Explainable Classification of Uncertain Time Series

Michael Franklin Mbouopda

UNIVERSITY CLERMONT AUVERGNE

Laboratory of Informatics, Modelling and Optimization of the Systems -
LIMOS - UMR 6168

DOCTORAL SCHOOL
SCIENCES FOR THE ENGINEER

Ph.D. thesis presented to obtain the title of

DOCTOR OF COMPUTER SCIENCE

Speciality : ARTIFICIAL INTELLIGENCE

EXPLAINABLE CLASSIFICATION OF UNCERTAIN TIME SERIES

Michael Franklin MBOUOPDA

Defense date: December 13, 2022

Jury

Anthony BAGNALL (R)	-	University of East Anglia
Sebastien DESTERCKE (R)	-	Heudiasyc, University of Technology of Compiegne
Elisa FROMONT (E)	-	IRISA, University of Rennes 1
Emmanuel GANGLER (E)	-	LPC, University Clermont Auvergne
David HILL (E)	-	LIMOS, University Clermont Auvergne
Themis PALPANAS (E)	-	LIPADE, Universite Paris Cite
Engelbert MEPHU NGUIFO (A)	-	LIMOS, University Clermont Auvergne

(R): Reviewer, (E): Examinator, (A): Advisor

Acknowledgments

This work would not have been possible without the support, in any kind, of many persons. I may not be able to cite all of these persons in this report, but here are some of them:

- My supervisor Engelbert MEPHU NGUIFO for guiding me since my Master internship in LIMOS up to the end of this thesis,
- the Miners team for constructive discussions during our meetings and all the joyful moments we had in and outside the lab,
- the members of the jury for accepting to assess this work and for their constructive feedback,
- my family for their inestimable support,
- my friends.

This work has been funded by the French Ministry of Higher Education, Research and Innovation, the LabEx IMobS3 and the CNRS PEPS project TransiXplore. Thank to the UEA & UCR Time Series Classification Repository which provides the datasets used for our experiments.

Abstract

Time series classification is one of the must studied and applied time series analysis tasks. Several methods have been proposed to perform this task accurately, efficiently and sometimes in an explainable way. However, situations where the time series are made of uncertain values are still under-explored although any physical measurement is subject to uncertainty. The existing works in this fields are based on uncertain similarity measures such as DUST, MUNICH, and FOTS which have the same main limitation of not propagating uncertainty to the next step of the classification process. This behavior causes the last parts of the process to treat the data as if they were certain while they are not, leading to untrustable predictions. This thesis tackles this limitation by proposing efficient, robust and explainable methods for uncertain time series classification (uTSC). We start by proposing a general framework for uncertain time series classification which propagates uncertainty from the beginning to the end of the process. Then, we instantiate this framework using uncertainty propagation arithmetic to propose the UST model which outperformed existing uTSC models while being explainable. We continue by improving the scalability of UST by proposing the SAST and the uSAST models. SAST is a novel accurate, scalable and interpretable method that we propose for time series classification. uSAST is the extension of SAST to uTSC. We show the effectiveness our methods on simulated datasets, on state-of-the-art datasets, and on a real-world uncertain time series dataset from the astrophysics domain. The source codes and the data used in the work are all available publicly.

Keywords: Time series, classification, uncertainty, explainability, shapelet, astrophysics, transient.

Résumé

La classification de séries temporelles est l'une des tâches d'analyse de séries temporelles les plus étudiées et les plus appliquées. Plusieurs méthodes performantes et des fois interprétables ont été proposées pour réaliser cette tâche. Cependant, les cas où les séries temporelles sont faites de valeurs incertaines restent sous-explorés, et ceci malgré que toute mesure physique soit sujette à incertitude. Les travaux existants dans ce domaine sont basés sur des mesures de similarité incertaine telles que DUST, MUNICH et FOTS qui partagent la principale limite de ne pas propager l'incertitude à la prochaine étape de la classification. Par conséquent, les dernières étapes du processus de classification ne sont pas conscientes du fait que les données sont incertaines et les traitent donc comme si elles étaient certaines, conduisant ainsi à des prédictions non fiables. Cette thèse a pour but de corriger cette limite en proposant des méthodes efficaces, robustes et interprétables pour la classification de séries temporelles incertaines. Nous commençons par proposer un cadre général pour la classification de séries temporelles incertaines qui propage l'incertitude du début à la fin du processus de classification. Nous instancions ensuite ce cadre en utilisant l'arithmétique de propagation de l'incertitude pour proposer la méthode UST qui a donné des résultats meilleurs que ceux données par les méthodes existantes de classification de séries temporelles incertaines tout en étant interprétable. Par la suite, nous améliorons le temps de calcul requis par UST en proposant les méthodes SAST et uSAST. SAST est une nouvelle approche performante, rapide et interprétable que nous avons proposé pour la classification de séries temporelles, et uSAST est son extension aux séries temporelles incertaines. Nous évaluons nos méthodes sur des jeux données simulées, sur des données de l'état de l'art et sur un jeu de données réel provenant du domaine de l'astrophysique. Les codes sources et les données utilisés dans ce travail sont rendus disponible sur internet.

Mots clés: Série temporelle, classification, incertitude, interprétabilité, shapelet, astrophysique, objet transitoire.

Contents

List of abbreviations vii

0 Introduction 1
- 0.1 Context . . . 1
- 0.2 Uncertainty, not a bad thing ! . . . 2
- 0.3 Main contributions . . . 4
- 0.4 Report structure . . . 5

1 Background and related work 7
- 1.1 Background . . . 7
  - 1.1.1 What is a time series . . . 7
  - 1.1.2 Notion of uncertainty . . . 8
  - 1.1.3 Time series classification . . . 12
- 1.2 Related work . . . 14
  - 1.2.1 Certain time series classification approaches . . . 14
  - 1.2.2 Uncertain time series classification approaches . . . 24
- 1.3 Our proposed general approach for uTSC . . . 28
- 1.4 Conclusion . . . 28

2 Uncertain Time Series Classification With Shapelet Transform 31
- 2.1 Introduction . . . 31
- 2.2 UED: a new uncertain dissimilarity measure . . . 32
- 2.3 UST: The uncertain shapelet transform classification . . . 35
  - 2.3.1 Definition of concepts . . . 35
  - 2.3.2 Uncertain shapelet transform classification . . . 37
  - 2.3.3 UST time complexity . . . 39
- 2.4 Experiments . . . 40
  - 2.4.1 Compared models . . . 40
  - 2.4.2 Datasets . . . 41
  - 2.4.3 Results . . . 42
- 2.5 Conclusion . . . 45

3 Scalable and Accurate Subsequence Transform for TSC 47
- 3.1 Introduction . . . 48
- 3.2 SAST: Scalable and Accurate Subsequence Transform . . . 49
  - 3.2.1 Reducing the number of shapelet candidates . . . 49
  - 3.2.2 Identify shapelets using feature importance analysis . . . 53
  - 3.2.3 Time series classification with SAST . . . 54
  - 3.2.4 SAST time complexity . . . 55
  - 3.2.5 Ensemble of SAST models . . . 56

3.3 Experiments . . . 56
3.3.1 Accuracy . . . 57
3.3.2 Scalability . . . 65
3.3.3 Interpretability . . . 67
3.4 Conclusion . . . 70

4 Explainable Classification of Astronomical uTS **75**
4.1 Introduction . . . 75
4.2 Uncertain Subsequence Transform Classification . . . 77
4.3 Experiment . . . 79
4.3.1 The PLAsTiCC dataset . . . 79
4.3.2 Results et discussion . . . 80
4.4 Conclusion and future directions . . . 87

