

One-Class Time Series Classification

Stefano Mauceri

M.Sc. University College Dublin
UCD Student Number: 15203663

The thesis is submitted to University College Dublin
in fulfilment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

School of Business

Head of School:
Prof. Anthony Brabazon

Research Supervisors:
Dr. James McDermott
Dr. James Sweeney
Dr. Miguel Nicolau

External Examiner:
Prof. Giuseppe Nicosia

Internal Examiner:
Dr. Paula Carroll

September, 2020

Statement of Original Authorship

I hereby certify that the submitted work is my own work, was completed while registered as a candidate for the degree stated on the Title Page, and I have not obtained a degree elsewhere on the basis of the research presented in this submitted work.

Abstract

This thesis contributes to the state of the art of time series classification and machine learning by investigating three novel data-driven representations for time series in the context of one-class classification. The one-class assumption is useful for all classification problems where only data of a single class is available for training a classifier, or those where it is not known if novel classes may appear at prediction time or what they could look like. Notable examples that can benefit from our research are: anomaly or novelty detection, fault detection, identity authentication, etc.






The common thread of our research is to represent time series as feature-vectors then used for classification. The features we extract are: (1) features constructed using dissimilarity measures; (2) features constructed using an evolutionary algorithm; (3) latent features constructed using neural networks. The proposed representations are thoroughly investigated in a variety of one-class classification experiments involving numerous benchmark methods, the 85 data-sets of the UCR/UEA archive and a data-set provided by ICON plc.

The key difference between one-class classification and binary or multi-class classification is in the amount of effort needed to gather training data. Binary and multi-class classifiers require exhaustively labelled training data. This can be difficult for problems where all but the samples of one class are scarcely available and ill-defined, e.g. anomaly detection. Or again, gathering labelled data can simply be impossible due to the cost of expert labour required to construct an appropriate data-set. Conversely, one-class classifiers are trained using only samples from a single class. We present a subject authentication problem through accelerometer data as a case study that motivates our research on one-class time series classification. We argue that it is not realistic to assume we can gather labelled training data that represent well both the subject of interest and a fixed population of “others”. Hence, the need to learn a classifier using data related to the subject of interest only.

We demonstrate that, with respect to the use of raw time series, feature-based representations allow substantial and compelling savings in terms of storage and computational requirements, facilitate the interpretability of the

solutions found, and enable visualisation of time series data-sets. We find that these advantages come at the cost of a slight loss in terms of classification performance with respect to a 1-nearest neighbour classifier on raw data. However, by examining data-sets one by one we detail how our representations can outperform raw time series. Furthermore, for some applications, e.g. embedded systems, storage and computational requirements may be more important than a slight loss in classification performance.

Acknowledgements

My deepest gratitude goes to my supervisors Dr. James McDermott , Dr. James Sweeney , and Dr. Miguel Nicolau , for their guidance and advice. The support of Prof. Michael O'Neill  is truly appreciated. Special thanks to all the colleagues of the [NCRA](#) lab for the good time. Finally, I would like to thank Dr. Alexandros Agapitos  for helpful discussions.

I must thank [ICON plc](#) for funding this research.

I must thank the DJEI/DES/SFI/HEA Irish Centre for High-End Computing ([ICHEC](#)) for the provision of computational facilities and support.

Last but not least, I must thank my wife, my parents, and my extended family for making the difference every day. Thanks to all my friends for the ninety craic. The best is yet to come.

To the quaint Emerald Isle

Contents

Statement of Original Authorship	i
Abstract	ii
Acknowledgements	iv
Publications Arising	ix
List of Figures	x
List of Tables	xiv
I Background	1
1 Introduction	2
1.1 Why is this Research Important?	2
1.2 One-Class Classification	3
1.3 Time Series Representations	5
1.4 Scientific Contributions	6
2 Related Work	8
2.1 Time Series Mining	8
2.2 Time Series Classification	10
2.2.1 Intra-Class Variance	12
2.2.2 One-Class Classification	12
2.2.3 Anomaly Detection	15
2.3 Instance-Based Classification	17
2.3.1 Neural Networks-Based Classification	18
2.4 Feature-Based Classification	21
2.4.1 Dissimilarity-Based Representations	23
2.4.2 Evolutionary Computation Techniques	25

3	Problem Statement and Experimental Design	27
3.1	Main Concepts	27
3.2	Experimental Data	28
3.2.1	The UCR/UEA Archive	29
3.2.2	Accelerometer Data	29
3.2.3	One-Class Labelling	31
3.3	Classifiers	32
3.4	Performance Evaluation	32
3.5	Dissimilarity Measures	33
3.6	Implementation Details	37
II	Experimental Research	39
4	Dissimilarity-Based Representations	40
4.1	Proposed Method	41
4.1.1	Overview	41
4.1.2	Prototype Methods	42
4.2	Results	44
4.2.1	Overview	44
4.2.2	Dissimilarity Measures and Prototype Methods	46
4.2.3	Dimensionality Reduction	49
4.2.4	Visual Exploration of Time Series Data-Sets	50
4.3	Conclusions	52
5	Feature-Based Representations via Grammatical Evolution	53
5.1	Proposed Method	54
5.1.1	Overview	55
5.1.2	Grammatical Evolution	56
5.1.3	Grammar and Primitives	57
5.1.4	Fitness Evaluation	59
5.2	Benchmark Methods	60
5.2.1	Random Search	61
5.2.2	1NN with DTW	61
5.2.3	Function and Sub-Sequence Selection	61
5.3	Experimental Design	62
5.3.1	Experimental Data	62
5.3.2	GE Configuration	63
5.3.3	Implementation Details	63
5.4	Results	63
5.4.1	Overview	64

5.4.2	Subject Authentication	66
5.4.3	Limitations	68
5.5	Feature-extractors and Interpretability	69
5.5.1	The Features to Extract	70
5.5.2	The Sub-sequences from which to Extract	74
5.6	Conclusions	76
6	Auto-Encoder-Based Representations	79
6.1	Proposed Method	80
6.1.1	Overview	81
6.1.2	Neural Network Architectures	82
6.2	Experimental Design	84
6.2.1	Dissimilarity Measures	84
6.2.2	Classification Framework and Data-Sets	85
6.2.3	Benchmark Methods	85
6.2.4	Implementation Details	85
6.3	Results	87
6.3.1	Overview	87
6.3.2	Learned DTW	95
6.3.3	Visual Exploration of Time Series Data-Sets	96
6.4	Conclusions	101
7	Comparing the Representations	102
7.1	Comparing Representations	102
7.1.1	Results	103
7.2	Combining Representations	116
7.2.1	Results	117
7.3	More Benchmark Methods	117
7.3.1	Results	118
7.4	Conclusions	119
III	Conclusions	120
8	Conclusions	121
8.1	Scientific Contributions	121
8.2	Limitations	124
8.3	Future Work	125
	Bibliography	128

Publications Arising

1. Stefano Mauceri, Louis Smith, James Sweeney, and James McDermott. Subject recognition using wrist-worn triaxial accelerometer data. In *International Workshop on Machine Learning, Optimization, and Big Data*, pages 574–585. Springer, 2017
2. Stefano Mauceri, James Sweeney, and James McDermott. One-class subject authentication using feature extraction by grammatical evolution on accelerometer data. In *International Conference on Metaheuristics and Nature Inspired Computing*, 2018
3. Stefano Mauceri, James Sweeney, and James McDermott. Dissimilarity-based representations for one-class classification on time series. *Pattern Recognition*, 100:107122, 2020

List of Figures

2.1	Time series intra-class variance. The solid line (—), and the dashed line (---) are used to show two time series that belong to the same class. However, in part (a) class membership is not immediately recognisable, unless an appropriate transform is applied as in part (b). (1) Amplitude invariance can be achieved by subtracting the mean, and dividing by the standard deviation (z-normalisation). (2) Linear trend invariance can be achieved by de-trending the time series (we use first order differences in our example). (3) Occlusion invariance can be achieved by filling the missing values (we use linear interpolation in our example). (4) Offset invariance can be achieved by subtracting the mean. (5) Phase invariance can be achieved by translating along the time-axis. (6) Uniform scaling invariance can be achieved by scaling the temporal index by a certain factor (1.5 in our example). (7) Warping invariance can be achieved by locally warping one time series to align with the other. In part (a), the dotted lines (····) show the alignment between the two time series as they were compared through a lock-step measure [250] (e.g. Euclidean distance). In part (b), the dotted lines (····) show the alignment between the two time series as they were compared through an algorithm defined to find the alignment that has minimum cumulative absolute difference (e.g. DTW).	13
2.2	Binary vs one-class classification. In part (a) a binary classifier (—) trained to separate positive samples from the negative ones (○/●) fails when a new class of negative samples (■) appears at prediction time. In part (b) a one-class classifier (---) trained on positive samples (○) only is robust against negative classes either known or not (●/■) at training time. .	14
3.1	Summary of characteristics of the data-sets in use.	29

3.2	Magnitude recordings for one day for one subject.	31
3.3	A randomly selected time series for each data-set in use. . . .	38
4.1	DBR. On the left-hand side there are two prototypes (P1, P2) and one time series (T) that we project into a 2-dimensional dissimilarity space shown on the right-hand side. For illustration the prototypes are projected in the dissimilarity space too. In the brackets it is shown how the new (x, y) coordinates are derived using a dissimilarity measure (d).	41
4.2	Prototype methods on 2-dimensional random data generated from a mixture of normal distributions with $n = 10$. ● Random data. □ Prototypes.	43
4.3	Scatter-plots of AUROC performance over all data-sets, analysed by dissimilarity measure for DBR _{100%} and RD _{100%}	46
4.4	Counts of data-sets where DBR _{100%} achieves a better performance than RD _{100%} (□) and vice-versa (■) for the EDR dissimilarity only.	47
4.5	Scatter-plots of AUROC performance over all data-sets for the pair (Manhattan, Percentiles) compared to all the other prototype methods paired with the same dissimilarity measure under DBR _{20%}	48
4.6	AUROC performance for all pairs of dissimilarity measures and prototype methods. In part (a) results are averaged over all data-sets; in part (b) are averaged only data-sets with average number of training samples greater than 39.	49
4.7	□ Training samples. ○ Positive test samples. ● Negative test samples. The shaded area gets darker as the distance from training samples increases. (a1) “FiftyWords”, ED, Closest, positive class: 6. (b1) “FiftyWords”, DTW, Closest, positive class: 6. (a2) “FacesUCR”, Ma., Centers-k-means, positive class: 2. (b2) “FacesUCR”, Ma., Percentiles, positive class: 2. (c) “FordA”, Ch., Support vectors, positive class: -1. (d) “ShapeletSim”, WD, Furthest, positive class: 0.	51
5.1	Grammar used to evolve the feature extractors.	58

5.2	Fitness evaluation of a feature-extractor F . First the feature-extractor is applied on each training/validation sample deriving the respective feature-based representation. The classifier in use is a 1NN-ED. The variable x corresponds the number of the feature we are extracting. The algorithm minimises the classification error ($1 - \text{AUROC}$) and the average squared Pearson correlation coefficient with previous features on training data ($\overline{R^2}(F)$).	60
5.3	Average AUROC of our algorithm for all data-sets (●). (a) AUROC per feature. (b) AUROC per sequence of features. Secondly, RS average AUROC for all data-sets (■).	66
5.4	Average AUROC “AccelerometerData” data-set, subjects 1 to 6. — Only subjects 1-6 are used during the feature-extraction phase. --- All subjects 1-9 are used during the feature-extraction phase.	68
5.5	(a) Average validation (■) and test (●) AUROC for all data-sets per feature. (b) Average validation and test AUROC for all data-sets per sequence of features.	69
5.6	A time series per each of the 6 classes of the “SyntheticControl” data-set.	72
5.7	Selection frequency of the grammar primitives for the first two features extracted from each of the 6 classes of the “SyntheticControl” data-set. The horizontal lines represent the bar heights we would observe if feature-extractors were generated at random i.e. without regard to fitness.	73
5.8	2D feature-based representation of each class of the “SyntheticControl” data-set. □ Training samples. ○ Normal test samples. ● Anomalous test samples. Shading represents distance from training set.	74
5.9	Selection frequency of each point within a time series of the “GunPoint” data-set. — Average time series from the data-set. --- Frequencies we would observe if feature-extractors were generated at random i.e. without regard to fitness. Frequencies as per our algorithm.	75
5.10	Sub-sequences of the “GunPoint” data-set related to the time interval $[0 : 40]$. Part (a) and (b) show the sub-sequences related to class 1 and 2 respectively.	76
5.11	Selection frequency of each point within a time series of the “AccelerometerData” data-set. The number in brackets in the top left corner of each plot corresponds to a different class of the data-set.	78

6.1	Diagram of the proposed convolutional auto-encoder for univariate time series.	84
6.2	Histograms (with a log vertical scale) of the differences between pairwise dissimilarities in the original space and in the latent space for all the considered data-sets. Numbers on the top-right corner corresponds to the mean and standard deviation of differences. Results are broken down by architecture and dissimilarity measure.	94
6.3	Three time series from the “BirdChicken” data-set (class 1) (a), and their latent representation according to a DissPCE-DTW (with 10 layers and latent dimensionality equal to $2 \times \log_2(L)$, where L is the time series length) (b).	97
6.4	Three time series from the “TwoLeadECG” data-set (class 1) (a), and their latent representation according to a DissPCE-DTW (with 10 layers and latent dimensionality equal to $2 \times \log_2(L)$, where L is the time series length) (b).	97
6.5	Three time series from the “Worms” data-set (class 1) (a), and their latent representation according to a DissPCE-DTW (with 10 layers and latent dimensionality equal to $2 \times \log_2(L)$, where L is the time series length) (b).	98
6.6	Three time series from the “Worms” data-set (class 1) (a), and their latent representation according to a DissPCE-DTW (with 10 layers and latent dimensionality equal to $2 \times \log_2(L)$, where L is the time series length) (b).	98
6.7	Dissimilarities of all training samples from the class medoid for three data-sets. For each class of each data-set samples are ordered according to their DTW dissimilarity from the class medoid in the original space. Then, using the ordering found the following dissimilarities from the class medoid are calculated. DTW dissimilarities in the original space (—), Euclidean distances in the original space (⋯⋯), and Euclidean distances in the latent space according to a DissPCE-DTW (with 10 layers and latent dimensionality equal to $2 \times \log_2(L)$, where L is the time series length) (---). The graph demonstrates that the correlation between DTW dissimilarities in the original space and dissimilarities in the latent space is not spurious due the ED acting as a confounding factor.	99
6.8	○ Positive samples. ● Negative samples. (a) “Adiac”, positive class: 9. (b) “DistalPhalanxTW”, positive class: 8. (c) “NonInvasiveFatalECGThorax1”, positive class: 33.	100

7.1	Critical difference diagram for the AUROC of all representations. Groups of representations that are not significantly different are connected by a horizontal bar. (a) Shows the performance of all the considered representations except FBRGE that is considered in (b).	112
7.2	Count of max AUROC performance by representation according to Table 7.1. RD (\square), DBR (\blacksquare), AEBR (\blacksquare).	113
7.3	Count of max AUROC performance by dissimilarity measure according to Table 7.1. DTW (\square), ED (\blacksquare), MSM (\blacksquare).	114
7.4	CC as a function of the time series length L . This is the CC to represent, and to classify with the considered approaches and a 1NN classifier equipped with the ED. — RD. --- DBR. --- FBRGE. AEBR.	115

List of Tables

3.1	“AccelerometerData” data-set: recorded variables.	31
3.2	Dissimilarity measures used in this work.	37
4.1	The table shows AUROC averaged across all the data-sets and rounded to the nearest integer.	45
5.1	Primitives used to evolve the feature extractors.	58
5.2	Average AUROC for the data-sets of the UCR/UEA repository rounded to the nearest integer. (a) Performance of every sequence of features, e.g. 1-10 means all 10 features. (b) Benchmark methods. Columns in bold relate to our best result (1-3) (best performance with lowest dimensionality), and the best benchmark method (1NN-DTW).	67
6.1	Considered neural network architectures. Acronyms that contains a “C” refer to convolutional networks, otherwise the network is dense. Acronyms that contains “DissP” refer to networks that use the $\mathcal{L}_{\text{DissP}}$ loss function. Finally, acronyms that contains a “V” refer to variational networks.	83
6.2	Summary of results. The table shows AUROC averaged across all the data-sets and rounded to the nearest integer. Table cells get darker as the AUROC increases. For each architecture, results are broken down by hyper-parameter configuration (number of layers, latent dimensionality where L is the time series length, and dissimilarity measure).	88

6.3	The table shows the Pearson correlation (R^2) of pairwise dissimilarities in the original space and in the latent space. Table cells get darker as the correlation increases. Data are divided in three groups: training data (TR), positive test data (TE ^P), and negative test data (TE ^N). For each architecture, results are broken down by hyper-parameter configuration (number of layers, latent dimensionality where L is the time series length, and dissimilarity measure).	91
6.4	The table shows the classification performance of a 1NN classifier on raw data for each of the three dissimilarity measures considered in the chapter: DTW, ED, and MSM. Also, the table shows the performance enabled by the latent representation of both a DissPCAE and DissPCE trained to preserve the considered dissimilarity measures. AUROC results, for all the 86 data-sets introduced in Section 3.2, are rounded to the nearest integer. Cells get darker as the AUROC increases. . . .	92
6.5	The table shows the Pearson correlation (R^2) between pairwise dissimilarities of raw time series measured according to different dissimilarity measures on different sub-sets of data. In the second part of the table is shown the correlation between pairwise dissimilarities of raw time series and their latent representation according to a DissPCE with 10 layers and latent dimensionality equal to $2 \times \log_2(L)$, where L is the time series length.	94
7.1	The table shows the AUROC, rounded to the nearest integer, for all the 86 data-sets introduced in Section 3.2. Cells get darker as the AUROC increases. Also, the table shows the CC, and the CR allowed by each representation. L is the time series length and l is the dimensionality of the representation. . . .	109
7.2	The table shows the CC to represent, and to classify with the considered approaches and a 1NN classifier equipped with the ED. L is the time series length, l is the dimensionality of the representation, and N is the number of training samples. . . .	115
7.3	The table shows AUROC averaged across all the data-sets and rounded to the nearest integer.	117
7.4	The table shows AUROC averaged across all the data-sets and rounded to the nearest integer. L is the time series length. . . .	119

Part I

Background

Chapter 1

Introduction

The aim of the present research project is to advance the field of time series classification with a focus on one-class classification. As elaborated below, the emphasis is not placed on specific classification algorithms, but rather on data representation.

This chapter is organised as follows. In Section 1.1, we explain why our research has several important practical applications. In Section 1.2 and 1.3 we provide some background about the core topics of our study. In Section 1.4, we list our scientific contributions.

1.1 Why is this Research Important?

There are many sources of unstructured digital data such as emails, mobile phones, online transactions, sensors, and social networks. New data is constantly generated resulting in massive volumes and varieties that are difficult to manage with the technologies currently available to most organisations. At the same time, organisations are in need of data for the purpose of knowledge extraction, intelligent and real-time decision making, and complex problem solving that can deliver consistent economic value. Data has become a valuable asset, however it is necessary to have the ability to analyse and make sense of this resource in order to address any useful task.

The goal of machine learning algorithms [38, 77, 124] is to automatically learn the characteristics of data and use them for future predictions with minimal human intervention. In the context of large volumes of data, machine learning algorithms require (1) efficient learning strategies, and (2) data representations. In this study we address both these aspects, and in doing so we focus on a specific data type, (3) time series. We now consider these three points in turn.

(1) In several real-world scenarios it is impossible to manually annotate data-sets with class labels due to their large scale, or other reasons. However, partially labelled data may still be useful. Techniques that leverage such data are referred to as semi-supervised learners [221]. These approaches combine aspects of both supervised and unsupervised learning. As most real-world data-sets are unlabelled and it is expensive or time consuming to label all data using domain experts, these approaches are in high demand. Many semi-supervised learning techniques have been proposed in the literature, however in this work we focus on one-class classification. As detailed below, the main assumption of one-class classification [126], closely related to the field of anomaly detection [30], is that training data belong to a single class.

(2) Data representation is a core issue in pattern recognition [61]. The representation is the way real-world objects and phenomena are numerically described such that they can be related to each other in some mathematical framework that allows learning and generalisation [58]. Data representation is crucial because it can dramatically impact the machine learning algorithm performance as well as the computational complexity required to solve a particular problem, and the interpretability of the solutions found.

(3) Time series are ubiquitous as they originate from any dynamic phenomenon whose state can be evaluated at different points in time. For instance, the CPU usage of a computer recorded at a certain frequency for one hour is a time series. However, although in our example we are tracking a single variable (the CPU usage), the dimensionality of a sample can escalate easily as we change the frequency parameter e.g. from minutes to seconds. The main characteristic of time series is that, in contrast to mere vectors, nearby values are correlated. In other words, past states of the considered phenomenon can be used to understand the present state, or forecast future ones. This has important implications for machine learning algorithms which need to account for the temporal ordering of time series for best results.

In summary, *why is this research important?* This research is important because it tackles real-world issues concerning the deployment of machine learning algorithms at scale (1), (2). Furthermore, our work deals with time series (3), a data type that is ubiquitous to any dynamic phenomena, and as per its intrinsic characteristics requires ad-hoc solutions.

1.2 One-Class Classification

There is a vast range of research topics concerning time series. In this study we focus on the field of time series classification, and representation. Specifically, as motivated by our review of related works, we focus on the one-class

time series classification problem that is significantly under-explored. In this section we explain why this is an interesting problem.

In machine learning, classification is the problem of learning a mapping function that assigns an object to the correct class. In the classical formulation, the learning process is said to be supervised in the sense that the learner is provided with a set of labelled training data. In these data each class is expected to be adequately sampled with a number of samples per class that is related to the complexity, and the dimensionality of the problem at hand.

These requirements of supervised classification are often difficult to satisfy in real-world applications. This is because it may be difficult or impossible to sample certain classes of a given problem. For instance, in the context of network security [29, 198] there may not be known examples of network intrusion, or there may be some but not representative of all types of intrusion, or again checking whether a suspicious network behaviour is related to an intrusion or not could require up to several hours of expert analysis with a consequent increase in terms of costs necessary to gather labelled data.

When dealing with problems where only the data related to a single class is available at training time, or when we have data from multiple classes but we expect to see data from novel classes at prediction time, one-class classification may be the right learning framework to use. In one-class classification a classifier is trained using only the samples of a single class. In other words, the goal is to learn a concept by only using examples of the same concept [102]. As a matter of fact, to distinguish an apple from another type of fruit humans do not need to be trained on all the types of fruit available on the planet. It is sufficient to see a few examples of the class apple to learn what is an apple and separate it from what is not.

Finally, another reason why we are interested in the one-class time series classification problem is its close relation to the anomaly detection problem. The goal of anomaly detection is to identify objects which do not conform to the expected (normal) behaviour [30]. It is complex, if not impossible, to answer the question *what is anomalous?* by enumerating all the possible ways in which an object could diverge from its expected pattern. It seems easier to answer the question by saying that whatever is not normal is anomalous. Thus, in both one-class classification and anomaly detection we have *only* data of a single (normal) class, and we do not know if novel (anomalous) classes may appear at prediction time or what they would look like.

1.3 Time Series Representations

Throughout classification, representation techniques are used to improve classification performance, to lower computational complexity, or to increase model interpretability. The latter two objectives sometimes result in lower predictive power. However, having models that have low computational complexity, or that readily offer insights in the relationships they have learned may be of primary importance for certain practical applications.

Well-established techniques (e.g. principal component analysis) which do not consider the temporal ordering inherent to time series produce representations that enable poor classification performance [53]. In this study, we propose three representation techniques defined to take account of the temporal information contained in time series. Besides that, our primary concern is to develop representations that can be useful in the one-class scenario. Furthermore, the techniques we propose achieve dimensionality reduction, and make both the problem and solution easier to understand.

We propose and thoroughly investigate three time series representations: (1) dissimilarity-based representations, (2) feature-based representations via grammatical evolution [185], and (3) auto-encoder-based representations.

(1) Dissimilarity-based representations transform a time series into a vector where each coordinate corresponds to its dissimilarity from some other time series. By selecting dissimilarity measures defined for time series it is possible to incorporate the temporal information into this representation.

(2) Feature-based representations via grammatical evolution transform a time series into a vector where each coordinate is the output of a function comprised of simple building blocks (e.g. mean, standard deviation, etc.) applied to a specific sub-sequence of the time series itself. As per its mechanisms, the algorithm selects the functions and sub-sequences which enable best classification performance. As pointed out by other studies [258], sub-sequences are key to the time series classification problem.

(3) The auto-encoder is a neural network designed to compress the input into a lower-dimensional space with minimal loss of information when the input is mapped back to the original space. In this case, we add a new constraint: we require the pairwise dissimilarities between time series in the lower-dimensional space to be equal to their values as measured by a dissimilarity measure in the original space. Thus, by selecting dissimilarity measures specifically defined for time series it is possible to incorporate the temporal information into this representation. In this way, we do not only learn a new representation, but also we learn how to approximate the dissimilarity measure used to derive the representation.

1.4 Scientific Contributions

Our contributions are grounded in novel ideas on one-class time series classification and representation. As detailed below, we evaluate a wide range of techniques. To adequately cover the variety and complexity of time series data we test each approach on a set of 86 time series classification problems.

In **Chapter 2**, we provide a broad review of the state of the art in time series classification. Furthermore, we expand the discussion to time series data mining with a specific focus on anomaly detection. Thus, this chapter can be considered as a knowledge repository on time series, and one of the most comprehensive discussions available in the field.

In **Chapter 3**, we provide some background on one-class time series classification, and we detail the experimental design common across all the experimental chapters. Along with the GitHub repository where our code is publicly available, this can be considered as a resource for researchers who want to study the one-class time series classification problem. Also, we introduce a proprietary data-set related to a biometric subject authentication problem where the objective is to confirm the identity of a person through the accelerometer time series generated by a wrist worn device. This problem can be seen as a motivational case study for the entire research project.

In **Chapter 4**, we evaluate the performance of a nearest neighbour classifier paired with an extensive set of dissimilarity measures on raw time series. This is a fundamental baseline for research on one-class time series classification that is here introduced for the first time. However, the more important contribution of this chapter is that for the first time in the one-class scenario, we evaluate and discuss a thorough set of methods to derive dissimilarity-based representations. This chapter is based on work published in the Elsevier Pattern Recognition Journal [170].

In **Chapter 5**, for the first time in the one-class scenario, we propose a feature extraction framework based on grammatical evolution. In the field of evolutionary computation there is a lack of consistent comparison of feature extraction methods for time series with relevant benchmarks. We address this gap, not only by testing our approach on a large set of problems, but also comparing our performance with that of several benchmark methods. Also, as the analysis of the literature reveals, it is not clear how to evolve multiple features that are not redundant, a problem we tackle through our fitness function. Finally, we use this technique, and a subject authentication case study to point out that features extracted in a one-class classification scenario are able to generalise to classes unseen during the feature extraction phase. This chapter is based on work published at the 7th International Conference on Meta-Heuristics and Nature Inspired Computing [169].

In **Chapter 6**, we combine the non-linear dimensionality reduction power of auto-encoders with elastic dissimilarity measures which are of central importance to the time series classification problem. We demonstrate how our approach can be used to reduce the dimensionality of raw time series while preserving pairwise dissimilarities between samples with respect to a measure of choice. This work is part of an ongoing journal paper draft.

In **Chapter 7**, we briefly compare and combine the different representations defined before. Finally, in **Chapter 8** we draw our conclusions about the overall research project, we summarise limitations, and highlight potential directions for future work.

Chapter 2

Related Work

In this chapter we discuss the scientific literature relevant to our research. After a brief introduction to the broad field of time series mining in Section 2.1, we turn our focus towards the time series classification problem examined in Section 2.2. We mention the problem of intra-class variance that motivates the need for elastic dissimilarity measures for time series in Section 2.2.1. The one-class classification problem, a central objective of our research, is considered in Section 2.2.2. We discuss the relation between one-class classification and anomaly detection in Section 2.2.3.

The rest of the chapter provides the background that relates each of our experiments to the corresponding area of research. The literature related to Chapter 4, where we investigate dissimilarity-based representations (DBR), is reviewed in Section 2.4.1. The literature related to Chapter 5, where we develop an evolutionary algorithm for feature extraction, is reviewed in Section 2.4.2. The literature related to Chapter 6, where we investigate auto-encoder-based representations, is reviewed in Section 2.3.1.

This chapter provides a broad discussion on time series classification with emphasis on the one-class assumption. To the best of our knowledge, there are very few works considering one-class time series classification. We hope that our study will motivate several directions for future research.

2.1 Time Series Mining

A time series is an ordered collection of measurements of a variable *usually* at equally spaced time intervals. In most cases the ordering is temporal, however in some applications the ordering is spatial [47]. Time series are common across every scientific field including: astronomy, biology, economics, finance, medicine, music, operational research, robotics, and telecommunica-

tions. Time series research has a history of at least 350 years [129]. Forecasting is often considered the main focus [48], however there are several other research avenues among which we list the main ones below.

Several time series mining tasks are related. For example, while the present research project focuses on time series classification, and representation, in future research some of the techniques we propose could be applied to anomaly detection, or clustering tasks. Again, *matrix profile* [260], arguably the most versatile algorithm in the field, can be used for several time series mining tasks: anomaly detection, classification, clustering, motif and rule discovery, segmentation, visualisation, and more.

- **Analysis:** studies the behaviour of a time series as a function of time [22] e.g. period, seasonality, spectrum, trend. This is a fundamental task in time series mining. The aim is to understand the past, and predict the future behaviour of any dynamic phenomenon.
- **Anomaly detection:** aims at finding patterns that exhibit a different behaviour than what is expected according to a base model [31].
- **Classification:** aims at finding a mapping function that, given a time series, it assigns it to one of a set of classes [84].
- **Clustering:** aims at identifying homogeneous groups in a collection of unlabelled time series [149].
- **Dissimilarity measures:** are functions defined to evaluate the dissimilarity between two time series. They are fundamental for a variety of time series mining tasks [2].
- **Forecasting:** concerns the prediction of future values, or characteristics (e.g. the trend) of a time series [48].
- **Indexing:** is the task of retrieving the most similar time series in a database given a query time series or sub-sequence [66].
- **Motif discovery:** concerns the detection of repeated patterns within a time series, or a collection of time series [159].
- **Pre-processing:** concerns a variety of techniques needed to deal with noise [140], missing values [177], and time series of different length [234].
- **Regression:** can be seen as a generalisation of forecasting in the sense that the target variable may belong or not the time series [233].
- **Representation:** concerns techniques that transform time series in order to ease, in some regards, a mining task [20].
- **Rule discovery:** is the problem of finding rules relating patterns within a time series to other patterns in the same time series, or to patterns in a different time series [44]. For instance, the relations between the channels of a multivariate time series can be modelled as a directed graph [257].
- **Segmentation:** focuses on how to segment a time series with minimum

loss in respect of a certain loss function [163]. A notable application is the detection of (often anomalous) events referred to as change-points [9].

- **Summarisation:** concerns techniques that provide a natural language description of time series to facilitate mining tasks [110].
- **Visualisation:** concerns the graphic representation of time series to facilitate mining tasks [78].

2.2 Time Series Classification

Time series are ubiquitous as they originate from any dynamic phenomenon whose state can be evaluated at different points in time. Typically, classification is the task of assigning a time series to one of a set of classes. Time series classification is a machine learning task with several practical applications as demonstrated by the variety of real-world problems included in the UCR/UEA archive [46] (Section 3.2.1). This is an archive of 128 data-sets that has proved essential for the progress of the field [12].

Over time, an extensive literature has developed on binary and multi-class time series classification [12]. However, we have found very few works that investigate one-class time series classification [170, 256] (Section 2.2.2).

The shape of a time series in terms of its peaks and troughs, ascending or descending slopes, is of fundamental importance to classify a time series. This is in line with the large research effort that is posed towards learning dissimilarity (distance) functions able to take into account invariance of transformations in the time-axis (Section 2.2.1) as per the widely used dynamic time warping (DTW) [205]. By treating a time series as a mere vector, the modeller is likely to discard information needed to discriminate between different classes. While the order-dependent nature of time series limits the performance of off-the-shelf classifiers unable to detect sequential structures, it also fuels the rise of several dedicated ones [12] whose usefulness is sometimes demonstrated only on just a limited number of cases [46].

The field of time series classification heavily relies on the k -nearest neighbour (k NN) classifier which is a simple and effective algorithm. For instance, the hierarchical vote collective of transformation-based ensembles (HIVE-COTE) [155], one of the best performing algorithms [12, 204] when evaluated on the data-sets of the UCR/UEA archive, relies on the 1-nearest neighbour classifier (1NN) for some of its components. Again, several works in the literature consider the 1NN classifier equipped with DTW as the standard benchmark method [12]. However, as a *lazy learner* [80], the nearest neighbour classifier does not have any training phase and it needs to scan through all the training samples each time we want to classify a new time

series. Moreover, some hyper-parameters like the neighbourhood size k , or the warping window size for the DTW dissimilarity, are usually selected by cross validation [235] further increasing computational complexity.

We identify two main approaches to time series classification: (1) instance-based methods (Section 2.3), and (2) feature-based methods (Section 2.4). (1) Instance-based methods require the whole time series to classify. The main example of instance-based classification is the k NN classifier equipped with a dissimilarity measure of choice. We also include in this category deep-learning methods (Section 2.3.1) that, for the most part, require the whole time series to classify [70]. (2) Feature-based methods represent time series in a vector space where therefore classification is carried out. A simple approach is to summarise a time series in a vector using its statistical moments e.g. mean, standard deviation, etc. Other researches, quite rightly, provide different ways to categorise approaches to time series classification [2, 12, 70]. Our idea is to separate approaches that learn a new representation of the time series (2), from those that not (1). Representations with good descriptive power (i.e. that allow high classification performance), allow us to discard raw time series and retain only their feature-based representation. This can offer substantial advantages, in terms of computational complexity, that can be fundamental to certain applications like classification in embedded devices that have limited computational and memory resources [21].

In general, our research efforts are directed towards feature-based representations for time series. Nevertheless, some of the methods we propose attempt to bridge instance and feature-based approaches. For instance, in Chapter 4 we represent time series as vectors of features where each coordinate corresponds to the dissimilarity of a time series from another time series of a set of carefully selected *prototypes*. In Chapter 6, we embed time series in a vector space where they are required to preserve their mutual dissimilarity as it is in the original space. Both methods transform time series in feature vectors but they try to preserve the neighbourhood structure of data that is fundamental to instance-based classification with the NN classifier.

The scope of our research is limited to offline univariate time series classification. While this is the most common setup for time series classification [12], there are also other variants. In the early time series classification problem [213] the goal is to classify a streamed time series as soon as possible. This approach is common in fault detection applications [239]. In ordinal time series classification, a relatively unexplored problem [95], the goal is to classify time series with ordinal labels. When at each time step we observe multiple variables we have a multivariate time series. Classification of multivariate time series is used in applications like activity recognition from wearable sensors [139], or electroencephalography (EEG) signal classi-

fication [227]. The most basic approach to deal with multivariate time series is to combine all the variables in a single one, thus obtaining an univariate time series [168]. However, there are several algorithms that first introduced for univariate time series have later been extended to the multivariate case: multivariate shapelets [85], multivariate DTW [91], etc.

2.2.1 Intra-Class Variance

Following the taxonomy proposed by Batista et al. [15], in Figure 2.1 we illustrate the problem of time series intra-class variance. The different types of intra-class variance are detailed in the figure caption.

Intra-class variance can be easily understood by borrowing an example from the computer vision domain [225]. An object, for instance a cup, can be rotated, or scaled, or translated but still is a cup. Similarly, time series can be affected by a number of transformations that make difficult to recognise that seemingly different samples actually belong to the same class. Conversely, in some cases class membership can be “easily” determined when invariance to such transformations is achieved. For instance, Bagnall et al. [12] point out that some authors report an error rate of 17% on the “Coffee” data-set (UCR/UEA archive) using a 1NN-DTW classifier. However, the error rate drops to 0% when time series are z-normalised [205].

All the data-sets of UCR/UEA archive are pre-processed to be invariant to most of the transformations listed in Figure 2.1 and this is not necessarily a good thing. For instance, while it is true that shape dissimilarity between time series can be better evaluated if all samples are z-normalised, it is possible that by removing mean, and standard deviation the modeller is omitting crucial information for the classification problem at hand. We believe that further research is needed to better evaluate the impact of pre-processing techniques on the performance of time series classifiers.

2.2.2 One-Class Classification

One-class classification [126], closely related to anomaly detection [30], is concerned with learning a classifier when only the data of a single class is available at training time, and/or at prediction time we might be exposed to classes that are unknown at training time. Usually, the class we learn is referred to as the positive (normal) while all the samples that fall outside it are allocated to a “generic” negative (anomalous) class. Some variants of this problem exploits a mix of positive and unlabeled data (PU learning) [247].

Figure 2.2 is designed to demonstrate the usefulness of the one-class assumption. In part (a), a binary classifier trained to separate positive/negative

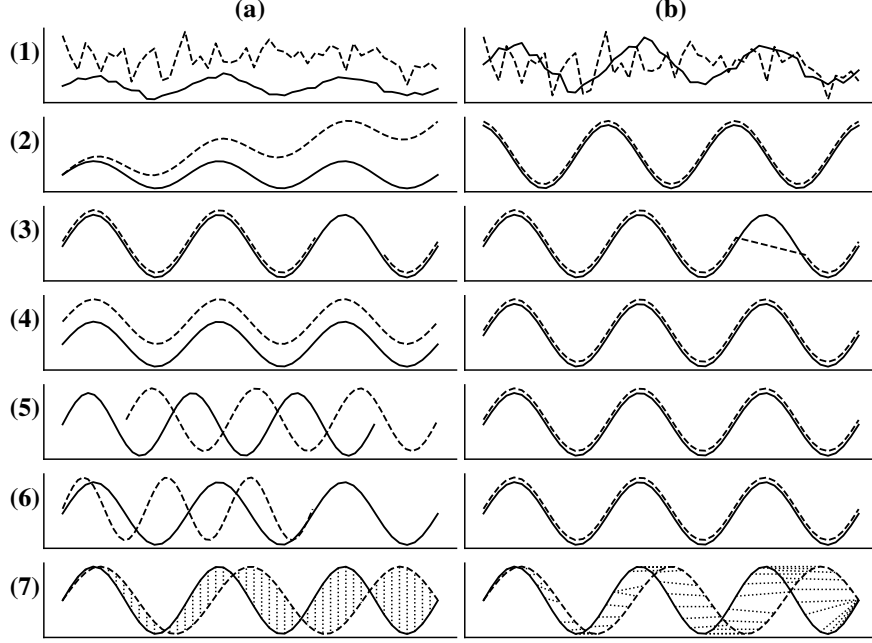


Figure 2.1: Time series intra-class variance. The solid line (—), and the dashed line (---) are used to show two time series that belong to the same class. However, in part (a) class membership is not immediately recognisable, unless an appropriate transform is applied as in part (b). (1) Amplitude invariance can be achieved by subtracting the mean, and dividing by the standard deviation (z-normalisation). (2) Linear trend invariance can be achieved by de-trending the time series (we use first order differences in our example). (3) Occlusion invariance can be achieved by filling the missing values (we use linear interpolation in our example). (4) Offset invariance can be achieved by subtracting the mean. (5) Phase invariance can be achieved by translating along the time-axis. (6) Uniform scaling invariance can be achieved by scaling the temporal index by a certain factor (1.5 in our example). (7) Warping invariance can be achieved by locally warping one time series to align with the other. In part (a), the dotted lines (\cdots) show the alignment between the two time series as they were compared through a lock-step measure [250] (e.g. Euclidean distance). In part (b), the dotted lines (\cdots) show the alignment between the two time series as they were compared through an algorithm defined to find the alignment that has minimum cumulative absolute difference (e.g. DTW).

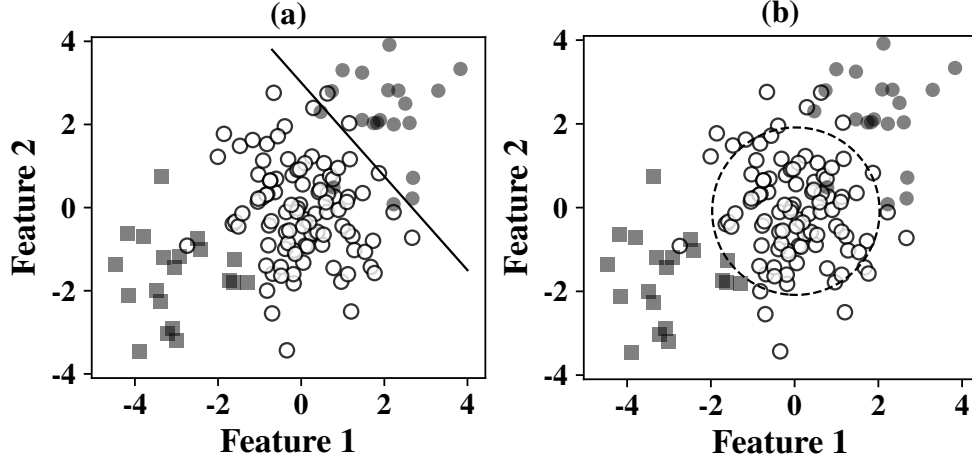


Figure 2.2: Binary vs one-class classification. In part (a) a binary classifier (—) trained to separate positive samples from the negative ones (\circ/\bullet) fails when a new class of negative samples (\blacksquare) appears at prediction time. In part (b) a one-class classifier (---) trained on positive samples (\circ) only is robust against negative classes either known or not (\bullet/\blacksquare) at training time.

samples fails when a new class of negative samples appears at prediction time. The same reasoning can be extended to a multi-class classifier. Conversely, in part (b) a one-class classifier trained on positive samples only is robust to negative classes either they are known or not at training time.

There are several real world applications where data from the negative class are difficult or impossible to obtain. For instance, in biometric subject authentication problems [168] it is relatively easy to collect some data from the “target subject” class, however it is less straightforward to collect data that are representative of the “not target subject” class. Other related applications are: fraud detection [197], machine fault detection [223], medical diagnosis [215], network intrusion [29], and signature verification [94].

The solutions to the one-class classification problem can be divided into three main categories: (1) boundary methods, (2) density methods, and (3) reconstruction methods [133]. (1) A classifier can learn a boundary that encloses the training data and allocate objects that fall outside the boundary to the negative class [237]. In this group can be found purely one-class learning algorithms such as support vector data description [237] or one-class support vector machines (OC-SVM) [217] but also the k NN classifier [125]. In this case, k NN considers whether the distance to the nearest training point exceeds a threshold. (2) Alternatively, a classifier can estimate the density of the training data and allocate objects that fall in low-density regions to

the negative class. For instance, this can be done using Gaussian-mixture models, or hidden Markov models. (3) Lastly, a classifier can learn a model to compress and reconstruct a particular class of data. The idea is that objects that are poorly reconstructed according to the estimated model do not belong to the positive class. This is the case when using auto-encoders, and variational auto-encoders for one-class classification [29].

2.2.3 Anomaly Detection

Over time, an extensive literature has developed on the topic of time series anomaly detection including a number of excellent surveys [30, 41, 73, 96].

