data Mining and Knowledge Discovery*, 15(2), 2007.

[44] J. Lines and A. Bagnall. Alternative quality measures for time series shapelets. In Hujun Yin, José A. F. Costa, and Guilherme Barreto, editors, *Intelligent Data Engineering and Automated Learning - IDEAL 2012*, pages 475–483, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[45] J. Lines, L. Davis, J. Hills, and A. Bagnall. A shapelet transform for time series classification. In *Proc. the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2012.

[46] J. Lines, S. Taylor, and A. Bagnall. Time series classification with HIVE-COTE: The hierarchical vote collective of transformation-based ensembles. *ACM Trans. Knowledge Discovery from Data*, 12(5), 2018.

[47] C. Liu, D. Springer, Q. Li, B. Moody, JR. Abad, F. Chorro, F. Castells, J. Roig-Millet, I. Silva, and A. Johnson. An open access database for the evaluation of heart sound algorithms. *Physiological Measurement*, 37(12):2181, 2016.

[48] J. Liu, L. Zhong, J. Wickramasuriya, and V. Vasudevan. uWave: accelerometer-based personalized gesture recognition and its applications. *Pervasive and Mobile Computing*, 5(6):657–675, 2009.

[49] C. Lubba, S. Sethi, P. Knaute, S. Schultz, B. Fulcher, and N. Jones. Catch22: Canonical time-series characteristics. *Data Mining and Knowledge Discovery*, 33(6):1821–1852, 2019.

[50] M. Malekzadeh, R. Clegg, A. Cavallaro, and H. Haddadi. Mobile sensor data anonymization. In *Proceedings of the International Conference on Internet*

of Things Design and Implementation, IoTDI '19, pages 49–58, New York, NY, USA, 2019. ACM.

[51] M. Middlehurst, J. Large, and A. Bagnall. The canonical interval forest (CIF) classifier for time series classification. In *proceedings of the IEEE International Conference on Big Data*, 2020.

[52] M. Middlehurst, J. Large, G. Cawley, and A. Bagnall. The temporal dictionary ensemble (TDE) classifier for time series classification. In *proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 2020.

[53] 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.

[54] T. Hinterberger I. Iversen-B. Kotchoubey A. Kübler J. Perelmouter E. Taub H. Flor N. Birbaumer, N. Ghanayim. A spelling device for the paralysed. *Nature*, 398(6725):297, 1999.

[55] T. L. Nguyen, S. Gsponer, and G. Ifrim. Time series classification by sequence learning in all-subsequence space. In *proceedings of 33rd IEEE International Conference on Data Engineering*, pages 947–958, 2017.

[56] T. Le Nguyen, S. Gsponer, J. Ilie, M. O'Reilly, and G. Ifrim. Interpretable time series classification using linear models and multi-resolution multi-domain symbolic representations. *Data Mining and Knowledge Discovery*, 33(4):1183–1222, 2019.

[57] T. Le Nguyen and G. Ifrim. Mrsqm: Fast time series classification with symbolic representations, 2022.

[58] E. Olivetti, SM. Kia, and P. Avesani. MEG decoding across subjects. In *Pattern Recognition in Neuroimaging, 2014 International Workshop*, pages 1–4, June 2014.

[59] A. Pasos-Ruiz and A. Bagnall. Dimension selection strategies for multivariate time series classification with hive-cotev2.0. volume 13812 of *Lecture Notes in Computer Science*, pages 133–147. Springer, 2022.

[60] A. Pasos-Ruiz, M. Flynn, and A. Bagnall. Benchmarking multivariate time series classification algorithms. *ArXiv e-prints*, ArXiv:2007.13156, 2020.

[61] J. Rodriguez, L. Kuncheva, and C. Alonso. Rotation forest: A new classifier ensemble method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(10):1619–1630, 2006.

[62] A. Pasos Ruiz, M. Flynn, Large J, M. Middlehurst, and A. Bagnall. The great multivariate time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, 35:401 – 449, 2021.

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

[64] P. Schäfer and M. Högqvist. 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*, pages 516–527, 2012.

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

[66] M. Shokoohi-Yekta, B. Hu, H. Jin, J. Wang, and E. Keogh. Generalizing DTW to the multi-dimensional case requires an adaptive approach. *Data Mining and Knowledge Discovery*, 31(1):1–31, 2017.

[67] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *The*

IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2015.

[68] D. Labisch T. Bierweiler. Four-tank batch process in smart automation. Technical report, 2021.

[69] J. Villar, P. Vergara, M. Menéndez, E. de la Cal, V. González, and J. Sedano. Generalized models for the classification of abnormal movements in daily life and its applicability to epilepsy convulsion recognition. *International Journal of Neural Systems*, 26(06):1650037, 2016.

[70] J. Wang, A. Balasubramanian, LM. de La Vega, J. Green, A. Samal, and B. Prabhakaran. Word recognition from continuous articulatory movement time-series data using symbolic representations. In *Proceedings of the 4th Workshop on Speech and Language Processing for Assistive Technologies*, pages 119–127, 2013.

[71] Z. Wang, W. Yan, and T. Oates. Time series classification from scratch with deep neural networks: A strong baseline. In *2017 international joint conference on neural networks*, pages 1578–1585, 2017.

[72] M. Wilhelm, D. Krakowczyk, F. Trollmann, and S. Albayrak. ERing: multiple finger gesture recognition with one ring using an electric field. In *Proceedings of the 2nd International Workshop on Sensor-based Activity Recognition and Interaction*, page 7. ACM, 2015.

[73] B. Williams, M. Toussaint, and A. Storkey. Extracting motion primitives from natural handwriting data. In *International Conference on Artificial Neural Networks*, pages 634–643. Springer, 2006.

[74] B. Williams, M. Toussaint, and A. Storkey. A primitive based generative model to infer timing information in unpartitioned handwriting data. In *IJCAI*, pages 1119–1124, 2007.

[75] B. Williams, M. Toussaint, and A. Storkey. Modelling motion primitives and

their timing in biologically executed movements. In *Proceedings of Advances in Neural Information Processing Systems 21*, pages 1609–1616, 2008.

[76] G. Batista A. Mafra-Neto E. Keogh Y. Chen, A. Why. Flying insect classification with inexpensive sensors. *Journal of insect behavior*, 27(5):657–677, 2014.

[77] K. Yang, H. Yoon, and C. Shahabi. CLeVer: A feature subset selection technique for multivariate time series. In *Advances in Knowledge Discovery and Data Mining*, pages 516–522, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

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

[79] X. Zhang, Y. Gao, J. Lin, and CT. Lu. TapNet: Multivariate time series classification with attentional prototypical network. In *proceedings of 34th AAAI conference on artificial intelligence*, 2020.