5 General conclusion and future directions **89**
5.1 Conclusion . . . 89
5.1.1 Main contributions . . . 90
5.1.2 Scientific valorization . . . 90
5.1.3 Open sourced codes and data . . . 90
5.2 Limitations and perspectives . . . 91
5.3 Recommendations . . . 92
5.4 List of publications . . . 92
5.4.1 First-authored publications . . . 92
5.4.2 Non first-authored publications . . . 93

Appendices **95**

A Appendix for chapter 2 **97**
A.1 Models performance on each dataset . . . 97
A.2 Scalability of SAST, SASTEN and SASTEN-A regarding the dataset size . . . 99

B Appendix for chapter 3 **101**
B.1 SAST-based methods results . . . 101
B.2 uSAST-based methods results . . . 102
B.3 Results of state-of-the-art models . . . 103

Bibliography **105**

List of abbreviations

DTW Dynamic Time Warping

ED Euclidean distance

FCN Fully connected convolutional neural network

FS Fast Shapelet

k-NN k Nearest Neighbor classifier

LCR Local Cascade Ensemble

LS Learning time series Shapelet

MLP Multi layer perceptron

PPV Proportion of Positive Values

ResNet Residual neural network

RISE Random Interval Spectral Ensemble

SAST Scalable and Accurate Subsequence Transform

SAX Symbolic Aggregate approXimation

SDT Shapelet Decision Tree

SFA Symbolic Fourrier Approximation

SIPTA Society for Imprecise Probabilities: Theories and Applications

SOTA State-of-the-art

SOTA State-of-the-art

STC Shapelet Transform Classifier

STSF Supervised time series forest

TDE Temporal Dictionary Ensemble

TS-CHIEF Time Series Combination of Heterogeneous andIntegrated Embedding Forest

TSC Time series classification

TSF Time Series Forest

UED Uncertain euclidean distance

uSAST Uncertain Scalable and Accurate Subsequence Transform

UST Uncertain Shapelet Transform

uTSC Uncertain time series classification

WEASEL Word ExtrAction for time SEries cLassication

List of Figures

1.1 A time series modeling the the evolution of daily new covid-19 cases in Africa . . . 8
1.2 A time series modeling the groundwater level in the french department Puy-de-dôme . . . 9
1.3 A multivariate time series modeling the successive positions of my right ankle when performing the Taekwondo roundhouse kick. . . . 9
1.4 Categorization of uncertainty types . . . 11
1.5 Illustration of uncertain time series . . . 12
1.6 Time series analysis tasks hierarchy . . . 13
1.7 Natural alignment of both time series. $ED(T_1, T_2) = 2.82$ . . . 15
1.8 Elastic alignment of both time series. $DTW(T_1, T_2) = 0$ . . . 16
1.9 Relevant intervals in a dataset (image obtained by annotating an original image from [Lines *et al.* 2018]) . . . 17
1.10 Overview of time series classification based on intervals . . . 17
1.11 Illustration of shapelets (image obtained by annotating an original image from [Lines *et al.* 2018]) . . . 18
1.12 Overview of time series classification based on shapelets . . . 19
1.13 Illustration of a dictionary dataset [Lines *et al.* 2018] . . . 20
1.14 SAX illustration [Lin *et al.* 2007]. The string representing the time series is **baabccbc**. . . . 21
1.15 Illustration of spectral dataset [Lines *et al.* 2018] . . . 21
1.16 Overview of existing time classification approaches . . . 24
1.17 Overview of existing uncertain time classification approaches . . . 28
1.18 Overview of the new proposed uncertain time classification approach 28

2.1 An illustration of a node in a shapelet decision tree for a binary time series classification . . . 36
2.2 Uncertain time series classification process . . . 39
2.3 Illustration of uncertainty for an instance from the Chinatown dataset. the uncertainty level is $c = 0.6$ . . . 43
2.4 Critical difference diagrams of UED-based models regarding the ordering strategy for some levels of uncertainty . . . 44
2.5 Critical difference diagrams of models for some levels of uncertainty . 45

3.1 Illustration of invariance in recognition. . . . 50
3.2 Three randomly selected instances from the Chinatown dataset. The instance on the right is probably from class 2 since it does not start with a valley . . . 51
3.3 Some randomly selected instances from the Chinatown dataset. The instance on the right is definitely from class 2 . . . 51

3.4 Top 5 shapelets extracted for each class of the Chinatown dataset by the shapelet transform algorithm. 52
3.5 Shapelets extracted by STC on the Chinatown dataset using a single randomly selected instance per class to generate shapelet candidates. 53
3.6 Overview of the SAST method 55
3.7 Comparison of STC-k to STC in terms of accuracy 58
3.8 Critical difference diagram between STC and STC-k 58
3.9 Critical difference diagram between approximated SAST models . . . 59
3.10 Pairwise comparison of model accuracies 59
3.11 Critical difference diagram between SAST models 60
3.12 Pairwise comparison of SAST, STC and STC-1 61
3.13 Critical difference diagram between SAST, STC and STC-1 61
3.14 Pairwise comparison of SAST, ELIS++, LS and FS 62
3.15 Critical difference diagram between SAST, LS, FS and ELIS++ . . . 62
3.16 SAST vs SOTA 63
3.17 SAST, STC-1 and STC percentage of wins per dataset type 64
3.18 SAST, ELIS++, LS and FS percentage of wins per dataset type considering the 35 datasets used in ELIS++ paper. 65
3.19 SAST, HIVE-COTE, TS-CHIEF and ROCKET percentage of wins per dataset type 66
3.20 Running time (in second) of each model 66
3.21 Top 5 shapelets learned by SAST-Ridge on Chinatown. 68
3.22 Top 5 shapelets learned by SAST-RF on Chinatown. 68
3.23 Explanation of SAST-Ridge predictions on two random test instances 69
3.24 Explanation of SAST-RF predictions on two test instances 69

4.1 Two supernova from PLAsTiCC. They look similar in terms of shapes although they are from distinct classes. 81
4.2 Local explainability of a Supernova Type Ia (top) and a Core-collapse Supernova Type II (bottom) 85
4.3 The top 20 most discriminative subsequences in the PLAsTiCC dataset 86

A.1 Running time in seconds of SAST, SASTEN and SASTEN-A regarding the number of time series 99

List of Tables

1.1 Summary of the current time series classification state-of-the-art (SOTA) . . . 25

2.1 Summary of the models that are compared in our experiments. . . . 41
2.2 Datasets Description . . . 42

3.1 List of models used in our experiments . . . 74
3.2 Average accuracy of each model per problem type . . . 74

4.1 Results on PLAsTiCC averaged over 3 runs. . . . 81
4.2 uSASTd performance regarding if the object are galactic or extra-galactic, are from the DDF or WFD. . . . 82
4.3 Results on PLAsTiCC averaged over 3 runs when uncertainty is ignored. 83
4.4 uSASTd vs SOTA results. . . . 83

A.2 Accuracy of models on 72 UEA & UCR datasets (average over 5 runs). The numbers are rounded at 2 decimals. . . . 97
A.2 Accuracy of models on 72 UEA & UCR datasets (average over 5 runs). The numbers are rounded at 2 decimals. . . . 98
A.2 Accuracy of models on 72 UEA & UCR datasets (average over 5 runs). The numbers are rounded at 2 decimals. . . . 99
A.1 Dataset identifiers, names, and types. Datasets marked with a star are those used for the experiment which results are presented in Section 3.3.1.2 and datasets marked with a *plus* sign are those used for the experiment which results are given in Section 3.3.1.4. . . . 100

B.1 SAST<X> models results on the PLAsTiCC dataset average over 3 runs. The stared row is the run that achieved the lowest cross entropy loss. . . . 101
B.2 uSAST<X> models results on the PLAsTiCC dataset average over 3 runs. The row with the star is the run that achieved the lowest cross entropy loss. Subsequence length: from 16 to 32 with a step of 1. . . . 102
B.3 uSAST<X> models results on the PLAsTiCC dataset average over 3 runs. The row with the star is the run that achieved the lowest cross entropy loss. Subsequence length: from 20 to 60 with a step of 10. . . . 103
B.4 Performance of state-of-the-art models average over 3 runs. MUSE uses a linear classifier. The row with the star is the run that achieved the lowest cross entropy loss . . . 103

Chapter 0

Introduction

This chapter gives the motivation of this thesis and summarizes its main contributions.