The way anomalies are modelled is crucial to every anomaly detection technique [89]. With respect to time series, if an anomaly is limited to a single time stamp this is referred to as a *point anomaly*. When an anomaly concerns a set of consecutive time stamps this is referred to as a *pattern anomaly* [36]. An anomaly may be modelled as a change in one or more key properties of a time series e.g. a shift of the mean value: this type of anomaly is known as a *change-point*. In general, the study of anomalies concerning values within a single time series is referred to as *within-signal* anomaly detection, a well-studied problem [31, 105, 259].

Alternatively, as in the present research, we might want to detect an anomalous time series with respect to a collection of time series: a variant of the time series classification problem [170]. This problem is known as *between-signal* anomaly detection, and it is relatively understudied [170].

The taxonomy of anomalies may be extended [120], and in general most authors attempt to provide their own. There is not a single algorithm able to detect all types of anomaly, neither it is obvious that e.g. a point anomaly is of interest to every practical application. Thus, the type of anomaly we aim to detect is inherently connected to the problem at hand and it impacts the design of the anomaly detection technique [35, 89]. In addition, the availability of labelled data, and the performance requirements of the problem at hand are other important factors to consider [30, 65, 89]. Finally, often it is not only needed to detect anomalies, but also to understand the causal relationships between variables that have led to the anomaly in order to deploy optimization, and fault management strategies [131].

According to a widely accepted definition anomalies are samples inconsistent with the rest of the data [92]. This inconsistency may not only concern anomalies and normal data; it may also be impossible to relate anomalies between them violating the clustering assumption required by supervised learning [220]. Thus, Chandola et al. [30] propose to learn from the normal data only, and to flag as anomalous everything that deviate from that.

Learning from a single class of data is a semi-supervised classification problem known as one-class classification [170]. This approach is relatively understudied while it is more common to tackle anomaly detection as an unsupervised problem. Unsupervised approaches are attractive because they allow to avoid the costs required to acquire labelled data. However, most unsupervised algorithms fails to adapt to previously unseen, but normal, data which are recognised as anomalies resulting in high false positive rates [36]. Thus, assuming there is enough data to approximate the true distributions of normal/anomalous patterns, than supervised approaches may be preferred. For instance, some network intrusion models detect anomalies using known attack patterns called *signature attacks* [34]. While this idea works against known attacks, it is weak in adversarial settings where new attack strategies are constantly developed [34]. Accounting of these considerations the active learning paradigm represents a way to balance the need for labelled data, and adaptability, while cutting the connected labelling costs [193].

A common assumption of several applications is that anomalies are rare in comparison to normal samples [30], thus the need to learn from imbalanced data [180]. Considering that the normal behaviour of a time series may change over time adaptive algorithms are required [6]. In this regard, streaming applications not only must be able to adapt to changing conditions, but also have to operate with little computational complexity in order to cope with the high velocity, and volume of streaming data [54].

Again, emerging internet of things (IoT) technologies are in need of low computational complexity anomaly detection techniques. According to the envisioned IoT revolution, in the near future all the electronic devices are going to be equipped with sensors/actuators and to be able to interact with each other [10]. Important applications in the healthcare domain aim to develop wearable devices for remote health monitoring, for instance, to monitor blood glucose levels [156], or for cardiac disease prevention [241]. Small (wearable) electronic devices have limited computational, memory, and power resources. As discussed throughout the thesis, to ease computational complexity is part of the scope of the one-class classification techniques we propose. Therefore, future research could extend our work to the anomaly detection domain.

Concluding, the case of univariate time series anomaly detection is arguably simpler to understand and to deal with than the multivariate counterpart [35]. For instance, given a univariate time series, a value statistically different from all the others can be classified as anomalous. The same idea cannot be so easily translated to multivariate data. In fact, the behaviour of a single time series may be uninformative if not analyzed in light of the relations that exist between all the time series of the multivariate signal [35]. Also, the amount of literature concerning multivariate time series is much

less than that regarding univariate time series. As an example, comparative studies investigating pre-processing techniques [140], or strategies to deal with missing values [234] have focused on univariate time series only so far.

2.3 Instance-Based Classification

In this section we provide an overview of instance-based time series classification. As techniques based on neural networks, part of this group, also concern one experimental chapter (Chapter 6) they are discussed in a dedicated section (Section 2.3.1).

- **Description:** This is a diverse group of approaches that require the whole time series to work and do not learn a new representation.
- **Pros:** Some instance-based approaches e.g. those that base their decision on the comparison of time series (or sub-sequences) through a dissimilarity measure of choice can facilitate problem and solution interpretability, and perform well with a limited amount of training data.
- **Cons:** Although several studies aim at speeding up existing algorithms computational complexity is arguably the main weakness. Secondly, most instance-based approaches do not learn a new representation, that as discussed in Section 2.4 can lead to several advantages.

One of the most common approaches to time series classification is to compare time series through a suitable dissimilarity measure, hence a time series is assigned to the class to which it is most similar. In general, the highest classification performance is allowed by those measures that are able to compensate for distortions in the time axis. Nevertheless, there are many other aspects to be considered. In this section we do not discuss the details of the different measures proposed in the literature. However, throughout the thesis we use several measures that are presented in Section 3.5. Furthermore, Abanda et al. [2] present a comprehensive survey on dissimilarity-based time series classification, while Paparrizos et al. [187] propose an excellent comparative study of 71 dissimilarity measures.

It is common to pair a dissimilarity measure of choice with the 1NN classifier, a simple and effective approach [12]. The 1NN classifier is a non-probabilistic classifier able to work well with a limited amount of training data. For instance, the 1NN classifier is able to achieve competitive performance on several problems where the average number of training samples per class is lower or equal to 10 regardless of time series lengths of hundreds or thousands of time steps [12]. This is a remarkable difference with respect

to modern machine learning algorithms (e.g. neural networks) that are expected to have poor generalisation capabilities with limited amounts of training data [16]. In other cases, researchers try to adapt existing kernel-based learning algorithms to time series classification, for instance by developing methods that first align time series using DTW and then calculate the value of the kernel function over the warping path [222].

Several state of the art algorithms for time series classification ensemble a variety of dissimilarity measures that together perform significantly better than any single measure. Lines et al. [154] ensemble 11 dissimilarity measures in an algorithm that is a core component of HIVE-COTE [155], one of the best classifiers when evaluated on the UCR/UEA archive. Tan et al. [235] propose a number of strategies to significantly decrease the training time required by Lines’ ensemble, its main weakness. In another attempt to create a scalable ensemble of dissimilarity measures Lucas et al. [161] propose a tree-ensemble. The authors introduce three novel splitting criteria specifically defined for time series. One of these calculates the dissimilarity of a time series from a randomly selected training sample according to a randomly selected measure. The others draw from dictionary-based approaches, and sub-sequences summary features (Section 2.4). Overall, this tree-based classifier allows better performance than HIVE-COTE.

For some problems class membership depends on specific sub-sequences that may occur anywhere along the time axis. Referred to as “shapelets”, these sub-sequences are maximally representative of a class according to a predefined criterion [258]. The presence of a shapelet within a time series is evaluated by sliding the shapelet along the time series and calculating the dissimilarity at each step. If the minimum dissimilarity found falls within a certain threshold the time series is assigned to the class related to the shapelet. Shapelet-based classification is an important area of research, for instance a shapelet-based representation named shapelet-transform [104] allows state of the art performance. However, the original shapelet-discovery algorithm is intractable for large data-sets, thus the majority of the research related to shapelets focuses on ways to speeding up the discovery phase [109].

Concluding, other approaches to time series classification are based on neural networks [64], discussed in Section 2.3.1, functional data analysis [195], hidden Markov models [248], and self-organising maps [188].

2.3.1 Neural Networks-Based Classification

The unrivalled success of neural networks in fields like computer vision [40], or natural language processing [26] has boosted the research about this family of algorithms in the time series classification domain, where however, most

of the progress has been made only in the last three years [71].

Fawaz et al. [70] evaluate several neural networks for supervised time series classification on all the data-sets of the UCR/UEA archive, including the multivariate ones. It is found that a residual neural network achieves the best performance on univariate time series. This architecture, originally proposed by Wang et al. [251], is essentially a convolutional neural network with so-called residual connections that allow the output of a given layer to jump over the upstream layer in order to be added to the input of a further layer. Residual connections, introduced in the computer vision domain, are important to train deep architectures and specifically to avoid the vanishing gradient problem [103]. The second best performing architecture in Fawaz's experiment is also a convolutional network [251]. Convolutional networks have shown to be particularly effective in detecting sequential structures in data [93]. This may explain their success with time series where temporal structures are often essential to identify the class membership. Also, Fawaz et al. [70] point out that convolutional networks can be trained in much less time than dense or recurrent networks, and this despite their suggestion to avoid pooling layers. In fact, it is argued that pooling layers may throw away important information in favour of a reduced model complexity that is not needed for time series that in contrast to images have one less dimension and thus require less computational resources.

Again, Fawaz et al. [71] expand the discussion about convolutional networks proposing an ensemble of inception modules [231] that slightly outperforms HIVE-COTE. The authors analyse the impact of hyper-parameters on performance, while it is difficult to draw general conclusions it is argued that the optimal number of filters or their length may be related to the number of training samples and the time series length respectively. In this regard, Rakhshani et al. [204] implement a neural architecture search algorithm based on differential evolution [202]. Hyper-parameter tuning (number of filters, activation function, etc.) allows the residual network investigated by Fawaz et al. [70] to outperform the state of the art ensemble HIVE-COTE. Not only convolutional networks, but also recurrent networks like those based on long short-term memory units (LSTM) are particularly suited to model sequential data [208]. Karim et al. [118], show that both convolutional and LSTM layers significantly contribute to the performance of their hybrid architecture. A number of studies consider different aspects of time series classification with neural networks like adversarial learning [69], data augmentation [112], few-shot learning [236], metric learning [32], and transfer learning [173]. Despite the recent progress, neural networks research for time series classification is far behind other fields e.g. computer vision [71]. On this point, we suggest that future research may investigate the features learned by the different con-

volitional layers of a network [135], and saliency maps for time series [224].

In the context of time series classification auto-encoders are used less frequently than end-to-end architectures. Typically, auto-encoders are used for unsupervised tasks like anomaly detection [242], dimensionality reduction [86], or clustering [178]. However some authors use of auto-encoders as supervised classifiers e.g. by adding a supervised loss on the middle layer [141], but we have not found an example of this kind about time series.

In contrast to other research areas of the time series classification domain we have not found any comprehensive study on auto-encoders that considers all the data-sets of the UCR/UEA archive. In Chapter 6 we address this gap. We evaluate different architectures, and we use the learned representations in a one-class classification scenario. In particular, we use auto-encoder and encoder-only architectures to learn a low-dimensional representation for time series that preserves samples' dissimilarities according to an elastic measures as for instance DTW. This is a task known as metric learning [56] that, in different ways, other studies have considered before.

Most research on metric learning for time series rely on siamese networks [25]. For instance, Pei et al. [190] use a recurrent network and a supervised loss function based on binary cross-entropy to learn a dissimilarity measure for time series. Again, Zheng et al. [261] use a hybrid network with both convolutional and dense layers, and a loss function inspired by neighbourhood component analysis [210] to learn a time series representation then used with a 1NN classifier. Their results are competitive with those of a 1NN classifier paired with different dissimilarity measures on raw data. Analogously to our approach, Utkin et al. [242] use a siamese auto-encoder to approximate in the learned space the Euclidean distances between time series in original space. They also add the reconstruction error and a contrastive term [98] to the loss function in the context of an anomaly detection problem. There are also some unsupervised approaches. Jansen et al. [115] use a triplet loss [219] and a convolutional network for unsupervised audio representation learning. Qiu et al. [203] use a dense neural network and a ranking loss [171] to approximate a correlation function for time series in the context of information retrieval.

Differently, other studies use the dissimilarity between samples in the learned space to induce a dissimilarity measure on the original space. For instance, Abid et al. [4] use an LSTM-based auto-encoder to reduce the dimensionality of time series. Then, the authors optimise the parameters of a warping function on raw time series in order to approximate their Euclidean distance in the learned space. Che et al. [32] use a dense network and a large margin approach to metric learning [253] to learn a representation for multivariate time series. Then, the authors use the squared Euclidean distance

between samples in the learned space to fit a custom warping function that can be used on raw data.

Concluding, other studies are closely related to our work, but do not use neural networks. Lods et al. [158] propose an algorithm based on stochastic gradient descent to embed time series in a shapelet-transform [104] vector space where the Euclidean distance between samples approximate their DTW dissimilarity in the original space. The authors argue that their algorithm is similar to a siamese network with a single convolutional layer. Although their approach could be used for classification the authors do not consider this task. Lei et al. [143] use matrix factorisation to learn a time series representation such that the dot-product between time series in the learned space approximates their DTW dissimilarity in the original space. However, as other dimensionality reduction techniques (e.g. t-distributed stochastic neighbour embedding (t-SNE) [165]) this approach does not learn a mapping function from the original to the learned space, thus it cannot be used for classification. Finally, Mei et al. [172] first align multivariate time series through DTW and then learn the parametrisation of a Mahalanobis dissimilarity measure.

2.4 Feature-Based Classification

In this section we provide an overview of feature-based time series classification. As DBR and evolutionary computation techniques, both part of this group also concern two experimental chapters (Chapters 4-5) they are discussed in dedicated sections (Sections 2.4.1-2.4.2).

- **Description:** Time series are transformed into feature-vectors which are used to complete the classification task through any off-the-shelf classifier.
- **Pros:** Most feature-based approaches facilitate problem and solution interpretability, and ease computational complexity.
- **Cons:** It is not trivial to identify distinctive features especially when a new problem is addressed. Most feature-based approaches are expected to be weak on problems where features can be shifted along the time axis.

Traditionally, dimensionality reduction can be regarded as the main objective of time series representation techniques [250]. Usually, dimensionality reduction is constrained to preserve some characteristics of a given dataset. For instance, representations that allow lower bounding of a dissimilarity measure are of particular interest in the field of information retrieval. This means that for a certain representation exists a dissimilarity measure in the induced lower dimensional space that approximates the dissimilarity

in the original space. Thus, it is possible to discard part of the comparisons needed to retrieve the nearest neighbour of a given query enabling savings in terms of computational complexity. Techniques like discrete Fourier transform (DFT) [39], and piecewise aggregate approximation (PAA) [122] have this property [250]. However, in Chapter 7 we show that these techniques as well as other general-purpose dimensionality reduction algorithms (i.e. not strictly defined to work with time series) like principal component analysis (PCA) [108], and kernel principal component analysis (KPCA) [216] achieve lower classification performance than the representations we propose. This is because feature-based representations are not only concerned with dimensionality reductions but also with classification performance.

Features are properties of an object that contribute to the way the object appears to the observer [60]; feature extraction aims at finding the most informative set of features for a certain task [97]. The feature-based approach to time series classification is convenient for a variety of reasons [63, 179, 255]. (1) To reduce data to a manageable size. This can mitigate the curse of dimensionality [17], reduce computational complexity, and allow visualisation of time series data-sets. (2) To highlight properties of a class of time series enabling understanding of the problem at hand. (3) To reduce the impact of noise and missing values. (4) To deal with time series of different length.

A feature-based representation affects the choice and the performance of the classifier and vice versa [57]. In fact, the quality of a representation may be overestimated if overfitting occurs, however, it may be underestimated if the classifier relies on assumptions that are violated by the representation (e.g. a normality assumption for a Gaussian-mixture classifier). In practice, expert practitioners first exploit their domain knowledge to extract a set of features. Then, they carry out a statistical analysis of the features before selecting an appropriate classifier. Conversely, the algorithm we propose in Chapter 5 extracts features that are not only suited for the problem at hand, but also for the classifier used in the extraction process.

There are two key questions that arise when dealing with feature-based time series classification. The first concerns the features to extract, and the second the sub-sequence from which to extract them. Neither answer is obvious. The feature extraction process can be manual [168], or automated [51]. Features can be as general as statistical moments, or more carefully designed for time series like the time-reversal asymmetry statistic [218].

Assuming that the most effective features are not known in advance, some authors propose to extract several features and then retain only the best ones. Deng et al. [51] propose a tree-ensemble classifier name time series forest (TSF) (similar to a random forest). In this case tree nodes calculate simple features (mean, variance, and slope) on randomly selected sub-sequences.

Building on this idea, Flynn [76] et al. propose a tree-ensemble classifier named contract random interval spectral ensemble (c-RISE). The difference with TSF is that c-RISE extracts spectral features (power spectrum and auto-correlation), and also it allows the user to constraint the training time.

Lubba et al. [160] consider 4791 features for time series. Of these, they retain only 22 features because of their classification performance and minimal redundancy. This subset named “catch22” (C22) is a compact and effective set of features for time series classification. Middlehurst et al. [174] combine C22 features and TSF in a classifier, referred to as canonical interval forest (CIF), that is better than both. Furthermore, as TSF is a component of the state of the art ensemble HIVE-COTE they have replaced TSF with CIF demonstrating a significant improvement of performance.

Dempster et al. [49] apply thousands of random convolutional filters on time series. Resulting feature maps are summarised through two ad-hoc statistics, then concatenated in vectors, and used for classification. This approach exploits the success of convolutional neural networks for time series classification [70]. Finally, the method relies on the implicit feature selection enabled by a ridge regression classifier in order to avoid overfitting. In contrast, it is not straightforward how to accomplish feature selection in the one-class scenario where there are no labelled data to evaluate the classification performance of a given classifier on a sub-set of features [116]. Future work may investigate one-class feature selection for time series.

Concluding, other approaches to transform time series into vectors of features include dictionary-based methods, and graph features. In dictionary-based methods, also known as bag-of-patterns (BOP) methods [151], time series are transformed into strings using symbolic aggregate approximation (SAX) [150]. Then, each feature corresponds to the frequency of occurrence of a specific sub-sequence (or word) within a time series. Some state of the art algorithms build on the BOP approach, thus the idea is to represent each time series as a histogram that counts the word frequencies, however the word code-book is built using an approach called symbolic Fourier approximation (SFA). This is the case of bag-of-SFA-symbols (BOSS) [211], and the more scalable word extraction for time series classification (WEASEL) [212]. Graph-based techniques first transform a time series into a graph, then features are extracted from this new representation [146]. Nevertheless, it is also possible to draw from the rich literature related to audio features [144].

2.4.1 Dissimilarity-Based Representations

The idea of DBR [191] is to describe one object through a number of pairwise comparisons with the elements of a set of reference objects. Thus, to represent

raw data in terms of dissimilarities it is required to identify a dissimilarity measure, and a prototype set. Hence, an object is represented as a vector where each coordinate corresponds to its dissimilarity to a given prototype.

DBR are used with a variety of data types: graphs [181], images [228], text [199], and time series [119]. In the context of time series classification, DBR can be considered as a hybrid between feature-based, and distance-based approaches. In fact, DBR account for the information about the dissimilarity of a sample with respect to other samples. This is equivalent to a 1NN on raw data when the prototype set includes all training samples.

Kate [119], investigates time series DBR using DTW finding that DBR performance is better than a 1NN-DTW on raw data. Also, the author shows that concatenating DBR, and SAX-BOP [151] representations further improves the performance. Expanding on Kate's work, Giusti et al. [87] investigate several DBR derived from a Cartesian product of four representations, and six dissimilarity measures showing that by concatenating different DBR it is possible to improve on the performance of a 1NN-DTW on raw data.

In terms of prototype selection strategies, Iwana et al. [111] propose to use AdaBoost [214] to select as prototypes those samples that have highest impact on performance. Again, Iwana et al. [113], note that when DTW is used to evaluate the dissimilarity between two time series all the information about the actual matching is wasted as only the sum of absolute differences between DTW-aligned time series is considered. Thus, the authors propose to use the point-wise absolute differences as a new representation. Jain and Spiegel [114] show that reducing the dimensionality of DBR through PCA has a positive impact on performance. However, they do not discuss how PCA could be used to extract a set of prototypes from a training set and this could be the object of future research. Petitjean et al. [194] use DTW barycenter averaging (DTW-BA) to identify class centroids for a nearest-centroid time series classifier showing good results. As discussed in Chapter 4, cluster centroids can be good prototypes. Thus, we argue that the use of DTW-BA in the context of DBR warrants further research. Finally, prototype selection strategies for DBR could draw from the related instance-based learning domain where the objective is to select the smallest set of training samples that enable the same (or higher) performance than the original set [24].

A number of authors have explored interesting variants. Spiegel [68] suggests that using time series that are completely unrelated to the problem at hand as prototypes may lead to improvements on performance. Hills et al. [104] use shapelets as prototypes to derive DBR. Buza et al. [27] use DTW and DBR to classify multivariate EEG time series.

In Chapter 4, we propose a comprehensive experiment on DBR. This is, to the best of our knowledge, the first time that DBR are considered in

the context of one-class time series classification. Not only do we evaluate and discuss a thorough set of methods to derive DBR, but also we introduce a prototype method named *percentiles* that shows remarkable performance. Furthermore, we evaluate the performance of a 1NN classifier paired with an extensive set of dissimilarity measures on raw time series. This is a fundamental baseline for research on one-class time series classification.

2.4.2 Evolutionary Computation Techniques

The majority of prior research concerning evolutionary techniques for time series classification has applied to ECG time series [74], and sensor time series for fault detection [148]. Some studies, discussed below, propose general purpose feature extraction algorithms [63, 101, 246].

Eads et al. [63] use grammatical evolution (GE) [185] to evolve a single population of feature extractors using a set of 25 primitives. Each feature extractor is able to target any sub-sequence and return a single scalar. This hill-climbing algorithm does not make use of any crossover operator. However, it allows modifications to the current solution (addition, deletion, mutation of feature extractors) only if changes increase the performance, or cause a negligible impact on performance but a decrease in run-time. Classification performance, tested on seven data-sets, is better than that of raw data.

Harvey and Todd [101] propose an algorithm based on genetic programming (GP) [132] where 35 primitives are used to evolve sets of feature extractors. Nearly all the primitives take a time series as input and output a transformed time series. To reduce time series to a single scalar each feature extractor must end with a summation. The authors provide a number of rules to reduce redundancy of final feature extractors and increase their interpretability. Performance is tested on simulated data.

Finally, Virgolin et al. [246] propose a GP-based algorithm for sequential feature construction but not on time series. While feature extraction concerns the extraction of features from raw data, feature construction concerns the transformation of existing features. They focus on the interpretability of the GP trees emphasising interpretability of the original features, and limiting the height of the trees. As discussed in Chapter 5, we point out that in the time series domain interpretability is aided by knowing which functions, and sub-sequences allow good classification performance.

The literature reveals a number of gaps that we seek to address in Chapter 5. There is a lack of consistent comparison of proposed methods with relevant benchmarks. We address this issue by evaluating our approach on 30 problems, all but one from the UCR/UEA archive. Also, we compare our results against a 1NN-DTW classifier considered as the standard bench-

mark in the literature [12]. It appears that it is not clear how to evolve multiple features that are not redundant. We tackle this problem by sequentially extracting the features, and requiring that their individual classification performance is maximised, while their average correlation with previous features is minimised. We analyse evolved solutions to expand the discussion on problem/solution interpretability enabled by evolutionary techniques. Finally, our algorithm allows modellers to extract features without requiring any previous knowledge of the field. This is in line with an increasingly popular research area known as automated machine learning (AutoML) [75]. The goal of AutoML is to automate the typical machine learning pipeline (e.g. data pre-processing, feature extraction, model and hyper-parameter selection) reducing the amount of human work required to deploy a model.

Chapter 3

Problem Statement and Experimental Design

The key idea of this research project is to investigate novel time series representations in the context of one-class classification. In the following chapters (Chapters 4, 5, 6) we present three different approaches to derive a time series representation. The objective of this chapter is to gather the background common across the different experimental chapters. This is not only to avoid unnecessary repetition, but also to facilitate understanding.

The chapter is organised as follows. We formalise the concepts of time series and one-class classification in Section 3.1. The data-sets in use are presented in Section 3.2. The classifiers are detailed in Section 3.3, while performance evaluation is discussed in Section 3.4. The dissimilarity measures needed for classification, and in some of the experiments are detailed in Section 3.5. Finally, some implementation details are provided in Section 3.6.

3.1 Main Concepts

In this section we concisely introduce the main concepts underpinning the whole body of work. Further background is provided throughout the thesis.

Time Series - A univariate time series T of length L is a collection of real values ordered according to an index i . Given a time series T , $\forall i : T_i \in \mathbb{R}$, $1 \leq i \leq L$, $T \in \mathbb{R}^L$.

Time Series Representation - Given a time series $T : T \in \mathbb{R}^L$, a new representation is achieved by applying a function \mathcal{R} on T with the objective to ease, in some regards, a certain data-driven task. \mathcal{R} is a mathematical function. \mathcal{R} may leave the dimensionality of T unchanged. However, often

representations are defined to map T to a new space \mathbb{R}^l with $l \ll L$. Finally, $\mathcal{R}(T)$ may preserve the sequential structure inherent to T or not. In other words, $\mathcal{R}(T)$ may still be a time series or not.

One-Class Classification - The main difference between binary and multi-class classification and one-class classification is in the training data. Binary and multi-class methods require labelled training data for all the classes that can occur at prediction time. Conversely, one-class methods are concerned with learning a classifier when all training data belong to a single class. Usually, the class learned through the training data is called the positive (normal) class while all the samples that fall outside it are allocated to a “generic” negative (anomalous) class.

Classes - We denote the positive class as C^{+1} , and the negative class as C^{-1} . Each class is associated with a label $c : c \in \{-1, +1\}$.

One-Class Classifier - A one-class classifier consists of two parts. First, given a time series T , or its representation $\mathcal{R}(T)$, a function \mathcal{F} assigns an anomaly score to it. Then, the identification function \mathcal{I} maps the score to a class label. The identification function returns $+1$ if the condition is true, and -1 otherwise. A common way to set the threshold h is to choose a value such that 95% of training data is correctly labelled as below the threshold, hence positive.

$$\begin{aligned} &\mathcal{I}(\mathcal{F}(T) < h) \text{ or} \\ &\mathcal{I}(\mathcal{F}(\mathcal{R}(T)) < h) \end{aligned}$$

3.2 Experimental Data

All the representations considered in this work (except for the approach discussed in Chapter 5) are evaluated on 86 data-sets. Of these, 85 belong to the UCR/UEA archive [46] (Section 3.2.1) while one is a proprietary data-set (Section 3.2.2). An overview over the different time series included in all the data-sets is shown in Figure 3.3. Nearly of all these data-sets are related to binary and multi-class problems, in Section 3.2.3 we explain how to adapt these data-sets to a one-class classification experiment. The classification performance for a given data-set corresponds to the average performance over all the classes.

3.2.1 The UCR/UEA Archive

The UCR/UEA archive [46] is an important resource for time series researchers. In 2018, the archive was expanded to 128 data-sets, however in this work we consider only its original body of 85 data-sets.

All the data-sets are partitioned into labelled training and test sets and have previously been examined in several binary and multi-class time series classification experiments [12]. All time series are univariate, contain only real numbers and no missing values, have a fixed length within a given data-set, and are z-normalised [205].

The data-sets possess different characteristics that, when needed, allow us to look at results from different angles. In Figure 3.1 it is shown how the data-sets can be divided in terms of application domain, time series length, and average number of training samples per class.

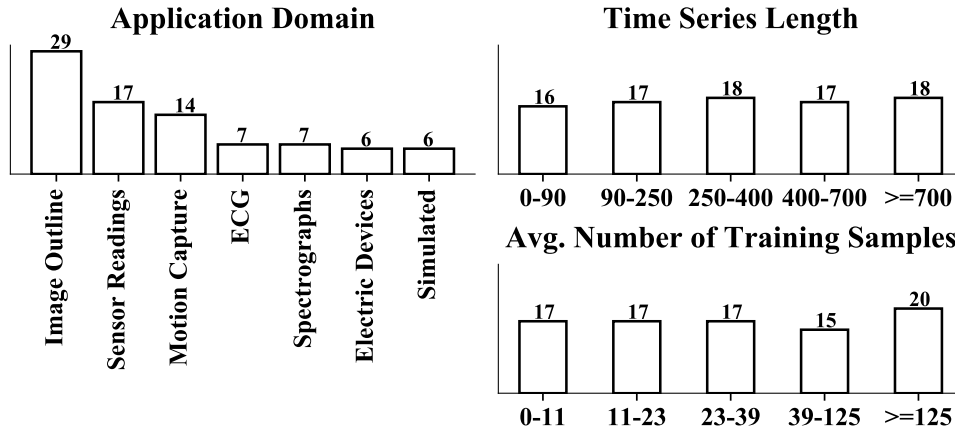


Figure 3.1: Summary of characteristics of the data-sets in use.

3.2.2 Accelerometer Data

The “AccelerometerData” data-set [168] is here introduced for the first time and it is not part of the UCR/UEA archive. The data-set is provided by ICON plc. as part of a research effort investigating subject authentication through accelerometer data. In a way, this application has motivated the whole body of work.

The aim of subject authentication is to confirm the identity of a person. There are several authentication technologies and applications [252]. In our case, clinical scientists would like to use accelerometer data for monitoring the efficacy of treatment options on movement disorders or the impact of

drugs on subjects' free living activity levels [121]. However, first they are in need of classification models to confirm that a given device is worn only by the intended subject for the whole trial period. Both errors and misconduct could invalidate studies of considerable cost and duration.

Accelerometer data come as a sequence of time-ordered real values, thus this problem falls within the time series classification domain: given a set of accelerometer time series we want to separate those that are generated by a specific subject from those that are not. A possibility would be to make use of a *supervised binary classification method* to separate those time series which belong to the intended subject from those which do not. Alternatively, the same data could be employed to implement a *supervised multi-class classification method*. In this case, a group of subjects would be involved simultaneously and the aim would be to recognise each single subject. However, both binary and multi-class methods assume, in a sense, a fixed population of "others", well-represented in the data, which is not realistic. Therefore, the best option may be a *semi-supervised one-class classification method* where learning is focused only on the intended subject.

A sample of nine volunteers (not enrolled in a clinical trial), composed of three females and six males, are each required to wear the mentioned device for a period of approximately 40 days in free living conditions. All participants are office workers based in the same location and working Monday to Friday. We retain only weekdays because weekend days are relatively few. In order to satisfy data privacy requirements of both the data controller (ICON plc.), and University College Dublin data has been anonymised: each time series is associated with its numerical class label only. In principle, it is possible that information like age, gender, or weekday could help in building better classifiers, however this is a topic for future research.

Data is collected through wrist-worn tri-axial accelerometers able to measure linear acceleration within a range of $\pm 16g$ per axis¹. The magnitude of the acceleration along the three axes is calculated as the square root of the sum of the square of the single accelerations and rounded to the closest integer. The time series are indeed a sequence of magnitude values at the resolution of one data point per minute over an entire day (24 hours). For each subject we have a total of 23 time series. Of these, using a random split, 18 are included in the training set, and five in the test set. All the variables recorded are shown in Table 3.1. ID is the subject unique and anonymous identifier. Axis_1,2,3 show acceleration. A graph of magnitude recordings for one typical day is shown in Figure 3.2.

First, data is collected at a frequency of 1000Hz. Then, in order to reduce

¹ For further information see ActiGraph: <http://www.actigraphcorp.com/>.

dimensionality to a manageable size, data is downloaded from the device and down-sampled to a frequency of 1/60Hz using ActiGraph proprietary software based on a band-pass filter. It may happen that a device runs out of memory and stop recordings as the internal memory is of just 4GB. Also, it may happen that some time series predominantly contain “non-wear time” i.e. time when the device is on but it is set down and stationary. We discard time series with a cumulative magnitude lower than 500,000g as these miss most of daily records. In other cases, where time series show a sub-sequence of missing values lower or equal to 90 minutes (presumably the time needed to download the data and recharge the device) we fill the missing sub-sequence with a copy of the values that precede it. This strategy for handling missing data may seem naive but it is adopted for its simplicity and because it is required only two to three times per subject.

ID	Date	Hour	Minute	Axis_1	Axis_2	Axis_3	Magnitude
Subject_1	2016-03-21	11	36	168	544	563	801

Table 3.1: “AccelerometerData” data-set: recorded variables.

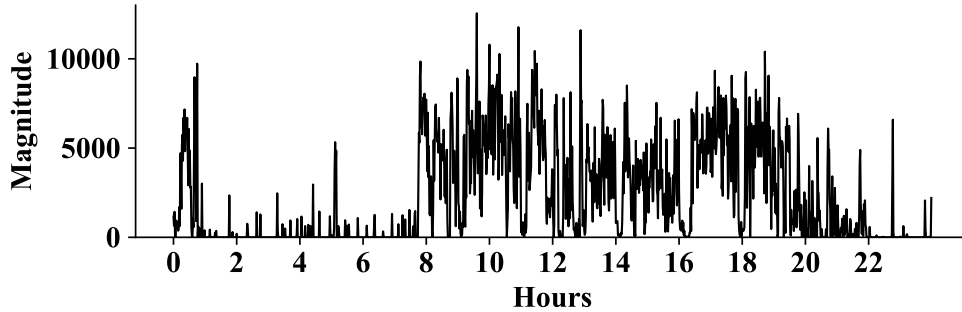


Figure 3.2: Magnitude recordings for one day for one subject.

3.2.3 One-Class Labelling

We adapt labelled binary and multi-class time series data-sets to the one-class classification scenario as follows. Given a data-set, while maintaining the original split between training/test data, each of the classes is considered in turn as the positive class (and so it is used for training) and all the others together become the negative class. The classification performance for a given

data-set corresponds to the average performance over the classes. When a validation set is needed we use a 2:1 split of the training data.

Although it may look like this prediction task suffers from a lack of balanced data, this is not the case. As stated in the extensive survey by Haixiang et al. [99], imbalanced learning occurs when in training data one or more classes have a much greater number of samples than the others. However, in the one-class scenario we are learning a single class.

3.3 Classifiers

We investigate several different representations, but in all cases we use the same classifier on top: a one-class 1-nearest neighbour classifier (1NN). We also use other classifiers as detailed in relevant chapters. The 1NN classifier scores a sample according to its distance from training data. Specifically, the 1NN considers the distance of a sample from its nearest training sample. Unless otherwise specified, the 1NN is equipped with the Euclidean distance.

The core idea is to use a simple non-parametric classifier to emphasise the quality of representations, and make minimal assumptions about class distributions. Furthermore, parametric classifiers are sensitive to hyper-parameters which, in principle, are difficult to tune because we assume only data of a single class are available for training. Thus, in contrast to a typical supervised classification scenario we cannot evaluate a model on a validation set.

3.4 Performance Evaluation

The classification performance of all the representations considered in this work is evaluated through the area under the receiver operating characteristic curve (AUROC). We believe the AUROC is particularly indicated for one-class time series classification experiments since it is insensitive to class imbalance. Furthermore, Gay et al. [83] advocate the use of the AUROC for binary and multi-class time series classification experiments too.

During the prediction phase, a sample is assigned a classification score according to the classifier in use. By imposing a threshold on the scores, a sample can be classified as either positive or negative. As mentioned before, in real-world applications for one-class classification a common way to set the threshold is to choose a value such that 95% of training data is correctly labelled as below the threshold, hence positive. However, this approach can require some fine tuning according to the specific problem at hand. In order

to avoid this and have a method that can be consistently applied across all data-sets and experiments we use the AUROC. The AUROC is obtained by computing the underlying area of a curve constructed by plotting the true positive rate against the false positive rate at various threshold settings. Threshold values are calculated as the midpoint between each pair of sorted classification scores.

3.5 Dissimilarity Measures

In this section we describe 12 dissimilarity measures used in this work. All the measures are collectively presented in Table 3.2. Some of these are true metrics e.g. the Euclidean dissimilarity, others are not e.g. dynamic time warping (DTW) [205]. In general, it is more important that a dissimilarity measure is able to detect the difference between two time series than that it is a true metric.

The first dissimilarity measures we use are derived from the family of ℓ^p norms and they are extensively studied in the time series classification literature where they are usually referred to as lock-step measures [250]. These measures assume both time series have the same length and they compare the i -th point of a time series with the i -th point of another. We evaluate the ℓ_1 (Manhattan), ℓ_2 (Euclidean), and ℓ_∞ (Chebyshev) norms. These measures can be criticised because of their inability to deal with transformations in the time-axis [15], however, they are efficient to compute and easy to interpret.

The cosine dissimilarity (Eq. 3.1) is a lock-step measure, hence is also vulnerable to time-axis transformations. It is used in the time series domain especially for fault diagnosis [162]. This function computes the cosine of the angle between two vectors and can be interpreted as proportional to the angle between the two vectors. The cosine dissimilarity is not a metric because it does not satisfy the triangle inequality. Given two time series T_1 and T_2 the cosine dissimilarity is calculated as follows:

$$\text{Cosine Diss.} = 1 - \frac{T_1 \cdot T_2}{\|T_1\|_2 \|T_2\|_2} \quad (3.1)$$

Some measures are referred to as elastic measures because they allow the comparison of one point of a time series to many points of another in order to find the “best” match [250]. We evaluate three elastic measures: DTW [205], the edit distance on real sequences (EDR) [33], and the move split merge dissimilarity (MSM) [230], which are all dynamic programming algorithms. DTW finds the alignment that minimises the cumulative sum of absolute differences between two time series. EDR counts the number of

times the absolute difference between two time series is greater than a certain threshold. Finally, MSM is similar to DTW, but also it can set a value in a time series as equal to either its preceding, or following value. In this way, the dissimilarity between matched values does not only depend on their absolute difference, but also on the absolute difference between their neighbours.

DTW (Eq. 3.2) is a distance measure of central importance in the literature related to time series mining [205]. It is used to find an optimal alignment between two time series, i.e. the alignment that allows the minimal distance. DTW provides a temporal alignment of two time series and is able to guarantee warping invariance [15], as discussed in Section 2.2.1. Furthermore, DTW can work with two time series of different length. DTW is not a metric because generally it does not satisfy the triangle inequality. To calculate the DTW distance between two time series T_1 and T_2 first we compute the cost matrix C i.e. for each pair of elements $i \in T_1$ and $j \in T_2$ we calculate the cost $C_{i,j} = |T_{1,i} - T_{2,j}|$. The objective is then to find a path through the distance matrix D with minimal cumulative cost by applying the following recurrence:

$$D_{i,j} = C_{i,j} + \min\{D_{i-1,j-1}, D_{i-1,j}, D_{i,j-1}\} \quad (3.2)$$

When the entire matrix D is filled we have $DTW\ Diss. = D_{L_1,L_2}$ (where L_1 and L_2 are the lengths of the two time series).

Generally, researchers use a hyper-parameter named “warping window size” to limit the maximum gap between two matched points; this avoids matching points that are too far apart in time, speeds up computation, and increases classification performance [250]. Unless otherwise motivated, we do not optimise the warping window size, usually selected through cross-validation, because our work focuses on one-class classification so we do not want to rely on data from other classes, as explained a few sections earlier.

Wang et al. [250] show that the performance of EDR (Eq. 3.3) is comparable to that of DTW for time series classification with a 1NN. As with DTW, EDR can work with two time series of different length. The EDR counts the number of times the absolute difference between two time series is greater than a certain threshold. To calculate the EDR, given two time series T_1 and T_2 , we first calculate a cost matrix C as done for DTW. Then we find the the minimum cost alignment by applying the following recurrence to the distance matrix D :

$$D_{i,j} = \min\{D_{i-1,j-1} + \text{penalty}, D_{i-1,j} + 1, D_{i,j-1} + 1\} \quad (3.3)$$

The penalty is equal to 0 if $C_{i,j} < \epsilon$, 1 otherwise. We set the threshold ϵ to $0.25 \times \sigma$ (where σ is the standard deviation of all the training time

series concatenated together) [33]. When the entire matrix D is filled we have $EDR\ Diss. = D_{L_1, L_2}$ (where L_1 and L_2 are the lengths of the two time series). The EDR violates the triangle inequality and so it is not a metric.

The MSM dissimilarity (Eq. 3.4) uses three editing operations (move, split, and merge) to calculate the cost needed to transform one time series into another. As opposed to other elastic measures e.g. DTW or EDR, MSM is a metric. Given two time series T_1 and T_2 , the MSM dissimilarity is calculated by applying the following recurrence on an appropriately initialised matrix D (we omit initialisation steps for brevity):

$$D_{i,j} = \min\{D_{i-1,j-1} + |T_{1,i} - T_{2,j}|, \\ D_{i-1,j} + Cost(T_{1,i}, T_{1,i-1}, T_{2,j}), \\ D_{i,j-1} + Cost(T_{2,j}, T_{1,i}, T_{2,j-1})\} \quad (3.4)$$

where the $Cost$ function is defined as follows:

$$Cost(T_{1,i}, T_{1,i-1}, T_{2,j}) = \begin{cases} c & \text{if } T_{1,i-1} \leq T_{1,i} \leq T_{2,j} \text{ or } T_{1,i-1} \geq T_{1,i} \geq T_{2,j} \\ c + \min\{|T_{1,i} - T_{1,i-1}|, |T_{1,i} - T_{2,j}|\} & \text{otherwise} \end{cases} \quad (3.5)$$

The parameter c corresponds to the cost required to make a split/merge operation (the cost of a move operation is equal to absolute value of the difference between the old and the new value). We set the parameter c as equal to $0.25 \times \sigma$ as done for the EDR dissimilarity. When the entire matrix D is filled we have $MSM\ Diss. = D_{L_1, L_2}$ (where L_1 and L_2 are the lengths of the two time series).

The auto-correlation dissimilarity [79] (Eq. 3.6) measures the correlation between time series values separated by a certain lag. In time series analysis and forecasting it is important to study the auto-correlation structure of a time series. For instance, this information is useful to estimate the parameters of auto-regressive and moving-average models. In this work, we compare the auto-correlation structure of two time series to see if this can result in an effective dissimilarity measure. Given two time series T_1 and T_2 of equal length L we calculate their auto-correlation for all the lags from 1 to $L-1$ obtaining two vectors of auto-correlation values R_1 and R_2 . The auto-correlation dissimilarity corresponds to the Euclidean distance between these two vectors multiplied by a vector of weights W (weights decrease with the lag and are calculated using a geometric progression in the interval $[1, 0.001]$). In their original formulation, the authors propose to use the squared Euclidean distance, however we use the Euclidean distance because

with the former, the auto-correlation dissimilarity is not a metric.

$$\text{Auto-correlation Diss.} = \|(R_1 - R_2) \times W\|_2 \quad (3.6)$$

The Gaussian kernel and the Sigmoid kernel dissimilarities are defined in Eq. 3.7 and Eq. 3.8 respectively. The Gaussian and the Sigmoid kernels measure the similarity between a given pair of inputs; we turn them into dissimilarity measures by a change of sign. These measures are not metrics because they violate the identity, and triangle inequality axioms. There are a number of studies that investigate the use of kernel methods for time series classification [2]. In Eq. 3.7, to calculate the Gaussian kernel dissimilarity we set the scale parameter $2\sigma^2 = L$ (where L is the time series length).

$$\text{Gaussian } k. \text{ Diss.} = -\exp\left(-\frac{\|T_1 - T_2\|_2}{2\sigma^2}\right) \quad (3.7)$$

The Sigmoid kernel dissimilarity (Eq. 3.8) takes two hyper-parameters: a and c . Långkvist et al. [138] explain in detail the implications of these hyper-parameters in the context of kernel methods. When $a > 0$, a can be interpreted as a scaling factor of the input data while c is a threshold that can be used to set the point where the function output is equal to 0. We set $a = 1/L$ (where L is the time series length), and $c = 0$.

$$\text{Sigmoid } k. \text{ Diss.} = -\tanh(a(T_1 \cdot T_2) + c) \quad (3.8)$$

Generally, the Kullback-Leibler dissimilarity (Eq. 3.9) [117] and the Wasserstein dissimilarity (Eq. 3.10) [245] are used to evaluate the difference between two probability distributions. These measures ignore the temporal structure of time series, treating each time series as a random variable. The Kullback-Leibler dissimilarity is not a metric because it violates the identity, symmetry, and triangle inequality axioms. A symmetric version [117] can be derived as follows: $KL_s(p_1, p_2) = KL(p_1, p_2) + KL(p_2, p_1)$.