Contents

0.1 Context 1
0.2 Uncertainty, not a bad thing ! 2
0.3 Main contributions 4
0.4 Report structure 5

0.1 Context

In this era of Big Data, machine learning (ML) has become ubiquitous in almost any aspect of human life. Whether it is in the field of transportation [Lopez Conde & Twinn 2019, Zheng *et al.* 2021], medicine [May 2021, Miller *et al.* 2022], industry or in physics [Boone 2019, Leoni *et al.* 2021]. ML is used not only to assist corresponding domain experts, but also to improve how the task is done and the service quality. ML is used to make predictions and to discover new knowledge from raw data. This is made possible by the data collection capabilities that are available today and the proliferation of tremendous techniques to analyze different types of data including tabular, image , video, audio and time series data. Most of the time however, and specifically for time series, the data are required to be precise for ML algorithms to perform well. This is not the case in every application as the data are sometimes uncertain because of many factors including noise, sensors precision, privacy preservation, and collection methods [Mazzi *et al.* 2019, Abdar *et al.* 2021]. Being able to analyze uncertain data is at least as important as being able to analyze certain data. The goal of this thesis is to build ML methods for the analysis of uncertain time series data. Although there are many types of uncertainty, in this work we focus on imprecision, a special type of uncertainty. There are three key properties that we would like our methods to have, namely:

- **Efficiency**: the methods should produce accurate results, uses as less resource as possible, and be competitive with state-of-the-art methods.

- **Robustness**: the methods should be resilient to the variation of data uncertainty.
- **Explainability**: the methods should be inherently explainable or explainable by other means.

Let us discuss these three properties in details and understand why they are important to have. Efficiency is the ability of a machine learning model to achieve good performance while being trained using a "small" quantity of memory and computation time [Hernandez & Brown 2020]. The magnitude of "small" quantity is application dependent and may vary a lot. For instance, a model that detects anomalous heartbeats should report anomalies as soon as they appear and not many hours after the patient cannot be saved anymore [Lu *et al.* 2022]. On the contrary, a model trained to detect plagiarism could take three hours to run. For Internet Of Things systems, which are governed by limited memory and computation power, efficiency is a must [Sliwa *et al.* 2020]. In any case, it is always good to have models that are efficient as inefficient ones have higher carbon footprint [Patterson *et al.* 2022].

The robustness property is another important qualifier of machine learning models and reflects its capability of achieving similar performance on both training and new data. This property is even more important nowadays as it has been proved that machine learning models can be fooled by malicious individuals (adversaries), noisy and uncertain data [Fawaz *et al.* 2019a, Yang *et al.* 2020]. In this work, the importance of this property is emphasized by the fact that we are dealing with uncertain data.

The explainability of a machine learning model is the ability to explain its decisions, to describe its weaknesses and strengths, and to convey an understanding of how it will behave in the future [DARPA]. Beyond understanding the model and increasing its faithfulness and adoption by humans, explainability helps in debugging machine learning models by revealing the features used by the model to make its predictions [Ribeiro *et al.* 2016]. In the case where the wrong features are used, the model can be modified correspondingly.

With these three characteristics, we would like to ensure in the first hand that the proposed methods work correctly and do what they have been built for using acceptable amount of resources. On the second hand, we would like the proposed methods to be adopted with confidence by domain experts as well as any end users.

0.2 Uncertainty, not a bad thing !

Uncertainty is ubiquitous in real life, the majority of the decisions we take everyday is based on uncertain knowledge. For instance, we choose how we dress regarding an uncertain and changing forecasting of the weather [Slingo & Palmer 2011]; we plan our future without having the certitude that we are going to live up to that future; we learn lot of things at school hoping that it will be useful someday in someway [Kauffman *et al.* 2022]. There are many of such examples. Similarly, any

collected data is associated with an uncertainty coming from the precision of sensor used, the environmental condition of the measurement, the source of the data, and other application-dependent constraints. It is sometimes possible to reduce the uncertainty, but it cannot be completely suppressed [Taylor 1996]. Therefore, uncertainty cannot be avoided and needs to be properly taken into account in machine learning algorithms as learning from uncertain data may lead to uncertain and inaccurate predictions.

Uncertainty is usually seen as a problem, a hindrance to learning from data. This is why it is usually handled during the preprocessing step, during which some assumptions are considered in order to get rid of uncertainty. The preprocessing of uncertainty requires domain knowledge, limiting the usability of this approach of uncertainty handling. Furthermore, assumptions made for getting rid of uncertainty actually add more uncertainty in the process as they are based on an incomplete knowledge of the system that generated the data.

Uncertainty comes with challenges for decision-making systems [Stanton & Roelich 2021], but it also brings some advantages. In particular, uncertain data can be used to model situations on which we have incomplete knowledge and on which we do not have a total control like modeling the climate change; they are more expressive and less prone to assumptions that may not hold in practice. Learning from uncertain data has gained a lot of interest recently. For instance, there is a community working on imprecise probabilities (SIPTA [SIPTA]) , there are international workshops organized for uncertain machine learning (Workshop on Uncertain Machine Learning [WUML 2020], Online Learning from Uncertain Data Streams [OLUD 2022]). For these communities, uncertainty is not a problem, but an additional input that should be taken into account in order to build more robust and trustable machine learning systems. These systems are expected to be aware of uncertainty, should be at least as effective as if there was no uncertainty, and should be used without requiring domain knowledge.

This thesis focuses on the classification of uncertain time series. Unlike, regular/certain time series which are generated from a stochastic process assumed to be completely known, uncertain time series are generated by processes that are only partially known. Some authors have worked on uncertain time series classification (uTSC) and have proposed different methods to perform this task. The main component of all these works is a similarity measure for uncertain data. These measures are named uncertain similarity measures [Dallachiesa *et al.* 2012]. They take as input two uncertain time series, and output a real number representing the similarity between the two objects. We claim in this work that modeling the similarity with a real number is enough for certain data, but is not sufficient for uncertain data. In fact, as the compared objects are uncertain, their similarity should also be uncertain. Existing works have never discussed how uncertain is the similarity computed by their uncertain measures, how this uncertainty might influence the final prediction, and even less, how this uncertainty could be computed. In this work, we address these limitations.

0.3 Main contributions

We mentioned in the previous section that existing uncertain time series classification methods share the same limitation of not providing the similarity uncertainty. In addition to this limitation, we identified three other lacks in the state-of-the-art of uncertain time series classification, namely the absence of published applications on real uncertain datasets and the difficulty to reproduce existing works. The four main contributions that we did throughout this thesis are guided by these limitations and the key properties presented in Section 0.1. These contributions are described in the following subsections:

Contribution 1: Uncertain shapelet transform

We first observed that with the existing uTSC approaches, uncertainty is not handled throughout the whole classification process. This is because the proposed uncertain similarities give the similarity without any information about the uncertainty on that similarity. This behavior is firstly not natural as the compared objects are uncertain. Secondly, it is misleading, as the end-user may think that the similarity between two uncertain objects is certain though it is not guaranteed. We tackled this limitation by proposing an explainable and accurate method for uTSC named UST, for Uncertain Shapelet Transform. UST is described in detailed in Chapter 2.

Contribution 2: Scalable subsequence transform

Shapelet approaches are known to be accurate and interpretable. However, they are computationally expensive. In this thesis, we proposed a new design of shapelet-based TSC, allowing us to significantly improve the scalability of shapelet approaches while slightly increasing the classification performance. Our method is named SAST for Scalable and Accurate Subsequence Transform and it is presented in Chapter 3. In Chapter 4, we extend SAST to uncertain time series classification.

Contribution 3: Real-word application

The authors of existing uTSC methods have limited their experiments on datasets with simulated uncertainties. This observation may question the usability of these methods in real world. In this thesis, we applied our method on real uncertain time series dataset. As described in Chapter 4, our method achieves good classification performance while being interpretable by astrophysicists.

Contribution 4: Reproducibility

The last contribution of this thesis is that all the datasets used, including the datasets with the simulated uncertainties that we created are publicly available. Elsewhere, the source codes of our experiments are accessible on public repositories. We wanted to make this work easily reproducible by anyone in order to facilitate

subsequent contributions to the field of uncertain time series classification in particular, and in the field of uncertain time series analysis in general.

0.4 Report structure

In the previous section, we gave our main contributions while specifying in which part of this report each contribution is detailed; however, we find it clearer and more informative to present the organization of this report in a dedicated section. We organized this report in 6 chapters: Chapter 0, this one, is actually the first and it gives this work's context, its motivations and goals, summarizes the main contributions and presents how this report is organized. A detailed background and related works of time series classification in the absence and presence of uncertainty is given in Chapter 1. For readers who are not familiar with time series classification or with uncertainty, we strongly suggest reading this chapter before the following ones as subsequent chapters use concepts described in Chapter 1. Next comes Chapter 2 which describes our first main contribution UST. The second main contribution, SAST, is described in Chapter 3 as our proposed solution to improve UST's time complexity. Chapter 4 extends SAST to uncertain time series and details its performance on a real uncertain time series dataset. Finally, a general conclusion and some possible future directions of this work are given in Chapter 5.

Chapter 1

Background and related work

In this chapter, we give in-depth background required for understanding this work.

Contents

1.1 Background 7
1.1.1 What is a time series 7
1.1.2 Notion of uncertainty 8
1.1.3 Time series classification 12
1.2 Related work 14
1.2.1 Certain time series classification approaches 14
1.2.2 Uncertain time series classification approaches 24
1.3 Our proposed general approach for uTSC 28
1.4 Conclusion 28

1.1 Background

1.1.1 What is a time series

A time series is a type of data that allows the modeling of the evolution of a phenomenon through time. Differently said, a time series is used to see how an object changes with time. A time series is formally defined as follows:

Definition 1.1 (Time series). *A time series (TS) is a finite sequence of objects ordered in time.*

$$T = (t_{d_1}, t_{d_2}, ..., t_{d_m}), \forall j \in [1, m], d_j \in \mathbb{D}, t_{d_j} \in \Omega, m \in \mathbb{N}, m \geq 1 \tag{1.1}$$

In the previous definition, $\mathbb{D}$ is a totally ordered set and for any pair of integers j_1 and j_2 such that $j_1 < j_2$, we have $d_{j_1} < d_{j_2}$. The set Ω is the domain of the objects whose evolution in time is tracked. The objects in Ω are generally of the same nature, for instance, it could be a set of numbers, images, videos, audio, text, etc. Finally, m is the length of the time series. When Ω is an ordered set, the time series can be represented as a line plot on a 2 dimensional space where the x-axis is labeled by sorted objects from the set $\mathbb{D}$ and the y-axis is labeled by objects from

Ω. In order to better understand this general definition of time series, concrete examples are shown on Figures 1.1 and 1.2.

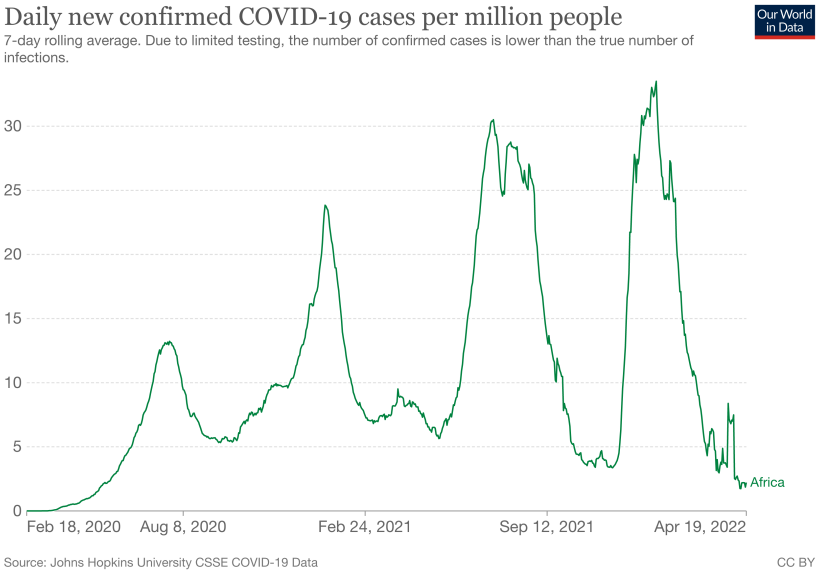


Figure 1.1: A time series modeling the the evolution of daily new covid-19 cases in Africa

Figure 1.1 is a time series of daily covid-19 cases in Africa from the 18th February 2020 to the 20th April 2022 as collected by the Our World in Data organization [Our World in Data 2022]. In this case the set $\mathbb{D}$ contains calendar days and the set Ω is the set of positive integers. Figure 1.2 is the time series of the groundwater level in the french department Puy-de-dôme collected by the France's public institution in earth science [BRGM]. In this case, the set $\mathbb{D}$ is still the calendar days, but the set Ω is the set of positive real numbers.

A time series could be either *univariate* or *multivariate*. It is univariate when only the evolution of a single variable is considered. This is the case for the time series on Figures 1.1 and 1.2. When the evolution of more than one variable is tracked simultaneously, the obtained time series is multivariate. A multivariate time series is shown on Figure 1.3, it models the successive positions in a 2 dimensional plan of my right ankle when I perform the Taekwondo round house kick. Here, two variables are tracked, the x-axis and the y-axis positions of my ankle.

In this thesis, we consider only the case where the set Ω is the set of real numbers. Therefore, we reduce the definition of time series to a finite set of numbers ordered in time.