To calculate the Kullback-Leibler dissimilarity, given two time series, first we merge them and find the bin edges from 10 equal-width bins. Then for each time series we count the bin frequencies p_1 and p_2 . We add 1 to each bin to avoid empty bins.

$$KL_s \text{ Diss.} = \sum_{i=1}^{10} p_{1,i} \log\left(\frac{p_{2,i}}{p_{1,i}}\right) + \sum_{i=1}^{10} p_{2,i} \log\left(\frac{p_{1,i}}{p_{2,i}}\right) \quad (3.9)$$

The first Wasserstein dissimilarity [186] between two one-dimensional distributions (Eq. 3.10), also known as earth mover's distance, considers not

only how different two probability distributions are, but also it takes into account the amount of “work” needed to transform one distribution into another. Adapted to the time series context the Wasserstein dissimilarity is not a metric because it violates the identity axiom. Given two time series we calculate their cumulative histograms, F_1 and F_2 , and a vector of the first order differences between the two time series I .

$$\text{Wasserstein Diss.} = \sum \left(|F_1 - F_2| \times I \right) \quad (3.10)$$

Dissimilarity Measure	Type	Complexity	Metric
Manhattan	Lock-step	$\mathcal{O}(L)$	Yes
Euclidean	Lock-step	$\mathcal{O}(L)$	Yes
Chebyshev	Lock-step	$\mathcal{O}(L)$	Yes
Cosine	Lock-step	$\mathcal{O}(L)$	No
DTW	Elastic	$\mathcal{O}(L^2)$	No
EDR	Elastic	$\mathcal{O}(L^2)$	No
MSM	Elastic	$\mathcal{O}(L^2)$	Yes
Auto-correlation	Statistical	$\mathcal{O}(L^2)$	Yes
Gaussian Kernel	Lock-step	$\mathcal{O}(L)$	No
Sigmoid Kernel	Lock-step	$\mathcal{O}(L)$	No
Kullback-Leibler	Probabilistic	$\mathcal{O}(L)$	No
Wasserstein	Probabilistic	$\mathcal{O}(L \log(L))$	No

Table 3.2: Dissimilarity measures used in this work.

3.6 Implementation Details

All the experiments are implemented in Python. There are three main libraries we use. (1) Scikit-learn [189] is used for the classifiers, some of the dissimilarity measures, and other utilities. (2) PonyGE2 [72] is used to implement the evolutionary algorithm proposed in Chapter 5. (3) Tensorflow [1] is used to implement the neural networks relevant to Chapter 6. The code related to published works is available on GitHub².

Most of the experiments are run on CPU nodes on the University College Dublin NCRA Galapagos server cluster³. The experiments related to convolutional auto-encoders described in Chapter 6 are run on GPU nodes of the DJEI/DES/SFI/HEA Irish Centre for High-End Computing (ICHEC)⁴.

² Code available at: <https://github.com/spaghetix/>.

³ NCRA lab: <http://ncra.ucd.ie/>.

⁴ ICHEC: <https://www.ichec.ie/>.

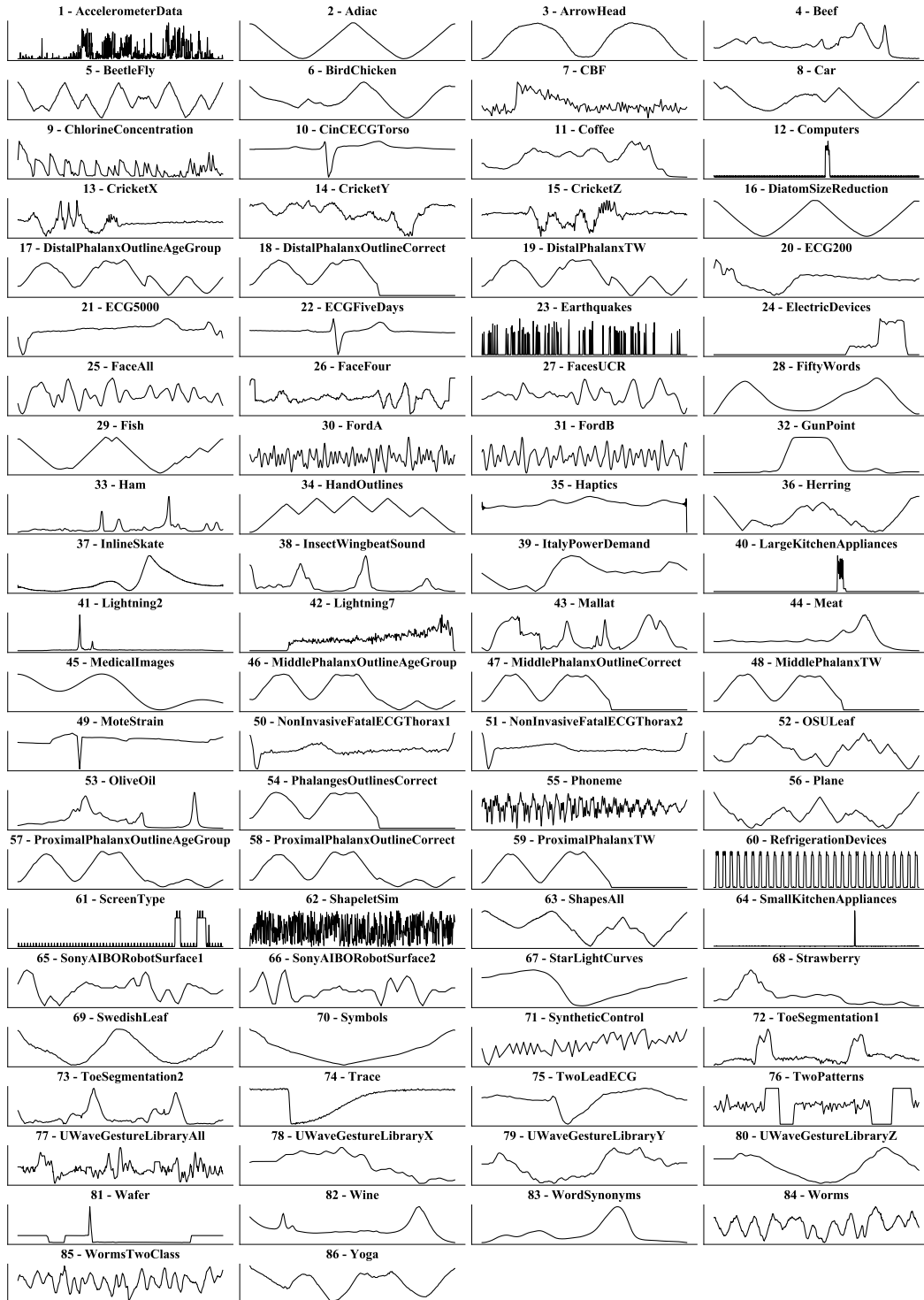


Figure 3.3: A randomly selected time series for each data-set in use.

Part II

Experimental Research

Chapter 4

Dissimilarity-Based Representations

In this chapter we investigate dissimilarity-based representations (DBR) [191] as a means to attain a vectorial representation of time series that, while preserving the information that allows classification, could enable scalable machine learning algorithms [106]. DBR are studied in the context of the little-explored one-class time series classification problem.

Our main contributions can be summarised as follows. As outlined in Chapter 2, there is no research using DBR for time series in the context of one-class classification, and very little using other representations. We address this gap evaluating a variety of DBR derived through a Cartesian product of 12 dissimilarity measures and eight “prototype methods”. Prototype methods are strategies designed to extract a subset of samples (prototypes) from a set of training samples. Subsequently, these prototypes are used along with a dissimilarity measure to derive the DBR. We benchmark the classification performance of DBR against raw data (RD). We find that the performance of DBR and RD is close. However, DBR have an advantage on problems where the class membership depends on the global time series shape. Also, DBR enable dimensionality reduction and so savings in terms of computational time, and visual exploration of time series data-sets.

The chapter is organised as follows. In Section 4.1, we show how to derive DBR of time series. In Section 4.2, we report and analyse our results. Finally, in Section 4.3 we summarise our conclusions and discuss future work.

4.1 Proposed Method

We provide an overview of the proposed method in Section 4.1.1. Then, we present all the considered prototype methods in Section 4.1.2.

4.1.1 Overview

As detailed below, we propose to represent a time series as a vector of dissimilarities from a set of carefully selected time series. Then, we use a simple 1NN classifier equipped with the Euclidean distance (1NN-ED-DBR) on this representation. The 12 dissimilarity measures we evaluate are introduced in Section 3.5. The prototype methods are presented in Section 4.1.2.

As shown in Figure 4.1, to represent RD in terms of dissimilarities we need a dissimilarity measure and a prototype set. Hence an object is represented as a vector where each coordinate corresponds to its dissimilarity to a given prototype. It is not required that a dissimilarity measure has the strict metric properties, however it is important that it is in monotonic relationship with whatever aspect of difference we want to capture. The prototype set can include all the elements of the training set or be constructed according to a user defined strategy. In other contexts, a prototype can be considered as the element of a set that is most similar to any of the elements of the set [191]. However, in previous work by Pekalska and others [147, 192] and in our work, prototypes are chosen on the basis of leading to useful dissimilarity representations, not necessarily on the basis of being typical.

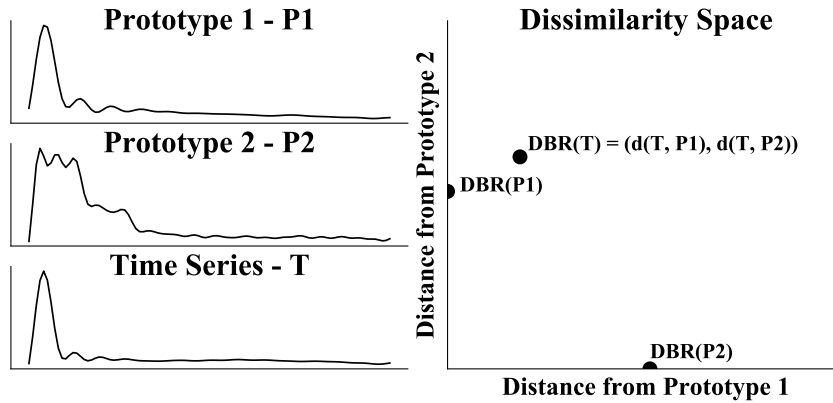


Figure 4.1: DBR. On the left-hand side there are two prototypes (P1, P2) and one time series (T) that we project into a 2-dimensional dissimilarity space shown on the right-hand side. For illustration the prototypes are projected in the dissimilarity space too. In the brackets it is shown how the new (x, y) coordinates are derived using a dissimilarity measure (d).

4.1.2 Prototype Methods

In this section we describe the eight prototype methods evaluated in this work. A visual overview of their functioning is provided in Figure 4.2.

There is relatively little literature on prototype methods in the context of DBR. This topic is often analysed in the context of nearest prototype classification [137]. The nearest neighbour rule suffers from several drawbacks such as high storage requirements, low efficiency at test time, and low noise tolerance, therefore it is argued that by reducing the training set to a smaller set of prototypes it is possible to mitigate all these weaknesses while improving generalisation performance [238]. Pekalska et al. [192] note that the set of prototypes should be large and diverse enough to be representative of the entire class we want to approximate; if these assumptions are not violated than even a random selection can lead to good classification performance.

The prototype methods can be divided into two groups: methods that select samples from the training data (1, 4, 5, 7, 8) [81], and methods that generate “synthetic” prototypes (2, 3, 6) [238]. All of the prototype methods can be used to obtain a variable number of prototypes through a user-defined hyper-parameter that we refer to as n . Since each prototype gives one dimension in the resulting representation, this allows the modeller to reduce any time series to any desired dimensionality. We study several prototype methods found in the literature and a new one named “percentiles”.

1. **Borders:** the first prototype to be selected is the training sample that on average is furthest from all the others. Next prototypes are the samples that on average are furthest from all the others used as prototypes.
2. **Centers Gaussian-mixture:** a Gaussian-mixture model with n components is fit on training data. The mean of each component is used as a prototype.
3. **Centers k-means:** a k -means model with $k = n$ is fit on training data. The centroid of each cluster is used as a prototype.
4. **Closest:** the n training samples that on average are closest to all the others are used as prototypes.
5. **Furthest:** the n training samples that on average are furthest from all the other training samples.
6. **Percentiles:** according to this prototype method, here introduced for the first time, we synthesize n prototypes by taking n sample-wise percentiles of the training data. E.g. for $n = 3$ we take the 0th, 50th, and 100th percentiles. For the special case $n = 1$ we take only the 50th percentile.

7. **Random:** n samples are randomly selected from training data with uniform probability.
8. **Support vectors:** We adapt the idea of using the support vectors of a support vector machine (SVM) classifier as prototypes, originally proposed by Li et al. [147] in the context of supervised classification, to the one-class classification framework. SVM classifiers find a decision boundary as a weighted combination of elements of the training set called support vectors. Similar to prototypes, support vectors are critical elements of the training set that enable the classification of an unknown object. We can see the dissimilarity-based approach to classification as a generalisation of the SVM algorithm where dissimilarity measures correspond to kernels and prototypes to support vectors.

According to this prototype method a one-class SVM (OC-SVM) model is fit on training data. The n closest support vectors to the decision boundary are used as prototypes. For this model we use a Gaussian kernel with scaling parameter $\gamma = 1/2\sigma^2$ set to $1/L$ (where L is the time series length). Also, we set $\nu = 1/N$ (where N is the number of training samples). The parameter ν represents a lower bound on the number of training samples that can be used as support vectors.

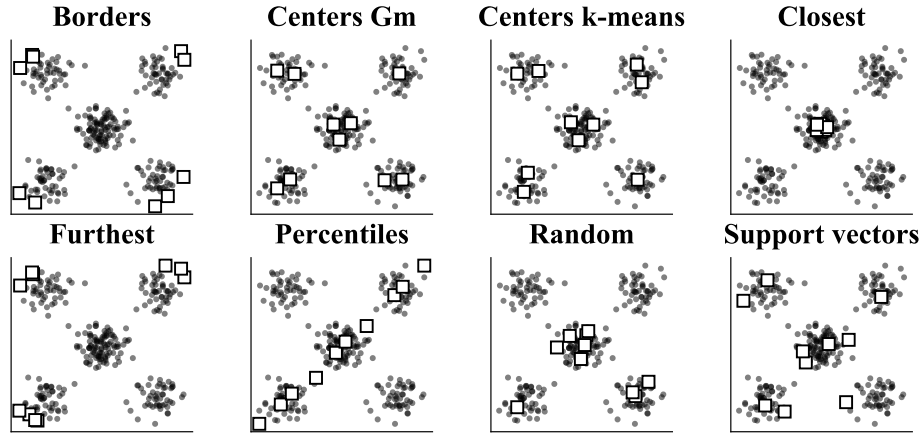


Figure 4.2: Prototype methods on 2-dimensional random data generated from a mixture of normal distributions with $n = 10$. ● Random data. □ Prototypes.

4.2 Results

An overview of the experimental results is provided in Section 4.2.1. Following, we elaborate on the impact of dissimilarity measures, and prototype methods (Sections 4.2.2). Finally, we demonstrate the ability of our approach in enabling dimensionality reduction (Sections 4.2.3), and visualisation of time series data-sets (Section 4.2.4).

4.2.1 Overview

All pairwise combinations of dissimilarity measures and prototype methods (Section 4.1) are evaluated on the 86 data-sets introduced in Section 3.2.

The classifier we use on the DBR is the 1NN classifier equipped with Euclidean distance (1NN-ED-DBR). Also, we evaluate the performance of the 1NN classifier on RD when this is equipped with each of the 12 dissimilarity measures considered (1NN-(·)-RD). A summary of results is shown in Table 4.1. For each dissimilarity measure and prototype method results are averaged over all the data-sets. The associated standard deviations are in the range [11, 16] as expected since some problems are far harder than others.

We evaluate the impact of the number of prototypes on the classification performance of 1NN-ED-DBR by varying $n = \{1, 2, 10\%, 20\%, 100\%\}$. For instance, when $n=1$ we use only one prototype; when $n=10\%$ we use a number of prototypes equal to $\lceil N \times 0.1 \rceil$ (where N is the number of training samples). When $n=100\%$ prototype methods are not needed because all the available training samples are considered as prototypes. For each value of n shown in the table there is also a row titled “RD”. In this row is shown the performance achieved using 1NN-(·)-RD with n randomly selected training samples. We use this approach as a point of comparison for the DBR performance.

The best performance (80% AUROC) is achieved in two cases by the dissimilarity measure MSM: for DBR_{100%}, and RD_{100%}. When $n = 100\%$ other dissimilarity measures like DTW, EDR, Euclidean, and Manhattan achieve a performance between 77-78% AUROC. Again for $n = 100\%$ the worst performance is achieved by Chebyshev and Kullback-Leibler (between 70-73% AUROC). It is not surprising that most dissimilarity measures achieve similar performance. As pointed out in other studies this can happen because strengths and weaknesses of individual measures are cancelled out by the variance over the large number of data-sets examined [250].

Concerning other values of n (1, 2, 10%, 20%), by comparing the average value of rows “Avg.” and “RD” we see that the performance of DBR and RD are close. However, the maximum performance of DBR (average value of row “Max”) is always superior to that of RD (except $n = 1$).

	AC	Ch.	Co.	DTW	EDR	ED	Gk	KL	MSM	Ma.	Sk	Wa.	Avg.
DBR₁													
Borders	63	60	61	63	59	60	60	59	61	60	61	60	61
Centers-Gm	66	65	68	67	64	67	67	62	66	68	68	64	66
Centers-k-means	66	65	68	67	64	67	67	62	66	68	68	64	66
Closest	66	62	66	68	64	64	63	65	66	65	66	64	65
Furthest	63	60	61	63	59	60	60	59	61	60	61	60	61
Percentiles	66	63	67	67	65	66	66	63	67	68	67	65	66
Random	65	61	64	66	63	62	62	62	65	63	64	62	63
Support vectors	64	60	63	66	61	61	61	63	65	61	64	63	63
Avg.	65	62	65	66	62	63	63	62	65	64	65	63	64
Max	66	65	68	68	65	67	67	65	67	68	68	65	66
RD₁	67	62	65	70	68	65	65	62	70	66	65	64	66
DBR₂													
Borders	71	66	69	71	68	68	67	64	70	70	70	68	68
Centers-Gm	70	68	72	72	69	71	71	65	72	72	72	68	70
Centers-k-means	71	68	72	72	69	72	72	65	72	73	73	69	71
Closest	68	63	68	69	66	64	64	66	67	65	69	65	66
Furthest	68	65	68	67	64	66	66	61	66	67	68	65	66
Percentiles	70	66	71	70	66	70	69	63	71	75	71	63	69
Random	69	64	68	70	67	66	65	65	68	66	69	67	67
Support vectors	70	64	68	71	67	66	66	65	69	67	69	67	67
Avg.	70	66	70	70	67	68	67	64	69	69	70	66	68
Max	71	68	72	72	69	72	72	66	72	75	73	69	71
RD₂	69	65	68	72	71	68	68	65	73	69	68	67	68
DBR_{10%}													
Borders	70	67	70	72	69	69	68	65	72	70	71	67	69
Centers-Gm	71	70	73	73	71	72	71	66	73	73	73	68	71
Centers-k-means	71	70	73	73	71	72	72	66	73	74	73	68	71
Closest	68	64	69	70	66	65	64	67	67	66	70	65	67
Furthest	69	66	69	70	67	68	67	63	69	68	70	65	68
Percentiles	70	68	71	73	70	71	69	67	73	75	72	67	70
Random	70	67	71	72	69	68	67	66	71	69	71	67	69
Support vectors	70	67	71	72	69	69	68	66	71	69	70	68	69
Avg.	70	67	71	72	69	69	68	66	71	70	71	67	69
Max	71	70	73	73	71	72	72	67	73	75	73	68	71
RD_{10%}	70	65	69	73	71	69	69	66	73	70	69	68	69
DBR_{20%}													
Borders	72	69	72	74	72	71	70	67	74	72	73	69	71
Centers-Gm	72	71	75	75	73	74	73	68	75	74	74	70	73
Centers-k-means	72	71	74	75	73	74	73	68	76	75	74	70	73
Closest	69	64	70	70	67	65	64	69	67	67	71	66	67
Furthest	71	68	73	72	70	70	69	66	72	70	73	69	70
Percentiles	71	69	72	74	72	71	70	68	75	76	72	69	72
Random	72	68	72	73	70	69	68	68	72	71	72	69	70
Support vectors	72	68	71	73	70	69	68	68	72	70	72	69	70
Avg.	71	69	72	73	71	70	69	68	73	72	73	69	71
Max	72	71	75	75	73	74	73	69	76	76	74	70	73
RD_{20%}	72	68	72	75	74	72	72	69	76	73	72	71	72
DBR_{100%}													
	74	73	75	78	78	76	75	70	80	77	75	73	75
RD_{100%}	75	72	77	78	77	77	77	71	80	78	77	74	76

Table 4.1: The table shows AUROC averaged across all the data-sets and rounded to the nearest integer.

4.2.2 Dissimilarity Measures and Prototype Methods

In Figure 4.3 performance of $\text{DBR}_{100\%}$ and $\text{RD}_{100\%}$ for each data-set is broken down by dissimilarity measure. Performance appear to be highly correlated and so nearly always tightly distributed along the diagonal.

Again in Figure 4.3, we note that for the EDR measure the performance of $\text{DBR}_{100\%}$ is higher than that of $\text{RD}_{100\%}$ in a good number of cases. Batista et al. [14] state that “several papers propose a method that is able to win on some, tie on many, and lose on some of the data-sets of the UCR/UEA archive and so authors claim that their method has some value. However, it is not useful to have an algorithm that can be accurate on some problems unless you can tell in advance on which problems it will be more accurate”.

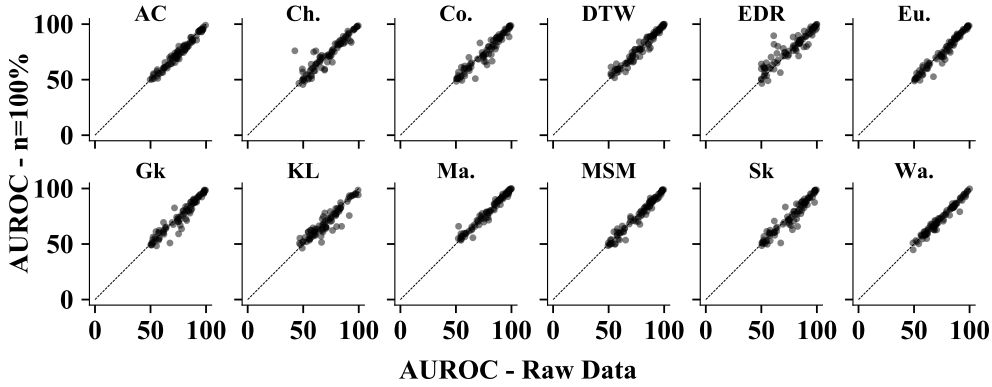


Figure 4.3: Scatter-plots of AUROC performance over all data-sets, analysed by dissimilarity measure for $\text{DBR}_{100\%}$ and $\text{RD}_{100\%}$.

In Figure 4.4, we select only the EDR dissimilarity, and we divide the 86 data-sets into two groups: a group where $\text{DBR}_{100\%}$ achieves better performance, and a group where $\text{RD}_{100\%}$ does. We further break down the two groups in terms of application domain, average number of training samples, and time series length. For each sub-group we test whether the difference between $\text{DBR}_{100\%}$ and $\text{RD}_{100\%}$ is significant or not using a Wilcoxon signed-rank test with a threshold on the p-value of $\alpha = 0.05$ [254]. We find that $\text{DBR}_{100\%}$ is significantly better than $\text{RD}_{100\%}$ when the application domain is “image outlines”. Duin and Pekalska [59] state that “dissimilarity based comparison of objects may be seen as the effort of transforming one structure into another”. Accounting for this point of view we argue that the EDR measure is successful because it asks the right question: how expensive is it to transform one object into the objects found in the set of prototypes? Finally, it is found that $\text{DBR}_{100\%}$ is significantly better than $\text{RD}_{100\%}$ when

the average number of training samples is greater than 125, and also better when the time series length is lower than 90.

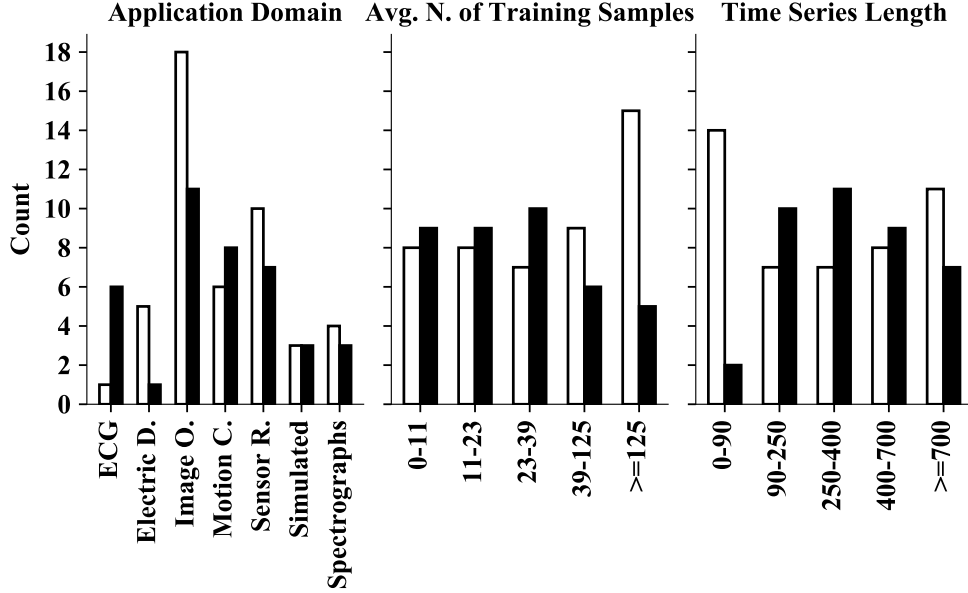


Figure 4.4: Counts of data-sets where DBR_{100%} achieves a better performance than RD_{100%} (\square) and vice-versa (\blacksquare) for the EDR dissimilarity only.

With respect to prototype methods, row-averages of Table 4.1 show that the best mean AUROC (73%) is achieved in DBR_{20%} by the prototype methods Centers-Gm and Centers-k-means. The effectiveness of these two prototype methods is consistent for other values of n .

In terms of the best pairing of dissimilarity measure and prototype method, the highest performance (76% AUROC) is achieved by MSM paired with Centers-k-means in DBR_{20%}, and by Manhattan paired with Percentiles again in DBR_{20%}. However, the latter pair shows a consistent performance for other values of n achieving 75% AUROC since $n = 2$.

In Figure 4.5, we investigate the interaction between the Manhattan dissimilarity and the prototype methods under DBR_{20%}. In particular, we want to understand how the performance of the pair Manhattan/Percentiles compares to other prototype methods paired with the Manhattan dissimilarity. It is noted that Percentiles is superior to all the other methods under the Manhattan dissimilarity, however all the other methods win for some data-sets confirming that they all have some potential value. Under the Manhattan dissimilarity, the difference between Percentiles and all the other prototype methods is significant according to a Wilcoxon signed-rank test ($\alpha = 0.05$).

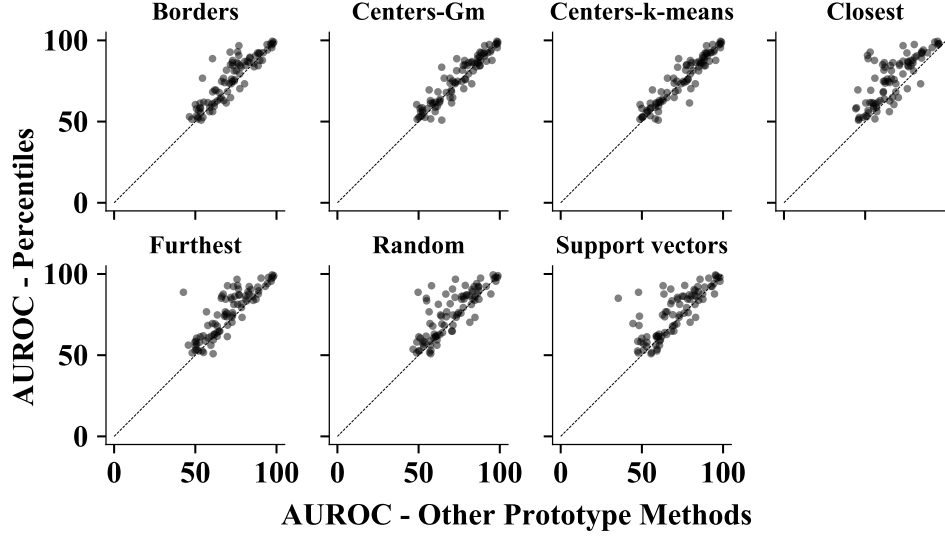


Figure 4.5: Scatter-plots of AUROC performance over all data-sets for the pair (Manhattan, Percentiles) compared to all the other prototype methods paired with the same dissimilarity measure under $\text{DBR}_{20\%}$.

The results discussed so far have considered a 1NN-ED classifier that achieves an average performance over all the variants $\text{DBR}_{1,2,10\%,20\%,100\%}$ equal to 68% AUROC. Now we discuss whether this choice has an important effect. In this regard we evaluate four other classifiers. (1) A centroid classifier assigns a score to a test sample that corresponds to its distance from the centroid of the training set. The average performance of this classifier over all the variants $\text{DBR}_{1,2,10\%,20\%,100\%}$ is equal to 63% AUROC. Notably, for DBR_2 when the representation is derived using the pair (Manhattan, Percentiles) this classifier achieves 75% AUROC. This good performance suggests that the representation is suitable for this classifier, and we know this classifier tends to do well if positive samples are projected towards the centre of a sphere and negative samples are projected far from it. (2) A kernel density estimation classifier (KDE) [28] which tends to mimic the performance of 1NN-ED. The average performance of this classifier over $\text{DBR}_{1,2,10\%,20\%,100\%}$ is equal to 65% AUROC. (3) A OC-SVM classifier that has shown substantially lower performance than 1NN-ED. The average performance of this classifier over $\text{DBR}_{1,2,10\%,20\%,100\%}$ is equal to 61% AUROC. (4) An isolation forest classifier (IF) [157] whose performance over $\text{DBR}_{1,2,10\%,20\%,100\%}$ is equal to 65% AUROC. Although the average performance of IF equals that of KDE the first do not exceed 70% AUROC for any pairing of dissimilarity measure and prototype method. While the centroid classifier is a non-parametric classi-

fier the other three are sensitive to the choice of hyper-parameters. Given the lower level of performance compared to the 1NN, and the challenges of hyper-parameter selection in the context of one-class classification, we decide not to investigate this further. Finally, we have evaluated some variants of the parameter k ($k=\{1,2,3,5\}$) for the k NN classifier but the performance of 1NN-ED is the best.

4.2.3 Dimensionality Reduction

As mentioned before, DBR allow us to reduce RD to vectors of arbitrary dimensionality by varying the number of prototypes. In Figure 4.6 (a), we can see that for all the pairs of dissimilarity measure/prototype method, performance tends to increase when we increase the number of prototypes. In this case a peak in performance is achieved when all the training samples are used as prototypes i.e. DBR_{100%}. However, if we look at Figure 4.6 (b) where we have considered only data-sets with an average number of training samples greater than 39, we can see that improvements in performance between DBR_{10%} and DBR_{100%} are relatively limited. Thus, for large data-sets using all the training samples as prototypes can lead to little or no improvement. This result can have a significant impact in terms of savings in computational time, especially when we use elastic dissimilarity measures such as DTW, EDR or MSM, subject to the number of prototypes we use, the length of the time series, and the number of test samples.

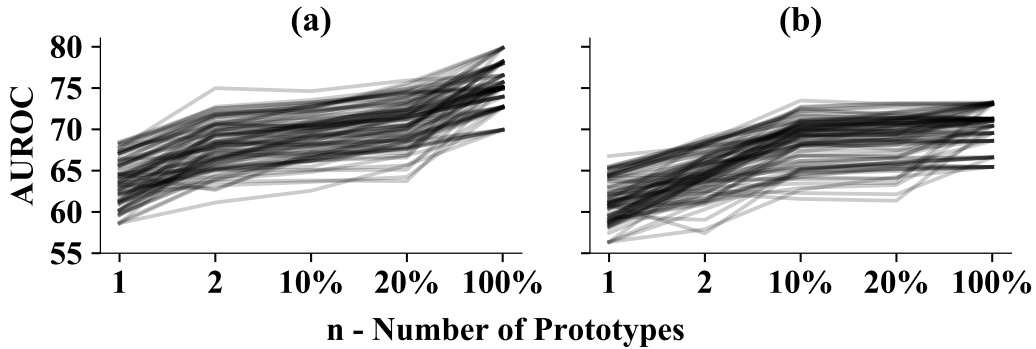


Figure 4.6: AUROC performance for all pairs of dissimilarity measures and prototype methods. In part (a) results are averaged over all data-sets; in part (b) are averaged only data-sets with average number of training samples greater than 39.

4.2.4 Visual Exploration of Time Series Data-Sets

Visualisation of time series data is important to enable understanding and decision making. It is particularly important in the context of anomaly detection where often modellers are not simply concerned with the detection of anomalies but are interested in understanding their dynamics and root causes, for instance in the context of fault detection, and medical diagnosis. By selecting only two prototypes DBR allow us to visualise a time series data-set in a 2-dimensional space as shown in Figure 4.7.

Sometimes, given the same data-set, the choice of the dissimilarity measure or the prototype method can make the difference in terms of performance. This aspect is not clear from Table 4.1 where results are averaged over all the data-sets. In part (a1)/(a2) of Figure 4.7, the positive and the negative class are not as easy to separate as in part (b1)/(b2). In part (a1) and (b1), two dissimilarity measures, ED and DTW, given the same prototype method (Closest) achieve a different performance (65% vs. 89%) on the “FiftyWords” data-set. Similarly, in (a2) and (b2), two prototype methods, Centers-Gm and Percentiles, given the Manhattan dissimilarity achieve a different performance (76% vs. 91%) on the “FacesUCR” data-set.

In both part (c) and (d) of Figure 4.7 are shown two examples where DBR achieve poor performance. The data-sets considered are “FordA” and “ShapeletSim”. From the experiment conducted by Bagnall et al. [12] we know that on both these problems shapelet-based and bag-of-patterns classifiers achieve a particularly good performance. Our understanding is that DBR are weak on problems where discriminative features are short distinctive sub-sequences (shapelets) or their frequencies (bag-of-patterns).

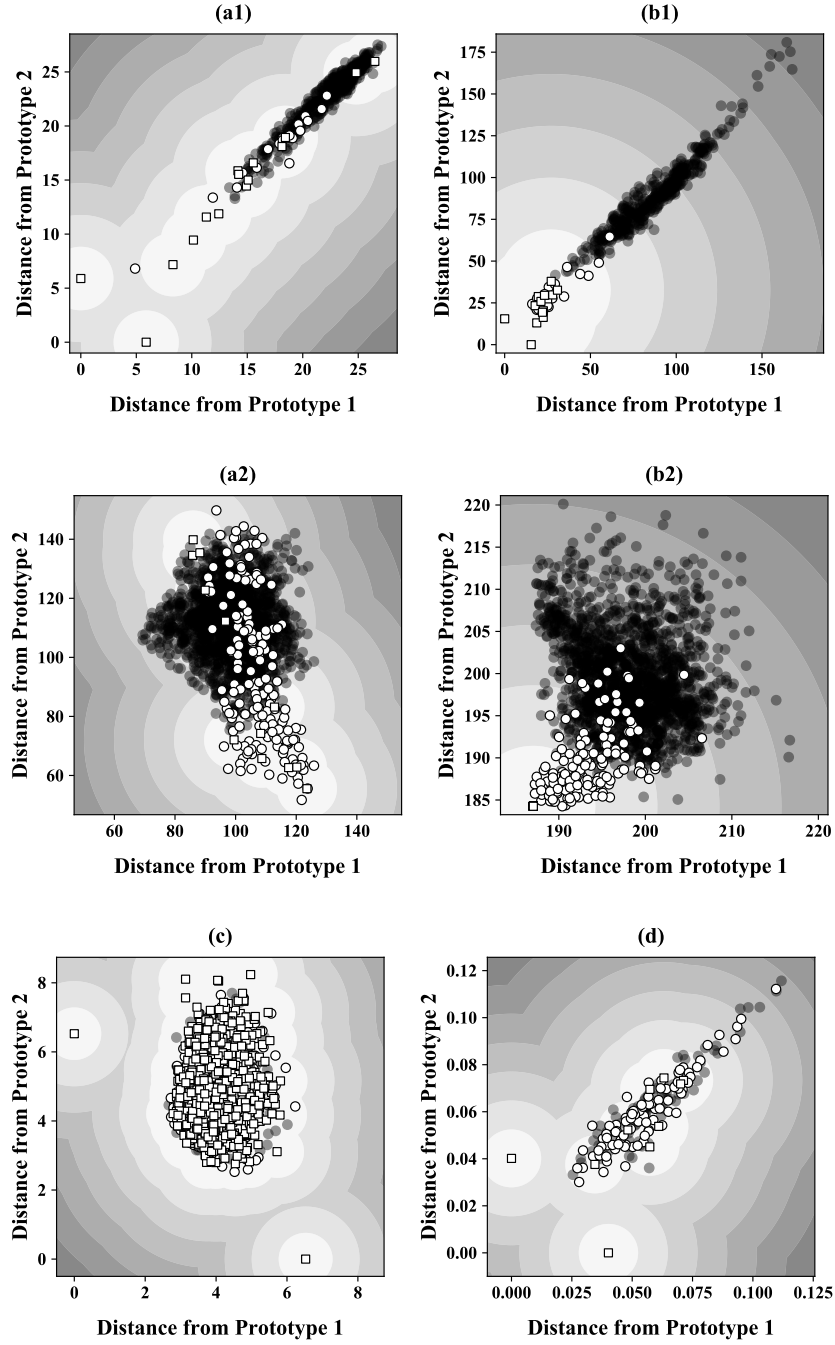


Figure 4.7: \square Training samples. \circ Positive test samples. \bullet Negative test samples. The shaded area gets darker as the distance from training samples increases. (a1) "FiftyWords", ED, Closest, positive class: 6. (b1) "FiftyWords", DTW, Closest, positive class: 6. (a2) "FacesUCR", Ma., Centers-k-means, positive class: 2. (b2) "FacesUCR", Ma., Percentiles, positive class: 2. (c) "FordA", Ch., Support vectors, positive class: -1. (d) "ShapeletSim", WD, Furthest, positive class: 0.

4.3 Conclusions

For the first time (to the best of our knowledge) we have conducted a comprehensive one-class classification experiment on the main archive of time series data-sets available in the literature (UCR/UEA archive).

We have investigated DBR as a methodology to derive a vector from a time series while maintaining information of its shape comparison to a set of carefully selected time series (prototypes) through a dissimilarity measure. In this regard, we have evaluated a Cartesian product of 12 dissimilarity measures, and 8 prototype methods (strategies to select prototypes).

We have argued that DBR are advantageous for problems where the class membership depends on the global time series shape. On the other hand, DBR are weak on problems where the class membership depends on short distinctive sub-sequences (shapelets), or their frequencies (bag-of-patterns).

As evidence from other studies confirm, there is no silver bullet that can achieve best performance over all problems. However, given the “right” pairing of dissimilarity measure and prototype method DBR can achieve high performance using only a sub-set of the available training samples (10-20%). This can guarantee dimensionality reduction and so improvements in terms of computational requirements. Finally, we have shown that by using only two prototypes this method can be used to visualise a data-set in a 2-dimensional space, enabling understanding of the problem at hand.

We have benchmarked the performance of DBR against a 1NN classifier equipped with each of the 12 dissimilarity measures considered on RD. As observed before in several binary and multi-class time series classification experiments this “simple” approach establishes a strong baseline.

DBR could enable the use of several off-the-shelf classifiers which generally perform poorly on raw time series. However, although we have evaluated five classifiers the 1NN is the one that has achieved the best performance. For three of the other classifiers, KDE, OC-SVM, and IF we believe this result is caused by the fact that we have used the same hyper-parameters throughout all the data-sets. This is because hyper-parameter selection is difficult in the context of one-class classification. In future we propose to investigate this problem. Again about hyper-parameter selection, we have not investigated how to select the best hyper-parameters for some of the dissimilarity measures in use. For instance, future research might examine how to select the best warping window for DTW when only samples from a single class are available for training (and validation). Also, future research is needed to extend our method to multivariate time series. Additionally, the possibility of ensembling multiple DBR, or to combine DBR with other representations warrants further investigation.

Chapter 5

Feature-Based Representations via Grammatical Evolution

Feature-based approaches to time series classification transform time series into feature-vectors that can be used with any off-the-shelf classifier. The basic assumption underlying the feature-based approach is that we can find an unequivocal “description” for a class of time series. Thus the question arises: how to determine a convenient description?

The feature extraction process should normally be customised to the problem at hand. Often, when researchers deal with a new problem they do not know which features could have the best discriminative power. Thus, a common approach is to construct an initial set of features and then select the subset that yields the best performance [51, 160]. Conversely, there is growing interest in algorithms that enable the data-driven discovery of useful features as made possible by deep learning methods [142]. A clear advantage is that modelers can redirect their efforts from the construction of the solution to the construction of the learning framework. While the former may be useful solely on a particular problem the latter may be effective on many.

We investigate grammatical evolution (GE) [185] as a means to attain a data-driven feature-based representation of time series. GE is an evolutionary computation technique related to genetic programming (GP) [132]. Our ultimate goal is to implement an automated feature extraction framework that could learn one feature at the time until a certain level of performance is achieved, or no further improvements are found. We exploit the flexibility allowed by GE as a learning framework. By starting from a set of “simple” functions e.g. mean, standard deviation, etc., GE allows the construction of complex features as required by the problem at hand. Each feature summarises a time series in a single scalar, thus by concatenating multiple features we can create representations of arbitrary dimensionality.

Previous research on time series classification has shown that class membership may depend on features related to the whole time series, or on features related to one or more of its sub-sequences [12]. We hypothesize that the same features extracted from a different time interval may lead to different performance. Taking this into account, we have enabled our algorithm to investigate both the features to extract and the sub-sequences from which to extract them. We demonstrate that these choices are of central importance. This is not only in terms of classification performance but also to allow understanding of the problem at hand.

Motivated by the requirements of a subject authentication problem we have recently investigated [168], the classification performance of extracted features is evaluated using a one-class classifier [170]. The aim of subject authentication is to confirm the identity of a person. Both binary and multi-class methods assume, in a sense, a fixed population of subjects, well-represented in the data, which is not realistic for this scenario. Thus the best option may be a one-class classifier tailored to the intended subject only.

Although we use a one-class classifier, our algorithm requires labelled data to evaluate the fitness function that drives the extraction process. This is in contrast with the requirements of one-class classification where only the samples of a single class are available before the test phase. However, we use our subject authentication problem to show how GE-evolved features are able to generalise to classes unseen during the feature extraction phase.

We compare our algorithm against a 1-nearest neighbour classifier equipped with dynamic time warping (DTW) [205] (1NN-DTW) on raw data, considered as the standard benchmark in the literature [12] (Section 5.2.2). Finally, we conduct an experimental analysis to demonstrate the impact of interval and function selection on performance (Section 5.5).

The chapter is organised as follows. In Section 5.1, we describe the overall GE-based algorithm for feature extraction. In Section 5.2 we describe the benchmark methods. In Section 5.3 we describe the experimental design. In Sections 5.4-5.5 we analyse our results. Finally, in Section 5.6 we summarise our conclusions and discuss future work.

5.1 Proposed Method

In this section we describe our evolutionary algorithm for feature extraction from time series based on GE. The core components of our approach are the grammar (Section 5.1.3), and the fitness function (Section 5.1.4).

5.1.1 Overview

Our algorithm relies on GE, a grammar-based form of GP. We choose GE to implement our algorithm because, as opposed to GP, it allows us to handle a mixture of data types. While the type constraint could be handled with other approaches e.g. strongly typed GP [175], we believe that the grammar is a particularly convenient way to express the syntax of admissible solutions, and also we want solutions to be readable Python code as this can facilitate understanding.

The grammar allows the modeller to exploit her/his domain knowledge, and to impose syntactical constraints to guide the search of feasible solutions. In this study we have used our knowledge of the TSC domain to define the proposed grammar. We have designed the grammar to be *balanced* rather than *explosive* [183], giving a bias towards short solutions. In many cases, the primitives included in the grammar are high-level functions specifically defined for time series e.g. the complexity estimate [14]. However, a practitioner specialising in a particular sub-domain of TSC could add further domain knowledge, e.g. spectral features known to be useful in that domain. In addition, the grammar allows the selection of any sub-sequence. Sub-sequences may be key to class membership as demonstrated by TSC algorithms like shapelets [258], or bag-of-patterns [152].

The solution space of our algorithm consists of feature-extractors. Each feature-extractor F is a function which takes as input a specific sub-sequence $[a : b]$ of a time series T e.g. $T[20 : 50]$. The output is a single scalar i.e. a feature. In Eq. 5.1 is shown an example of a feature-extractor. Note that all the primitives included in a given feature-extractor are applied to the same sub-sequence.

$$F = \text{Mean}(T[20 : 50]) \times \text{Skewness}(T[20 : 50]) \quad (5.1)$$

Our algorithm outputs a single feature-extractor. Thus, to extract multiple features we have to run the algorithm multiple times. Sequential extraction allows control over the “quality” of all features. If we were to evolve a number of features all at once we would not know which ones are contributing to classification performance, or which ones are even harming it. Furthermore, by extracting one feature at a time we are oriented towards finding the minimum number of features for the problem at hand. In fact, we can extract one feature at a time and stop as soon as we meet the required level of performance, or we do not observe any meaningful improvement. On the other hand, our greedy approach to feature extraction may not find the global optimum. The alternative to evolve a set of feature-extractors at once, already considered by Eads et al. [63], warrants further investigation.

Concluding, the fitness function requires each feature to have good classification performance, while being minimally correlated with previous features. While the first component of the fitness function is defined to extract features of good predictive power, the second one weakens the redundancy of extracted features.

5.1.2 Grammatical Evolution

GE is an evolutionary algorithm used for several machine learning tasks like AutoML [67], automatic program synthesis [226], feature extraction as in the present study, and it is also used in other fields like bioinformatics [176], finance [167], etc. Similarly to GP, GE exploits Darwinian evolution ideas to implement a search heuristic. Through an iterative process, automatically created candidate solutions are combined and tweaked until the process ends and the best solution found is returned.

The main difference between GP and GE is that while the former evolves solutions as executable trees, the latter evolves solutions as arrays of integers that are then mapped to executable code in an arbitrary language according to the rules written in an ad-hoc grammar. As a consequence most of the operators defined to select, combine, and modify solutions in GP are different from those used in GE [207]. Furthermore, and more importantly to the present discussion, GE allows modellers to enforce the structure of solutions through the grammar. This not only ensure the validity of evolved solutions that will not cause errors when executed, but also allows modellers to change the structure of admissible solutions by simply making changes to the plain text grammar. For instance, one could exploits hundreds of time series functions readily available in freeware software packages [37] as plug-in components that can be inserted in our grammar discussed in Section 5.1.3.

The first step required by GE is the creation of the grammar. The grammar contains the rules to generate candidate solutions for the problem at hand using the Backus–Naur syntax [130]. Each component of the solution is enclosed in angular brackets $\langle \cdot \rangle$. Proceeding left to right from the starting point ($\langle F \rangle$ in our grammar), the algorithm selects one of the available alternatives for each component encountered. Specifically, the algorithm selects the i^{th} alternative with $i = (c \bmod n)$, where c is an integer, and n the number of alternatives. At each choice the algorithm consumes one of the integers contained in the array (c). The main steps of the GE algorithm are listed below, more discussion of the algorithm and variants are provided in previous work [185].

1. Generate a set of solutions at random.

2. Map solutions from arrays of integers to executable binary trees.
3. Score each solution according to a fitness function (Section 5.1.4).
4. Combine and tweak solutions (Section 5.3.2).
5. Stop if the termination criteria is met, otherwise go to Step 2.

5.1.3 Grammar and Primitives

The grammar guiding the search of feature-extractors is shown in Figure 5.1. The general idea is to compose a feature-extractor $\langle F \rangle$ by selecting its individual components one at the time. Each component, enclosed in the angle brackets ($\langle \cdot \rangle$), requires a choice from a set of predefined equally probable alternatives, separated by pipe symbols ($|$). Finally, the actual extraction is carried out by a wrapper function *Extract* that simply takes the input arguments and arranges them in the form shown in Eq. 5.1, returning a feature. Note that T (a time series) is not enclosed in the angle brackets. This is because T is an argument that we pass to the function at call time.

As said before, a feature-extractor consists of a function $\langle \text{fun} \rangle$ applied to a sub-sequence of a time series T . To select a specific sub-sequence we need to choose a lower bound $\langle \text{lb} \rangle$ and an upper bound $\langle \text{ub} \rangle$. These indices are chosen from the range $[1, L]$, where L corresponds to the time series length. Since a requirement is that the lower bound is below the upper bound, if this condition is violated $\langle \text{lb} \rangle$ and $\langle \text{ub} \rangle$ are swapped. The grammar allows $\langle \text{ub} \rangle$ to have a *None* value. In this case if $\langle \text{bool} \rangle$ is *True* we select the sub-sequence $[1 : \langle \text{lb} \rangle]$, if *False* the sub-sequence $[\langle \text{lb} \rangle : L]$.

A function $\langle \text{fun} \rangle$ can use one or more elements of the $\langle \text{primitive} \rangle$ set. Multiple primitives are related through the operators $\{+, -, *, AQ\}$ contained in $\langle \text{op} \rangle$. In $\langle \text{op} \rangle$, *AQ* (analytic quotient) is a function defined to perform division while avoiding division by 0 error [182]. This is achieved by transforming a divisor d into $\sqrt{a + d^2}$. We set $a = 1$ as suggested by the authors of this function.

In $\langle \text{fun} \rangle$, $\langle \text{primitive} \rangle$ is repeated twice while $\langle \text{op} \rangle$ only once. This is done to make the selection of $\langle \text{primitive} \rangle$ twice as likely than $\langle \text{op} \rangle$ thus interrupting the recursion triggered by the $\langle \text{op} \rangle$ rule. In fact, as mentioned before each feature-extractor is represented as a list of integers each of which allows only one choice from the grammar. If all the integers are used up before the feature-extractor is completed this is considered an invalid solution.

The set of 17 primitives, presented in Table 5.1, includes functions able to take a sub-sequence as input and output a single scalar. These are among the simplest and most commonly used functions in feature-based time series classification.

Primitive	Description
Mean	-
Standard deviation	-
Median	-
Skewness	-
Kurtosis	-
Max	-
Min	-
Above0	Number of values above 0.
Below0	Number of values below 0.
AbsoluteEnergy	Sum of the absolute value.
AR	Coefficient of an auto-regressive model of order 1.
Autocorrelation	Auto-correlation at lag 1.
CE	Measure of shape complexity known as “complexity estimate” [14].
FFT	Amplitude associated with the largest coefficient of a fast Fourier transform.
LinearTrend	Slope of a linear regression model.
MeanChanges	Mean of first order differences.
TRAS	Measure of non-linearity known as “time reversal asymmetric statistic” [218].

Table 5.1: Primitives used to evolve the feature extractors.

```

<F> ::= Extract(T, <lb>, <ub>, <bool>, <fun>)

<fun> ::= <primitive> | <primitive> | <op>

<primitive> ::= Above0 | AbsoluteEnergy | AR | Autocorrelation |
               Below0 | CE | FFT | Kurtosis | LinearTrend |
               Max | Mean | MeanChanges | Median | Min |
               Skewness | Std | TRAS

<op> ::= (<fun>+<fun>) | (<fun>-<fun>) |
         (<fun>×<fun>) | AQ(<fun>, <fun>)

<lb> ::= 1 | 2 | ... | L

<ub> ::= <lb> | None

<bool> ::= True | False

```

Figure 5.1: Grammar used to evolve the feature extractors.

5.1.4 Fitness Evaluation

The fitness function, shown in Eq. 5.2, drives the search through the solution space. The fitness score is crucial because it influences the probability of a feature-extractor to be selected, breed with other feature-extractors, and thus pass on to future generations. The fitness evaluation process is illustrated in Figure 5.2. Our goal is to sequentially extract features that are all able to achieve a good classification performance individually, while having minimum linear dependence between them.

As an example, suppose we want to extract two features for a certain classification problem. This means we have to run our algorithm $x = 2$ times with each run outputting a single feature-extractor. The first feature-extractor is evolved aiming at the minimisation of the classification error on a validation set. The classification error is defined as $1 - \text{AUROC}$. On the other hand, the second feature-extractor is evolved aiming for both minimisation of the classification error on a validation set, and minimisation of the average squared Pearson correlation coefficient with previous features $\overline{R^2}(F)$. This coefficient is calculated as follows. We take the feature-based representation of training data according to a candidate feature-extractor. Then, we calculate its squared Pearson correlation with the feature-based representation of the same data according to each of the feature-extractors outputted from previous runs (\hat{F}_i). Finally, we calculate the average value. Both the classification error, and the average squared Pearson correlation coefficient have magnitude in the range $[0, 1]$ thus they are commensurate in scale, and have the same impact on the fitness score. This process is repeated three times using 3-folds cross-validation. Thus, the final fitness score of a feature-extractor corresponds to the average score of the 3-folds.

In summary, after all the feature-extractors of an initial population are assigned a fitness score, the crossover and the mutation operators are applied to create a new generation. The evolutionary process continues until a certain number of generations is reached. At that point the feature-extractor with the lowest fitness is considered the final solution of the algorithm.

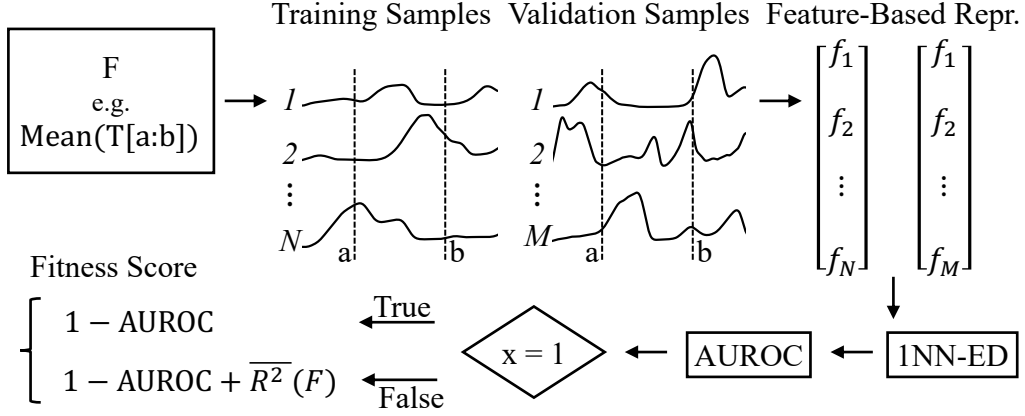


Figure 5.2: Fitness evaluation of a feature-extractor F . First the feature-extractor is applied on each training/validation sample deriving the respective feature-based representation. The classifier in use is a 1NN-ED. The variable x corresponds the number of the feature we are extracting. The algorithm minimises the classification error ($1 - \text{AUROC}$) and the average squared Pearson correlation coefficient with previous features on training data ($\overline{R^2}(F)$).

$$\text{Fitness}(F) = \begin{cases} 1 - \text{AUROC}, & \text{if } x = 1 \\ 1 - \text{AUROC} + \overline{R^2}(F), & \text{otherwise} \end{cases} \quad (5.2)$$

$$\overline{R^2}(F) = \frac{1}{x-1} \sum_{i=1}^{x-1} R^2(F, \hat{F}_i) \quad (5.3)$$

$$R^2(F, \hat{F}_i) = \left[\frac{\text{cov}(F, \hat{F}_i)}{\sigma(F) \times \sigma(\hat{F}_i)} \right]^2 \quad (5.4)$$

In Eq. 5.4 $\text{cov}(\cdot)$ and $\sigma(\cdot)$ represent the covariance and the standard deviation functions respectively.

5.2 Benchmark Methods

In this section we describe the benchmark methods we compare with. We consider a random search (Section 5.2.1), a distance-based approach (Section 5.2.2), and a feature-based approach with some variants (Section 5.2.3).

5.2.1 Random Search

Random search (RS) is often used to evaluate whether the evolutionary process brings any benefit over the random generation of solutions. Thus, it disentangles the effect of the grammar from the effect of the search process.

Following the rules of our grammar, the RS creates a number of feature-extractors ($\# \text{ generations} \times \text{population size}$, Section 5.3.2) with no duplicates allowed. Feature-extractors are evaluated through the fitness function, and the best one is selected to benchmark our algorithm. As per our algorithm, the RS is repeated multiple times in order to extract multiple features.

5.2.2 1NN with DTW

The 1NN-DTW classifier (with warping window set through cross-validation [206]) is considered the standard benchmark in the time series classification literature [12]. Unlike all the other approaches used in this study 1NN-DTW does not involve feature-vectors but raw time series. This is a dissimilarity-based classification approach that compares two time series using DTW. In the one-class classification framework each test sample receives a classification score equal to the DTW distance to its nearest training sample.

For the warping window size we have tested all the values in the range $[1, 0.1 \times L]$ (where L is the time series length) using a 10-fold cross-validation on validation data.

5.2.3 Function and Sub-Sequence Selection

As discussed in Section 5.1, our algorithm can dynamically decide both the features to extract and the sub-sequences from which to extract them. A parallel can be drawn with the manual approach where features and sub-sequences can be selected according to fixed user-defined strategies. To investigate the behaviour of our data-driven algorithm we evaluate a Cartesian product of the alternatives mentioned below.

In terms of features to be extracted we consider two sets. (1) The 17 primitives used in our algorithm (Section 5.1.3) and (2) the C22 features discussed in Section 2.4. Overall, C22 features are different and more complex with respect to our primitives.

In terms of segmentation strategies we consider three approaches. (1) We extract the features from the whole time series. (2) We segment time series into a number of (approximately) equally sized sub-sequences. Then, features are extracted from each sub-sequence and concatenated into a feature-vector. (3) We segment time series using a change point model. Generally

speaking, change point models are used to divide a time series into distinct homogeneous sub-sequences [9]. We use the bottom-up segmentation algorithm proposed by Keogh et al. [123]. In order to define the number of sub-sequences and their boundaries for a given data-set we implement the following algorithm. Given a training set we select 50% of the samples at random. We segment each sample using the bottom-up algorithm and we record the number of sub-sequences found. Then, we compute the average number of sub-sequences (rounding to the nearest integer). Continuing, we re-run the segmentation algorithm on the same samples imposing as stopping criterion the average number of sub-sequences found before. Finally, we average resulting indices.

5.3 Experimental Design

We provide an overview of the data-sets used in Section 5.3.1. A summary of the main components of the evolutionary framework is provided in Section 5.3.2. We report some implementation details in Section 5.3.3.

5.3.1 Experimental Data

We test our feature extraction algorithm on 30 data-sets. Of these, 29 are selected from the UCR/UEA archive, while one is a proprietary data-set [168]. This data-set concerns a subject authentication problem through accelerometer data collected using wrist-worn devices.

When carrying out a time series classification experiment it is good practice to consider all the data-sets of the UCR/UEA archive [46]. This is to avoid spurious results, and allow a thorough comparison between different studies. However, we consider only 29 data-sets selected as follows: from the original body of 85 data-sets of the archive, first we filter out those where there is at least one class with less than 17 training samples. Then we sort the remaining ones according to their total number of samples and select the smallest 29. The rationale is that our algorithm requires a validation set to evaluate the quality of candidate feature-extractors during the evolutionary process. Thus, we want to ensure that there is always a sufficient amount of samples to allow a meaningful training/validation split. Specifically, we use a 2:1 split of training data. Of Furthermore, we prefer smaller data-sets as our current implementation is very expensive in terms of runtime.

5.3.2 GE Configuration

The evolutionary process is configured with parameters that are common across the field [200]. The algorithm follows a generational approach with a population of 500 individuals (feature-extractors) for 40 generations. In the GP literature it is most common to use 50 generations [132], however 40 is also often used [43]. In their review of the GP literature, Poli et al. [200] state that the number of generations typically falls in the range [10, 50], where the most productive search is usually performed.

The population is randomly initialised with uniform probability. Genomes are initialised with length 200, while the max length is set to 500. Individuals are recombined with each other using two-point crossover with probability 0.8, and no wraps are allowed. Given two individuals, the crossover operator creates two new individuals by randomly selecting two different points on each genome. Then, the initial and final sections of one genome are merged with the mid section of the other. We increase the selection pressure setting the tournament size to five. After the crossover phase, the resulting population is subject to the mutation operator. We use the int-flip mutation operator that may change with probability 0.01 each gene of a given individual. Elitism is used to preserve the best individual through the generations.

5.3.3 Implementation Details

The experiment took about 5,000 CPU hours. Our current implementation is expensive especially if compared with the benchmark methods which take just a few hours to run. However, it is not straightforward to draw conclusions about the computational complexity, and scalability of our algorithm. First, for each data-set we run the algorithm 30 times for statistical purposes. However, this would not be required in real-world use. Continuing, runtime depends on several implementation details (e.g. the primitives included in the grammar) which could be reconsidered in future research. Finally, the expensive feature extraction phase is balanced by savings during the prediction phase. In fact, during the prediction phase our algorithm allows the classifier to work with low-dimensional feature-based representations. In contrast, 1NN-DTW always requires the whole time series to work.

5.4 Results

An overview of the experimental results is provided in Section 5.4.1. In Section 5.4.2, we present an excursus on the generalisation capabilities of our algorithm when tested on classes that are not available during the extraction

phase. Algorithm limitations are discussed in Section 5.4.3. Following, we unfold the discussion on feature extractors in Section 5.5).

5.4.1 Overview

Classification performance for the data-sets of the UCR/UEA archive is shown in Table 5.2. Results related to our subject authentication problem are discussed in Section 5.4.2. For each data-set we extract 10 features and we repeat this process for 30 independent runs. We standardise features to have zero mean and unit variance (with reference to the training set).

We compare the performance of the features evolved through our algorithm with that of two benchmark sets of features. Of these, PR is the set of primitives used in the grammar, while C22 is the set of features selected by Lubba et al. [160]. These features are used to derive feature-based representations of time series according to three different strategies. (1) Features are extracted from the whole time series (columns 1-PR/1-C22). (2) Time series are broken down into 5 adjacent and equally sized sub-sequences. Then, features are extracted from each sub-sequence and grouped together in a feature-vector (columns 5ES-PR/5ES-C22). (3) Time series are broken down in a number of adjacent sub-sequences using a change point model [123]. Then, features are extracted from each sub-sequence and grouped together in a feature-vector (columns CPM-PR/CPM-C22). We observe that the average number of sub-sequences per data-set found through the change point model is 5 and this is why we use this number in item (2). Concluding, we compare against a dissimilarity-based classification approach (column 1NN-DTW).

Our algorithm is superior to any benchmark method. Our algorithm gives better results than 1NN-DTW. It also improves on the C22 features considered the state of the art for feature-based time series classification [160].

We consider column 1-3 our best result (in the trade-off between best average performance subject to the lowest dimensionality achievable, which is our aim), while 1NN-DTW is the best benchmark method for comparison. We find that the difference between column 1-3 and 1NN-DTW is statistically significant using a Wilcoxon signed-rank test with a p-value threshold of $\alpha = 0.05$ [254]. We observe that the p-values decrease monotonically as additional features are added, as we would expect. If a Bonferroni-Holm correction for multiple testing [3] is applied, statistical significance is achieved for all tests from 1-2 onward. As 1-3 is the point with the largest reduction in p-value relative to the preceding feature number choice (1-3, p-value=0.007 vs. 1-2, p-value=0.024), we select this as our optimal result in the performance vs. dimensionality reduction trade-off.

Comparing the average performance of columns 1-3 and 1-PR in Table 5.2,

we can see that our algorithm is able to exploit and combine the functions included in the set of primitives in a way that allows +17% AUROC relative to the use of the functions alone. Furthermore, our algorithm achieves the highest average performance using only 3 features. This gives the time series representation with lowest dimensionality. In fact, PR and C22 require at least 17 and 22 features respectively, while 1NN-DTW requires the whole time series. Dimensionality has an impact on storage requirements, and computational complexity at prediction time.

To extract features from a segmented time series has a positive impact on performance. We note a +11/10% AUROC comparing the average performance of column 1-PR with that of columns 5ES-PR/CPM-PR. Conversely, the same does not hold comparing 1-PR with C22. In that case segmentation has little or no impact on performance. This seems to depend on the nature of features. The PR set mainly includes simple statistical moments like mean, standard deviation etc. Although these features are easy to compute, their descriptive power is weak when they are extracted from the whole time series. Moreover, all time series of the UCR/UEA archive are z-normalised so they all have 0 mean and a standard deviation of 1. When a time series is segmented, PR features have more descriptive power.

On the other hand, C22 features are more complex, thus they are able to capture some distinctive characteristics even when they are extracted from the whole time series. In order to evaluate if 1-C22 features are overfitting we measure their performance on validation data. The average performance across all data-sets on validation data is equal to 66% AUROC. Considering that the performance of 1-C22 features on test data is 70% AUROC we can conclude that these features are not overfitting.

On average, the performance achieved by segmenting a time series in equally sized sub-sequences is nearly the same as that achieved by segmenting through a more complex model, as we can see comparing columns 5ES-PR/5ES-C22 with columns CPM-PR/CPM-C22. It is difficult to explain such a result within this experimental study. Investigating this result in more detail is the focus of further research. However segmentation in equally sized segments can be considered as a strong baseline.

Our method converges to the maximum performance with only three features. In Figure 5.3 we show the average AUROC for all data-sets of our algorithm (●). As shown in part (a), the “quality” of features decreases by 4% from the first to the third feature. The performance remains steady from the third to the tenth feature (when rounded to the nearest integer). This demonstrates that individually features maintain a similar level of performance. In part (b), we can see that by combining multiple features the AUROC increases by 2% between the first and the third feature. Then we

can observe a weak but positive trend. Features extracted by our algorithm are not explicitly required to improve the overall classification performance when grouped with those that have been extracted before. This objective is pursued indirectly by minimising the linear correlation between features.

Secondly, Figure 5.3 shows RS average AUROC for all data-sets (■). In order to speed-up the computation we do not repeat the RS for 30 runs, as done for our algorithm. The variance between RS runs is expected to be low because the sample size is large, and there is no danger of an “unlucky” initial population, as in GE itself. Also, as we have a large enough number of data-sets, the variance across data-sets makes our results statistically significant. Our algorithm is better than RS. However, RS performance rivals 1NN-DTW, a strong baseline. This demonstrates that both our grammar and the search contribute strongly to performance.

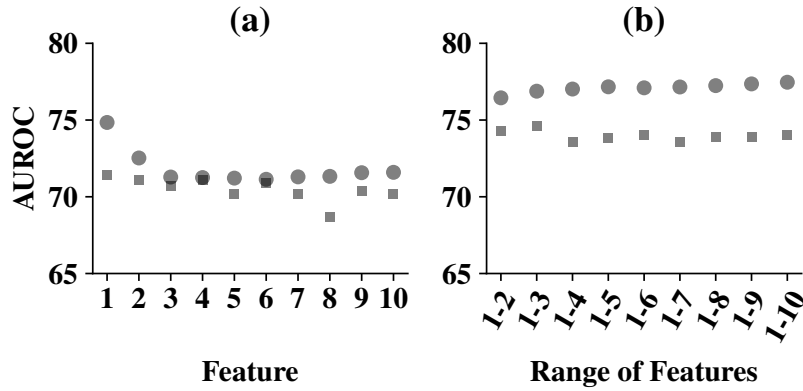


Figure 5.3: Average AUROC of our algorithm for all data-sets (●). (a) AUROC per feature. (b) AUROC per sequence of features. Secondly, RS average AUROC for all data-sets (■).

5.4.2 Subject Authentication

This data-set, which we refer to as “AccelerometerData”, concerns a subject authentication problem we have recently investigated [168]. To collect this data a group of nine subjects wore a watch-like tri-axial accelerometer for about a month in free living conditions. Each time series corresponds to the average acceleration per minute over an entire day.

Features extracted by our algorithm enable good classification performance when tested on data that include subjects not available during the

Data-set	(a)										(b)				
	1	1-2	1-3	1-4	1-5	1-6	1-7	1-8	1-9	1-10	1-PR	5ES-PR	CPM-1-PR	5ES-C22	CPM-1NN-C22 DTW
Coffee	91	98	99	99	99	99	99	99	98	98	61	93	92	95	92
Computers	65	67	66	65	65	64	64	64	64	64	51	52	53	56	51
DistalPhalanxOutlineAgeGroup	81	80	80	80	81	79	79	79	78	78	59	60	66	70	61
DistalPhalanxTW	81	81	81	81	82	82	82	81	82	82	73	80	79	83	74
ECG200	80	82	83	84	85	86	86	86	87	87	56	80	76	72	80
Earthquakes	67	68	68	68	68	68	69	69	69	70	50	56	54	56	55
Fish	80	85	87	88	89	90	90	90	90	90	61	85	85	84	87
GunPoint	91	87	86	87	88	89	92	92	92	93	59	78	70	80	82
Ham	60	65	67	68	69	69	69	70	70	69	51	53	58	55	56
Haptics	63	63	63	63	64	64	64	64	64	65	51	63	62	60	56
Herring	51	53	53	53	53	54	53	53	53	53	50	50	49	51	55
LargeKitchenAppliances	75	78	79	79	78	78	78	77	77	77	51	62	58	71	61
Lightning2	64	70	72	74	75	75	74	74	74	74	56	64	67	66	57
Meat	95	97	98	97	97	97	97	96	96	96	74	98	98	85	86
MiddlePhalanxOutlineAgeGroup	59	58	58	58	58	57	57	57	56	57	48	54	62	58	57
MiddlePhalanxTW	71	72	72	72	72	72	72	72	73	73	72	74	72	73	71
OSULeaf	78	80	81	81	81	81	80	80	80	80	65	81	83	83	81
ProximalPhalanxOutlineAgeGroup	91	90	87	86	85	82	81	80	80	80	65	68	81	79	77
ProximalPhalanxTW	91	92	92	93	93	93	93	93	93	93	79	85	87	85	80
RefrigerationDevices	64	64	64	64	64	64	64	64	64	65	50	57	56	63	61
ScreenType	58	60	61	61	61	61	60	60	60	60	50	53	51	57	55
SmallKitchenAppliances	79	81	81	80	80	80	80	80	80	80	52	66	68	73	68
SyntheticControl	99	99	99	99	99	99	100	100	100	100	96	99	99	99	76
ToeSegmentation1	64	66	68	69	69	70	71	72	72	72	59	63	64	65	62
ToeSegmentation2	74	77	78	79	80	79	79	79	79	80	63	62	62	57	62
Trace	100	100	100	100	100	100	100	100	100	100	96	98	96	96	94
Wine	56	58	58	59	59	58	59	59	60	61	53	56	61	47	50
Worms	70	73	73	74	75	75	75	75	76	75	53	74	76	84	81
WormsTwoClass	63	65	64	64	63	63	63	64	64	65	49	60	61	70	70
Average	74	76	77	77	77	77	77	77	77	77	60	70	71	71	69
															73

Table 5.2: Average AUROC for the data-sets of the UCR/UEA repository rounded to the nearest integer. (a) Performance of every sequence of features, e.g. 1-10 means all 10 features. (b) Benchmark methods. Columns in bold relate to our best result (1-3) (best performance with lowest dimensionality), and the best benchmark method (1NN-DTW).

extraction phase. In other words, features learned to distinguish subject 1 from subjects 2-6 are also effective to distinguish subject 1 from subjects 7-9.

Figure 5.4 shows the average classification performance achieved by our algorithm on a subset of subjects (1-6). The solid line (—) represents the performance achieved when, during the feature-extraction phase, we use a validation set that includes only subjects 1-6. The dashed line (---) represents the performance achieved when, during the feature-extraction phase, we use a validation set that includes all subjects (1-9). In both cases, extracted features are evaluated on a test set including all subjects (1-9). The two lines follow almost the same trajectory. Excluding the first feature, the average absolute difference between the two lines is equal to 0.3% AUROC.

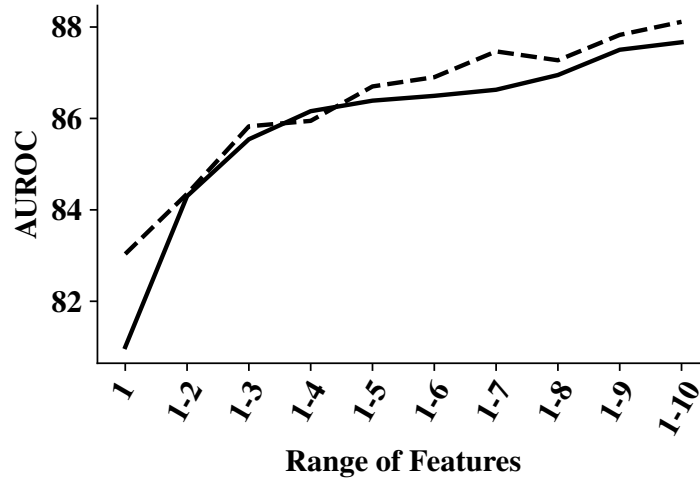


Figure 5.4: Average AUROC “AccelerometerData” data-set, subjects 1 to 6. — Only subjects 1-6 are used during the feature-extraction phase. --- All subjects 1-9 are used during the feature-extraction phase.

5.4.3 Limitations

Our algorithm reveals a number of limitations. The first limitation concerns the runtime required by our current implementation. Although, as discussed in Section 5.3.3, there are several aspects to consider, overall the algorithm would greatly benefit from any substantial improvement of its runtime.

Continuing, we observe that the algorithm tends to overfit. As shown in Figure 5.5 part (a), average AUROC of features 1 to 10 on validation data is approximately 11% higher than on test data. This value drops to 6% if we consider sequences of features, as shown in part (b). The performance per

feature seems to follow the same pattern on both validation and test data. Conversely, the performance per group of features exhibits a weak negative trend on validation data, and a weak positive trend on test data showing that together multiple features generalise better. Also, the performance of our algorithm remains better than baselines on test data. While overfitting is a problem it also presents an opportunity to improve our results through regularisation strategies, as we plan to do in future work.

Finally, one possible weakness in our fitness function is that as the number of extracted features increases the correlation penalty is averaged over an increasing number of features as well. While this could make the penalty very small, after a large number of feature-extractors have been created, we observe that a small number of features (e.g. 3 in Table 5.2) achieves best average performance.

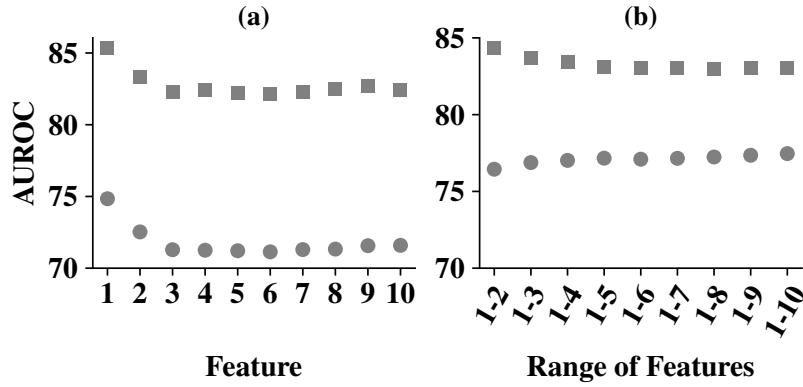


Figure 5.5: (a) Average validation (■) and test (●) AUROC for all data-sets per feature. (b) Average validation and test AUROC for all data-sets per sequence of features.

5.5 Feature-extractors and Interpretability

In this section we show how the evolutionary process leads to the selection of specific features and sub-sequences according to the problem at hand. This not only enables good classification performance but also understanding.

The development of interpretable machine learning models is an important research topic [55]. Interpretability is not only essential to machine learning practitioners, e.g. understanding a model allows understanding of its limitations, but also legislation can require model interpretability. For

instance, the General Data Protection Regulation act¹ introduced in the European Union gives to citizens a “right to explanation” with respect to decisions taken by algorithms using their data.

Several researchers have shown that GP can be used to enable interpretability of so-called black-box models (models that are not easy to interpret) [246]. Others have shown that evolutionary algorithms can be as effective as black-box models with the advantage of being more interpretable [145].

Our algorithm allows interpretability of the classification outcome in several ways. In Sections 5.5.1-5.5.2, we expand the discussion about feature-extractors and interpretability in the context of TSC, however here we provide some more general considerations. First, as the search for feature-extractors is driven by the classification performance they allow through a 1NN classifier, it is expected that our approach considers a sample to be normal if it is “close enough” to other samples we already know to be normal (i.e. training data). This type of decision by analogy resembles human thinking [184]. Continuing, extracted features are immediately interpretable as they are expressed as readable Python code. Also, as extracted features are composed of high-level functions e.g. mean, standard deviation, etc. predictions made by our algorithm can be explained in the context of the problem at hand. Finally, as our algorithm gives good classification performance with two or three features, it allows useful visualisation of time series data-sets (Figure 5.8), and data visualisation is considered key to interpretability [145]. On the other hand, as the number of primitives included in a feature-extractor increases they become more difficult to interpret (Section 5.5.1). As suggested by Lensen et al. [145] in the context of data visualisation, future work may investigate multi-objective fitness functions able to trade-off between classification performance and complexity of the solutions found.

5.5.1 The Features to Extract

We use the “SyntheticControl” data-set [7] as a case study to illustrate the ability of our algorithm to discover the right features for the problem. As shown in Figure 5.6 this data-set contains 6 different classes of simulated time series. In (1) time series are generated by sampling from a Normal distribution. In (2) time series are generated by sampling from a Normal distribution and adding a cyclic component through a sine function. In (3) and (4) time series are generated by sampling from a Normal distribution

¹<https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32016R0679&qid=1603268249651>

and adding or subtracting a trend component. In (5) and (6) time series are generated by sampling from a Normal distribution and adding or subtracting a trend component only after or before a certain time point.

We count the selection frequency of each function and operator used in the first two feature-extractors evolved for each class of the “SyntheticControl” data-set. Without loss of generality we focus on the first two feature-extractors because this eases our discussion and enables visualisation as shown in Figure 5.8. Feature-extractors used to generate Figure 5.8 are listed below. In the text T represents a time series and L its length. Feature extractors are composite functions that can be read and interpreted by humans. They can be as short as to include only one or two primitives. On the other hand, longer feature extractors may be broken down in order to evaluate the impact of single components. In principle, the maximum number of primitives to be included in a feature extractor may be limited by the user.

- **Feature 1**

- **Class (1):** $\text{Min}(T[1:59]) + \text{CE}(T[1:59])$
- **Class (2):** $\text{AQ}(\text{FFT}(T[1:57]), \text{Kurtosis}(T[1:57]))$
- **Class (3):** $\text{AQ}(\text{Mean}(T[9:L]) * \text{STD}(T[9:L]) - \text{Max}(T[9:L]), \text{STD}(T[9:L]))$
- **Class (4):** $\text{Min}(T[4:34])$
- **Class (5):** $\text{Mean}(T[1:50]) * \text{AQ}(\text{Min}(T[1:50]), \text{Mean}(T[1:50]))$
- **Class (6):** $\text{AQ}(\text{Below0}(T[36:L]), \text{LinearTrend}(T[36:L]) * \text{Min}(T[36:L]) * \text{Median}(T[36:L]) - \text{TRAS}(T[36:L]))$

- **Feature 2**

- **Class (1):** $\text{Above0}(T[1:53]) - \text{AQ}(\text{AR}(T[1:53]), \text{Max}(T[1:53]))$
- **Class (2):** $\text{AQ}(\text{FFT}(T[1:34]), \text{AQ}(\text{AR}(T[1:34]) + \text{FFT}(T[1:34])), \text{AbsSum}(T[1:34]) * \text{AQ}(\text{FFT}(T[1:34]), \text{AQ}(\text{AR}(T[1:34]) + \text{LinearTrend}(T[1:34]) * \text{LinearTrend}(T[1:34]), \text{TRAS}(T[1:34]))$
- **Class (3):** $\text{AQ}(\text{Max}(T[1:34]), \text{AQ}(\text{Min}(T[1:34]), \text{AQ}(\text{Mean}(T[1:34]) * \text{Skewness}(T[1:34]) + \text{Max}(T[1:34]) * \text{AR}(T[1:34]), \text{Median}(T[1:34])))$
- **Class (4):** $\text{Above0}(T[22:L]) - \text{AQ}(\text{AQ}(\text{AQ}(\text{Above0}(T[22:L]), \text{Autocorrelation}(T[22:L])), \text{AbsSum}(T[22:L]) - \text{LinearTrend}(T[22:L]) + \text{Min}(T[22:L])), \text{Mean}(T[22:L])) - \text{AQ}(\text{Max}(T[22:L]), \text{Min}(T[22:L]) + \text{LinearTrend}(T[22:L]))$
- **Class (5):** $\text{Max}(T[7:L]) * \text{Mean}(T[7:L])$
- **Class (6):** $\text{Below0}(T[7:22]) - \text{AQ}(\text{AR}(T[7:22]) * \text{Max}(T[7:22]), \text{AQ}(\text{AQ}(\text{AQ}(\text{Min}(T[7:22]) * \text{Autocorrelation}(T[7:22]), \text{Below0}(T[7:22])), \text{AQ}(\text{MeanChanges}(T[7:22]), \text{CE}(T[7:22]))), \text{CE}(T[7:22]) - \text{AbsSum}(T[7:22]) + \text{Mean}(T[7:22]))$

In Figure 5.7 we can see that the algorithm selects specific groups of functions and operators for each class of the data-set. As reflected by the bar heights, the algorithm finds that class 1 is mainly characterised by the coefficient of an auto-regressive model (AR), and a measure of complexity (CE). The cyclic component of class 2 is captured through the FFT and the auto-correlation functions. Classes 3 and 4 requires similar features like max, min, and the mean. In addition, class 4 is characterised by the linear trend. Finally, classes 5 and 6 require functions like linear trend, max, min, mean, and number of values above and below zero. Finally, the algorithm most often relates multiple functions through the AQ operator for all classes.

Concluding, as mentioned before, features are standardised (with reference to the training set) hence as shown in Figure 5.8 our algorithm tends to concentrate the samples of the normal class around the origin even though this is not explicitly required. This aspect shows that our algorithm could also work well using “simple” classifiers, for instance a radial basis function classifier which has the advantage of requiring only a single hyper-parameter, i.e. a distance threshold from the origin.

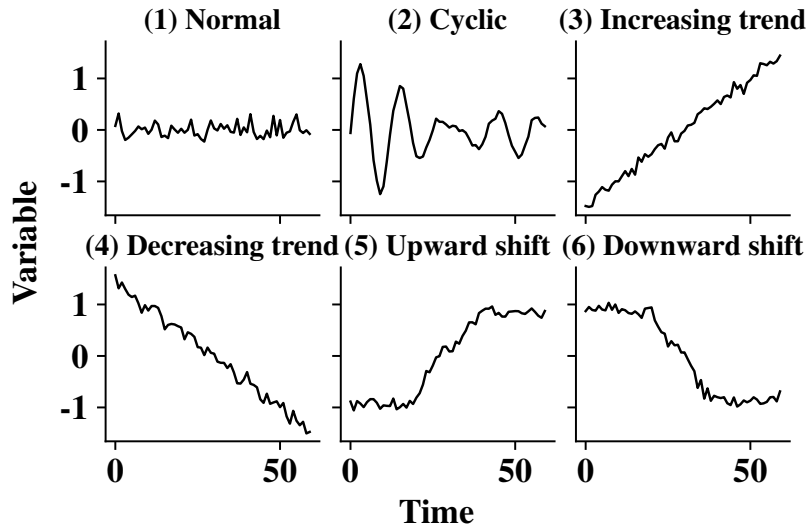


Figure 5.6: A time series per each of the 6 classes of the “SyntheticControl” data-set.