1.1.2 Notion of uncertainty

Any measurement is subject to uncertainty, and unlike error which can be avoided by being careful, uncertainty cannot be avoided [Taylor 1996]. It can be reduced

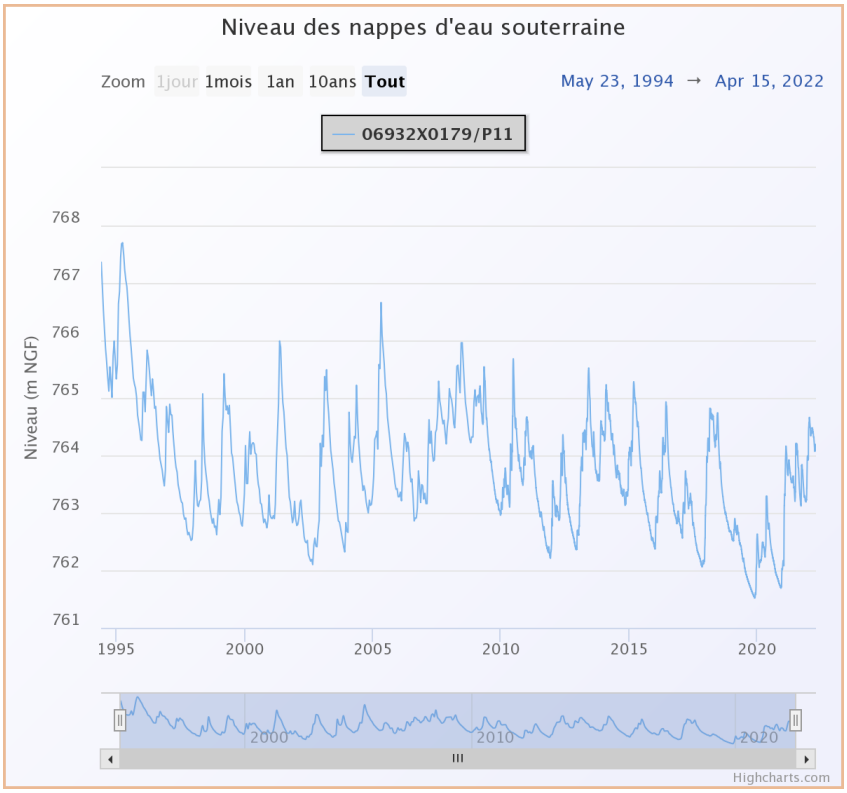


Figure 1.2: A time series modeling the groundwater level in the french department Puy-de-dôme

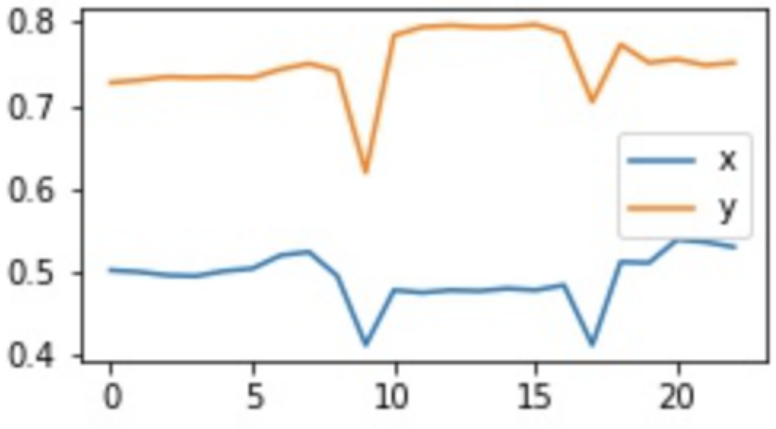


Figure 1.3: A multivariate time series modeling the successive positions of my right ankle when performing the Taekwondo roundhouse kick.

to a certain level but it cannot be eliminated. Many factors can lead to uncertain measurements including the sensitivity and the precision of the sensor used to make the measurement, the environmental conditions in which the measurement is done and the privacy preservation constraints. To make this clear, let's assume we want to know the height of a person who is 500 meters far away, we could estimate that he measures between 150 centimeters and 160 centimeters with some level of confidence given our experience. The uncertainty here is due to the fact that the person is far away, and hence, it is difficult to make a more precise estimation of its height. This uncertainty can be reduced by getting closer to the person. A more precise height can be obtained using a meter, but the precision of this measurement will still be limited by the graduation of the meter. A meter graduated in centimeter will give less precise measurements than a meter graduated in millimeter. It is difficult and even impossible in some applications to obtain the required level of certainty. Therefore, it is necessary to build ML tools that could work well despite the uncertainty in the data.

There exists two types of uncertainty: aleatoric and epistemic. Also called statistical, aleatoric uncertainty is due to the unknowns that differ each time a measurement is made. A typical source of aleatoric uncertainty is the random seed used for random numbers generation in computer science. Epistemic uncertainty, which is also called systematic uncertainty is due to things that should be known in principle, but are not in practice. The uncertainty in the measurement of a person height as described in the previous paragraph is epistemic as it is possible to measure a person height with precision using an appropriate tool. Some subtypes of epistemic uncertainty are imprecision, incompletion and unreliability. Imprecision is when there are many possible outputs for the same measurement and the exact output is unknown. An example is to give an interval in which a person height is known to be instead of a crisp real number. Incompletion is when some measurement are unknown or missing. A typical example is missing values in datasets. Unreliability is when it is not guarantee at 100% if the output of a measurement is correct or not. Figure 1.4 summarizes the categorization of the different types of uncertainty that have just been described.

Besides, being either epistemic or aleatoric, uncertainty has other facets [Destercke 2022]. For instance, it can be either objective or subjective depending if it can be measured objectively or only depends on the agent that quantify it; if an entire population is concerned by the uncertainty, it said generic, otherwise it is singular; another aspect is the uncertainty reducibility or non-reducibility.

There are two ways to represent uncertainty in the literature of uncertain time series: the multiset and the probability density function (PDF) representations [Dallachiesa *et al.* 2012, Siyou Fotso *et al.* 2020].

Definition 1.2 (PDF-based uncertain value). *The probability density function representation of an uncertain value x is given as a probability density function with known mean and maximal deviation from that mean. The mean is the best estimate*

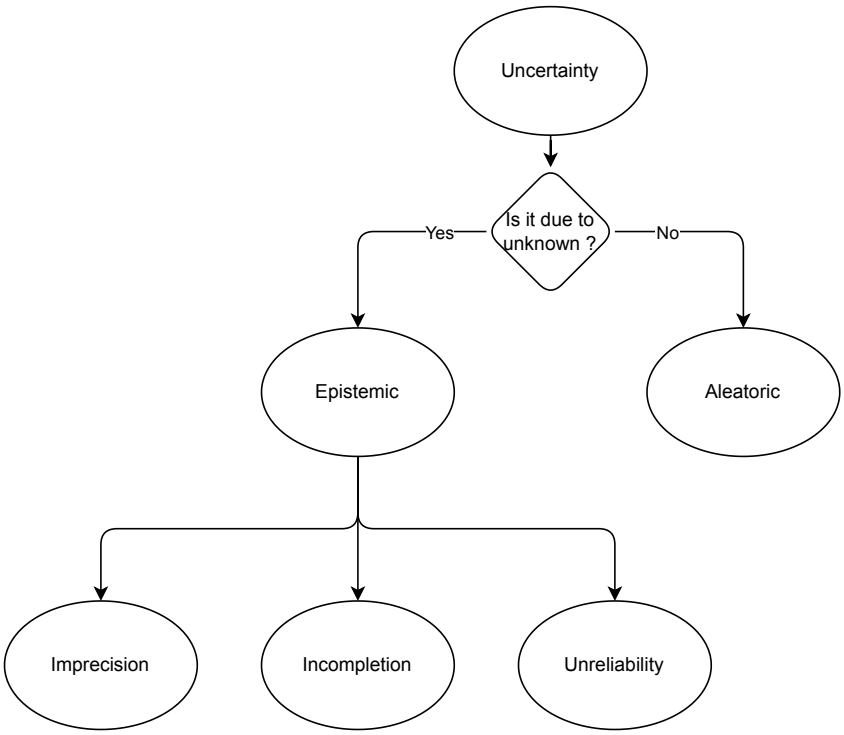


Figure 1.4: Categorization of uncertainty types

of the value and the deviation is the maximal possible error on that estimate.

$$x = \hat{x} \pm \delta x, \hat{x} \in \mathbb{R}, \delta x \in \mathbb{R}_+ \tag{1.2}$$

Therefore, the exact value is somewhere in the interval $[\hat{x} - \delta x, \hat{x} + \delta x]$.