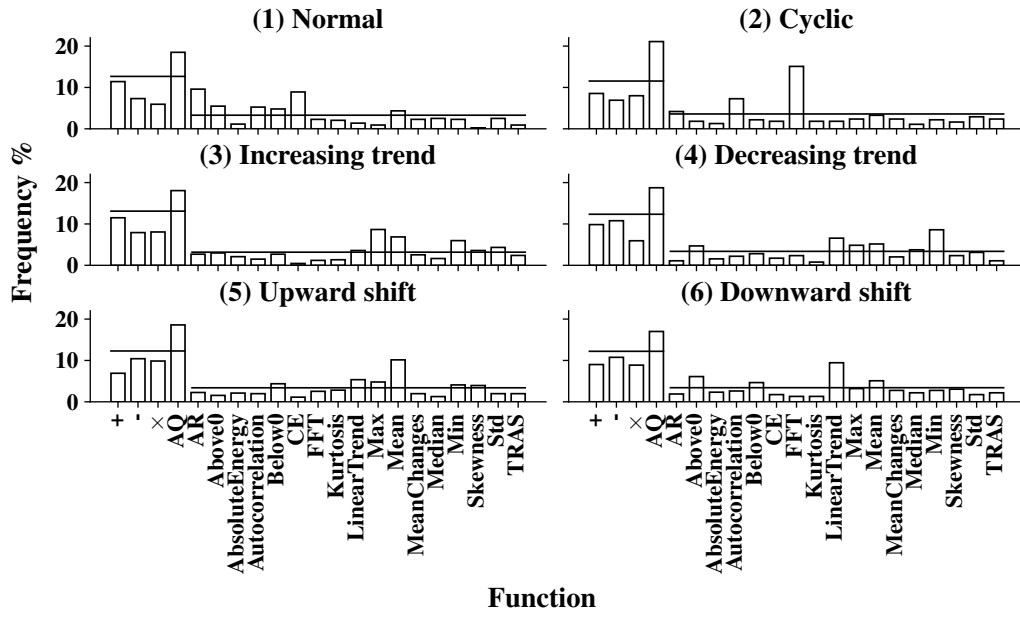


Figure 5.7: Selection frequency of the grammar primitives for the first two features extracted from each of the 6 classes of the "SyntheticControl" data-set. The horizontal lines represent the bar heights we would observe if feature-extractors were generated at random i.e. without regard to fitness.

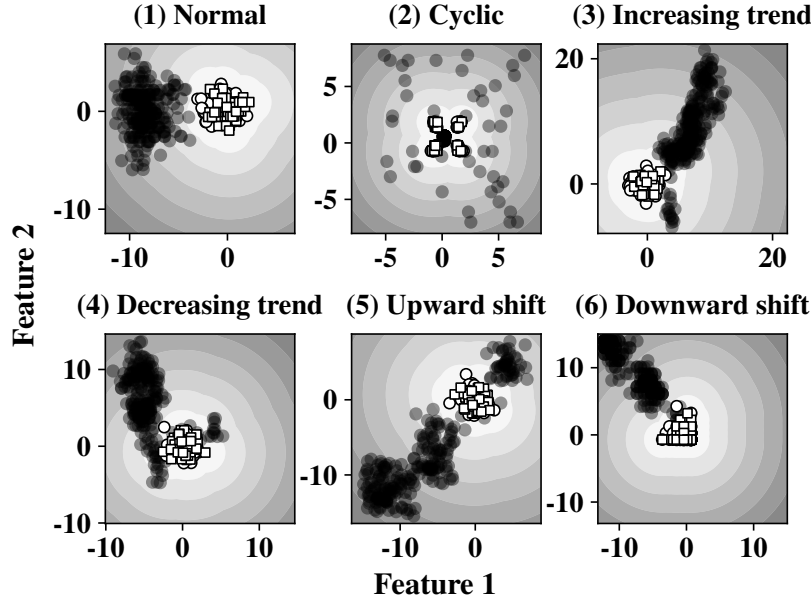


Figure 5.8: 2D feature-based representation of each class of the “SyntheticControl” data-set. □ Training samples. ○ Normal test samples. ● Anomalous test samples. Shading represents distance from training set.

5.5.2 The Sub-sequences from which to Extract

To offer insight into which sub-sequences are most useful we consider the “GunPoint” data-set [258], and our “AccelerometerData” data-set [168].

Figure 5.9 is derived by averaging over the two classes of the “GunPoint” data-set. In this case, our algorithm over-selects two specific sub-sequences. The most frequently selected one corresponds to the sub-sequence underlying the large peak at the begin of the dotted line. The other corresponds to the sub-sequence underlying the small peak towards the end of the dotted line. Ye and Keogh [258] demonstrate that the time series of this data-set can be classified with high performance using “shapelets”. Shapelets are sub-sequences that are maximally representative of a class according to a pre-defined criterion [258]. In another study related to shapelets Hills et al. [104] show that there are two important sub-sequences in the “GunPoint” problem, the same are identified by our algorithm. Also, they report that the top five most important shapelets extracted by their algorithm are related to the end of the time series. However, our algorithm over-selects the sub-sequences at the beginning of the time series. This means that our algorithm is able to target sub-sequences that are relevant for the problem, but overall it works

in a way that is different from shapelets. Thus, our representation could be a good addition to a representation based on shapelet-transform [104].

In Figure 5.10, we show all the sub-sequences of the “GunPoint” data-set related to the time interval $[0 : 40]$. This is the most selected time interval according to the analysis related to Figure 5.9. Sub-sequences are divided per class: part (a) and (b) of the figure show the sub-sequences related to class 1 and 2 respectively. The figure provides insights into the sub-sequences that are important for classification according to our algorithm. Visual assessment of the sub-sequences allows us to claim that most time series could be correctly classified if we were able to catch shape dissimilarities between the two classes in the considered time interval. As shown in Table 5.2, our algorithm achieves approximately 90% AUROC on this data-set. Thus, not only our algorithm is able to detect important sub-sequences, but also it is able to extract useful features.

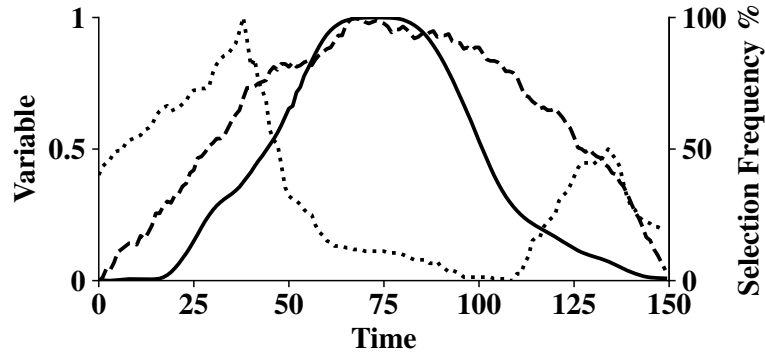


Figure 5.9: Selection frequency of each point within a time series of the “GunPoint” data-set. — Average time series from the data-set. --- Frequencies we would observe if feature-extractors were generated at random i.e. without regard to fitness. Frequencies as per our algorithm.

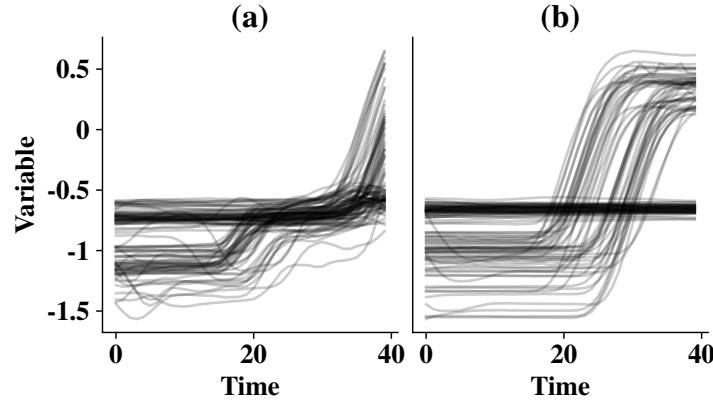


Figure 5.10: Sub-sequences of the “GunPoint” data-set related to the time interval $[0 : 40]$. Part (a) and (b) show the sub-sequences related to class 1 and 2 respectively.

Finally, in Figure 5.11 considers all the nine classes of the “AccelerometerData” data-set. We can see that each person is characterised by her/his activity during a specific part of the day. In the time axis 0 corresponds to 0am and 1440 to 11.59pm. Most subjects are characterised by their activity between 6.30-8.30am, presumably the time they wake up and go to work. Two subjects are characterised by their activity during 0-6.30am (1,6). Also, for three subjects (6,7,9) the activity between 9-11pm seems to be important for their classification.

5.6 Conclusions

We propose a data-driven evolutionary algorithm for feature extraction from time series. The goal is to find a low-dimensional feature-based representation that enables good one-class classification performance with minimum human intervention.

The search for suitable solutions is guided by our grammar specifically defined for time series. In fact, our algorithm can select both the features to extract, and the sub-sequences from which to extract them. Both aspects can be key to TSC problems, as revealed by several related studies.

Evolved features are composed of high-level functions e.g. mean, standard deviation, etc. By choosing the number of features to be extracted a high-dimensional time series can be reduced to a feature-vector of arbitrary dimensionality. This not only lowers computational complexity at prediction time, but also allows the visualisation of time series data-sets. Furthermore,

the analysis of the functions/sub-sequences that are most frequently selected during the evolutionary process enables understanding about the problem at hand. Finally, our algorithm enables better classification performance than considered benchmark methods.

Future work could investigate further the generalization capabilities of our algorithm in a one-class classification scenario where new classes appear at prediction time. Another avenue of research could investigate other primitives to be included in the grammar. A remaining issue of our algorithm is the tendency to overfit. This suggests that our results could be improved, an interesting topic for future work.

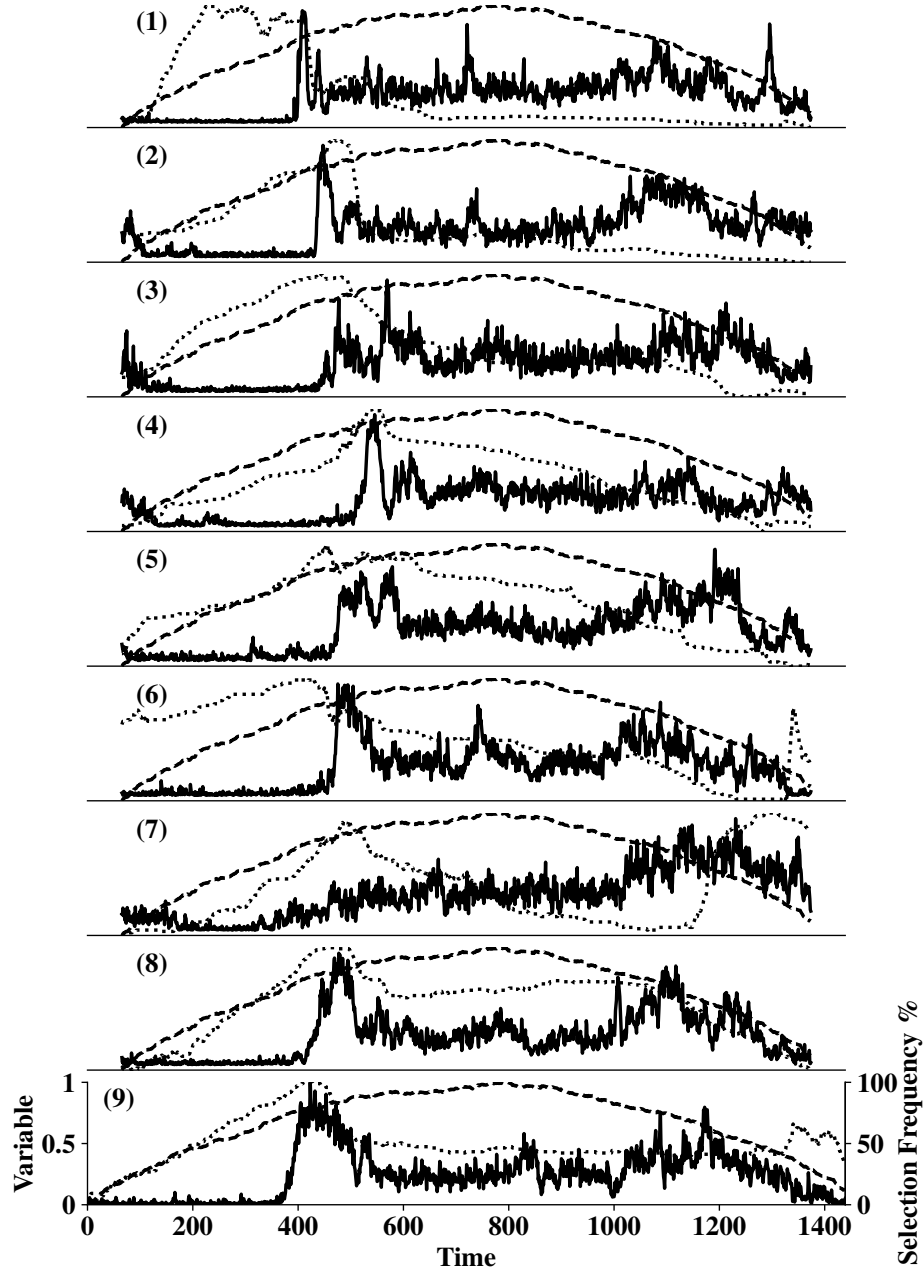


Figure 5.11: Selection frequency of each point within a time series of the “AccelerometerData” data-set. The number in brackets in the top left corner of each plot corresponds to a different class of the data-set.

Chapter 6

Auto-Encoder-Based Representations

In this chapter we investigate simple neural networks as tools for time series representation learning. We use both auto-encoder and encoder-only architectures to learn a representation that not only is of substantially lower dimensionality with respect to the time series length, but also is able to preserve the complex non-Euclidean neighbourhood structure of the original space according to a dissimilarity measure of choice e.g. DTW [205]. Our approach to representation learning [19] draws from multi-dimensional scaling (MDS) [42], and Siamese networks [25].

In our approach, we leverage neural networks as function approximators [107] to learn elastic dissimilarity measures (Section 6.2.1) that are central to time series classification [12], as already discussed in Chapter 2 and in Section 3.5. We require simple neural networks (Section 6.1.2) to map time series to a latent space where the Euclidean distance (ED) between samples approximates their dissimilarity in the original space according to a measure of choice. Then, we use the latent representations in the context of one-class classification [170] (Section 6.2.2) with a 1NN classifier equipped with the ED. In principle, if pairwise dissimilarities in the original space were perfectly preserved in the latent space our approach would allow the same classification performance as a 1NN classifier on raw data when equipped with the same measure used to derive the representation.

To preserve the topological structure of data is fundamental for time series classification [2], and this is best informed by non-Euclidean measures like DTW [12]. The measurement of the dissimilarity between time series is required by several other time series mining applications like clustering [5], indexing [203], motif discovery [153].

In particular, elastic measures, like DTW, are of interest because of their

ability to align two time series that are misaligned or warped with respect to each other, leading to better classification performance than lock-step measures like ED [12]. We pose here an interesting new challenge for neural networks: to learn elastic dissimilarity measures for time series. This is an ambitious objective as most elastic measures are dynamic programming algorithms able to deal with misalignment that can appear everywhere along the time axis and with varying degrees of warping. Studies in many different applications have shown that neural networks can approximate dynamic programming algorithms [201], but not on elastic measures for time series. At the present stage, we are not learning elastic dissimilarity measures in their general form, but we are learning an embedding that preserves pairwise dissimilarities in the context of a given data-set. Nevertheless, our work sets an important first step towards neural approximation of elastic dissimilarity measures for time series. This idea, can allow nearest neighbour classification, a fundamental approach to time series classification [2, 187], at scale and in resource-constrained environments.

Our main contributions can be summarised as follows. As outlined in Section 2.3.1, to the best of our knowledge there is no comprehensive study investigating auto-encoders as tools for representation learning in the context of time series classification. We address this gap by evaluating a variety of auto-encoder and encoder-only architectures on all the 85 data-sets of the first version of the UCR/UEA archive [46] plus a proprietary data-set [168]. We demonstrate that our approach, designed to achieve a dissimilarity-preserving transformation (isometry), gives the possibility of computing geometric information of interest in the learned coordinate space. In fact, our work lies at the intersection of manifold [164] and metric learning [56]. Specifically, we show that learned representations allow classification performance that is close to that of raw data, but in a space of substantially lower-dimensionality. This implies remarkable savings in terms of computational and storage requirements for nearest neighbour time series classification. Finally, our approach enables visual exploration of time series data-sets.

The chapter is organised as follows. In Section 6.1, we present our approach to time series representation learning. In Section 6.3, we report and analyse our results. Finally, in Section 6.4 we summarise our conclusions and discuss future work.

6.1 Proposed Method

We provide an overview of the proposed method in Section 6.1.1. Then, we present all the considered neural network architectures in Section 6.1.2.

6.1.1 Overview

We propose to embed time series in a latent space where pairwise Euclidean distances between samples are equal to pairwise dissimilarities in the original space, for a given dissimilarity measure. To this end we use auto-encoder and encoder-only neural networks.

An auto-encoder is an unsupervised neural network that learns to compress and reconstruct data with minimal distortion [13]. It consists of two components that are connected through a middle layer. Usually, the middle layer is of substantially lower dimensionality than the input. The first component is referred to as the *encoder* (f), and its purpose is to compress the input. The second component is referred to as the *decoder* (g), and its purpose is to reconstruct the compressed input. In encoder-only architectures there is no decoder.

Given an input time series T the auto-encoder outputs a time series $T' = g(f(T))$. Typically, the auto-encoder is trained so that the mean squared error (MSE) between the input and the output is minimised. This loss function is referred to as the reconstruction loss and it is presented in Eq. 6.1 for a time series T of length L .

$$\mathcal{L}_{\text{Rec}} = \frac{1}{L} \|T - T'\|_2^2 \quad (6.1)$$

Our main objective is to preserve pairwise dissimilarities between time series in the latent space for a given dissimilarity measure (d). Specifically, we want to approximate d in the latent space using the ED. This requires us to define another loss function that is presented in Eq. 6.2 for all the pairs i, j of a training batch of n time series T and their latent representation $t = f(T)$. We refer to this loss function as the dissimilarity-preserving loss. This is equivalent to the MSE between the dissimilarity matrix induced by d in the original space, and the dissimilarity matrix induced by the ED in the latent space.

$$\mathcal{L}_{\text{DissP}} = \frac{1}{n^2} \sum_{i=1}^n \sum_{j=1}^n (d(T_i, T_j) - \|t_i - t_j\|_2)^2 \quad (6.2)$$

We use \mathcal{L}_{Rec} and $\mathcal{L}_{\text{DissP}}$ together or separately in a variety of experiments that are detailed in Section 6.2. It is common practice to add a regularisation term to the loss function of a neural network to penalise large weights, but we avoid this to better isolate the effect of \mathcal{L}_{Rec} and $\mathcal{L}_{\text{DissP}}$ on performance.

6.1.2 Neural Network Architectures

We evaluate several neural network architectures that are collectively presented in Table 6.1. The first distinction we consider is between auto-encoder and encoder-only architectures. As stated before, our main concern is to embed time series in a low-dimensional space where pairwise dissimilarities are preserved and approximated by the ED. To this end we take inspiration from the typical auto-encoder architecture, but in principle we could achieve the same objective using only the encoder part. Thus, we evaluate both alternatives: auto-encoder architectures where the loss function is equal to the sum of \mathcal{L}_{Rec} and $\mathcal{L}_{\text{DissP}}$, and encoder-only architectures where the loss function is given by $\mathcal{L}_{\text{DissP}}$ only.

Continuing, we compare dense and convolutional networks. Dense networks are the most typical architecture [90]. A dense network is just a sequence of dense layers organised following the instructions provided in Section 6.2.4. Differently, convolutional networks are less straightforward to design because a variety of architectures are possible. We illustrate our convolutional architecture in Figure 6.1.

When building a convolutional auto-encoder for univariate time series we have to consider two main issues: (1) how to reduce input dimensionality, and (2) how to deal with multiple filters.

(1) An auto-encoder gradually compresses the input down to the middle layer. One can design a convolutional auto-encoder using only convolutional layers, and can reduce input dimensionality by setting the stride parameter to a value that is greater than one [229]. Another possibility is to use no padding, so that the kernel size causes the reduction. Alternatively, input dimensionality can be reduced using pooling layers. As shown in Figure 6.1, we use max-pooling layers. This type of layer can help translation invariance [90] that can be useful in the context of time series classification [258]. The (approximate) inverse of a max-pooling layer is the unpooling layer [11] which we use in the decoder part.

(2) A convolutional layer with a single filter transforms a time series into another time series by applying a sliding filter. Thus, by using several filters a univariate time series is transformed into a multivariate one with a number of channels equal to the number of filters applied. When it comes to the latent layer, we may wish to reduce to a univariate time series, and we can do this by applying a 1×1 convolution [232]. Alternatively, we could concatenate all the channels to a single vector and use a dense layer to achieve the desired dimensionality. However, we avoid this as the dense layer would add a large number of extra weights to be learned.

Finally, we use two dense layers, one in the encoder, and one in the de-

coder. In the encoder, one dense layer is needed after the last convolutional layer to ensure that the input is compressed to the desired latent dimensionality. The output of this layer is the latent representation that we use in our classification experiments. Thus, our latent representation is not a time series. In the decoder, the first layer is dense and it is needed to recover the dimensionality of the last convolutional layer of the encoder. However, it is possible to recover the original time series length only by doubling the dimensionality of the last convolutional layer of the encoder the same number of times as we halve it by pooling in the encoder, which in our implementation is at most two.

In addition to dense and convolutional architectures, we evaluate a variational auto-encoder [128]. This neural network is designed to learn the probability density function of the input data [128]. In contrast, non-variational architectures discussed before do not have a probabilistic foundation. The variational auto-encoder relies on Bayesian inference and it assumes that the relation between input data T and its latent representation t is described by a distribution p_θ . Specifically, $p_\theta(t)$ is the prior distribution, $p_\theta(T|t)$ is the distribution of the input given the latent representation (the likelihood), and $p_\theta(t|T)$ is the distribution of the latent space given the input (the posterior). The objective of the network is to learn the parametrisation θ . However, instead of learning $p_\theta(t|T)$ that is typically intractable, the posterior distribution is approximated using a Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$. The loss function of the variational auto-encoder is shown in Eq. 6.3.

$$\mathcal{L}_{\text{VAE}} = \mathcal{L}_{\text{Rec}} + \frac{1}{2} \mathbb{E}_{t \sim \mathcal{N}(\mu, \sigma^2)} [\mu^2 + \sigma^2 - 1 - \log(\sigma^2)] \quad (6.3)$$

Architecture	Loss Function
AE	\mathcal{L}_{Rec}
CAE	\mathcal{L}_{Rec}
VAE	\mathcal{L}_{VAE}
CVAE	\mathcal{L}_{VAE}
DissPAE	$\mathcal{L}_{\text{Rec}} + \mathcal{L}_{\text{DissP}}$
DissPCE	$\mathcal{L}_{\text{Rec}} + \mathcal{L}_{\text{DissP}}$
DissPE	$\mathcal{L}_{\text{DissP}}$
DissPCE	$\mathcal{L}_{\text{DissP}}$

Table 6.1: Considered neural network architectures. Acronyms that contains a “C” refer to convolutional networks, otherwise the network is dense. Acronyms that contains “DissP” refer to networks that use the $\mathcal{L}_{\text{DissP}}$ loss function. Finally, acronyms that contains a “V” refer to variational networks.

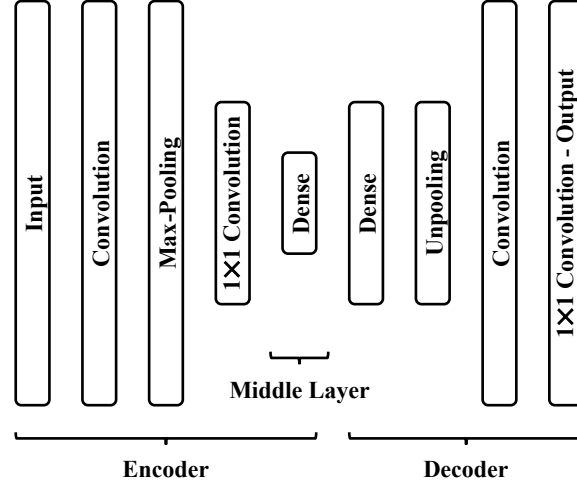


Figure 6.1: Diagram of the proposed convolutional auto-encoder for univariate time series.

6.2 Experimental Design

In this section we provide all the necessary background needed to implement our experiments. In Section 6.2.1, we present the considered dissimilarity measures. In Section 6.2.2, we present the classification task we are investigating, and the data-sets in use. In Section 6.2.3, we discuss how we benchmark our results. Finally, in Section 6.2.4, we provide all the implementation details related to the considered neural network architectures.

6.2.1 Dissimilarity Measures

We try to learn an embedding that preserves pairwise dissimilarities according to each of these three measures: DTW, ED, and move-split-merge (MSM). DTW, and MSM are elastic dissimilarity measures able to account for distortions in the time axis in order to find the alignment that allows the minimum dissimilarity between two time series. DTW is of interest as it is extensively studied in the literature [209]. On the other hand, MSM has demonstrated slightly better performance in both supervised and semi-supervised time series classification experiments [12, 170]. While DTW is not a metric as it violates the triangle inequality, MSM is [230]. The ED is also a metric, and a useful baseline [187]. Further details about these measures are provided in Section 3.5.

6.2.2 Classification Framework and Data-Sets

In this study, first, only the samples related to a single class are used to learn the representation. Then, in the learned space, only the samples of the class used to derive the representation are used to train the one-class classifier in use. The classifier is a one-class 1NN, i.e. it is a threshold on the ED distance to the nearest neighbour in the latent space. We evaluate classification performance using the AUROC.

We evaluate all the architectures on the 85 data-sets of the first version of the UCR/UEA archive plus a proprietary data-set [168] related to a subject authentication problem through accelerometer data. All the data-sets are partitioned into labelled training and test sets and have previously been examined in several supervised time series classification experiments [12]. All the time series are univariate, contain only real numbers, have a fixed length within a given data-set, and are z-normalised [205]. We normalise all the time series to unit norm as we notice that this has a positive impact on training time and no major impact on performance. Also, we normalise the dissimilarity matrix of training data to unit norm. This helps to align the scale of the ED between latent representations with the original dissimilarities we aim to approximate.

6.2.3 Benchmark Methods

As mentioned before, if pairwise dissimilarities in the original space were perfectly preserved in the latent space our approach would allow the same classification performance as a 1NN classifier on raw data when equipped with the same measure used to derive the representation. Hence, we consider the 1NN classifier on raw data our *natural* benchmark.

6.2.4 Implementation Details

In this section we describe all the implementation details regarding the architectures used in this study. Some design choices depend on the fact that we need to evaluate all the architectures on 86 data-sets of time series of different length. For instance, as discussed below, we use at most two max-pooling layers in convolutional architectures. This is because halving some time series more than twice may result in an output length that is too short for any meaningful convolutional filter length.

For each architecture we investigate different numbers of layers: $\{1, 3, 10\}$. In the case of dense architectures, the number of layers is counted as the number of dense layers between the input and the middle layer inclusive. In the

case of convolutional architectures, the number of layers is counted as the number of convolutional layers between the input and the middle layer. Other types of layer, either max-pooling, 1×1 convolution, and dense layers are excluded from this count (Figure 6.1). For what it concerns the latent dimensionality we investigate the following alternatives: $\{2, \lceil \log_2(L) \rceil, \lceil 2 \times \log_2(L) \rceil\}$.

In dense architectures the number of neurons per layer is determined using the integer part of a geometric progression between the input size and the required latent dimensionality. In convolutional architectures we reduce input dimensionality using max-pooling layers with kernel length of two, and stride of two. The max-pooling function takes the maximum value over non-overlapping windows of size two. The unpooling function simply repeats each value twice. We use two max-pooling/unpooling layers if the number of network layers is set to a value that is greater than one, a single max-pooling/unpooling layer otherwise. As each max-pooling layer halves its input length, if such length is not divisible by two the max-pooling function crops the input by one on the right hand side. This cause a mismatch between the input and the output length of an auto-encoder: a time series of uneven length would be reconstructed as a time series of even length. In order to avoid this issue for all the convolutional networks, considering that we have at most two max-pooling layers, if $x = (L \bmod 4)$ is not zero than we crop x time steps from the right hand side of the time series. As x can be at most three we argue that this pre-processing step has little impact on performance.

We place the first max-pooling layer after the first convolutional layer. The second max-pooling layer is placed after the convolutional layer which is halfway through the layers. Each convolutional layer uses zero padding (to ensure that output has the same size as input [62]), and has 16 filters. The kernel length is set to 3% of the length of the input to the layer, rounded down (with minimum kernel length set to three).

Our implementation is similar to that of related works [70, 118]. For each data-set we report the model that gives the best AUROC performance [170] of five runs [118]. Results per architecture reported in Section 6.3 relate to the average AUROC across all data-sets. The activation function in use is the hyperbolic tangent, but no activation is used on the middle layer. Weights are initialised using the Glorot initialisation scheme [88]. As mentioned before, we avoid weight regularisation to better isolate the effect of the considered loss functions. All the architectures are trained for 1000 epochs via gradient descent using the Adam optimiser [127], and mini-batches of 16 samples. The learning rate is set to 0.001 and it is reduced by a factor of 0.5 whenever the training loss does not improve for 100 consecutive epochs down to a minimum of 0.0001 (the minimum improvement is set to 0.0001).

6.3 Results

An overview of the experimental results is provided in Section 6.3.1. In Section 6.3.2, we show that our algorithm can approximate DTW. Finally, we demonstrate that our approach can enable visualisation of time series data-sets (Section 6.3.3).

6.3.1 Overview

We begin our analysis with Table 6.2 that shows results in terms of AUROC averaged across all the considered data-sets for all the architectures and the hyper-parameter configurations. It appears that the number of network layers does not have a major impact on performance, however performance increases as the latent dimensionality increases peaking at the largest value of this hyper-parameter that is $2 \times \log_2(L)$. In dimensionality reduction it is important to know the intrinsic dimensionality of data [165], that can be seen as the minimum number of dimensions needed to describe data and avoid pathological results. Table 6.2 shows that increasing the latent dimensionality has a positive impact on performance. We relate the latent dimensionality with the time series length through the \log_2 function, however more advanced approaches are available in the literature and their investigation warrants further research [240]. The best performance we achieve is 77% AUROC. For what it concerns our dissimilarity-preserving architectures this result is achieved by different architectures with multiple hyper-parameter configurations. Conversely, among the typical auto-encoders only the CAE achieves 77% AUROC. In terms of architectures there is not any that stands out to be the most consistent across different hyper-parameter configurations as we can see from the column “Avg.” of Table 6.2.

To better understand our results we need to compare them with the performance of raw data. As mentioned before, if pairwise dissimilarities in the original space were perfectly preserved in the latent space our approach would allow the same classification performance of a 1NN classifier on raw data when equipped with the same measure used to derive the representation. For the three considered dissimilarity measures, raw data allows the following performances: 79% AUROC with DTW, 77% AUROC with ED, 80% AUROC with MSM (Table 6.4). Our dissimilarity-preserving architectures achieve at most 77% AUROC with each of the three dissimilarity measures. We are very close to the performance of raw data for DTW, and we achieve the same performance as raw data with the ED. However, we are not able to achieve the same performance of MSM by a 3% margin.

As stated before, our objective is to embed time series in a low-dimensional

# Layers	1	3	10	1	3	10	1	3	10	
Latent Dim.	2			$\log_2(L)$			$2 \times \log_2(L)$			
Architecture										Avg.
AE	70	69	68	75	75	73	76	76	74	73
CAE	69	69	68	76	75	74	77	76	75	73
VAE	70	70	72	75	74	74	75	75	75	73
CVAE	66	66	65	69	68	69	70	69	69	68
Avg.	69	69	68	74	73	73	75	74	73	

Architecture	DTW									Avg.
DissPAE	69	70	69	75	76	75	76	76	76	74
DissPCE	68	68	68	76	76	75	77	77	76	74
DissPE	69	69	69	75	75	75	76	76	76	74
DissPCE	68	69	68	75	76	76	76	77	77	74
Avg.	69	69	69	75	76	75	76	77	76	

Architecture	ED									Avg.
DissPAE	68	69	69	76	76	75	76	76	76	73
DissPCE	68	68	68	76	76	75	77	77	76	73
DissPE	68	69	69	75	75	75	75	76	75	73
DissPCE	67	68	67	76	76	76	76	77	77	73
Avg.	68	69	68	76	76	75	76	77	76	

Architecture	MSM									Avg.
DissPAE	69	69	69	76	76	75	76	76	76	74
DissPCE	68	68	68	76	76	75	77	77	76	73
DissPE	69	70	69	75	75	75	76	76	76	73
DissPCE	68	68	67	76	76	76	76	76	77	73
Avg.	69	69	68	76	76	75	76	76	76	

Table 6.2: Summary of results. The table shows AUROC averaged across all the data-sets and rounded to the nearest integer. Table cells get darker as the AUROC increases. For each architecture, results are broken down by hyper-parameter configuration (number of layers, latent dimensionality where L is the time series length, and dissimilarity measure).

space where pairwise dissimilarities are preserved and approximated by the ED. To evaluate how well we achieve this goal we divide data into three subsets: training data (TR), positive test data (TE^P), and negative test data (TE^N). Then, we calculate the Pearson correlation (R^2) between pairwise dissimilarities in the original space and in the latent space. Dissimilarities in the original space are calculated according to a measure of choice, while dissimilarities in the latent space are calculated using the ED. Even if typical auto-encoders (e.g. AE) are not explicitly required to preserve pairwise dissimilarities it is still possible to correlate pairwise dissimilarities in their latent spaces (in terms of ED) with pairwise dissimilarities in the original space (in terms of a dissimilarity measure of choice e.g. DTW). All results are presented in Table 6.3.

First we consider the correlation of pairwise dissimilarities of training data in the original space and in the latent space (TR-TR). Results show that for each dissimilarity measure our architectures achieve an average R^2 that ranges in $[0.81, 0.92]$. Best average results are achieved by encoder-only architectures with the DissPCE getting the maximum R^2 of 0.95. For what it concerns typical architectures that are not trained to preserve pairwise dissimilarities the average R^2 is around 0.6 for the AE/CAE, and around 0.35 for the VAE/CVAE. However, there is one notable exception, the R^2 of the AE/CAE with respect to the distortion of ED. These two architectures seem to preserve ED well ($R^2 \sim 0.85$) without being explicitly required to. Finally, larger networks with more layers, and higher latent dimensionality show better performance.

We can see that for both pairwise dissimilarities between training data and positive test data (TR- TE^P), and training data and negative test data (TR- TE^N), our architectures preserve dissimilarities better than typical auto-encoders by a large margin. For TR- TE^P with DTW and MSM the average R^2 of our architectures ranges in $[0.54, 0.73]$, while for typical auto-encoders it is in $[0.2, 0.48]$. For TR- TE^N with DTW and MSM the average R^2 of our architectures ranges in $[0.44, 0.64]$, while for typical auto-encoders it is in $[0.25, 0.41]$. Again, the AE/CAE preserve ED well without being explicitly required to, and achieve the same R^2 as our architectures. While encoder-only architectures show an advantage at preserving pairwise dissimilarities for training data, it is convolution that makes the difference on test data. In fact, the DissPCE/DissPCE achieve an R^2 that is about 10/15% higher than the DissPAE/DissPE on both positive and negative test data. As per training data, larger networks with more layers, and higher latent dimensionality show better performance. Finally, the DissPCE is the architecture that best preserves dissimilarities on both positive and negative test data, thus we refer to this as our best architecture as it preserves well pairwise dissimilarities on

training data, but also is able to generalise to unseen test data.

In order to provide further details, in Table 6.4 (a)-(b) we show the classification performance, in terms of AUROC per data-set, of a DissPCE and a DissPCE, both with 10 layers and latent dimensionality equal to $2 \times \log_2(L)$, where L is the time series length. We show the performance of these architectures for each dissimilarity measure considered in the chapter. Also, for each dissimilarity measure we show the performance of a 1NN classifier on raw data. Although performances per data-set are highly correlated, for some data-sets it happens that the DissPCE/DissPCE improve on the performance of raw data. This could be due to the interaction of the dissimilarity-preserving and the reconstruction losses. Also, it is possible that elastic measures (DTW, MSM) overfit for some data-sets while this is somehow prevented by our architectures.

In Figure 6.2 for each dissimilarity-preserving architecture and each dissimilarity measure we show the histogram of the differences between pairwise dissimilarities in the original space and in the latent space for all the considered data-sets and all the data (train and test). Pairwise dissimilarities in both the original and the latent space are independently z-normalised in order to be on the same scale. The DissPCE seems to be the architecture that best approximates dissimilarities as differences are mostly distributed around zero. The DissPCE shows similar distributions of differences but slightly more spread. When differences are below zero, it means that architectures tend to expand pairwise dissimilarities, conversely, when differences are above zero, it means that architectures tend to compress them. For the ED and MSM, dense architectures show a long tail on the right-hand side above zero where differences are positive, but most differences are negative. Considering DTW, it seems that all architectures tend to compress dissimilarities as most differences are positive.

Finally, in Table 6.5, we show the correlation between pairwise dissimilarities of raw time series measured according to different dissimilarity measures and averaged across all the considered data-sets. Both DTW and MSM are dynamic programming algorithms defined to find the minimum cost alignment between two time series. They differ slightly in the way they evaluate the cost of aligning two unaligned points. Conversely, the ED is quite different from DTW and MSM as it just compares two time series in lock-step. Thus, we can see that the correlation between DTW and MSM is much higher than their correlation with the ED (+0.26 on average). Considering the ED as a baseline to approximate DTW and MSM we can see, in the second part of Table 6.5, that our DissPCE allows better R^2 . The R^2 allowed by our DissPCE is on average +0.18 larger for DTW, and +0.14 larger for MSM, with respect to the R^2 between the ED and DTW and MSM.

# Layers	1	3	10	1	3	10	1	3	10	Av.																					
	Latent Dim.			$\log_2(L)$			$2 \times \log_2(L)$			Av.																					
	Architecture			DTW			ED			MSM			Av.																		
	1	3	10	1	3	10	1	3	10	1	3	10	1	3	10	1	3	10	1	3	10										
TR-1R	AE	.58	.56	.54	.66	.65	.61	.67	.66	.62	.62	.79	.78	.73	.92	.88	.82	.94	.91	.85	.85	.57	.56	.54	.67	.67	.63	.69	.69	.65	.63
	CAE	.59	.58	.58	.66	.64	.64	.66	.65	.66	.63	.81	.79	.78	.9	.88	.87	.91	.9	.89	.86	.58	.58	.57	.68	.66	.66	.69	.68	.69	.64
	VAE	.2	.21	.22	.29	.3	.3	.33	.34	.35	.28	.21	.23	.24	.3	.31	.32	.35	.36	.37	.3	.23	.23	.25	.34	.35	.35	.39	.4	.41	.33
	CVAE	.25	.27	.26	.41	.45	.46	.5	.53	.53	.41	.27	.3	.29	.47	.48	.51	.54	.56	.58	.44	.29	.31	.29	.48	.51	.51	.56	.59	.6	.46
	DissPAE	.81	.78	.8	.86	.85	.88	.86	.86	.89	.84	.83	.83	.84	.92	.91	.92	.93	.91	.92	.89	.75	.73	.74	.84	.84	.86	.84	.85	.88	.81
	DissPCAE	.79	.79	.8	.85	.87	.88	.87	.89	.89	.85	.84	.84	.84	.85	.89	.92	.93	.93	.93	.9	.74	.74	.74	.82	.84	.85	.86	.86	.87	.81
	DissPE	.85	.87	.89	.92	.94	.95	.91	.95	.95	.91	.8	.83	.85	.89	.92	.93	.9	.94	.94	.89	.79	.81	.83	.89	.82	.92	.93	.9	.94	.88
	DissPCE	.88	.89	.9	.94	.94	.95	.93	.94	.95	.92	.84	.84	.85	.93	.93	.94	.94	.94	.94	.9	.82	.83	.84	.92	.93	.93	.92	.93	.94	.9
	Avg.	.62	.62	.62	.7	.71	.71	.72	.73	.73		.67	.68	.68	.78	.78	.78	.81	.81	.8		.6	.6	.6	.7	.71	.72	.73	.74	.75	
TR-1E ^a	AE	.46	.44	.42	.52	.5	.46	.52	.52	.48	.48	.73	.71	.67	.86	.8	.74	.89	.84	.78	.78	.46	.44	.42	.52	.5	.47	.52	.52	.49	.48
	CAE	.46	.46	.44	.51	.49	.49	.5	.5	.5	.48	.74	.71	.69	.82	.79	.76	.84	.82	.79	.77	.46	.45	.44	.51	.49	.49	.51	.5	.5	.48
	VAE	.14	.13	.17	.23	.21	.2	.25	.22	.21	.2	.18	.16	.2	.27	.24	.23	.31	.27	.25	.23	.17	.16	.18	.26	.23	.23	.29	.26	.24	.22
	CVAE	.15	.14	.14	.25	.26	.26	.32	.32	.32	.24	.16	.15	.15	.26	.27	.28	.33	.32	.33	.25	.17	.16	.16	.28	.29	.28	.35	.35	.34	.26
	DissPAE	.54	.53	.61	.55	.53	.64	.53	.52	.63	.56	.63	.64	.72	.67	.64	.75	.66	.63	.74	.68	.5	.5	.55	.53	.52	.59	.52	.52	.6	.54
	DissPCAE	.62	.62	.61	.68	.7	.69	.7	.71	.67	.71	.67	.74	.73	.73	.81	.81	.81	.83	.82	.79	.57	.56	.56	.63	.64	.66	.66	.66	.62	
	DissPE	.53	.57	.64	.53	.56	.66	.51	.54	.64	.57	.59	.61	.71	.59	.62	.72	.58	.61	.71	.64	.5	.52	.61	.51	.53	.61	.5	.53	.6	.54
	DissPCE	.68	.69	.69	.74	.75	.75	.75	.77	.77	.73	.72	.73	.73	.81	.81	.82	.82	.83	.84	.79	.62	.63	.66	.69	.71	.7	.71	.72	.68	
	Avg.	.45	.45	.47	.5	.5	.52	.51	.51	.53		.56	.55	.57	.64	.62	.64	.65	.64	.66		.43	.43	.45	.49	.49	.5	.51	.51	.52	
TR-1E ^b	AE	.38	.35	.33	.45	.43	.38	.47	.45	.4	.4	.65	.61	.57	.8	.73	.66	.84	.77	.69	.7	.34	.32	.31	.41	.39	.36	.43	.42	.38	.37
	CAE	.37	.36	.34	.43	.42	.41	.45	.44	.44	.41	.63	.61	.56	.75	.71	.68	.77	.75	.71	.68	.34	.33	.31	.4	.39	.4	.42	.41	.42	.38
	VAE	.19	.19	.21	.28	.26	.27	.3	.3	.27	.25	.21	.21	.25	.31	.29	.3	.34	.32	.3	.28	.21	.21	.22	.31	.29	.29	.34	.32	.3	.28
	CVAE	.21	.21	.2	.31	.3	.31	.35	.34	.34	.28	.2	.19	.18	.27	.27	.28	.31	.3	.31	.26	.24	.23	.22	.34	.33	.34	.39	.37	.37	.31
	DissPAE	.43	.44	.49	.46	.46	.55	.46	.46	.54	.48	.52	.54	.62	.59	.57	.66	.59	.57	.65	.59	.4	.39	.43	.44	.46	.47	.45	.46	.48	.44
	DissPCAE	.5	.49	.48	.58	.6	.58	.61	.63	.63	.57	.61	.62	.6	.71	.73	.72	.76	.75	.69	.42	.42	.42	.42	.51	.51	.5	.55	.55	.49	.45
	DissPE	.44	.47	.52	.46	.49	.55	.46	.48	.55	.49	.48	.51	.59	.53	.55	.63	.53	.56	.63	.56	.39	.41	.46	.44	.45	.48	.45	.47	.48	.45
	DissPCE	.57	.57	.56	.65	.67	.67	.67	.69	.69	.64	.59	.6	.59	.73	.74	.75	.75	.76	.77	.7	.49	.5	.52	.58	.59	.59	.61	.67	.61	.57
	Avg.	.39	.38	.39	.45	.45	.46	.44	.47	.48		.49	.49	.49	.59	.57	.59	.61	.6	.6		.35	.35	.36	.43	.43	.43	.45	.45	.45	.45

Table 6.3: The table shows the Pearson correlation (R^2) of pairwise dissimilarities in the original space and in the latent space. Table cells get darker as the correlation increases. Data are divided in three groups: training data (TR), positive test data (TE^{P}), and negative test data (TE^{N}). For each architecture, results are broken down by hyper-parameter configuration (number of layers, latent dimensionality where L is the time series length, and dissimilarity measure).