Definition 1.3 (Multiset uncertain value). *An uncertain value can be given as a set of all the possible exact values. These values could be equiprobable or not.*

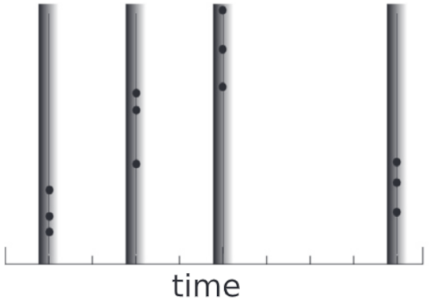
$$x = \{x_1, x_2, ..., x_s\}, \forall i, x_i \in \mathbb{R}, s \in \mathbb{N}_+ \tag{1.3}$$

The exact value is either one of the values in the set or a value close to one of the values in the set.

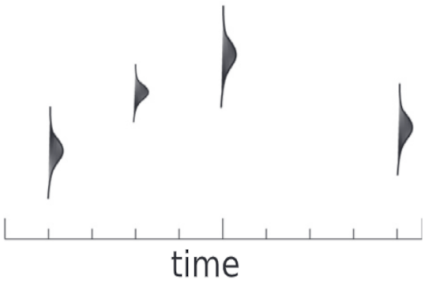
Now that we know what is an uncertain value, we can easily define what an uncertain time series is.

Definition 1.4 (Uncertain time series). *An uncertain time series (uTS) is a time series of uncertain values.*

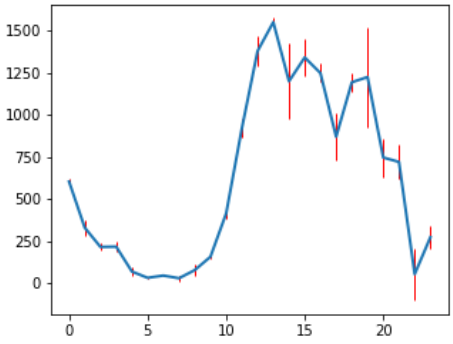
Figure 1.5 illustrates some uncertain time series. The first two figures are from [Siyou Fotso *et al.* 2020] and show a multiset uncertain time series (Figure 1.5a) and a probability density function uncertain time series (Figure 1.5b). The last two figures use the PDF representation, the blue lines are obtained by connecting the best estimates, these lines can be seen as the best estimate of the uncertain time series. The red bars are the possible deviation from the best estimates. Unlike Figure 1.5b, the probability distributions are unknown on the last two figures.



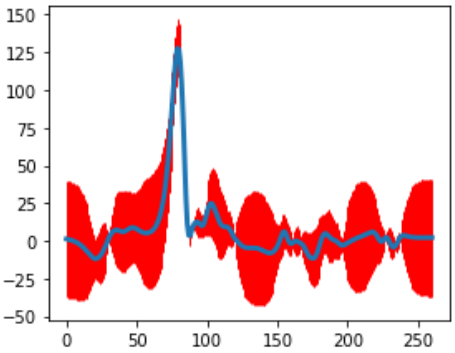
(a) A multiset uncertain time series



(b) A PDF uncertain time series



(c) A simulated uTS



(d) A uTS from the PLAsTiCC dataset [Allam Jr *et al.* 2018]

Figure 1.5: Illustration of uncertain time series

1.1.3 Time series classification

Time series analysis includes several tasks including the four classical machine learning tasks which are classification [Ye & Keogh 2009b, Bagnall *et al.* 2017], clustering[Ulanova *et al.* 2015, Siyou Fotso *et al.* 2020], regression [Tan *et al.* 2021], and dimensionality reduction [Lewandowski *et al.* 2010, Lin *et al.* 2007]. Others time series analysis tasks are forecasting [Hewamalage *et al.* 2022, Mbouopda 2022, Mbouopda *et al.* 2022], novelty detection [Ma & Perkins 2003], anomaly detection [Nakamura *et al.* 2020, Audibert *et al.* 2020], motif discovery [Yeh *et al.* 2018] and querying [Ding *et al.* 2008, Yagoubi *et al.* 2018]. These tasks can be classified as either supervised or unsupervised as shown on Figure 1.6.

This thesis focuses on the classification task of time series whose values are uncertain. The presence of uncertainty makes this task more complicated, since even for humans, it is not easy to take decisions when the possessed information are inaccurate. Because the existing methods for the classification of uncertain time series are inspired from *certain* time series classification methods, we present the later methods first.