Data-set	1NN-DTW	DissPCAE-DTW	DisPCE-DTW	INN-ED	DissPCAE-ED	DisPCE-ED	1NN-MSM	DissPCAE-MSM	DisPCE-MSM
AccelerometerData									
Adiac	77	65	57	82	59	63	77	61	65
ArrowHead	91	91	90	90	88	88	91	91	90
Beef	74	76	78	77	77	77	79	79	78
BeetleFly	78	81	82	81	81	81	79	83	83
BirdChicken	70	83	80	71	71	68	80	82	81
Car	73	64	71	58	74	65	77	65	64
CBF	72	68	80	77	75	77	85	77	77
ChlorineConcentration	100	70	73	94	75	76	100	72	73
CinCEGTorso	65	67	68	67	66	68	65	66	69
Coffee	81	91	91	98	96	95	84	90	91
Computers	96	97	99	96	96	96	95	96	95
CricketX	55	55	55	52	53	53	52	54	55
CricketY	90	78	80	83	80	83	89	79	79
CricketZ	91	83	82	87	82	83	90	82	78
DiatomSizeReduction	90	80	79	82	79	80	89	80	77
DistalPhalanxOutlineAgeGroup	99	100	100	100	100	100	100	100	100
DistalPhalanxOutlineCorrect	66	65	65	62	64	64	64	62	66
DistalPhalanxTW	58	59	60	58	60	59	58	59	61
Earthquakes	82	80	80	79	79	81	80	78	80
ECG2000	57	80	80	50	85	82	56	83	82
ECG5000	70	78	77	82	79	80	75	77	77
ECGFiveDays	80	76	83	78	84	85	80	83	84
ElectricDevices	78	56	60	83	59	56	79	54	59
FaceAll	74	71	74	75	66	71	74	63	75
FaceFour	94	86	85	92	85	87	95	85	84
FacesUCR	91	86	84	89	89	90	96	80	93
FiftyWords	96	78	81	89	79	81	96	78	83
Fish	95	84	87	89	83	85	95	83	83
FordA	87	86	86	86	85	85	92	88	87
FordB	52	60	58	57	60	56	52	59	59
GunPoint	57	56	53	55	53	55	55	55	56
Ham	88	88	86	85	86	84	97	86	88
HandOutlines	52	52	54	52	54	53	54	57	55
Haptics	70	68	69	74	62	76	71	57	75
Herring	60	60	59	61	61	62	62	61	61
InlineSkate	53	59	62	52	56	56	55	54	59
InsectWingbeatSound	67	64	65	62	63	62	70	63	64
ItalyPowerDemand	78	90	89	91	91	90	85	89	90
LargeKitchenAppliances	82	85	92	89	91	89	86	83	86
Lightning2	74	60	62	58	58	59	65	64	60
Lightning7	74	64	62	63	63	68	71	64	66
Mallat	87	76	72	74	72	76	88	76	78
	99	97	97	99	98	98	98	98	97

Table 6.4 (a): The table shows the classification performance of a 1NN classifier on raw data for each of the three dissimilarity measures considered in the chapter: DTW, ED, and MSM. Also, the table shows the performance enabled by the latent representation of both a DissPCAE and DisPCE trained to preserve the considered dissimilarity measures. AUROC results, for all the 86 data-sets introduced in Section 3.2, are rounded to the nearest integer. Cells get darker as the AUROC increases.

Data-set	INN-DTW	DissPCAE-DTW	DissPCE-DTW	INN-ED	DissPCAE-ED	DissPCE-ED	INN-MSM	DissPCAE-MSM	DissPCE-MSM
Meat	97	99	100	98	98	98	98	98	99
MedicalImages	93	84	84	90	84	86	91	85	86
MiddlePhalanxOutlineAgeGroup	52	60	59	55	58	57	55	57	61
MiddlePhalanxOutlineCorrect	61	61	65	64	63	66	64	63	67
MiddlePhalanxTW	71	74	73	73	75	73	73	75	73
MoreStrain	78	70	71	75	69	87	85	72	82
NonInvasiveFetalECGThorax1	97	96	97	98	98	98	97	97	97
NonInvasiveFetalECGThorax2	98	98	98	99	98	98	98	98	98
OliveOil	90	80	78	90	80	79	89	81	75
OSULeaf	86	95	90	80	97	96	93	90	94
PhalangesOutlinesCorrect	56	58	57	58	58	56	57	58	57
Phoneme	72	55	54	57	52	53	69	53	54
Plane	100	99	99	100	100	100	100	99	99
ProximalPhalanxOutlineAgeGroup	67	75	75	72	75	75	77	75	72
ProximalPhalanxOutlineCorrect	58	63	62	62	62	62	59	62	64
ProximalPhalanxTW	88	89	90	86	86	85	88	89	89
RefrigerationDevices	58	57	55	53	52	53	58	55	55
ScreenType	55	56	52	51	52	53	55	52	52
ShapletSim	71	53	55	51	51	58	74	55	55
ShapesAll	95	92	92	92	92	92	97	90	89
SmallKitchenAppliances	57	62	62	50	56	53	54	59	59
SonyAIBORobotSurface1	78	80	76	75	74	76	78	77	74
SonyAIBORobotSurface2	78	79	80	79	83	79	81	79	82
StarLightCurves	91	92	92	92	92	92	92	91	92
Strawberry	86	88	83	86	88	85	87	88	87
SwedishLeaf	85	81	86	86	84	86	92	83	86
Symbols	99	97	96	97	96	95	99	97	96
SyntheticControl	99	87	94	89	87	91	98	84	93
ToeSegmentation1	65	61	68	62	57	59	69	57	63
ToeSegmentation2	67	73	69	75	69	71	75	78	67
Trace	99	91	93	94	94	93	98	93	93
TwoLeadECG	87	72	79	78	74	80	90	72	76
TwoPatterns	100	97	71	94	97	91	97	97	77
UWaveGestureLibraryAll	96	98	96	98	98	98	97	98	95
UWaveGestureLibraryX	85	86	85	86	86	87	86	85	81
UWaveGestureLibraryY	87	87	87	87	86	88	87	86	82
UWaveGestureLibraryZ	84	84	83	84	84	84	84	84	80
Wafer	93	99	94	98	99	97	95	100	93
Wine	59	62	60	57	62	59	60	63	61
WordSynonyms	90	75	75	80	76	75	91	73	72
Worms	72	67	72	64	67	64	78	70	74
WormsTwoClass	61	61	60	54	57	59	67	58	59
Yoga	69	70	67	69	70	68	73	70	65
Avg.	79	76	77	77	76	77	80	76	77

Table 6.4 (b)

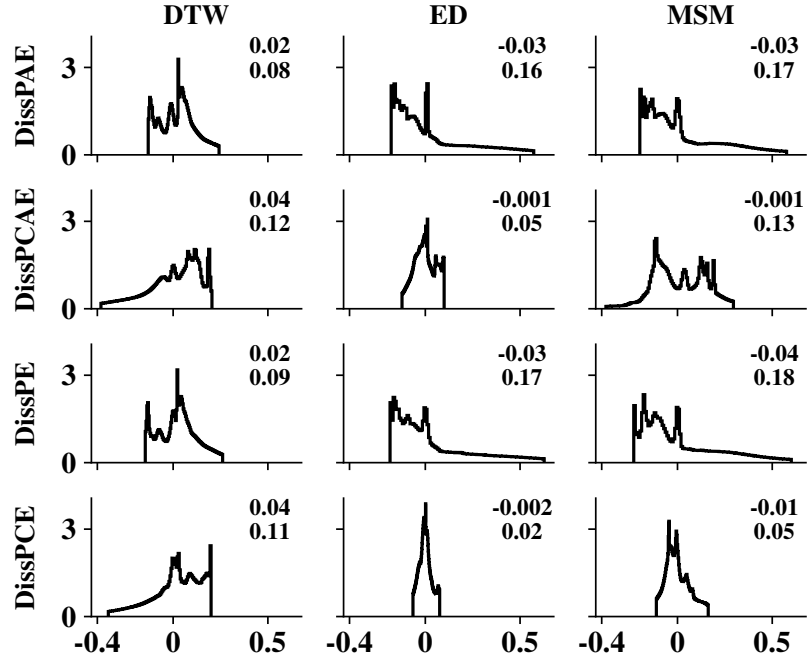


Figure 6.2: Histograms (with a log vertical scale) of the differences between pairwise dissimilarities in the original space and in the latent space for all the considered data-sets. Numbers on the top-right corner corresponds to the mean and standard deviation of differences. Results are broken down by architecture and dissimilarity measure.

Dissimilarities	TR-TR	TR-TE ^P	TR-TE ^N	Avg.
DTW-ED	0.71	0.58	0.56	0.62
DTW-MSM	0.91	0.88	0.85	0.88
ED-MSM	0.74	0.59	0.52	0.62
Architecture	TR-TR	TR-TE ^P	TR-TE ^N	Avg.
DissPCE-DTW	0.95	0.77	0.69	0.8
DissPCE-MSM	0.94	0.72	0.62	0.76

Table 6.5: The table shows the Pearson correlation (R^2) between pairwise dissimilarities of raw time series measured according to different dissimilarity measures on different sub-sets of data. In the second part of the table is shown the correlation between pairwise dissimilarities of raw time series and their latent representation according to a DissPCE with 10 layers and latent dimensionality equal to $2 \times \log_2(L)$, where L is the time series length.

6.3.2 Learned DTW

In this section we illustrate some examples of how time series appear in the latent space (Figures 6.3-6.6), and how dissimilarities in the original space compare to dissimilarities in the latent space (Figure 6.7). We draw these data-sets from among those where the difference, in terms of AUROC, between 1NN-DTW and 1NN-ED on raw data is largest (Tables 6.4 (a)-(b)), in order to select data-sets where DTW has made the difference.

In Figures 6.3-6.6 part (a) we show three time series for a given class of a given data-set. Specifically, we plot the class medoid in terms of DTW dissimilarity (—), its nearest neighbour (\cdots), and its furthest neighbour (\cdots). Then, in part (b) we show the latent representation of these time series according to a DissPCE trained to preserve DTW dissimilarities, except for Figure 6.6 where we use a DissPCAE. For all the architectures we use the configuration with 10 layers, and latent dimensionality equal to $2 \times \log_2(L)$, where L is the time series length.

For all the architectures we propose the latent representation is the output of a dense layer with no activation function. Thus, it is not possible to interpret this embedding as a time series. However, as shown in Figure 6.3 close-by samples of the same class show a similar pattern in their latent representation. In principle, for DTW-preserving architectures we may expect to observe latent representations that account for distortions in the axis, thus are aligned although the input time series are not. In practice it is difficult to evaluate whether this is the case or not because networks do not learn the general DTW function, but only its behaviour in the context of a given class of a given data-set. Also, we notice that the DissPCE tends to represent time series as tightly packed around a certain point as shown in Figures 6.4-6.5. This is not ideal, but also it is not a malfunction as the correlation between pairwise dissimilarities in the original space and in the latent space remains high (i.e. around 0.9 for the considered data-sets). Furthermore, we notice that this behaviour does not occur with the DissPCAE. Comparing Figure 6.5 with Figure 6.6 we can see that the latent representations of the same time series appear tightly packed for the DissPCE, but are clearly separated for the DissPCAE. It is expected that this is the effect of the reconstruction loss. In any case, further research is needed to understand what these latent representations can reveal about original time series besides their dissimilarity to other samples.

In Figure 6.7 we propose an analysis that is somehow equivalent to the analysis of the correlation between pairwise dissimilarities in the original space and in the latent space discussed in Section 6.3.1. However, now we provide some visual insights by focusing only on the training data of each

data-set considered in this section. For each class of each data-set we select the class medoid in terms of DTW dissimilarity. Then, we order all the class samples according to their increasing DTW dissimilarity from the class medoid in the original space. Finally, we use the same ordering to plot the Euclidean distances of samples from the medoid in the original space, and in the latent space according to a DissPCE-DTW. First, we notice that DTW dissimilarities, and Euclidean distances in the original space follow a different pattern. Also, and more importantly, our DissPCE-DTW is able to closely approximate DTW dissimilarities in the latent space. However, the observed correlation between pairwise DTW dissimilarities in the original space and in the latent space could be spurious due to the ED acting as a confounding factor, but Figure 6.7 shows that this is not the case.

6.3.3 Visual Exploration of Time Series Data-Sets

Visualisation of time series data is important to enable understanding and decision making. It is particularly important in the context of several applications where modellers are not simply concerned with class labels but are interested in knowing the causes of the classification outcome (e.g. medical diagnosis through sensor readings [52]). In Figure 6.8, we show that our approach allows us to visualise time series data-sets in a 2-dimensional space.

As shown in Tables 6.2-6.3, when we set the latent dimensionality to two we achieve results, both in terms of AUROC and preservation of pairwise dissimilarities, that are inferior to that allowed by a larger latent dimensionality. However, considering the case with latent dimensionality of size two and 10 layers, we notice that for each dissimilarity-preserving architecture top 30 best performances by data-set achieve an average AUROC that is greater than 80%. This is a good result that implies that a network has learned an embedding that is useful for classification. Thus, in Figure 6.8, we show the 2D representation of three data-sets for each dissimilarity-preserving architecture. We compare our approach with two strong dimensionality reduction techniques i.e. MDS, and t-distributed stochastic neighbour embedding (t-SNE) [165]. For all the approaches we calculate pairwise dissimilarities using DTW on all the training and test data related to a given data-set where a single class is considered as positive and all the others are considered as negative. Also, for each approach we show the correlation (R^2) between pairwise dissimilarities in the original space and in the latent space. All the approaches allow a good separation between the positive and the negative class with the exception of MDS. However, our dissimilarity-preserving architectures have an advantage as they always allow the highest correlation between dissimilarities in the original space and in the latent space.

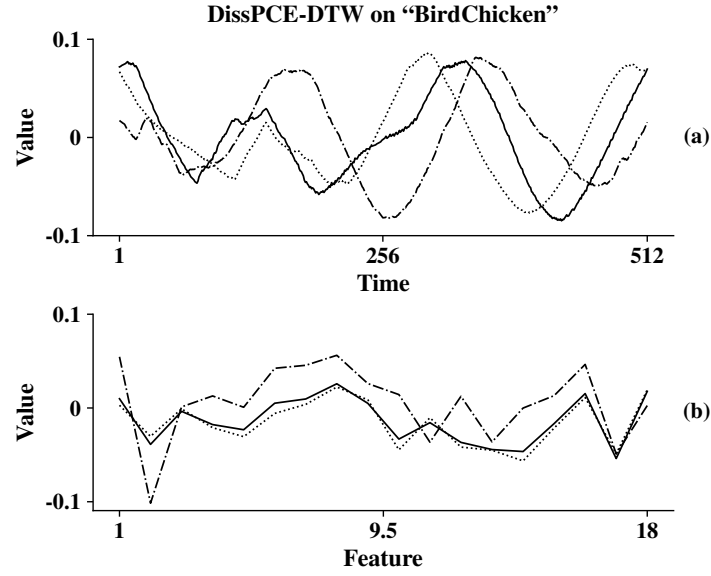


Figure 6.3: Three time series from the "BirdChicken" data-set (class 1) (a), and their latent representation according to a DissPCE-DTW (with 10 layers and latent dimensionality equal to $2 \times \log_2(L)$, where L is the time series length) (b).

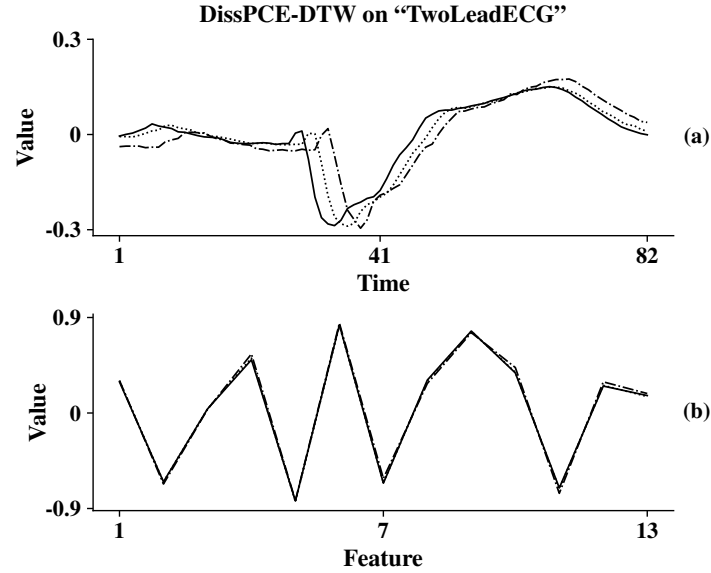


Figure 6.4: Three time series from the "TwoLeadECG" data-set (class 1) (a), and their latent representation according to a DissPCE-DTW (with 10 layers and latent dimensionality equal to $2 \times \log_2(L)$, where L is the time series length) (b).

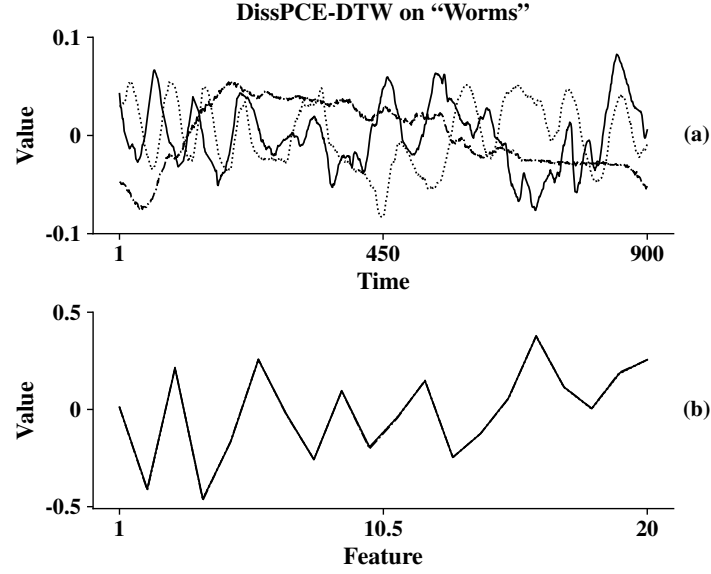


Figure 6.5: Three time series from the “Worms” data-set (class 1) (a), and their latent representation according to a DissPCE-DTW (with 10 layers and latent dimensionality equal to $2 \times \log_2(L)$, where L is the time series length) (b).

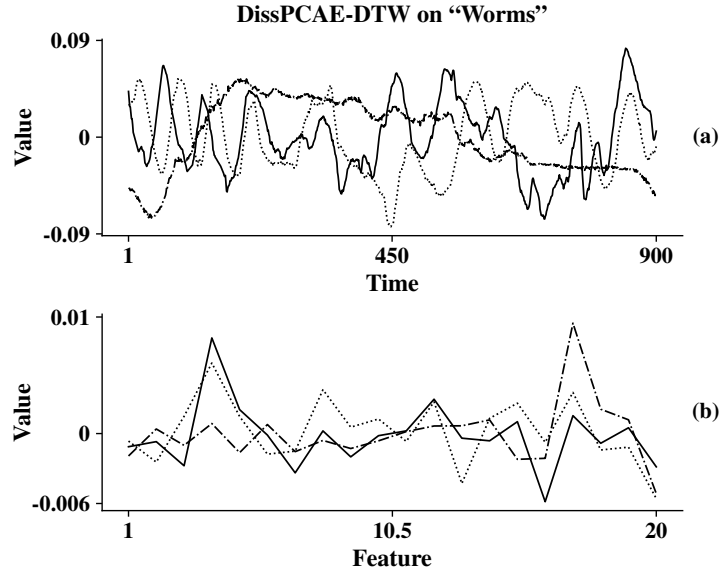


Figure 6.6: Three time series from the “Worms” data-set (class 1) (a), and their latent representation according to a DissPCE-DTW (with 10 layers and latent dimensionality equal to $2 \times \log_2(L)$, where L is the time series length) (b).

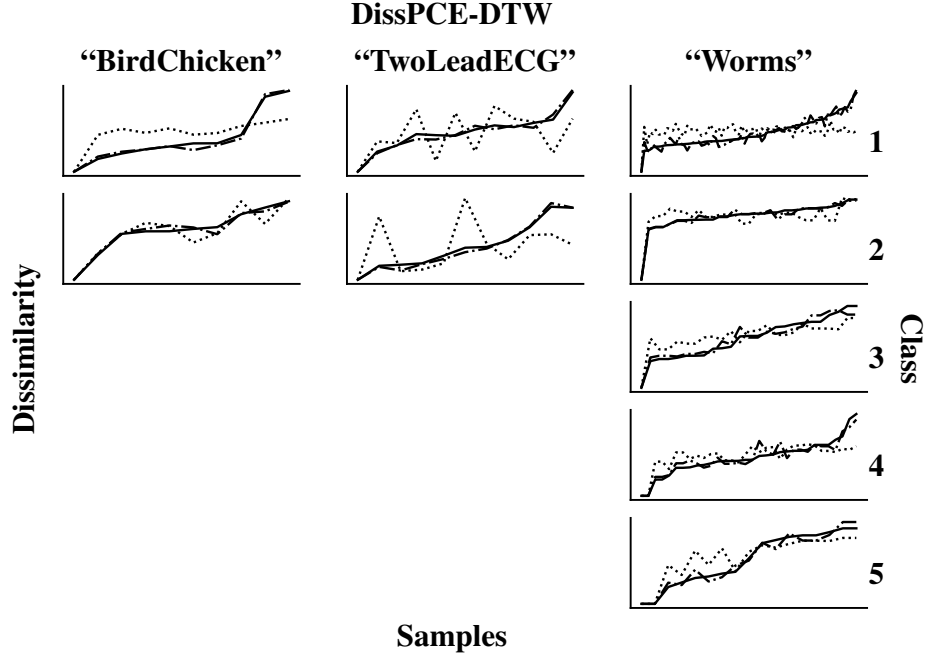


Figure 6.7: Dissimilarities of all training samples from the class medoid for three data-sets. For each class of each data-set samples are ordered according to their DTW dissimilarity from the class medoid in the original space. Then, using the ordering found the following dissimilarities from the class medoid are calculated. DTW dissimilarities in the original space (—), Euclidean distances in the original space (····), and Euclidean distances in the latent space according to a DissPCE-DTW (with 10 layers and latent dimensionality equal to $2 \times \log_2(L)$, where L is the time series length) (- - -). The graph demonstrates that the correlation between DTW dissimilarities in the original space and dissimilarities in the latent space is not spurious due the ED acting as a confounding factor.

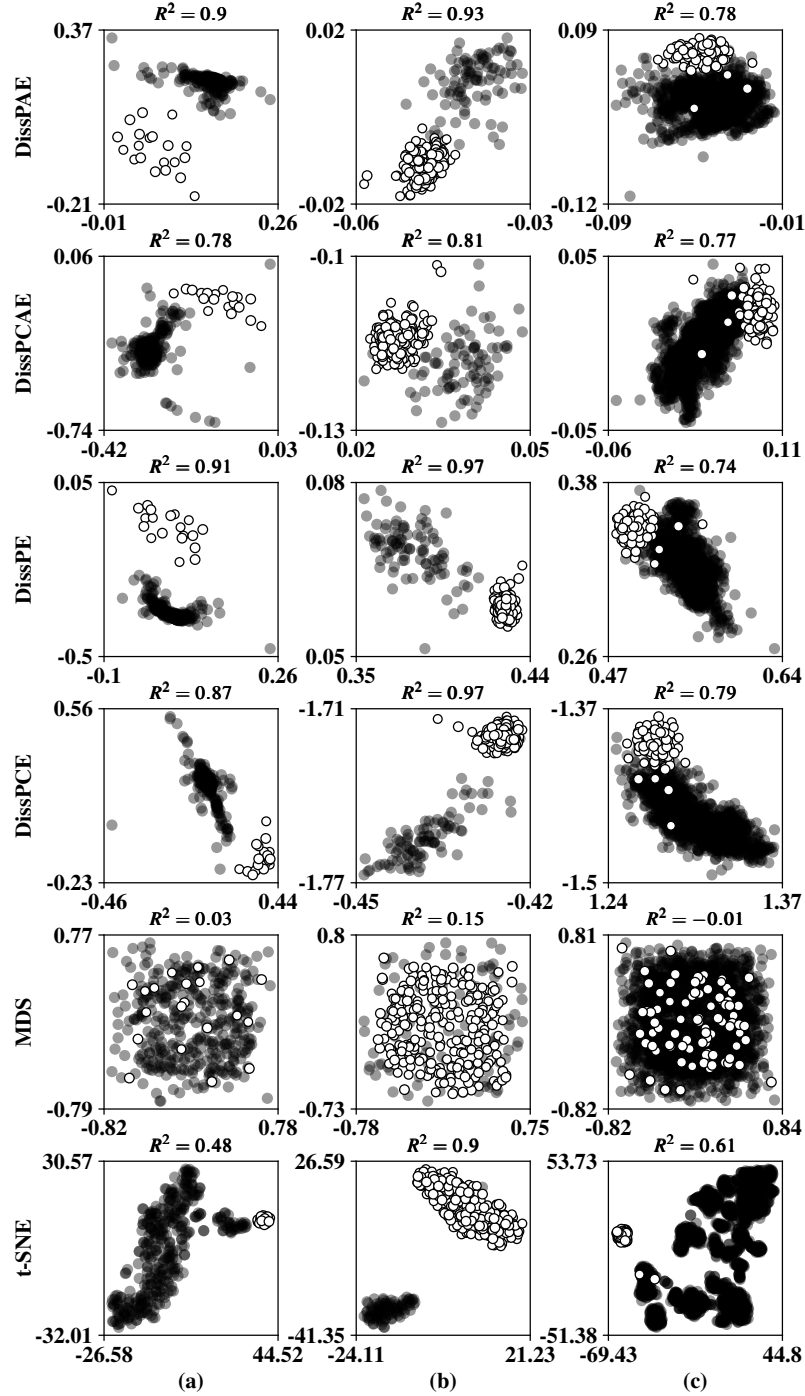


Figure 6.8: ○ Positive samples. ● Negative samples. (a) “Adiac”, positive class: 9. (b) “DistalPhalanxTW”, positive class: 8. (c) “NonInvasiveFatalECGThorax1”, positive class: 33.

6.4 Conclusions

We propose to embed time series in a latent space where pairwise ED between samples are equal to pairwise dissimilarities in the original space, for a given elastic measure. To this end we use simple auto-encoder and encoder-only neural networks. We show that our approach allows savings in terms of computational complexity, with limited impact on classification performance. Specifically, as learned representations have generally remarkably lower dimensionality than the raw data our approach reduces storage requirements. Also, as the geometric structure of data is preserved in latent spaces and approximated by the ED, it is possible to evaluate the dissimilarity between two time series for elastic measures (e.g. DTW) with linear complexity.

We show that dissimilarities between samples in the original space are highly correlated with ED between samples in the latent spaces learned through our approach for both training and test data. Also, we show that the AUROC enabled by representations learned through our approach is close to that achieved by a 1NN classifier on raw data when equipped with the same measure used to derive the representation. The 1NN on raw data sets an upper bound to the maximum performance we could achieve with our approach if pairwise dissimilarities in the original space were perfectly preserved in the latent space. Overall these results confirm that we have succeeded in preserving pairwise dissimilarities in latent spaces.

We evaluate several alternatives in terms of network type (dense, convolutional, and variational), and loss functions (reconstruction loss, and dissimilarity-preserving loss). We find that classical architectures like AE/-CAE preserve the ED between samples well without being explicitly required to. However, in relation to elastic measures like DTW or MSM, dissimilarity-preserving convolutional networks are the best at preserving pairwise dissimilarities. Specifically, the DissPCE is the best architecture in this sense.

The analysis of latent representations reveals that encoder-only architectures tend to represent time series as tightly packed around a certain point. This is not a malfunction as pairwise dissimilarities are well preserved. Further research is needed to understand what these latent representations can reveal about original time series besides their dissimilarity to other samples. However, our approach can provide useful insights by enabling the 2D visualisation of time series. In fact, in a good number of cases latent dimensionalities of size two allow a good separation between positive and negative data enabling 2D visualisation of time series.

Future work could extend our approach to multivariate time series by adapting the convolutional layers to 2D data, and using a suitable dissimilarity measure as for instance multivariate DTW [91].

Chapter 7

Comparing the Representations

In this chapter we provide a concise overview of the different time series representations described in Chapters 4-5-6. In Section 7.1, we report a selection of experimental results to allow a comparative analysis of different representations. In Section 7.2, we propose a small experiment to demonstrate how combining different representations can lead to better classification performance. Finally, we evaluate some common dimensionality reduction techniques in Section 7.3, and we summarise the chapter in Section 7.4.

7.1 Comparing Representations

In this section we compare the different representations discussed throughout the thesis. In our comparative analysis we consider three main aspects: (1) the AUROC performance, (2) the computational complexity (CC) both to achieve the representation, and to classify with a 1-nearest neighbour (1NN) classifier, (3) the compression rate (CR) allowed by the representation as defined in Eq. 7.1 where L is the time series length, and l is the dimensionality of the representation.

$$\text{CR} = \left[1 - \frac{l}{L} \right] \times 100 \quad (7.1)$$

In each experimental chapter (Chapters 4-5-6) we have proposed a different time series representation evaluating several variants. To improve clarity, we report, here, and specifically in Tables 7.1 (a)-(b)-(c), only the most important results. First, we report results related to raw data (RD) as this is a fundamental benchmark for our work. For RD, we do not have a CR as, in fact, we are using the entire time series, thus $l = L$. In relation to dissimilarity-based representations (DBR) we consider the variant

DBR_{20%} because it is the best in the classification performance vs. dimensionality reduction trade-off. In DBR_{20%}, the representation has dimensionality $l = 0.2 \times N$ where N is the number of training samples. Within the variant DBR_{20%}, we select the prototype method “Centers-k-means” because it allows the best average performance across the considered dissimilarity measures. In respect of feature-based representations via grammatical evolution (FBRGE), we select the first three extracted features, thus $l = 3$. Again, this is our best result in the classification performance vs. dimensionality reduction trade-off. Finally, in regard of auto-encoder-based representations (AEBR) we report results related to the “DissPCE” architectures discussed in Section 6.3 as they are the best in preserving pairwise training samples dissimilarities in the latent space. All the selected “DissPCE” architectures allow the same dimensionality $l = 2 \times \log_2(L)$, and have 10 layers.

Most of the representations we propose, as well as time series classification with RD, require a dissimilarity measure to work. In the present discussion we consider three dissimilarity measures. The first one is dynamic time warping (DTW) as this is a measure of central importance for time series classification [12]. Then, we consider the Euclidean distance (ED) as this is one of the simplest measures, but still able to compete with far more complex ones. Finally, we consider the move-split-merge dissimilarity (MSM) as this measure allows the highest classification performance we have recorded.

7.1.1 Results

In Tables 7.1 (a)-(b)-(c) is shown the AUROC performance for all the 86 data-sets introduced in Section 3.2. The tables’ cells get darker as the AUROC increases. Also, the tables show the CC, and the CR allowed by each representation. Where applicable results are broken down by dissimilarity measure. The columns “Min” and “Max” show the minimum and maximum performance on a given data-set regardless of the representation.

Results averaged across all data-sets are shown in Table 7.1 (c). Specifically, the row “Avg.” shows results averaged across all the 86 data-sets, on the other hand, the row “Avg. FBRGE” shows results averaged across the subset of 30 data-sets used in Chapter 5 where we have presented our evolutionary algorithm for feature extraction. The same applies to the rows “St. Dev.”/“St. Dev. FBRGE” and “Skew.”/“Skew. FBRGE” that show the standard deviation and the skewness of results respectively.

The distributions of AUROC performances across data-sets are approximately symmetric. In fact, the skewness of each representation is within the range ± 0.5 (row “Skew.”). The same applies to the subset of problems considered in Chapter 5 where only DBR-DTW appear to be moderately skewed

to the right (row “Skew. FBRGE”). Average performances range between 74% AUROC (DBR-ED) and 80% AUROC (RD-MSM), and between 68% AUROC (DBR-ED) and 77% AUROC (FBRGE) for the subset of problems considered in Chapter 5. A detailed discussion about average AUROC performances is provided in the rest of this section. Finally, AUROC performances across data-sets show large variations indicating that for each distribution some problems are more difficult than others. While most representations have a standard deviation around 14/15% AUROC, AEBr-DTW-MSM and FBRGE have a slightly lower standard deviation around 13% AUROC.

The difference between the best and the worst AUROC performance, shown by the “Min” and “Max” columns, can be as little as 1% AUROC as on the data-set “Plane”, or quite large as 38% AUROC as on the data-set “CinCECGTorso”. This demonstrates that some problems are “simple” to solve, thus the difference between the best and the worst performance is almost nothing. In contrast, for other problems the “right” representation can have a remarkable impact on classification performance.

Concluding, in regard of the CR, FBRGE allow the best average result equal to 98.5% with the lowest standard deviation equal to 1.5%. The second best CR is allowed by AEBr with an average of 96.2% and a standard deviation of 3.1% across all data-sets.

- AUROC Performance

The shading in Table 7.1 reveals that for a given data-set different representations achieve similar AUROC confirming that some problems are just more difficult than others. Nevertheless, a representation can do much better than the others on a case by case basis. AUROC scores between representations are highly correlated with a minimum Pearson correlation of 0.8.

As shown in Table 7.1 (c), RD achieves the best classification performance. For the three dissimilarity measures considered i.e. DTW, ED, MSM, the AUROC is equal to 78%, 77%, and 80% respectively. The second best representation is given by AEBr where results by dissimilarity measure are 77%, 77%, and 77% AUROC. In relation to DTW, the difference is limited to 1%, for the ED we have a tie, however for the MSM dissimilarity the difference is quite large and equal to 3% AUROC in favour of RD. Considering DBR_{20%}, results by dissimilarity measure are 75%, 74%, and 76% AUROC. While these results appear to be lower with respect to RD it is important to recall that the variant DBR_{100%} achieves results by dissimilarity measure equal to 78%, 76%, and 80% AUROC, thus equalling RD. Nevertheless, there are two reasons for us to report results related to DBR_{20%}, and not to DBR_{100%} here. First, DBR_{20%} allow a much higher CR than DBR_{100%} (87.2% vs. 35.8%). Also, and more importantly, DBR_{100%} do not require any prototype method as the representation is derived using all the available

training samples. Conversely, $\text{DBR}_{20\%}$ require a prototype method to derive the representation easing the CC needed to represent as well as to classify. Taking into account scalability, we prefer to focus on $\text{DBR}_{20\%}$, although further research is needed to investigate prototype methods that can improve on using all the available training samples (i.e. $\text{DBR}_{100\%}$). Finally, considering FBRGE we report an AUROC performance of 77% that is at least 4% higher than any other representation on the specific subset of data-sets. Also, this representation allows the highest CR of 98.5%. Although this is a positive outcome further research is needed to make this approach completely unsupervised as at the present time the class label is required during the feature extraction phase.

Continuing our discussion about AUROC performance, in Figure 7.1, we show the critical difference diagram [50] commonly used in the time series classification literature to compare the performance of multiple classifiers over multiple data-sets [12]. In this case, we consider each representation paired with the 1NN classifier as a different classifier. For each representation the average rank is calculated with regard to the AUROC over all the data-sets, then results are arranged on a line where lowest rank is best. Finally, group of representations that are not significantly different in their ranking are connected by a horizontal bar. Significance is evaluated using the two tailed Wilcoxon signed rank test with Holm's correction and $\alpha = 0.05$ [18, 82]. Figure 7.1 (a) shows the performance of all the considered representations except FBRGE that is evaluated in Figure 7.1 (b) where only the relevant subset of data-sets is considered. In both parts of the figure ranks are in line with the average results shown at the bottom of Table 7.1 (c). For instance, Figure 7.1 (a) shows that RD performs better than AEBR, that in turn performs better than DBR. Also, the figure shows that RD paired with the MSM dissimilarity is significantly better than the other representations; there is not a significant difference between all the remaining variants of RD and the AEBR; DBR fall behind. Figure 7.1 (b) shows that there is a group of representations that not perform significantly worse than FBRGE that allow the best performance. Finally, all the representations except FBRGE have not a significantly different performance.

We continue our analysis considering the maximum AUROC score per data-set. As illustrated in Figures 7.2-7.3, we group AUROC scores in two ways: (1) by representation (RD, DBR, AEBR) regardless of the dissimilarity measure; (2) by dissimilarity measure (DTW, ED, MSM) regardless of the representation. In both cases, we count the number of times a given group achieves the maximum score, then we break results down by data-set characteristics i.e. application domain, time series length, and average number of training samples (Section 3.2.1). We do not include FBRGE in this

analysis as this approach is evaluated only on a subset of data-sets.

In Figure 7.2 we consider grouping by representation. RD allow the maximum score nearly as frequently as DBR and AEBR combined (41 times vs. 21 and 24 times). RD seem to work well with ECG, image outlines, and simulated data. AEBR seem to work well with spectrographs data. It is difficult to draw any conclusion about the impact of the time series length as there is not a clear pattern. Finally, in terms of number of training samples it seems that the ability of RD to allow the maximum score decreases as the number of training samples increases, while it is the opposite for DBR. Conversely, the ability of AEBR to allow the maximum score seems to be steady with respect to the number of training samples.

In Figure 7.3 we consider grouping by dissimilarity measure. MSM allows the maximum score nearly as frequently as the other two measures combined. In fact, MSM allows the maximum score 40 times, DTW 25 times, and the ED 21 times. MSM seems to work well with image outlines, and motion captures; the ED with ECG data. DTW seems to be the most consistent measure as it allows the maximum score at least a few times for almost every application domain. Finally, it is difficult to draw any conclusion about the impact of the time series length, and the average number of training samples, as there is not a clear pattern.

- Computational Complexity

For each of the considered approaches we show the CC to both represent and classify in Table 7.2. At prediction time when paired with a 1NN classifier all our representations allow linear CC in the new dimensionality $l \ll L$. In fact, all the representations we propose allow the 1NN classifier to work with the ED that has linear CC in the number of dimensions. In contrast, approaches based on raw data and elastic dissimilarity measures like DTW or MSM have quadratic CC in the time series length $L \gg l$.

The CC of representing with DBR depends on the dissimilarity measure in use, and the number of prototypes. Thus, for measures like DTW and MSM the CC is quadratic in the number of prototypes l , that is the number that determine the dimensionality of the representation. In contrast, the CC to represent with DBR is linear in l if we use the ED. To represent with AEBR we need to consider only the CC of the encoder as the decoder is not needed to this end. In this case, we consider the CC of the encoder to be equal to the CC of the first convolutional layer that is the layer with highest CC. The CC of a 1-dimensional convolutional layer with stride equal to 1, zero-padding, and f filters of size k is equal to $\mathcal{O}(f \times k \times L)$ [244] for a time series of length L . According to our implementation we have $f = 16$ and $k = 0.03 \times L$, thus substituting and removing constants this would yield a quadratic CC. However, this CC is not realistic as it is common practice to

have $k = \{3, 5, 8\}$ [70], thus we consider the CC of representing with AEBR linear. Finally, to represent with FBRGE the CC is equal to $\mathcal{O}(L \times \log(L))$ that corresponds to the highest CC in the set of primitives, namely the CC of a fast Fourier transform.

In Figure 7.4, we show the CC of each approach as a function of the time series length L . The CC shown in the figure corresponds to the sum of the CC needed to represent and to classify. The number of training samples N , and the dimensionality of the representation l are derived in accordance to the actual use cases presented in Table 7.1. Thus, we set $N = 100$ as it is roughly equal to average number of training samples across all the data-sets, continuing, for DBR we have $l = 0.2 \times N$, for AEBR we have $l = 2 \times \log_2(L)$, and for FBRGE we have $l = 3$. For the representations that need a dissimilarity measure we assume a quadratic CC in L as required by DTW and MSM. The figure shows that RD and DBR have highest CC although the latter is slightly less requiring. On the other hand, FBRGE and AEBR have a similar CC that is several orders of magnitude lower than RD.

- Compression Rate

RD achieve, although by a limited margin, the best AUROC performance. Classification with the 1NN classifier and RD requires the entire time series to work. Vice versa, all the representations we propose make it possible to work with a highly compressed version of RD. The DBR_{20%} allow a CR of 87.2%, furthermore this number rises to 92.7% if we exclude two cases where the DBR has actually increased the dimensionality with respect to RD. As mentioned before, the FBRGE allow the highest CR of 98.5%. Finally, the AEBR allow a CR of 96.4%. These CR are remarkably high especially if compared to the limited loss of AUROC performance, that is at most 3%.

For some applications, e.g. embedded systems [8, 23], storage and computational savings enabled by our representations may be more important than a slight loss in classification performance. Branco et al. [23] report that most embedded devices have flash memory and RAM in the order of kilobytes. In this regard, using AEBR-DTW as an example, we observe that the average data-set size drops from 1,657kb to 35kb, with the largest data-set reduced from 21,164kb to 94kb. This is considering the training data of all classes of a given data-set stored in NumPy format [100]. We do not consider the impact of time series length on storage requirements as these two quantities are not strongly correlated in the data-sets of the UCR/UEA archive. Finally, although the CR allowed by our representations are high, even higher CR may be required by extremely large data-sets.

- A Note on the “AccelerometerData” Data-Set

Concluding, we comment on the data-set “AccelerometerData”. This data-set, presented in Section 3.2.2, is related to a subject authentication

problem through accelerometer data collected using wrist-worn devices. Embedded devices have often quite limited memory and computational resources. Thus, it is a very good outcome that our representations not only allow the best classification performance on this problem (83% and 85% AUROC for DBR and FBRGE respectively) but also a high CR equal to 99.8%.

Classification performance allows us understanding how well recorded accelerometer data can be used to solve the subject authentication problem at hand. In addition to classification performance, interpretability of learned features is important in order to explain why a certain model has come out with a certain outcome. Individual features of DBR, or FBRGE give some insights. For instance, in Chapter 5, through the analysis of evolved feature-extractors, we have shown that each subject is characterised by her/his activity during a specific part of the day. Future research may leverage our representations to gain further insights through the analysis of the distribution of individual features. In this respect, it would be good to have more samples and meta-data like age, gender, occupation, commute type, etc. In that case even non-interpretable features like those provided by AEBR could give useful insights.

Representation		RD				DBR				FBRGE		AEBR				
CC	$\mathcal{O}(L^2N)$	$\mathcal{O}(LN)$		$\mathcal{O}(L^2N)$		$\mathcal{O}(L^2I) + \mathcal{O}(lN)$				$\mathcal{O}(L\log(L)) + \mathcal{O}(lN)$		$\mathcal{O}(L) + \mathcal{O}(lN)$				
		DTW	ED	MSM	CR	DTW	ED	MSM	CR	CR	DTW	ED	MSM	CR	Min	Max
AccelerometerData	77	82	77	99.8	62	83	71	99.8	85	99.8	57	63	65	99.3	57	85
	91	90	91	98.8	85	87	90	98.8			90	88	90	95.8	85	91
	74	77	79	99.0	68	71	72	99.0			78	77	78	96.8	68	79
ArrowHead	78	82	79	99.7	80	78	78	99.7			82	81	82	98.1	78	82
	70	70	80	99.6	53	72	71	99.6			80	68	80	98.2	53	80
BeetleFly	72	58	77	99.6	60	56	69	99.6			71	65	64	98.2	56	77
	72	77	85	99.5	63	67	73	99.5			80	86	77	98.4	63	86
BirdChicken	100	94	100	98.4	91	88	94	98.4			73	76	73	94.5	73	100
	65	67	65	81.2	54	52	54	81.2			68	68	69	95.6	52	69
ChlorineConcentration	81	98	84	99.9	60	74	60	99.9			91	95	91	99.3	60	98
	96	96	95	99.0	92	92	91	99.0	99	99	99	96	95	97.1	91	99
CinCECGTorso	55	52	52	96.5	56	51	55	96.5	66	99.6	55	53	55	98.7	51	66
	90	83	89	97.8	80	77	81	97.8			80	80	79	97.3	77	90
CricketX	91	87	90	97.8	82	83	83	97.8			82	83	78	97.3	78	91
	90	82	89	97.8	80	77	82	97.8			79	80	77	97.3	77	90
CricketY	95	95	96	99.8	93	95	91	99.8			100	100	100	97.6	91	100
	66	62	64	88.5	67	68	66	88.5	80	96.2	65	64	66	92.1	62	80
DistalPhalanxOAG	58	58	58	65.5	60	63	57	65.5			60	59	61	92.1	57	63
	82	79	81	94.2	82	79	82	94.2	81	96.2	80	80	80	92.1	79	82
DistalPhalanxOC	57	50	56	97.3	59	54	54	97.3	68	99.4	80	82	82	98.2	50	82
	70	82	75	89.6	63	79	70	89.6	83	96.9	77	80	77	93.1	63	83
Earthquakes	80	78	80	85.7	78	78	81	85.7			83	85	84	94.9	78	85
	78	83	79	98.2	58	62	53	98.2			60	56	59	94.8	53	83
ECG5000	74	75	74	-165.6	78	70	79	-165.6			74	71	75	93.1	70	79
	94	92	95	93.9	89	85	90	93.9			85	87	84	94.6	84	95
FaceAll	91	89	96	99.7	83	88	82	99.7			84	90	93	97.6	82	96
	96	89	96	97.9	84	80	80	97.9			81	81	83	94.6	80	96
FacesUCR	95	89	95	99.3	88	86	85	99.3			87	85	83	97.0	83	95
	87	86	92	98.9	78	78	84	98.9	87	99.4	86	85	87	98.1	78	92
FiftyWords																
Fish																
FordA	52	57	52	73.6	61	57	66	73.6			58	56	59	98.2	52	66

Table 7.1 (a): The table shows the AUROC, rounded to the nearest integer, for all the 86 data-sets introduced in Section 3.2. Table cells get darker as the AUROC increases. Also, the table shows the CC, and the CR allowed by each representation. L is the time series length and l is the dimensionality of the representation.

Representation		RD			DBR				FBRGE		AEBR				
		$\mathcal{O}(L^2N)$		$\mathcal{O}(LN)$	$\mathcal{O}(L^2I) + \mathcal{O}(LN)$		CR	$\mathcal{O}(L\log(L)) + \mathcal{O}(LN)$		DTW	ED	MSM	CR		
CC		DTW	ED	MSM	DTW	ED	MSM	CR							
Data-set														Min	Max
FordB		57	55	55	55	48	58	83.8			53	55	56	48	58
GunPoint		88	85	97	75	80	86	96.7	86	98	86	84	88	75	97
Ham		52	52	54	56	51	52	97.4	67	99.3	54	53	55	51	67
HandOutlines		70	74	71	68	70	67	98.6			69	76	75	67	76
Haptics		60	61	62	60	59	62	99.4	63	99.7	59	62	61	59	63
Herring		53	52	55	60	51	53	98.8	53	99.4	62	56	59	51	62
Inlineskate		67	62	70	61	57	63	99.9			65	62	64	57	70
InsectWingbeatSound		78	91	85	75	89	77	98.4			89	90	90	75	91
ItalyPowerDemand		82	89	86	79	88	86	71.7			92	89	86	79	92
LargeKitchenAppliances		74	58	65	72	56	59	96.5	79	99.6	62	59	60	56	79
Lightning2		74	63	71	67	58	68	99.1	72	99.5	62	68	66	58	74
Lightning7		87	74	88	79	75	84	99.4			72	76	78	72	88
Mallat		99	99	98	99	99	96	99.9			97	98	97	96	99
Meat		97	98	98	94	94	93	99.1	98	99.3	100	98	99	93	100
MedicalImages		93	90	91	88	90	88	92.3			84	86	86	84	93
MiddlePhalanxOAG		52	55	55	54	58	54	87.2	58	96.2	59	59	61	52	61
MiddlePhalanxOC		61	64	64	61	65	63	63.5			65	66	67	61	67
MiddlePhalanxTW		71	73	73	72	73	71	93.5	72	96.2	73	73	73	71	73
MoteStrain		78	75	85	81	73	73	97.6			71	87	82	71	87
NonInvasiveFatalECGT1		97	98	97	96	98	96	98.9			97	98	97	96	98
NonInvasiveFatalECGT2		98	99	98	97	98	97	98.9			98	98	98	97	99
OliveOil		90	90	89	86	87	85	99.7			78	79	75	75	90
OSULeaf		86	80	93	73	76	78	98.5	81	99.3	90	96	94	73	96
PhalangesOC		56	58	57	57	60	58	-125.0			57	56	57	56	60
Phoneme		72	55	70	65	50	65	99.9			54	53	54	50	72
Plane		100	100	100	99	99	100	97.9			99	100	99	99	100
ProximalPhalanxOAG		67	72	67	70	75	69	66.8	87	96.2	75	75	72	67	87
ProximalPhalanxOC		58	62	59	59	59	58	25.0			62	62	64	58	64
ProximalPhalanxTW		88	86	88	88	87	88	91.5	92	96.2	90	85	89	85	92
RefrigerationDevices		58	53	58	60	49	58	96.5	64	99.6	55	53	55	49	64

Table 7.1 (b)

Representation		RD				DBR				FBRGE		AEBR			
CC	Data-set	$\mathcal{O}(L^2N)$		$\mathcal{O}(LN)$		$\mathcal{O}(L^2D) + \mathcal{O}(LN)$				$\mathcal{O}(L\log(L)) + \mathcal{O}(LN)$					
		DTW	ED	MSM	$\mathcal{O}(L^2N)$	DTW	ED	MSM	CR	DTW	ED	MSM	CR	Min	Max
ScreenType ShapeletSim ShapesAll SmallKitchenAppliances SonyAIBORobotSurface1 SonyAIBORobotSurface2 StarLightCurves Strawberry SwedishLeaf Symbols SyntheticControl ToeSegmentation1 ToeSegmentation2 Trace TwoLeadECG TwoPatterns UWaveGestureLibraryAll UWaveGestureLibraryX UWaveGestureLibraryY UWaveGestureLibraryZ Wafer Wine WordSynonyms Worms WormsTwoClass Yoga	55	51	55	53	53	56	96.5	61	99.6	52	53	52	98.7	51	61
	71	51	74	59	52	71	99.6			55	58	55	98.2	51	74
	95	92	97	91	90	91	99.6			92	92	89	98.2	89	97
	57	50	54	70	51	57	96.5	81	99.6	62	53	59	98.7	50	81
	78	75	78	71	63	80	97.1			76	76	74	91.2	63	80
	78	79	81	66	68	63	95.7			80	79	82	90.7	63	82
	91	92	92	93	93	95	93.5			92	92	92	99.0	91	95
	86	86	87	83	83	83	84.3			83	85	87	96.6	83	87
	85	86	92	87	86	91	94.8			86	86	86	94.5	85	92
	99	97	99	96	94	97	99.8			96	95	96	97.8	94	99
	99	89	98	96	86	97	83.3	99	95	94	91	93	90.2	86	99
	65	62	69	57	61	63	98.6	68	98.9	68	59	63	97.1	57	69
	67	75	75	56	66	54	99.0	78	99.1	69	71	67	97.5	54	78
	99	94	98	96	92	94	98.2	100	98.9	93	93	93	97.1	92	100
	87	78	90	68	62	71	97.1			79	80	76	92.2	62	90
	100	94	97	100	94	89	60.9			71	91	77	94.5	71	100
	96	98	97	93	98	89	97.6			96	98	95	99.0	89	98
	85	86	86	89	88	87	92.9			85	87	81	97.4	81	89
	87	87	87	88	88	87	92.9			87	88	82	97.4	82	88
	84	84	84	86	85	86	92.9			83	84	80	97.4	80	86
	93	98	95	95	98	94	34.2			94	97	93	95.2	93	98
	59	57	60	59	57	60	97.5	58	98.7	60	59	61	96.6	57	61
	90	80	91	73	74	72	99.2			75	75	72	97.0	72	91
	72	64	78	69	60	79	99.7	73	99.7	72	64	74	98.9	60	79
61	54	67	60	59	70	99.1	64	99.7	60	59	59	98.9	54	70	
69	69	73	70	68	74	93.0			67	68	65	97.9	65	74	
Avg.	78	77	80	75	74	76				77	77	77	96.2	70	84
St. Dev.	14.6	15.3	14.7	14.2	15.1	13.8				13.7	14.5	13.5	3.1	14.7	12.6
Skew.	-0.23	-0.3	-0.42	0.15	-0.06	-0.09				-0.07	-0.19	-0.09	-1.84	0.2	-0.45
Avg. FBRGE	71	69	73	69	68	70	95.1	77	98.5	72	71	72	96.4	65	79
St. Dev. FBRGE	15.1	15.6	15.7	13.2	14.4	14.2	6.8	13.3	1.5	14.6	15	14.5	2.9	14.3	13.2
Skew. FBRGE	0.5	0.37	0.39	0.84	0.38	0.51	-2.95	0.16	-1.06	0.52	0.44	0.46	-0.94	0.82	0.13

Table 7.1 (c)

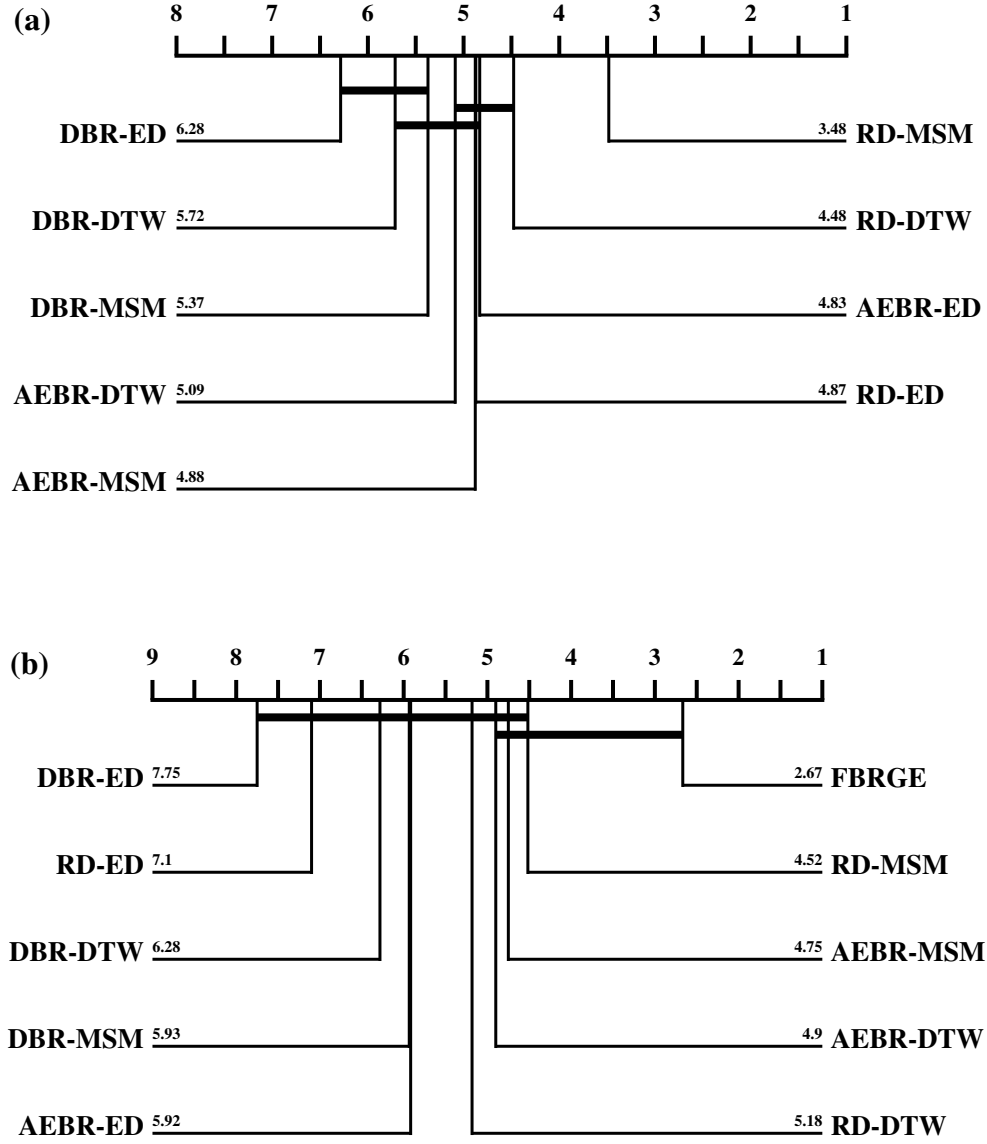


Figure 7.1: Critical difference diagram for the AUROC of all representations. Groups of representations that are not significantly different are connected by a horizontal bar. (a) Shows the performance of all the considered representations except FBRGE that is considered in (b).

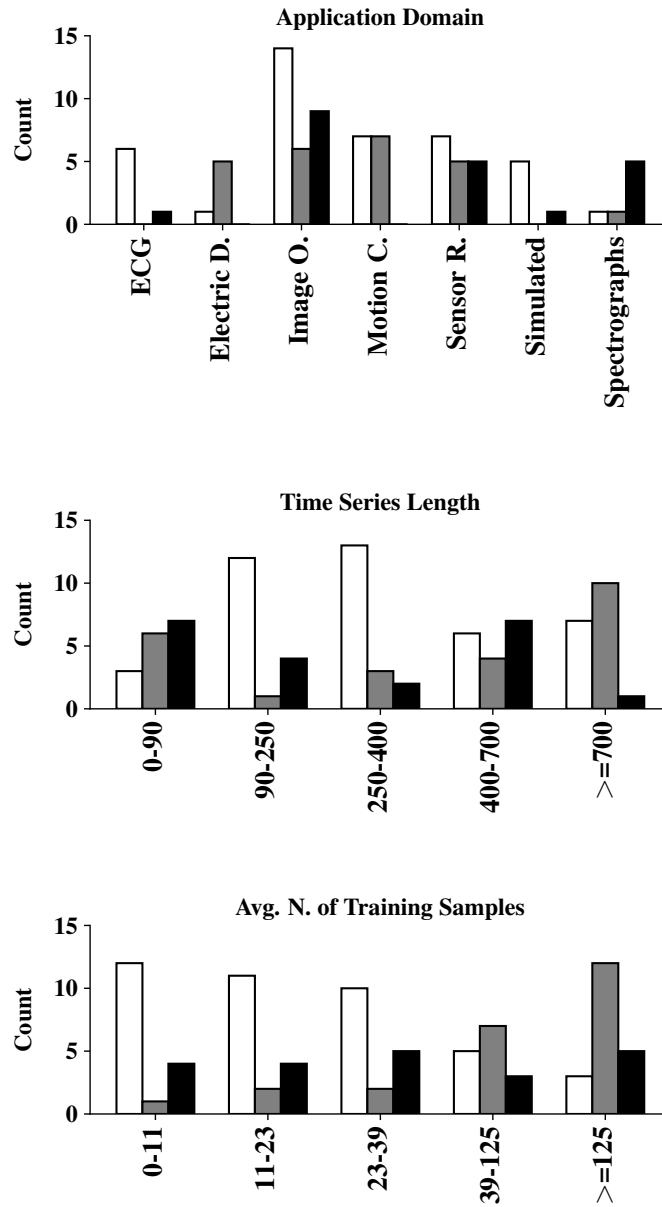


Figure 7.2: Count of max AUROC performance by representation according to Table 7.1. RD (□), DBR (■), AEBR (■).

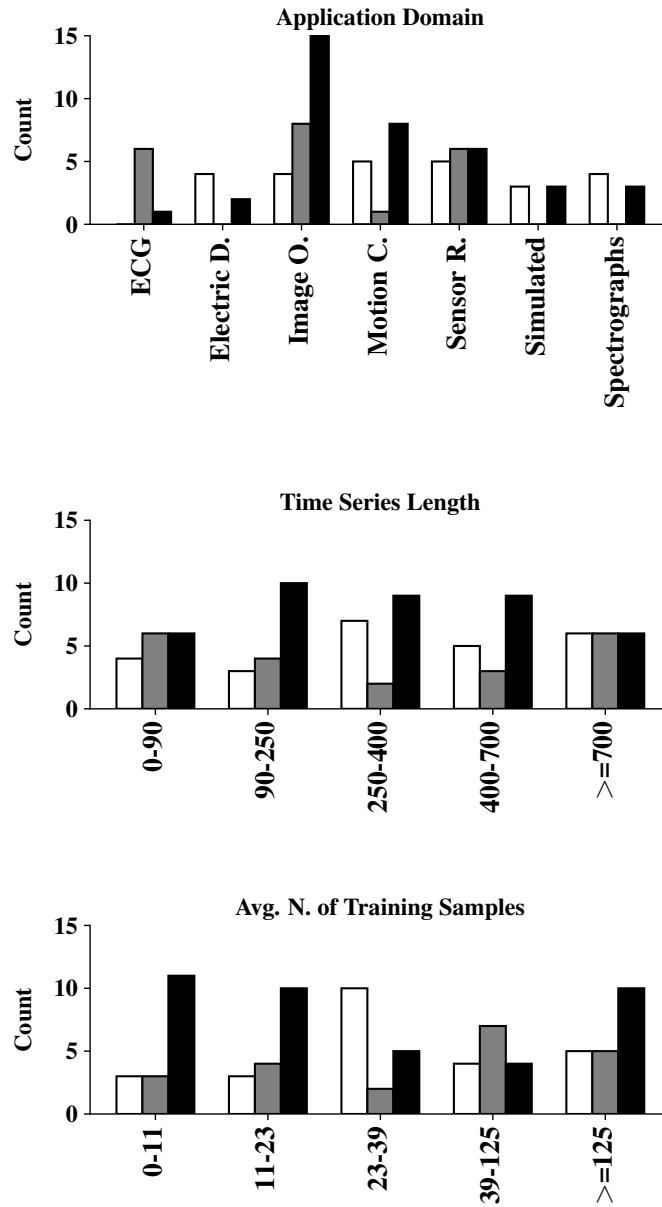


Figure 7.3: Count of max AUROC performance by dissimilarity measure according to Table 7.1. DTW (\square), ED (\blacksquare), MSM (\blacksquare).

	To Represent			To Classify		
	Dissimilarity Measure					
Repr.	DTW	ED	MSM	DTW	ED	MSM
RD	-	-	-	$\mathcal{O}(L^2 \times N)$	$\mathcal{O}(L \times N)$	$\mathcal{O}(L^2 \times N)$
DBR	$\mathcal{O}(L^2 \times l)$	$\mathcal{O}(L \times l)$	$\mathcal{O}(L^2 \times l)$	$\mathcal{O}(l \times N)$	$\mathcal{O}(l \times N)$	$\mathcal{O}(l \times N)$
AEBR	$\mathcal{O}(L)$	$\mathcal{O}(L)$	$\mathcal{O}(L)$	$\mathcal{O}(l \times N)$	$\mathcal{O}(l \times N)$	$\mathcal{O}(l \times N)$
FBRGE	$\mathcal{O}(L \times \log(L))$			$\mathcal{O}(l \times N)$		

Table 7.2: The table shows the CC to represent, and to classify with the considered approaches and a 1NN classifier equipped with the ED. L is the time series length, l is the dimensionality of the representation, and N is the number of training samples.

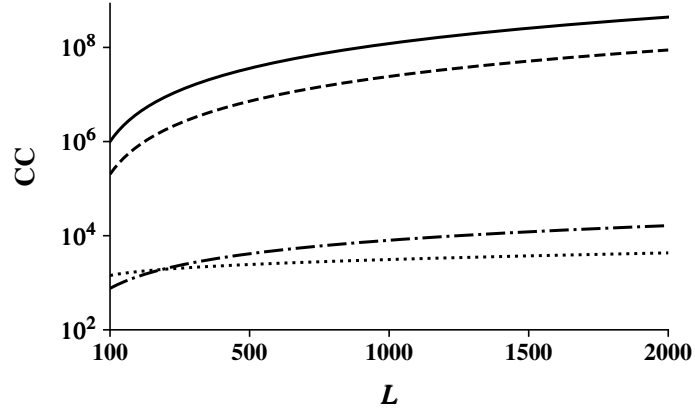


Figure 7.4: CC as a function of the time series length L . This is the CC to represent, and to classify with the considered approaches and a 1NN classifier equipped with the ED. — RD. --- DBR. -.- FBRGE. AEBR.

7.2 Combining Representations

In this thesis we have presented three novel algorithms for time series representation (Chapters 4-5-6). For each approach we have evaluated several variants gathering a remarkable number of representations. In the field of time series classification ensemble methods have often led to better performance than the single algorithms combined [155, 161, 235]. In this section, we propose a small experiment to show that ensembling different representations, in the context of one-class time series classification, can lead to better performance than any representation alone, and it is worth further research.

Diversity is a key aspect of ensemble schemes as samples that are incorrectly classified by some classifiers can be correctly classified by others in such a way that the voting system allows a performance that is greater than that of any single classifier [243]. Some ensembles allow diversity using different classifiers, some train the same classifier on different subsets of data, others, as in our approach, train the same classifier on different subsets of features [166]. In this experiment we select two representations and we compare their individual and combined performance. The ensemble schemes we implement are arguably the simplest possible. (1) First, we concatenate the two representations into a single vector that then is used for classification. We refer to this approach as “Concatenation”. (2) Then, we average the classification scores achieved by the two representations independently and we calculate the AUROC using the resulting average score. We refer to this approach as “Averaging”. Considering the “Concatenation” approach we notice a major limitation, by concatenating an increasing number of representations, we increase data dimensionality, and so we may incur in the curse of dimensionality [17].

We select two representations that are conceptually distinct. We select the DBR “Centers-k-means-DTW” from Chapter 4. This representation achieves 72% AUROC on its own. We select the prototype method “Centers-k-means” as it is the one that achieves the highest average AUROC across different dissimilarity measures (Table 4.1, DBR₂). Then, we select the AE-based representation “DissPCE-DTW” from Chapter 6. This representation achieves 68% AUROC on its own. We select the dissimilarity-preserving architecture that has best preserved pairwise dissimilarities on training data. The representations we select are both 2-dimensional as we want them to have equal contribution towards the dissimilarity estimation carried out by the 1NN classifier. We use two representations derived through the DTW dissimilarity because this is a measure of central importance in the time series classification domain [12]. We do not include our evolutionary algorithm in this experiment as that algorithm exploits the class label to derive the

representation, and thus it is different from the other methods which are completely unsupervised. Furthermore, this algorithm is evaluated only on a subset of data-sets.

7.2.1 Results

The two ensembling schemes described before are evaluated on the 86 data-sets introduced in Section 3.2 using a 1NN classifier. All the features are standardised to have zero mean and unit variance (with reference to the training set). Results, in terms of AUROC averaged across all data-sets, are shown in Table 7.3. Both the ensemble schemes we evaluate are able to improve on the performance of a single representation on its own. Specifically, Concatenation enables +3% AUROC, and Averaging enables +2% AUROC with respect to the best representation (Centers-k-means-DTW). Although there is a large difference in the “quality” of the representations we ensemble (72% vs. 68% AUROC), together they improve on their individual performance proving that they have learned different characteristics of data.

Representation		Ensemble Scheme	
Centers-k-means-DTW	DissPCE-DTW	Concatenation	Averaging
72	68	75	74

Table 7.3: The table shows AUROC averaged across all the data-sets and rounded to the nearest integer.

7.3 More Benchmark Methods

In this section we evaluate the classification performance of four common dimensionality reduction techniques. All the selected techniques are unsupervised thus they can be used in the one-class scenario without requiring any adjustment. Although we have already evaluated all our experiments against several benchmark methods, now our goal is to understand how off-the-shelf and general purpose techniques compare against our algorithms that are specifically defined to enable good classification performance.

The techniques we consider are: discrete Fourier transform (DFT) [39], piecewise aggregate approximation (PAA) [122], principal component analysis (PCA) [108], and kernel principal component analysis (KPCA) [216].

PCA is a widely used technique which aims at reducing the dimensionality of data with minimal loss of information. PCA uses an orthogonal

transformation to convert possibly correlated variables into a set of linearly uncorrelated variables called principal components. This transformation is defined in such a way that the first principal component accounts for as much as possible of the variability in the data. Each subsequent component in turn accounts for as much as possible of the variability in the data under the constraint that it is orthogonal to the preceding components. Standard PCA is not effective if data has a structure which cannot be well represented in a linear subspace. Thus, KPCA allows us to generalize PCA to nonlinear dimensionality reduction using kernel methods. In our implementation we use KPCA with radial basis function kernel.

DFT decomposes a time series into a finite sum of sinusoidal components. The coefficients of these components are the DFT of the time series. The DFT has the same dimensionality of the original time series, however, in most applications is selected only the subset of coefficients with largest magnitude. The coefficients with largest magnitude allow the reconstruction of the original time series with *little* loss of information through an operation called inverse Fourier transform. The inverse transform expresses a time series as a sum of sines and cosines waves sampled at frequencies that are multiples of $2 \times \frac{\pi}{L}$ (where L is the time series length). Thus, we think of the DFT as a change of basis from the time domain to the frequency domain.

PAA breaks down a time series in adjacent, and equally sized sub-sequences and then takes their mean value. When the time series cannot be exactly divided in the desired number of sub-sequences adjacent sub-sequences share some points proportionally to their length.

7.3.1 Results

We evaluate the algorithms discussed above at different levels of compression on all the 86 data-sets introduced in Section 3.2 using a 1NN classifier. Specifically, we consider all values $\{2, 0.1 \times L, 0.2 \times L\}$ (where L is the time series length). All the features are standardised to have zero mean and unit variance (with reference to the training set). Results, in terms of AUROC averaged across all data-sets, are shown in Table 7.4.

Results appear to be weak when compared with those achieved by our algorithms. While the best performance in Table 7.4 is equal to 68% AUROC, our algorithms achieve a performance well above 70% AUROC in the majority of cases. However, this is not surprising. In fact, algorithms like PCA and KPCA are not defined to work with time series, thus they are not able to account of the temporal ordering of this particular data type. Conversely, algorithms like DFT and PAA are more suitable for time series, but as they work on one time series at a time they are not able to learn any characteristic

about a data-set as a whole.

	Dimensionality		
Algorithm	2	$0.1 \times L$	$0.2 \times L$
DFT	66	66	66
KPCA	65	66	66
PAA	61	64	64
PCA	66	68	68

Table 7.4: The table shows AUROC averaged across all the data-sets and rounded to the nearest integer. L is the time series length.

7.4 Conclusions

In this chapter we compare the different representations discussed throughout the thesis considering three main aspects: the AUROC, the CC, and the CR.

Our best performance in terms of AUROC is 3% lower than the best performance allowed by RD. Nevertheless, we show that on a case by case basis our representations can do better than RD providing some insights into the underlying reasons [14]. In terms of CC, we show that all our representations have lower CC than RD at prediction time. While the CC required to represent and classify a sample using DBR is close to that of RD, FBRGE and AEBr have a CC that is several orders of magnitude lower. Also, our representations allow a CR that spans from 87.2% to 98.5%. These results are practically relevant. In fact, while it is extremely easy to acquire new time series e.g. through sensors, it is less straightforward to store and mine this data type in its raw form.

Secondly, we show that it is possible even with simple ensemble schemes to combine multiple representations and improve their individual classification performance. This provides a good starting point for further research. Finally, we show that some common dimensionality reduction techniques allow a much lower classification performance than our representations.

Part III

Conclusions

Chapter 8

Conclusions

In this chapter we summarise the research carried out in this thesis. In Section 8.1, we discuss our scientific contributions. We conclude by discussing the main limitations of our work in Section 8.2, and a number of avenues for future research in Section 8.3.

8.1 Scientific Contributions

The primary objective of this thesis is to investigate the one-class time series classification problem. To the best of our knowledge we are the first to thoroughly explore this topic. Thus, one of the major contributions of this work is to investigate this useful variant of the classification problem. The common thread of our research is to represent time series as feature-vectors then used for classification. We demonstrate that, with respect to the use of raw time series, feature-based representations can reduce storage requirements, ease computational complexity, facilitate the interpretability of the solutions found, and enable visualisation of time series data-sets.

We benchmark our work against a 1NN classifier, paired with several dissimilarity measures, on raw time series. In particular, the 1NN classifier paired with DTW is a fundamental approach to time series classification [2, 12]. We find that the advantages of feature-based representations come at the cost of a slight loss in terms of average classification performance across all the considered data-sets. However, the difference in terms of classification performance between our representations and raw data is found to be statistically significant only for the MSM measure, as shown in Figure 7.1. Furthermore, by examining data-sets one by one it can be shown that our representations can outperform raw time series. For some applications, e.g. embedded systems [8, 23], storage and computational sav-

ings may be more important than a slight loss in classification performance (Section 7.1.1). Not to mention that the approach we propose in Chapter 5, feature-based representations via grammatical evolution, outperforms the 1NN-DTW on raw data by a 4% margin in terms of AUROC.

The representations we propose are: (1) dissimilarity-based representations (DBR) (Chapter 4), (2) feature-based representations via grammatical evolution (Chapter 5), and (3) auto-encoder-based representation (Chapter 6). We now highlight the main contributions of each chapter.

In **Chapter 2**, we have provided a broad review of the state of the art in time series classification. Furthermore, we have expanded the discussion on time series data mining with a specific focus on anomaly detection showing how this problem relates to one-class time series classification. This chapter can be considered as a knowledge repository on time series, and one of the most comprehensive discussions available in the field.

In **Chapter 3**, we have detailed the experimental design underlying our one-class time series classification experiments. We have discussed how to adapt existing resources (benchmark data-sets and classifiers) to the one-class scenario. Along with the GitHub repository where our code is publicly available, this can be considered as a great resource for researchers who want to investigate one-class time series classification.

In **Chapter 4**, we have evaluated the performance of a 1NN classifier paired with an extensive set of dissimilarity measures on raw time series. This is a fundamental baseline for research on one-class time series classification that is here introduced for the first time. However, the more important contribution of this chapter is that for the first time in the one-class scenario, we have evaluated and discussed a thorough set of methods to derive DBR. Our results have shown that DBR are particularly suited for problems where class membership depends on the global time series shape. Also, we have found that DBR perform better than raw data as the number of training samples increases (Figure 4.4 and Figure 7.3). We have introduced a prototype method named “percentiles” that allows good classification performance with just two features. Finally, we have demonstrated that DBR can allow the 2D visualisation of time series data-sets, and in doing so we have shown that the right combination of dissimilarity measure and prototype method can have a substantial impact on classification performance.

In **Chapter 5**, for the first time in the one-class scenario, we have proposed a feature extraction framework based on grammatical evolution. In the field of evolutionary computation there is a lack of consistent comparison of feature extraction methods for time series with relevant benchmarks. We have addressed this gap, not only by testing our approach on a large set of problems, but also comparing our performance with that of several bench-

mark methods. Also, as the analysis of the literature reveals, it is not clear how to evolve multiple features that are not redundant, a problem we have tackled through our fitness function. We have shown that the classification performance allowed by the features extracted by our algorithm is superior to that of all the benchmark methods. We have shown how evolved features offer insights on the sub-sequences and functions that are relevant to the classification problem at hand, and allow the 2D visualisation of time series data-sets. Finally, using our subject authentication case study, we have shown that features extracted in a one-class classification scenario are able to generalise to classes unseen during the feature extraction phase.

In **Chapter 6**, we have combined the non-linear dimensionality reduction power of auto-encoders with elastic dissimilarity measures which are of central importance to the time series classification problem. We have demonstrated how our approach can be used to reduce the dimensionality of raw time series while preserving pairwise dissimilarities between samples with respect to a measure of choice. We have demonstrated that this approach does not only allow a low-dimensional embedding useful for classification, but also allows us to approximate the dissimilarity measure used during the training phase. Once again we have carried extensive analysis beyond raw performance, including in particular 2D visualisation of time series data-sets.

In **Chapter 7**, we have compared the different representations introduced throughout the thesis. We have shown that feature-based representations can reduce, by several orders of magnitude, the computational complexity required by a 1NN classifier on raw time series paired with elastic dissimilarity measures, a strong baseline [12]. In addition, we have shown that our representations can reduce the dimensionality of raw time series by 87.2%-98.5%, with a limited loss in terms of classification performance. These are remarkable results that can allow the deployment of machine learning approaches to time series classification at scale. In fact, while the time series classification literature is richly populated by algorithms that have quadratic computational complexity or worse [12], we have shown that our feature-based representation can be used with a 1NN classifier paired with the Euclidean distance that has linear complexity in the representation dimensionality. Finally, we have shown that it is possible to improve the classification performance of single representations even through simple ensemble schemes. Also, we have shown that dimensionality reduction techniques that do not account for the temporal ordering of time series (e.g. PCA), or are not data-driven (e.g. PAC), give substantially lower classification performance than our representations.

8.2 Limitations

Our research has several limitations that we discuss in this section. All the data-sets of the UCR/UEA archive we use are z-normalised, and their un-normalised version is not available (Section 3.2.1). The rationale is that this transformation enables effective shape comparison of two time series by removing shifts in the mean value, and noise. However, it can be shown that other pre-processing techniques may lead to significant improvements of classification performance [187]. We have not been able to test this hypothesis, therefore we cannot draw any conclusion on how our algorithms would be affected by different pre-processing techniques.

In almost every experiment we have evaluated our algorithms over 86 data-sets. Performance evaluation over multiple data-sets is fundamental for machine learning research. In our work, this is important to identify if a given representation is more suited for general learning than another. However, while designing experiments that can be deployed over all the considered data-sets we may have disregarded some data-set specific peculiarities that, if took into account, may have led to better performance.

The one-class assumption requires that only the data related to a single class is available at training time. Thus, due to the lack of validation data, we have not been able to tune the hyper-parameters of some of the dissimilarity measures used in some of our experiments (Chapters 4-6). This may be regarded as a limitation of our work as hyper-parameter tuning is known to significantly improve performance in the supervised classification domain [12]. We have not been able to overcome this issue as there is a lack of methods for unsupervised hyper-parameter tuning of dissimilarity measures [187]. However, the impact on our findings is limited. This is because results related to classification with raw data described in Chapter 4 agree with related results described in the supervised classification domain e.g. the MSM dissimilarity achieves best performance [230]. Similarly, in Chapter 6 the objective is to evaluate whether auto-encoders can approximate elastic dissimilarity measures or not. The fact that the hyper-parameters of these measures are not optimised is marginal to this question.

Again, due to the one-class assumption we have not been able to tune the hyper-parameters related to some classifiers considered in some of our experiments e.g. the OC-SVM classifier. This may have led to sub-optimal performance for these classifiers. Hyper-parameter tuning in one-class classification is a known challenge [249], that however has only marginally affected our research. This is because most of our research relies on the evaluation of the dissimilarity between time series that is best assessed through the 1NN classifier. Although we have evaluated the performance of other classifiers

to make our research more comprehensive, the non-parametric 1NN classifier is the algorithm that is most suited for our experiments and it has no hyper-parameters (Section 3.3).

Concluding, despite a large investigation in this area, we have been able to provide only weak guidelines to decide when and why a certain representation is expected to enable better classification performance than another given some properties like: application domain, length of the time series, or number of training samples (Figure 3.1). As discussed in Chapter 2, time series class membership may depend on a variety of reasons that are disentangled from the elements of differentiation at our disposal (e.g. time series length).

8.3 Future Work

In this section we gather all the considerations related to future research as they have arisen from each experimental chapter. In addition, we add some final thoughts to account for the present body of work as a whole. We believe that each limitation discussed in the previous section (Section 8.2) is not just an *absolute limit* but rather a good opportunity for improvement.

In **Chapter 4** we have investigated dissimilarity-based representations for one-class time series classification, evaluating a large number of dissimilarity measures and prototype methods. Future research may consider novel dissimilarity measures and prototype methods that frequently appear in the literature [136, 187]. Also, our study has pointed out the lack of methods for unsupervised tuning of dissimilarity measures hyper-parameters. The same finding is confirmed by Paparrizos et al. [187] in a comprehensive study on dissimilarity measures for time series. We believe this will be an important area for future research. Finally, the possibility of ensembling multiple dissimilarity-based representations warrants further investigation. This idea is supported by other studies that have combined multiple dissimilarity measures, demonstrating a significant improvement of performance [154, 235].

In **Chapter 5** we have developed an evolutionary algorithm for automated feature extraction. The algorithm combines a number of primitive functions to create new functions which are used to extract features from time series. The algorithm evaluates the quality of a candidate solution according to its classification performance on validation data through a 1NN classifier equipped with the Euclidean distance. In this regard, it would be interesting to investigate the effect of different classifiers on extracted features. A remaining issue of our algorithm is the tendency to overfit. This suggests that our results could be improved through regularisation strategies, thus this is an interesting topic for future work.

In **Chapter 6** we have investigated auto-encoders as tools to reduce the dimensionality of time series while preserving pairwise dissimilarities between samples. We have evaluated a variety of architectures, however there are several alternatives that are unexplored [70]. Again, Rakhshani et al. [204] show that neural architecture search [196] can have a significant impact on the performance of supervised neural network classifiers for time series. Future research may investigate neural architecture search in the context of semi-supervised and unsupervised time series representation learning.

In **Chapter 7** we have compared the time series representations proposed throughout the thesis. Secondly, we have shown that combining multiple representations through simple ensemble schemes can have a positive impact on classification performance. This result, extensively discussed in the supervised time series classification domain [12], warrants further research. By leveraging our time series representations, future research could further develop feature selection or construction methods, or ensemble learning techniques for one-class time series classification.

Finally, we point out six major topics that could fuel the research in the field of one-class time series classification, and as far as we know, no previous research has investigated. (1) It would be of interest to extend our methods and research on one-class classification to multivariate time series. (2) In line with AutoML research [75], we believe that any effort made to automate the one-class time series classification pipeline (e.g. pre-processing, representation, model, and hyper-parameter selection) would be valuable. (3) One-class classifiers can be used to solve binary or multi-class problems. For instance, Krawczyk et al. [134] show that the decomposition of multi-class classification problems into a number of one-class problems has some advantages with respect to other related strategies. Thus, we suggest this is a promising direction for future research. (4) One-class classification may be useful to detect anomalous time series within a given data-set. However, for matters of time we have not presented any study specifically focused on time series anomaly detection. This is an interesting topic for future work. (5) The literature related to supervised time series classification is rich in classifiers specifically designed for time series [12]. Future research should certainly try to adapt the most successful classifiers from the supervised to the one-class domain. In particular, online learners [106] may make a difference in terms of scalability of time series classification solutions. (6) Lastly, considering how easily sensors can generate time series we believe that learning strategies able to leverage the abundance of data, whether labelled or not, like active learning, or transfer learning could prove beneficial to several practical applications e.g. active anomaly detection [45]. In the context of transfer learning for supervised time series classification, and using the

data-sets of the UCR/UEA archive, it has been shown that the choice of the source and target data-set can have a significant impact on classification performance [68]. Further research is needed to develop transfer learning approaches in the context of one-class time series classification. Also, it would be of interest to investigate which application domain (Section 3.2.1) could work best as source domain for classification tasks related to wearable sensor data as per our subject identification problem (Section 3.2.2). In this regard, some data-sets related to human movement can be found in the version of the UCR/UEA archive considered in this thesis (“GunPoint”, “InlineSkate”, “ToeSegmentation”, “UWaveGestureLibrary”, etc.), and some more have been recently added.

Bibliography

- [1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.
- [2] Amaia Abanda, Usue Mori, and Jose A Lozano. A review on distance based time series classification. *Data Mining and Knowledge Discovery*, pages 1–35, 2018.
- [3] Hervé Abdi. Holm’s sequential bonferroni procedure. *Encyclopedia of research design*, 1(8):1–8, 2010.
- [4] Abubakar Abid and James Y Zou. Learning a warping distance from unlabeled time series using sequence autoencoders. In *Advances in Neural Information Processing Systems*, pages 10547–10555, 2018.
- [5] Saeed Aghabozorgi, Ali Seyed Shirkhorshidi, and Teh Ying Wah. Time-series clustering—a decade review. *Information Systems*, 53:16–38, 2015.
- [6] Subutai Ahmad, Alexander Lavin, Scott Purdy, and Zuha Agha. Un-supervised real-time anomaly detection for streaming data. *Neurocomputing*, 262:134–147, 2017.
- [7] Robert J Alcock, Yannis Manolopoulos, et al. Time-series similarity queries employing a feature-based approach. In *7th Hellenic conference on informatics*, pages 27–29, 1999.

- [8] Cesare Alippi. *Intelligence for embedded systems*. Springer, 2014.
- [9] Samaneh Aminikhanghahi and Diane J Cook. A survey of methods for time series change point detection. *Knowledge and information systems*, 51(2):339–367, 2017.
- [10] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: A survey. *Computer networks*, 54(15):2787–2805, 2010.
- [11] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495, 2017.
- [12] Anthony Bagnall, Jason Lines, Aaron Bostrom, James Large, and Eamonn 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.
- [13] Pierre Baldi. Autoencoders, unsupervised learning, and deep architectures. In *Proceedings of ICML workshop on unsupervised and transfer learning*, pages 37–49, 2012.
- [14] Gustavo EAPA Batista, Eamonn J Keogh, Oben Moses Tataw, and Vinicius MA De Souza. Cid: an efficient complexity-invariant distance for time series. *Data Mining and Knowledge Discovery*, 28(3):634–669, 2014.
- [15] Gustavo EAPA Batista, Xiaoyue Wang, and Eamonn J Keogh. A complexity-invariant distance measure for time series. In *Proceedings of the 2011 SIAM international conference on data mining*, pages 699–710. SIAM, 2011.
- [16] Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019.
- [17] Richard Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966.
- [18] Alessio Benavoli, Giorgio Corani, and Francesca Mangili. Should we really use post-hoc tests based on mean-ranks? *The Journal of Machine Learning Research*, 17(1):152–161, 2016.