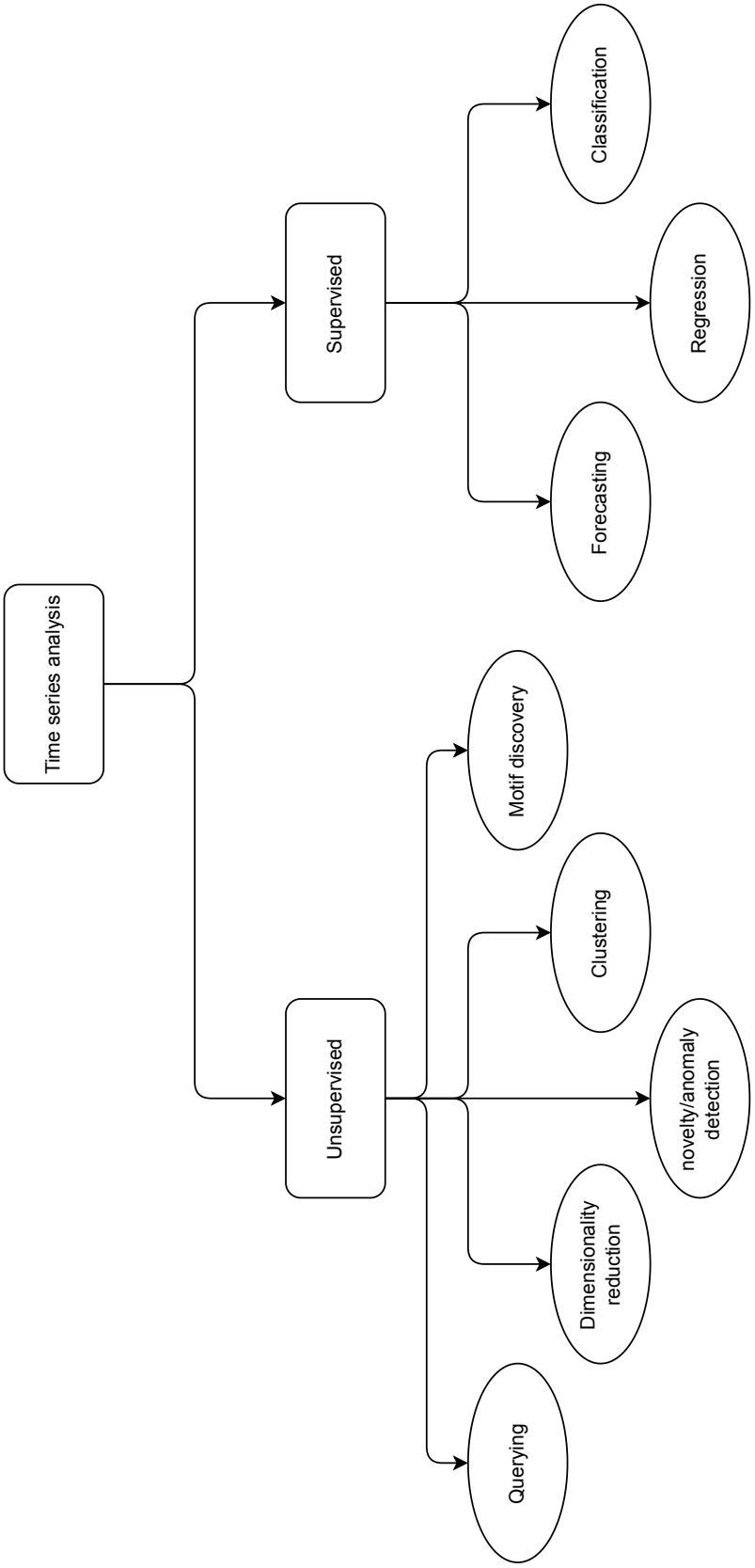


Figure 1.6: Time series analysis tasks hierarchy

1.2 Related work

1.2.1 Certain time series classification approaches

This subsection describes the existing approaches to perform time series classification when uncertainty is assumed to be absent or negligible: this is what we call `certain` or `regular` time series classification.

Definition 1.5 (Time series classification). *Let* $D = \{(T_1, c_1), (T_2, c_2), ..., (T_n, c_n)\}$ *be a dataset, where each* T_i *is a time series and* c_i *the associated class label taken from a finite set of discrete classes* $C = \{c_1, c_2, ..., c_{n_c}\}$ *(*n_c *is the number of classes and* $C \subset \mathbb{N}$*). The classification task for this dataset consists of learning a function* $\hat{f}$ *(also called a classifier) such that:*

$$\hat{f}(T_i) = c_i \tag{1.4}$$

Once the function is learned, it can be applied to new time series to automatically predict their class labels. In practice, learning the exact function $\hat{f}$ is generally a difficult problem depending on the complexity of the relationship between the time series and the classes. Instead, an approximated function f, close as possible to $\hat{f}$ is learned. The quality of the approximation is measured using a loss function l which computes how far is the approximation from the exact function. One of the most used loss functions is the categorical cross entropy defined as follows:

Definition 1.6 (Cross entropy). *The cross entropy loss of a classifier* f *on a given time series* T_i *is defined as follows:*

$$l(T_i, c_i) = c_i \log f(T_i) \tag{1.5}$$

The overall loss is obtained by averaging the losses for every time series in the dataset. Definitions 1.5 and 1.6 remain valid when the time series are uncertain.

Time series classification has been used in several practical applications including human activity recognition [Le Nguyen *et al.* 2019], phoneme detection [Hamooni & Mueen 2014], astronomical transients classification [Möller & de Boissière 2020, Moss 2018] and irrigation management [Zhao *et al.* 2016]. Diverse approaches has been developed in the literature to perform time series classification. Regarding the discriminative features used to classify the data, the existing methods have been grouped in five categories which are: whole series, interval, shapelet, dictionary and spectral approaches[Bagnall *et al.* 2017]. In order to include the methods proposed subsequently, we add three other categories, namely: subsequence, hybrid and deep learning approaches.

Whole series

Whole series approaches classify time series using the k-Nearest Neighbor (k-NN) classifier. In particular, a new time series is affected to the class of its first nearest

neighbor. The neighborhood is defined using a distance that measures the dissimilarity between two times series. The Euclidean distance is one of the most used distances in machine learning, and time series classification is not an exception.

Definition 1.7 (Euclidean distance (ED)). *Given two times series T_1 and T_2 of same length m, the Euclidean distance between them is defined as follows:*

$$ED(T_1, T_2) = \sqrt{\sum_{i=1}^{m}(t_{1i} - t_{2i})^2} \tag{1.6}$$

where t_{1i} and t_{2i} are the respective values of T_1 and T_2.

In practice, the square root can be omitted to save computation since its effect is only to change the dissimilarity scale. The main limitation of ED appears when the compared time series are not aligned, for instance when there is a time shift. To illustrate this situation, let us consider the the time series $T_1 = (0, 0, 0, 2, 2, 2, 0, 0, 0, 0)$ and $T_2 = (0, 0, 0, 0, 2, 2, 2, 0, 0, 0)$. It can be seen that T_2 is obtained by shifting T_1 one time step to the right. However the ED between them is not 0, meaning that this time series are not similar. The reason behind this behavior is that ED is computed assuming that the time series are naturally aligned as depicted in Figure 1.7. Time series are rarely perfectly aligned in practice, and particularly for long time series.

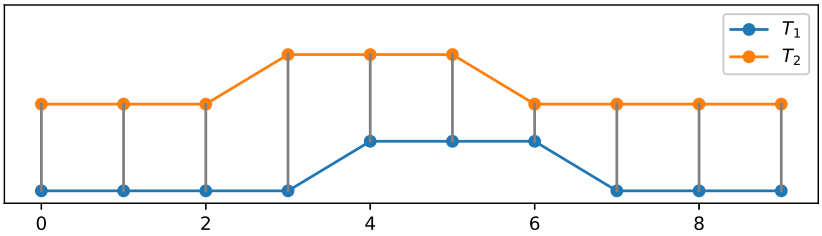


Figure 1.7: Natural alignment of both time series. $ED(T_1, T_2) = 2.82$

To handle situations where the time series does not follow a natural alignment, researchers has proposed to minimize the Euclidean distance by finding the optimal alignment. The resulted distance is called **elastic distances** in the state-of-the-art. Dynamic time warping (DTW) is one of the most used elastic distance in time series analysis. DTW has been proposed in the context of speech recognition, it stretches and realign one speech signal to perfectly match another one [Sakoe & Chiba 1978]. The optimal alignment found using DTW is illustrated on Figure 1.8. The Euclidean distance computed with respect to this alignment is 0, meaning that these two time series are exactly the same.

Combining *DTW* and the 1-NN has been the state-of-the-art approach to classify time series [Bagnall *et al.* 2017] for many years. Different variants of DTW has also been proposed, in particular the constrained DTW which accounts for

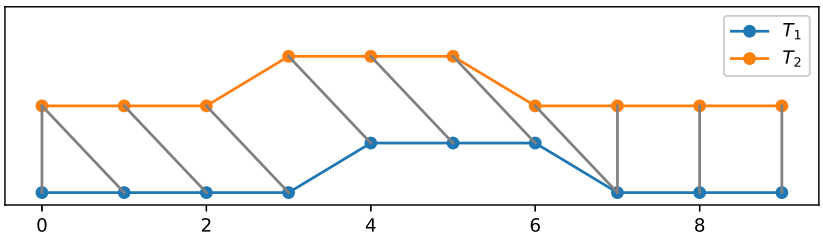


Figure 1.8: Elastic alignment of both time series. $DTW(T_1, T_2) = 0$

a parameter called the warping window in order to reduce the computation time complexity [Sakoe & Chiba 1978, Ratanamahatana & Keogh 2004]. Beside DTW, there exists other elastic distances that has been used in time series analysis, but DTW performs better. Unlike ED which has a linear time complexity, elastic distances generally have a quadratic time complexity as the optimal alignment is obtained using dynamic programming. Some pruning and early abandon techniques has been proposed to accelerate this computation [Herrmann & Webb 2021]. Given the success of ensemble techniques in machine learning, and the fact that different distances computes different dissimilarities, whole series approaches has been combined in order to improve the classification. Elastic Ensemble (`EE`), proposed in [Lines *et al.* 2018] is one of these ensemble methods. EE's time complexity has been significantly reduced in [Tan *et al.* 2020] to obtain the Fast Elastic Ensemble (`FastEE`).