- [19] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.
- [20] Vineetha Bettaiah and Heggere S Ranganath. An analysis of time series representation methods: data mining applications perspective. In *Proceedings of the 2014 ACM Southeast Regional Conference*, page 16. ACM, 2014.
- [21] Davis Blalock, Samuel Madden, and John Guttag. Sprintz: Time series compression for the internet of things. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2(3):1–23, 2018.
- [22] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.
- [23] Sérgio Branco, André G Ferreira, and Jorge Cabral. Machine learning in resource-scarce embedded systems, fpgas, and end-devices: A survey. *Electronics*, 8(11):1289, 2019.
- [24] Henry Brighton and Chris Mellish. Advances in instance selection for instance-based learning algorithms. *Data mining and knowledge discovery*, 6(2):153–172, 2002.
- [25] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a "siamese" time delay neural network. In *Advances in neural information processing systems*, pages 737–744, 1994.
- [26] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.
- [27] Krisztian Buza, Júlia Koller, and Kristóf Marussy. Process: projection-based classification of electroencephalograph signals. In *International Conference on Artificial Intelligence and Soft Computing*, pages 91–100. Springer, 2015.
- [28] Van Loi Cao, Miguel Nicolau, and James McDermott. One-class classification for anomaly detection with kernel density estimation and genetic programming. In *European Conference on Genetic Programming*, pages 3–18. Springer, 2016.

- [29] Van Loi Cao, Miguel Nicolau, and James McDermott. Learning neural representations for network anomaly detection. *IEEE Transactions on Cybernetics*, 49(8):3074–3087, 2018.
- [30] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):15, 2009.
- [31] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection for discrete sequences: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 24(5):823–839, 2012.
- [32] Zhengping Che, Xinran He, Ke Xu, and Yan Liu. Decade: a deep metric learning model for multivariate time series. In *KDD workshop on mining and learning from time series*. sn, 2017.
- [33] Lei Chen, M Tamer Özsu, and Vincent Oria. Robust and fast similarity search for moving object trajectories. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 491–502. ACM, 2005.
- [34] Weiwei Chen, Fangang Kong, Feng Mei, Guiqin Yuan, and Bo Li. A novel unsupervised anomaly detection approach for intrusion detection system. In *2017 IEEE 3rd international conference on big data security on cloud (bigdatasecurity), IEEE international conference on high performance and smart computing (hpsc), and IEEE international conference on intelligent data and security (ids)*, pages 69–73. IEEE, 2017.
- [35] Haibin Cheng, Pang-Ning Tan, Christopher Potter, and Steven Klooster. Detection and characterization of anomalies in multivariate time series. In *Proceedings of the 2009 SIAM international conference on data mining*, pages 413–424. SIAM, 2009.
- [36] Dhruv Choudhary, Arun Kejariwal, and Francois Orsini. On the runtime-efficacy trade-off of anomaly detection techniques for real-time streaming data. *arXiv preprint arXiv:1710.04735*, 2017.
- [37] Maximilian Christ, Nils Braun, Julius Neuffer, and Andreas W Kempa-Liehr. Time series feature extraction on basis of scalable hypothesis tests (tsfresh—a python package). *Neurocomputing*, 307:72–77, 2018.
- [38] M Bishop Christopher. *PATTERN RECOGNITION AND MACHINE LEARNING*. Springer-Verlag New York, 2016.

- [39] William T Cochran, James W Cooley, David L Favin, Howard D Helms, Reginald A Kaenel, William W Lang, George C Maling, David E Nelson, Charles M Rader, and Peter D Welch. What is the fast fourier transform? *Proceedings of the IEEE*, 55(10):1664–1674, 1967.
- [40] Taco S Cohen, Mario Geiger, Jonas Köhler, and Max Welling. Spherical cnns. *arXiv preprint arXiv:1801.10130*, 2018.
- [41] Andrew Cook, Göksel Mısırlı, and Zhong Fan. Anomaly detection for iot time-series data: A survey. *IEEE Internet of Things Journal*, 2019.
- [42] Michael AA Cox and Trevor F Cox. Multidimensional scaling. In *Handbook of data visualization*, pages 315–347. Springer, 2008.
- [43] Wei Cui, Anthony Brabazon, and Michael O’Neill. Evolving efficient limit order strategy using grammatical evolution. In *IEEE Congress on Evolutionary Computation*, pages 1–6. IEEE, 2010.
- [44] Gautam Das, King-Ip Lin, Heikki Mannila, Gopal Renganathan, and Padhraic Smyth. Rule discovery from time series. In *KDD*, volume 98, pages 16–22, 1998.
- [45] Shubhomoy Das, Md Rakibul Islam, Nitthilan Kannappan Jayakodi, and Janardhan Rao Doppa. Active anomaly detection via ensembles: Insights, algorithms, and interpretability. *arXiv preprint arXiv:1901.08930*, 2019.
- [46] Hoang Anh Dau, Anthony Bagnall, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Annh Ratanamahatana, and Eamonn Keogh. The ucr time series archive. *IEEE/CAA Journal of Automatica Sinica*, 6(6):1293–1305, 2019.
- [47] Luke M Davis, Barry-John Theobald, Jason Lines, Andoni Toms, and Anthony Bagnall. On the segmentation and classification of hand radiographs. *International journal of neural systems*, 22(05):1250020, 2012.
- [48] Jan G De Gooijer and Rob J Hyndman. 25 years of time series forecasting. *International journal of forecasting*, 22(3):443–473, 2006.
- [49] Angus Dempster, François Petitjean, and Geoffrey I Webb. Rocket: Exceptionally fast and accurate time series classification using random convolutional kernels. *Data Mining and Knowledge Discovery*, 34(5):1454–1495, 2020.

-
- [50] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, 7(Jan):1–30, 2006.
 - [51] Houtao Deng, George Runger, Eugene Tuv, and Martyanov Vladimir. A time series forest for classification and feature extraction. *Information Sciences*, 239:142–153, 2013.
 - [52] Nilanjan Dey, Amira S Ashour, Simon James Fong, and Chintan Bhatt. *Wearable and Implantable Medical Devices: Applications and Challenges*. Academic Press, 2019.
 - [53] Hui Ding, Goce Trajcevski, Peter Scheuermann, Xiaoyue Wang, and Eamonn Keogh. Querying and mining of time series data: experimental comparison of representations and distance measures. *Proceedings of the VLDB Endowment*, 1(2):1542–1552, 2008.
 - [54] Gregory Ditzler and Robi Polikar. Incremental learning of concept drift from streaming imbalanced data. *IEEE transactions on knowledge and data engineering*, 25(10):2283–2301, 2012.
 - [55] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.
 - [56] Yueqi Duan, Jiwen Lu, Jianjiang Feng, and Jie Zhou. Deep localized metric learning. *IEEE Transactions on Circuits and Systems for Video Technology*, 28(10):2644–2656, 2017.
 - [57] Richard O Duda, Peter E Hart, and David G Stork. *Pattern classification*. John Wiley & Sons, 2012.
 - [58] Robert PW Duin and Elżbieta Pekalska. Open issues in pattern recognition. In *Computer Recognition Systems*, pages 27–42. Springer, 2005.
 - [59] Robert PW Duin and Elżbieta Pekalska. Structural inference of sensor-based measurements. In *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*, pages 41–55. Springer, 2006.
 - [60] Robert PW Duin and Elżbieta Pekalska. The dissimilarity representation for structural pattern recognition. In *Iberoamerican Congress on Pattern Recognition*, pages 1–24. Springer, 2011.
 - [61] Robert PW Duin, Fabio Roli, and Dick de Ridder. A note on core research issues for statistical pattern recognition. *Pattern recognition letters*, 23(4):493–499, 2002.

- [62] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*, 2016.
- [63] Damian Eads, Karen Glocer, Simon Perkins, and James Theiler. Grammar-guided feature extraction for time series classification. In *Proceedings of the 9th Annual Conference on Neural Information Processing Systems (NIPS'05)*, 2005.
- [64] Saba Ale Ebrahim, Javad Poshtan, Seyedh Mahboobeh Jamali, and Nader Ale Ebrahim. Quantitative and qualitative analysis of time-series classification using deep learning. *IEEE Access*, 2020.
- [65] Andrew Emmott, Shubhomoy Das, Thomas Dietterich, Alan Fern, and Weng-Keen Wong. A meta-analysis of the anomaly detection problem. *arXiv preprint arXiv:1503.01158*, 2015.
- [66] Bahaeddin Eravci and Hakan Ferhatosmanoglu. Diverse relevance feedback for time series with autoencoder based summarizations. *IEEE Transactions on Knowledge and Data Engineering*, 30(12):2298–2311, 2018.
- [67] Suilan Estevez-Velarde, Yoan Gutiérrez, Yudián Almeida-Cruz, and Andrés Montoyo. General-purpose hierarchical optimisation of machine learning pipelines with grammatical evolution. *Information Sciences*, 2020.
- [68] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Transfer learning for time series classification. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 1367–1376. IEEE, 2018.
- [69] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Adversarial attacks on deep neural networks for time series classification. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2019.
- [70] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, 33(4):917–963, 2019.
- [71] Hassan Ismail Fawaz, Benjamin Lucas, Germain Forestier, Charlotte Pelletier, Daniel F Schmidt, Jonathan Weber, Geoffrey I Webb, Lhassane Idoumghar, Pierre-Alain Muller, and François Petitjean. Incep-

- tiontime: Finding alexnet for time series classification. *arXiv preprint arXiv:1909.04939*, 2019.
- [72] Michael Fenton, James McDermott, David Fagan, Stefan Forstenlechner, Erik Hemberg, and Michael O'Neill. Ponyge2: Grammatical evolution in python. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 1194–1201. ACM, 2017.
- [73] Gilberto Fernandes, Joel JPC Rodrigues, Luiz Fernando Carvalho, Jalal F Al-Muhtadi, and Mario Lemes Proença. A comprehensive survey on network anomaly detection. *Telecommunication Systems*, 70(3):447–489, 2019.
- [74] Enrique Fernández-Blanco, Daniel Rivero, Marcos Gestal, and Julián Dorado. Classification of signals by means of genetic programming. *Soft Computing*, 17(10):1929–1937, 2013.
- [75] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. Efficient and robust automated machine learning. In *Advances in neural information processing systems*, pages 2962–2970, 2015.
- [76] Michael Flynn, James Large, and Tony 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*, pages 381–392. Springer, 2019.
- [77] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.
- [78] Tak-chung Fu. A review on time series data mining. *Engineering Applications of Artificial Intelligence*, 24(1):164–181, 2011.
- [79] Pedro Galeano and Daniel Pella Peña. Multivariate analysis in vector time series. *Resenhas do Instituto de Matemática e Estatística da Universidade de São Paulo*, 4(4):383–403, 2000.
- [80] Eric K Garcia, Sergey Feldman, Maya R Gupta, and Santosh Srivastava. Completely lazy learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(9):1274–1285, 2010.
- [81] Salvador Garcia, Joaquin Derrac, Jose Cano, and Francisco Herrera. Prototype selection for nearest neighbor classification: Taxonomy and

- empirical study. *IEEE transactions on pattern analysis and machine intelligence*, 34(3):417–435, 2012.
- [82] Salvador Garcia and Francisco Herrera. An extension on “statistical comparisons of classifiers over multiple data sets” for all pairwise comparisons. *Journal of machine learning research*, 9(Dec):2677–2694, 2008.
- [83] Dominique Gay and Vincent Lemaire. Should we reload time series classification performance evaluation? (a position paper). *arXiv preprint arXiv:1903.03300*, 2019.
- [84] Pierre Geurts. Pattern extraction for time series classification. In *European Conference on Principles of Data Mining and Knowledge Discovery*, volume 1, pages 115–127. Springer, 2001.
- [85] Mohamed F Ghalwash and Zoran Obradovic. Early classification of multivariate temporal observations by extraction of interpretable shapelets. *BMC bioinformatics*, 13(1):195, 2012.
- [86] Nikolaos Gianniotis, Sven D Kügler, Peter Tiño, and Kai L Polsterer. Model-coupled autoencoder for time series visualisation. *Neurocomputing*, 192:139–146, 2016.
- [87] Rafael Giusti, Diego F Silva, and Gustavo EAPA Batista. Improved time series classification with representation diversity and SVM. In *2016 15th IEEE International Conference on Machine Learning and Applications*, pages 1–6. IEEE, 2016.
- [88] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.
- [89] Markus Goldstein and Seiichi Uchida. A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. *PloS one*, 11(4):e0152173, 2016.
- [90] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>.
- [91] Tomasz Górecki and Maciej Łuczak. Multivariate time series classification with parametric derivative dynamic time warping. *Expert Systems with Applications*, 42(5):2305–2312, 2015.

- [92] Frank E Grubbs. Procedures for detecting outlying observations in samples. *Technometrics*, 11(1):1–21, 1969.
- [93] Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Gang Wang, Jianfei Cai, et al. Recent advances in convolutional neural networks. *Pattern Recognition*, 77:354–377, 2018.
- [94] Yasmine Guerbai, Youcef Chibani, and Bilal Hadjadji. The effective use of the one-class svm classifier for handwritten signature verification based on writer-independent parameters. *Pattern Recognition*, 48(1):103–113, 2015.
- [95] David Guijo-Rubio, Pedro Antonio Gutiérrez, Anthony Bagnall, and César Hervás-Martínez. Ordinal versus nominal time series classification.
- [96] Manish Gupta, Jing Gao, Charu C Aggarwal, and Jiawei Han. Outlier detection for temporal data: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 26(9):2250–2267, 2013.
- [97] Isabelle Guyon and André Elisseeff. An introduction to feature extraction. In *Feature extraction*, pages 1–25. Springer, 2006.
- [98] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742. IEEE, 2006.
- [99] Guo Haixiang, Li Yijing, Jennifer Shang, Gu Mingyun, Huang Yuanyue, and Gong Bing. Learning from class-imbalanced data: Review of methods and applications. *Expert Systems with Applications*, 73:220–239, 2017.
- [100] Charles R. Harris, K. Jarrod Millman, St’efan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre G’erard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

- [101] Dustin Y Harvey and Michael D Todd. Automated feature design for numeric sequence classification by genetic programming. *IEEE Transactions on Evolutionary Computation*, 19(4):474–489, 2015.
- [102] Haibo He and Yunqian Ma. *Imbalanced learning: foundations, algorithms, and applications*. John Wiley & Sons, 2013.
- [103] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [104] Jon Hills, Jason Lines, Edgaras Baranauskas, James Mapp, and Anthony Bagnall. Classification of time series by shapelet transformation. *Data Mining and Knowledge Discovery*, 28(4):851–881, 2014.
- [105] Victoria Hodge and Jim Austin. A survey of outlier detection methodologies. *Artificial intelligence review*, 22(2):85–126, 2004.
- [106] Steven CH Hoi, Doyen Sahoo, Jing Lu, and Peilin Zhao. Online learning: A comprehensive survey. *arXiv preprint arXiv:1802.02871*, 2018.
- [107] Kurt Hornik, Maxwell Stinchcombe, Halbert White, et al. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.
- [108] Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417, 1933.
- [109] Lu Hou, James T Kwok, and Jacek M Zurada. Efficient learning of timeseries shapelets. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 1209–1215, 2016.
- [110] Shima Imani and Eamonn Keogh. Natura: Towards conversational analytics for comparing and contrasting time series. In *Companion Proceedings of the Web Conference 2020*, pages 46–47, 2020.
- [111] Brian Kenji Iwana, Volkmar Frinken, Kaspar Riesen, and Seiichi Uchida. Efficient temporal pattern recognition by means of dissimilarity space embedding with discriminative prototypes. *Pattern Recognition*, 64:268–276, 2017.
- [112] Brian Kenji Iwana and Seiichi Uchida. An empirical survey of data augmentation for time series classification with neural networks. *arXiv preprint arXiv:2007.15951*, 2020.

- [113] Brian Kenji Iwana and Seiichi Uchida. Time series classification using local distance-based features in multi-modal fusion networks. *Pattern Recognition*, 97:107024, 2020.
- [114] Brijnesh Jain and Stephan Spiegel. Dimension reduction in dissimilarity spaces for time series classification. In *International Workshop on Advanced Analysis and Learning on Temporal Data*, pages 31–46. Springer, 2015.
- [115] Aren Jansen, Manoj Plakal, Ratheet Pandya, Daniel PW Ellis, Shawn Hershey, Jiayang Liu, R Channing Moore, and Rif A Saurous. Unsupervised learning of semantic audio representations. In *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 126–130. IEEE, 2018.
- [116] Young-Seon Jeong, In-Ho Kang, Myong-Kee Jeong, and Dongjoon Kong. A new feature selection method for one-class classification problems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(6):1500–1509, 2012.
- [117] Yoshihide Kakizawa, Robert H Shumway, and Masanobu Taniguchi. Discrimination and clustering for multivariate time series. *Journal of the American Statistical Association*, 93(441):328–340, 1998.
- [118] Fazle Karim, Somshubra Majumdar, and Houshang Darabi. Insights into lstm fully convolutional networks for time series classification. *IEEE Access*, 7:67718–67725, 2019.
- [119] Rohit J Kate. Using dynamic time warping distances as features for improved time series classification. *Data Mining and Knowledge Discovery*, 30(2):283–312, 2016.
- [120] Ravneet Kaur and Sarbjeet Singh. A survey of data mining and social network analysis based anomaly detection techniques. *Egyptian informatics journal*, 17(2):199–216, 2016.
- [121] Louise A Kelly, Duncan GE McMillan, Alexandra Anderson, Morgan Fippinger, Gunnar Fillerup, and Jane Rider. Validity of actigraphs uniaxial and triaxial accelerometers for assessment of physical activity in adults in laboratory conditions. *BMC medical physics*, 13(1):5, 2013.
- [122] Eamonn Keogh, Kaushik Chakrabarti, Michael Pazzani, and Sharad Mehrotra. Dimensionality reduction for fast similarity search in large

- time series databases. *Knowledge and information Systems*, 3(3):263–286, 2001.
- [123] Eamonn Keogh, Selina Chu, David Hart, and Michael Pazzani. Segmenting time series: A survey and novel approach. In *Data mining in time series databases*, pages 1–21. World Scientific, 2004.
- [124] Murphy Kevin. Machine learning: a probabilistic perspective, 2012.
- [125] Shehroz S Khan and Amir Ahmad. Relationship between variants of one-class nearest neighbors and creating their accurate ensembles. *IEEE Transactions on Knowledge and Data Engineering*, 30(9):1796–1809, 2018.
- [126] Shehroz S Khan and Michael G Madden. One-class classification: taxonomy of study and review of techniques. *The Knowledge Engineering Review*, 29(3):345–374, 2014.
- [127] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [128] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.
- [129] Judy L Klein. *Statistical visions in time: a history of time series analysis, 1662-1938*. Cambridge University Press, 1997.
- [130] Donald E Knuth. Backus normal form vs. backus naur form. *Communications of the ACM*, 7(12):735–736, 1964.
- [131] Satoru Kobayashi, Kazuki Otomo, Kensuke Fukuda, and Hiroshi Esaki. Mining causality of network events in log data. *IEEE Transactions on Network and Service Management*, 15(1):53–67, 2017.
- [132] John R Koza. *Genetic programming: on the programming of computers by means of natural selection*, volume 1. MIT press, 1992.
- [133] Bartosz Krawczyk and Michał Woźniak. Dynamic classifier selection for one-class classification. *Knowledge-Based Systems*, 107:43–53, 2016.
- [134] Bartosz Krawczyk, Michał Woźniak, and Francisco Herrera. On the usefulness of one-class classifier ensembles for decomposition of multi-class problems. *Pattern Recognition*, 48(12):3969–3982, 2015.

- [135] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
- [136] Ludmila I Kuncheva. Prototype classifiers and the big fish: The case of prototype (instance) selection. *IEEE Systems, Man, and Cybernetics Magazine*, 6(2):49–56, 2020.
- [137] Ludmila I Kuncheva and James C Bezdek. Nearest prototype classification: Clustering, genetic algorithms, or random search? *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 28(1):160–164, 1998.
- [138] Martin Långkvist, Lars Karlsson, and Amy Loutfi. A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognition Letters*, 42:11–24, 2014.
- [139] Oscar D Lara and Miguel A Labrador. A survey on human activity recognition using wearable sensors. *IEEE Communications Surveys and Tutorials*, 15(3):1192–1209, 2013.
- [140] James Large, Paul Southam, and Anthony Bagnall. Can automated smoothing significantly improve benchmark time series classification algorithms? In *International Conference on Hybrid Artificial Intelligence Systems*, pages 50–60. Springer, 2019.
- [141] Lei Le, Andrew Patterson, and Martha White. Supervised autoencoders: Improving generalization performance with unsupervised regularizers. In *Advances in Neural Information Processing Systems*, pages 107–117, 2018.
- [142] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.
- [143] Qi Lei, Jinfeng Yi, Roman Vaculin, Lingfei Wu, and Inderjit S Dhillon. Similarity preserving representation learning for time series clustering. In *IJCAI*, volume 19, pages 2845–2851, 2019.
- [144] Raphael Lenain, Jack Weston, Abhishek Shivkumar, and Emil Fristed. Surfboard: Audio feature extraction for modern machine learning. *arXiv preprint arXiv:2005.08848*, 2020.
- [145] Andrew Lensen, Bing Xue, and Mengjie Zhang. Genetic programming for evolving a front of interpretable models for data visualization. *IEEE Transactions on Cybernetics*, 2020.

- [146] Daoyuan Li, Jessica Lin, Tegawendé François D Assise Bissyande, Jacques Klein, and Yves Le Traon. Extracting statistical graph features for accurate and efficient time series classification. In *21st International Conference on Extending Database Technology*, 2018.
- [147] Yuanguai Li, Zhonghui Hu, Yunze Cai, and Weidong Zhang. Support vector based prototype selection method for nearest neighbor rules. In *International Conference on Natural Computation*, pages 528–535. Springer, 2005.
- [148] Linxia Liao. Discovering prognostic features using genetic programming in remaining useful life prediction. *IEEE Transactions on Industrial Electronics*, 61(5):2464–2472, 2014.
- [149] T Warren Liao. Clustering of time series data—a survey. *Pattern recognition*, 38(11):1857–1874, 2005.
- [150] Jessica Lin, Eamonn Keogh, Stefano Lonardi, and Bill Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, pages 2–11. ACM, 2003.
- [151] Jessica Lin, Rohan Khade, and Yuan Li. Rotation-invariant similarity in time series using bag-of-patterns representation. *Journal of Intelligent Information Systems*, 39(2):287–315, 2012.
- [152] Jessica Lin and Yuan Li. Finding structural similarity in time series data using bag-of-patterns representation. In *International conference on scientific and statistical database management*, pages 461–477. Springer, 2009.
- [153] Michele Linardi, Yan Zhu, Themis Palpanas, and Eamonn Keogh. Matrix profile goes mad: variable-length motif and discord discovery in data series. *DATA MINING AND KNOWLEDGE DISCOVERY*, 2020.
- [154] Jason Lines and Anthony Bagnall. Time series classification with ensembles of elastic distance measures. *Data Mining and Knowledge Discovery*, 29(3):565–592, 2015.
- [155] Jason Lines, Sarah Taylor, and Anthony Bagnall. Time series classification with hive-cote: The hierarchical vote collective of transformation-based ensembles. *ACM Transactions on Knowledge Discovery from Data*, 12(5):52, 2018.

- [156] Luca Lipani, Bertrand GR Dupont, Floriant Doungmene, Frank Marken, Rex M Tyrrell, Richard H Guy, and Adelina Ilie. Non-invasive, transdermal, path-selective and specific glucose monitoring via a graphene-based platform. *Nature nanotechnology*, 13(6):504–511, 2018.
- [157] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422. IEEE, 2008.
- [158] Arnaud Lods, Simon Malinowski, Romain Tavenard, and Laurent Am-saleg. Learning dtw-preserving shapelets. In *International Symposium on Intelligent Data Analysis*, pages 198–209. Springer, 2017.
- [159] JLEKS Lonardi and Pranav Patel. Finding motifs in time series. In *Proc. of the 2nd Workshop on Temporal Data Mining*, pages 53–68, 2002.
- [160] Carl H. Lubba, Sarab S. Sethi, Philip Knaute, Simon R. Schultz, Ben D. Fulcher, and Nick S. Jones. catch22: Canonical time-series characteristics. *Data Mining and Knowledge Discovery*, August 2019.
- [161] Benjamin Lucas, Ahmed Shifaz, Charlotte Pelletier, Lachlan O’Neill, Nayyar Zaidi, Bart Goethals, François Petitjean, and Geoffrey I Webb. Proximity forest: an effective and scalable distance-based classifier for time series. *Data Mining and Knowledge Discovery*, 33(3):607–635, 2019.
- [162] Bo Luo, Haoting Wang, Hongqi Liu, Bin Li, and Fangyu Peng. Early fault detection of machine tools based on deep learning and dynamic identification. *IEEE Transactions on Industrial Electronics*, 66(1):509–518, 2019.
- [163] Ge Luo, Ke Yi, Siu-Wing Cheng, Zhenguo Li, Wei Fan, Cheng He, and Yadong Mu. Piecewise linear approximation of streaming time series data with max-error guarantees. In *2015 IEEE 31st International Conference on Data Engineering*, pages 173–184. IEEE, 2015.
- [164] Yunqian Ma and Yun Fu. *Manifold learning theory and applications*. CRC press, 2011.
- [165] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

- [166] Utthara Gosa Mangai, Suranjana Samanta, Sukhendu Das, and Pinaki Roy Chowdhury. A survey of decision fusion and feature fusion strategies for pattern classification. *IETE Technical review*, 27(4):293–307, 2010.
- [167] Carlos Martín, David Quintana, and Pedro Isasi. Grammatical evolution-based ensembles for algorithmic trading. *Applied Soft Computing*, 84:105713, 2019.
- [168] Stefano Mauceri, Louis Smith, James Sweeney, and James McDermott. Subject recognition using wrist-worn triaxial accelerometer data. In *International Workshop on Machine Learning, Optimization, and Big Data*, pages 574–585. Springer, 2017.
- [169] Stefano Mauceri, James Sweeney, and James McDermott. One-class subject authentication using feature extraction by grammatical evolution on accelerometer data. In *International Conference on Metaheuristics and Nature Inspired Computing*, 2018.
- [170] Stefano Mauceri, James Sweeney, and James McDermott. Dissimilarity-based representations for one-class classification on time series. *Pattern Recognition*, 100:107122, 2020.
- [171] Brian McFee and Gert R Lanckriet. Metric learning to rank. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 775–782, 2010.
- [172] Jiangyuan Mei, Meizhu Liu, Yuan-Fang Wang, and Huijun Gao. Learning a mahalanobis distance-based dynamic time warping measure for multivariate time series classification. *IEEE transactions on Cybernetics*, 46(6):1363–1374, 2015.
- [173] Amiel Meiseles and Lior Rokach. Source model selection for deep learning in the time series domain. *IEEE Access*, 8:6190–6200, 2020.
- [174] Matthew Middlehurst, James Large, and Anthony Bagnall. The canonical interval forest (cif) classifier for time series classification. *arXiv preprint arXiv:2008.09172*, 2020.
- [175] David J Montana. Strongly typed genetic programming. *Evolutionary computation*, 3(2):199–230, 1995.
- [176] Jason H Moore and Moshe Sipper. Grammatical evolution strategies for bioinformatics and systems genomics. In *Handbook of Grammatical Evolution*, pages 395–405. Springer, 2018.

- [177] Steffen Moritz and Thomas Bartz-Beielstein. imputets: time series missing value imputation in r. *The R Journal*, 9(1):207–218, 2017.
- [178] S Mostafa Mousavi, Weiqiang Zhu, William Ellsworth, and Gregory Beroza. Unsupervised clustering of seismic signals using deep convolutional autoencoders. *IEEE Geoscience and Remote Sensing Letters*, 16(11):1693–1697, 2019.
- [179] Alex Nanopoulos, Rob Alcock, and Yannis Manolopoulos. Feature-based classification of time-series data. *International Journal of Computer Research*, 10(3):49–61, 2001.
- [180] Krystyna Napierala and Jerzy Stefanowski. Types of minority class examples and their influence on learning classifiers from imbalanced data. *Journal of Intelligent Information Systems*, 46(3):563–597, 2016.
- [181] Michel Neuhaus and Horst Bunke. *Bridging the gap between graph edit distance and kernel machines*, volume 68. World Scientific, 2007.
- [182] Ji Ni, Russ H Driberg, and Peter I Rockett. The use of an analytic quotient operator in genetic programming. *IEEE Transactions on Evolutionary Computation*, 17(1):146–152, 2012.
- [183] Miguel Nicolau and Alexandros Agapitos. Understanding grammatical evolution: Grammar design. In *Handbook of Grammatical Evolution*, pages 23–53. Springer, 2018.
- [184] Robert M Nosofsky. Attention, similarity, and the identification–categorization relationship. *Journal of experimental psychology: General*, 115(1):39, 1986.
- [185] Michael O’Neill and Conor Ryan. Grammatical evolution. In *Grammatical Evolution*, pages 33–47. Springer, 2003.
- [186] Victor M Panaretos and Yoav Zemel. Statistical aspects of wasserstein distances. *Annual Review of Statistics and Its Application*, 2018.
- [187] John Paparrizos, Chunwei Liu, Aaron J Elmore, and Michael J Franklin. Debunking four long-standing misconceptions of time-series distance measures. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 1887–1905, 2020.
- [188] John L Pearce, Lance A Waller, Howard H Chang, Mitch Klein, James A Mulholland, Jeremy A Sarnat, Stefanie E Sarnat, Matthew J

- Strickland, and Paige E Tolbert. Using self-organizing maps to develop ambient air quality classifications: a time series example. *Environmental Health*, 13(1):56, 2014.
- [189] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12(Oct):2825–2830, 2011.
- [190] Wenjie Pei, David MJ Tax, and Laurens van der Maaten. Modeling time series similarity with siamese recurrent networks. *arXiv preprint arXiv:1603.04713*, 2016.
- [191] Elzbieta Pekalska and Robert PW Duin. Dissimilarity representations allow for building good classifiers. *Pattern Recognition Letters*, 23(8):943–956, 2002.
- [192] Elzbieta Pekalska, Robert PW Duin, and Pavel Paclík. Prototype selection for dissimilarity-based classifiers. *Pattern Recognition*, 39(2):189–208, 2006.
- [193] Davi Pereira-Santos, Ricardo Bastos Cavalcante Prudêncio, and André CPLF de Carvalho. Empirical investigation of active learning strategies. *Neurocomputing*, 326:15–27, 2019.
- [194] François Petitjean, Germain Forestier, Geoffrey I Webb, Ann E Nicholson, Yanping Chen, and Eamonn Keogh. Dynamic time warping averaging of time series allows faster and more accurate classification. In *2014 IEEE international conference on data mining*, pages 470–479. IEEE, 2014.
- [195] Florian Pfisterer, Laura Beggel, Xudong Sun, Fabian Scheipl, and Bernd Bischl. Benchmarking time series classification–functional data vs machine learning approaches. *arXiv preprint arXiv:1911.07511*, 2019.
- [196] Hieu Pham, Melody Y Guan, Barret Zoph, Quoc V Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. *arXiv preprint arXiv:1802.03268*, 2018.
- [197] Clifton Phua, Vincent Lee, Kate Smith, and Ross Gayler. A comprehensive survey of data mining-based fraud detection research. *arXiv preprint arXiv:1009.6119*, 2010.

- [198] Stjepan Picek, Erik Hemberg, Domagoj Jakobovic, and Una-May O'Reilly. One-class classification of low volume dos attacks with genetic programming. In *Genetic Programming Theory and Practice XV*, pages 149–168. Springer, 2018.
- [199] Roberto HW Pinheiro, George DC Cavalcanti, and Ren Tsang. Combining dissimilarity spaces for text categorization. *Information Sciences*, 406:87–101, 2017.
- [200] Riccardo Poli, William B Langdon, Nicholas F McPhee, and John R Koza. *A field guide to genetic programming*. Lulu. com, 2008.
- [201] Warren B Powell. Perspectives of approximate dynamic programming. *Annals of Operations Research*, 241(1-2):319–356, 2016.
- [202] Kenneth V Price. Differential evolution. In *Handbook of Optimization*, pages 187–214. Springer, 2013.
- [203] Han Qiu, Hoang Thanh Lam, Francesco Fusco, and Mathieu Sinn. Learning correlation space for time series. *arXiv preprint arXiv:1802.03628*, 2018.
- [204] Hojjat Rakhshani, Hassan Ismail Fawaz, Lhassane Idoumghar, Germain Forestier, Julien Lepagnot, Jonathan Weber, Mathieu Brévigliers, and Pierre-Alain Muller. Neural architecture search for time series classification.
- [205] Thanawin Rakthanmanon, Bilson Campana, Abdullah Mueen, Gustavo Batista, Brandon Westover, Qiang Zhu, Jesin Zakaria, and Eamonn Keogh. Addressing big data time series: Mining trillions of time series subsequences under dynamic time warping. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 7(3):10, 2013.
- [206] Chotirat Ann Ratanamahatana and Eamonn Keogh. Making time-series classification more accurate using learned constraints. In *Proceedings of the 2004 SIAM international conference on data mining*, pages 11–22. SIAM, 2004.
- [207] Conor Ryan, Michael O'Neill, and JJ Collins. *Handbook of Grammatical Evolution*. Springer, 2018.
- [208] Hasim Sak, Andrew W Senior, and Françoise Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. 2014.

- [209] Yoshifumi Sakai and Shunsuke Inenaga. A reduction of the dynamic time warping distance to the longest increasing subsequence length. *arXiv preprint arXiv:2005.09169*, 2020.
- [210] Ruslan Salakhutdinov and Geoff Hinton. Learning a nonlinear embedding by preserving class neighbourhood structure. In *Artificial Intelligence and Statistics*, pages 412–419, 2007.
- [211] Patrick Schäfer. The boss is concerned with time series classification in the presence of noise. *Data Mining and Knowledge Discovery*, 29(6):1505–1530, 2015.
- [212] Patrick Schäfer and Ulf 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, 2017.
- [213] Patrick Schäfer and Ulf Leser. Teaser: Early and accurate time series classification. *Data Mining and Knowledge Discovery*, pages 1–27, 2020.
- [214] Robert E Schapire. Explaining adaboost. In *Empirical inference*, pages 37–52. Springer, 2013.
- [215] Thomas Schlegl, Philipp Seeböck, Sebastian M Waldstein, Ursula Schmidt-Erfurth, and Georg Langs. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In *International Conference on Information Processing in Medical Imaging*, pages 146–157. Springer, 2017.
- [216] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Kernel principal component analysis. In *International conference on artificial neural networks*, pages 583–588. Springer, 1997.
- [217] Bernhard Schölkopf, Robert C Williamson, Alex J Smola, John Shawe-Taylor, and John C Platt. Support vector method for novelty detection. In *Advances in neural information processing systems*, pages 582–588, 2000.
- [218] Thomas Schreiber and Andreas Schmitz. Discrimination power of measures for nonlinearity in a time series. *Physical Review E*, 55(5):5443, 1997.
- [219] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

- [220] Matthias Seeger. Learning with labeled and unlabeled data (technical report). *Edinburgh University*, 2000.
- [221] Razieh Sheikhpour, Mehdi Agha Sarram, Sajjad Gharaghani, and Mohammad Ali Zare Chahooki. A survey on semi-supervised feature selection methods. *Pattern Recognition*, 64:141–158, 2017.
- [222] Kejian Shi, Hongyang Qin, Chijun Sima, Sen Li, Lifeng Shen, and Qianli Ma. Dynamic barycenter averaging kernel in rbf networks for time series classification. *IEEE Access*, 7:47564–47576, 2019.
- [223] Hyun Joon Shin, Dong-Hwan Eom, and Sung-Shick Kim. One-class support vector machines—an application in machine fault detection and classification. *Computers & Industrial Engineering*, 48(2):395–408, 2005.
- [224] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.
- [225] Bharat Singh and Larry S Davis. An analysis of scale invariance in object detection snip. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3578–3587, 2018.
- [226] Dominik Sobania and Franz Rothlauf. Challenges of program synthesis with grammatical evolution. In *European Conference on Genetic Programming (Part of EvoStar)*, pages 211–227. Springer, 2020.
- [227] Sumit Soman et al. High performance eeg signal classification using classifiability and the twin svm. *Applied Soft Computing*, 30:305–318, 2015.
- [228] Lauge Sørensen, Marco Loog, Pechin Lo, Haseem Ashraf, Asger Dirksen, Robert PW Duin, and Marleen De Bruijne. Image dissimilarity-based quantification of lung disease from ct. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 37–44. Springer, 2010.
- [229] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.
- [230] Alexandra Stefan, Vassilis Athitsos, and Gautam Das. The move-split-merge metric for time series. *IEEE transactions on Knowledge and Data Engineering*, 25(6):1425–1438, 2012.

- [231] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alex Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. *arXiv preprint arXiv:1602.07261*, 2016.
- [232] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.
- [233] Chang Wei Tan, Christoph Bergmeir, Francois Petitjean, and Geoffrey I Webb. Time series regression. *arXiv preprint arXiv:2006.12672*, 2020.
- [234] Chang Wei Tan, Francois Petitjean, Eamonn Keogh, and Geoffrey I Webb. Time series classification for varying length series. *arXiv preprint arXiv:1910.04341*, 2019.
- [235] Chang Wei Tan, François Petitjean, and Geoffrey I Webb. Fastee: Fast ensembles of elastic distances for time series classification. *Data Mining and Knowledge Discovery*, 34(1):231–272, 2020.
- [236] Wensi Tang, Lu Liu, and Guodong Long. Interpretable time-series classification on few-shot samples. *arXiv preprint arXiv:2006.02031*, 2020.
- [237] David MJ Tax and Robert PW Duin. Support vector data description. *Machine learning*, 54(1):45–66, 2004.
- [238] Isaac Triguero, Joaquín Derrac, Salvador Garcia, and Francisco Herrera. A taxonomy and experimental study on prototype generation for nearest neighbor classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(1):86–100, 2012.
- [239] Kwok L Tsui, Nan Chen, Qiang Zhou, Yizhen Hai, and Wenbin Wang. Prognostics and health management: A review on data driven approaches. *Mathematical Problems in Engineering*, 2015, 2015.
- [240] Claudio Turchetti and Laura Falaschetti. A machine learning method to determine intrinsic dimension of time series data. In *2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pages 303–307. IEEE, 2017.

- [241] Arijit Ukil, Soma Bandyopadhyay, Chetanya Puri, and Arpan Pal. Iot healthcare analytics: The importance of anomaly detection. In *2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA)*, pages 994–997. IEEE, 2016.
- [242] Lev V Utkin, Vladimir S Zaborovsky, Alexey A Lukashin, Sergey G Popov, and Anna V Podolskaja. A siamese autoencoder preserving distances for anomaly detection in multi-robot systems. In *2017 International Conference on Control, Artificial Intelligence, Robotics & Optimization (ICCAIRO)*, pages 39–44. IEEE, 2017.
- [243] Merijn Van Erp, Louis Vuurpijl, and Lambert Schomaker. An overview and comparison of voting methods for pattern recognition. In *Proceedings Eighth International Workshop on Frontiers in Handwriting Recognition*, pages 195–200. IEEE, 2002.
- [244] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.
- [245] Cédric Villani. *Topics in optimal transportation*. American Mathematical Soc., 2003.
- [246] Marco Virgolin, Tanja Alderliesten, and Peter AN Bosman. On explaining machine learning models by evolving crucial and compact features. *Swarm and Evolutionary Computation*, 53:100640, 2020.
- [247] Haishuai Wang, Qin Zhang, Jia Wu, Shirui Pan, and Yixin Chen. Time series feature learning with labeled and unlabeled data. *Pattern Recognition*, 89:55–66, 2019.
- [248] Jiang Wang, Zicheng Liu, Ying Wu, and Junsong Yuan. Learning actionlet ensemble for 3d human action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 36(5):914–927, 2013.
- [249] Siqi Wang, Qiang Liu, En Zhu, Fatih Porikli, and Jianping Yin. Hyperparameter selection of one-class support vector machine by self-adaptive data shifting. *Pattern Recognition*, 74:198–211, 2018.
- [250] Xiaoyue Wang, Abdullah Mueen, Hui Ding, Goce Trajcevski, Peter Scheuermann, and Eamonn Keogh. Experimental comparison of representation methods and distance measures for time series data. *Data Mining and Knowledge Discovery*, 26(2):275–309, 2013.

- [251] Zhiguang Wang, Weizhong Yan, and Tim Oates. Time series classification from scratch with deep neural networks: A strong baseline. In *2017 International joint conference on neural networks (IJCNN)*, pages 1578–1585. IEEE, 2017.
- [252] James Wayman, Anil Jain, Davide Maltoni, and Dario Maio. An introduction to biometric authentication systems. *Biometric Systems*, pages 1–20, 2005.
- [253] Kilian Q Weinberger and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10(2), 2009.
- [254] Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics bulletin*, 1(6):80–83, 1945.
- [255] Seunghye J Wilson. Data representation for time series data mining: time domain approaches. *Wiley Interdisciplinary Reviews: Computational Statistics*, 9(1), 2017.
- [256] Akihiro Yamaguchi and Takeichiro Nishikawa. Oclts: One-class learning time-series shapelets. *International Journal of Data Mining Science*, 1(1):24–32, 2019.
- [257] Shanchao Yang, Jing Liu, Kai Wu, and Mingming Li. Learn to generate time series conditioned graphs with generative adversarial nets. *arXiv preprint arXiv:2003.01436*, 2020.
- [258] Lexiang Ye and Eamonn 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.
- [259] Chin-Chia Michael Yeh, Yan Zhu, Liudmila Ulanova, Nurjahan Begum, Yifei Ding, Hoang Anh Dau, Diego Furtado Silva, Abdullah Mueen, and Eamonn Keogh. Matrix profile i: all pairs similarity joins for time series: a unifying view that includes motifs, discords and shapelets. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on*, pages 1317–1322. IEEE, 2016.
- [260] Chin-Chia Michael Yeh, Yan Zhu, Liudmila Ulanova, Nurjahan Begum, Yifei Ding, Hoang Anh Dau, Zachary Zimmerman, Diego Furtado Silva, Abdullah Mueen, and Eamonn Keogh. Time series joins, motifs, discords and shapelets: a unifying view that exploits the matrix profile. *Data Mining and Knowledge Discovery*, 32(1):83–123, 2018.

- [261] Yi Zheng, Qi Liu, Enhong Chen, J Leon Zhao, Liang He, and Guangyi Lv. Convolutional nonlinear neighbourhood components analysis for time series classification. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 534–546. Springer, 2015.