Interval

Whole series approaches assume that the whole time series is important to achieve the classification. It is said that these approaches perform classification regarding the global features of the time series. By doing so, the computational time is high. On the contrary, interval approaches hypothesize that only a limited portion of the time series are relevant for classification. For instance, consider the time series dataset on Figure 1.9. It can be realized that the classification could be done regarding only the subsequences that are inside the gray rectangles, the remaining parts of the time series do not bring more information as these parts are similar regardless of the time series' class. Interval approaches are suitable for these type of datasets.

An interval approach for time series classification works in tree steps: intervals selection, features computation, and finally, classification. Relevant intervals identification is generally done randomly or using an heuristic. The next step after the selection of relevant intervals is the computation of a set of statistics on these intervals. In particular the mean, the standard deviation, the slope, the median, etc are computed for each interval and for each time series in the dataset. The last step is the actual classification using any supervised classifier on the computed statistics. This process in summarized on Figure 1.10.

State-of-the-art methods that follow the interval based approach are the time

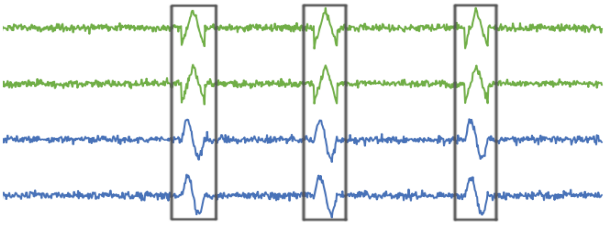


Figure 1.9: Relevant intervals in a dataset (image obtained by annotating an original image from [Lines *et al.* 2018])

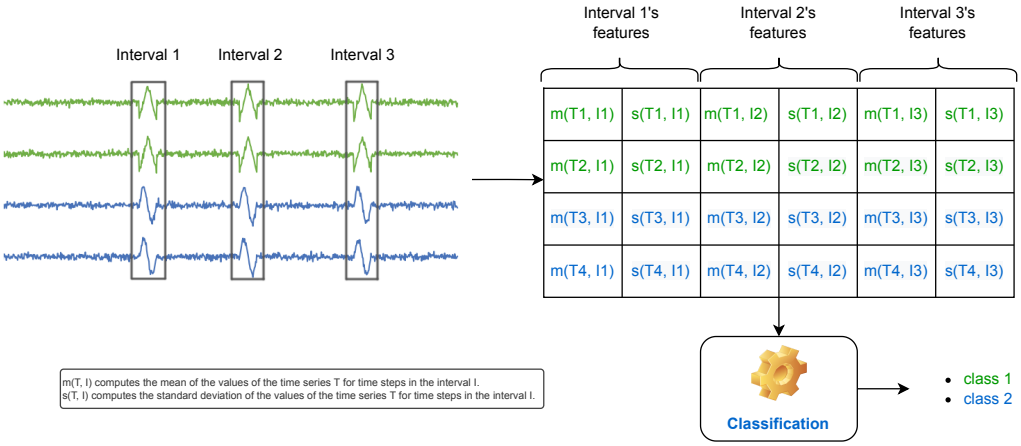


Figure 1.10: Overview of time series classification based on intervals

series forest (`TFS`) [Deng *et al.* 2013]. As `TSF` computes basic statistic, the method `CIF` has been proposed [Middlehurst *et al.* 2020a]. `CIF` uses 22 statistics specifically designed for time series and named as "catch22" [Lubba *et al.* 2019]. The supervised time series forest (`STSF`) selects the relevant intervals using a supervised approach in order to improve the classification performance [Cabello *et al.* 2020].

Shapelet

A Shapelets is a time series primitive proposed by [Ye & Keogh 2009b] as an effective and interpretable feature for time series classification. Simply said, a shapelet is subsequence (a set of consecutive values in a time series) that is characteristic of a class in a dataset. This notion is illustrated on Figure 1.11 on which the shapelets are the subsequences located in the rectangles. It can be seen that these subsequences are enough to perform the classification effectively as the green shapelets are only present in the green time series, while the blue shapelets are only present in the blue ones. Therefore, similarly to interval approaches, shapelet methods assume that only a subpart of the time series are relevant for the classification. However, unlike intervals whose locations are fixed once for all, shapelets can appear at any location on the time series: shapelet are *phase-independent*, while interval are *phase-*

dependent.

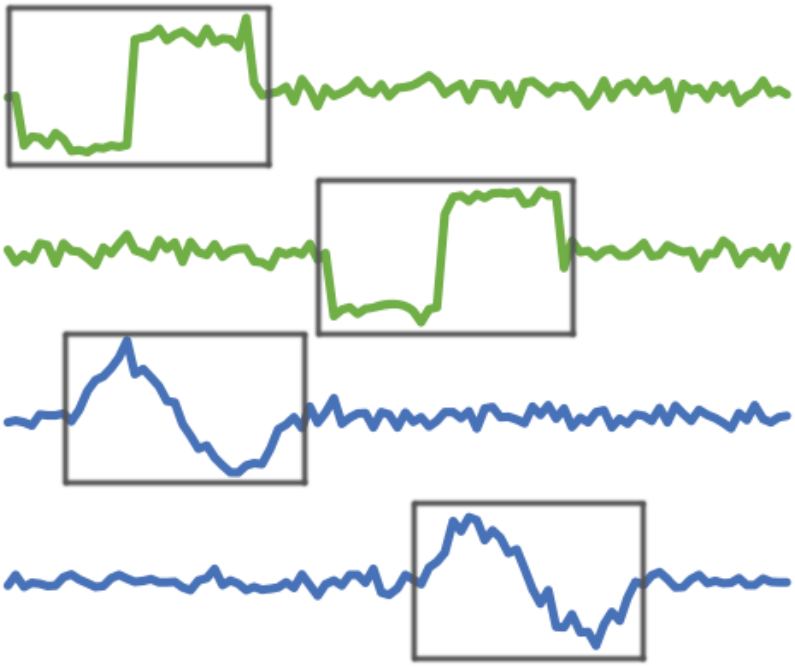


Figure 1.11: Illustration of shapelets (image obtained by annotating an original image from [Lines *et al.* 2018])

The classification of a time series using a shapelet approach is done with respect to the similarity between the time series and the shapelets. More specifically, if a time series contains a subsequence that is similar to a shapelet, then it is consider as being from the same class as the shapelet. The similarity is computed using a distance function, generally the Euclidean distance, but any distance can be also used. A shapelet approach performs classification in three steps: shapelets extraction, shapelets transformation, and finally, the actual classification. Extracting shapelets consists in finding the top most relevant shapelets from the training set. This is done by generating all the subsequences from the dataset, then computing the information gain obtained by splitting the dataset with respect to each subsequence (the split is done by creating two groups: one containing time series that are the most similar to the subsequence, and the other containing time series that are the less similar to the subsequence). Finally, the subsequences with the highest information gain are selected as shapelets. Then comes the second step which consists of transforming the original time series dataset to a tabular dataset. This is achieved by replacing each time series in the dataset by the vector of its distances to the selected shapelets. The final step is the training of any supervised classifier on the the transformed dataset. Figure 1.12 summarizes how time series classification with shapelets is performed.

This approach is said to be interpretable since the selected shapelet are mean-

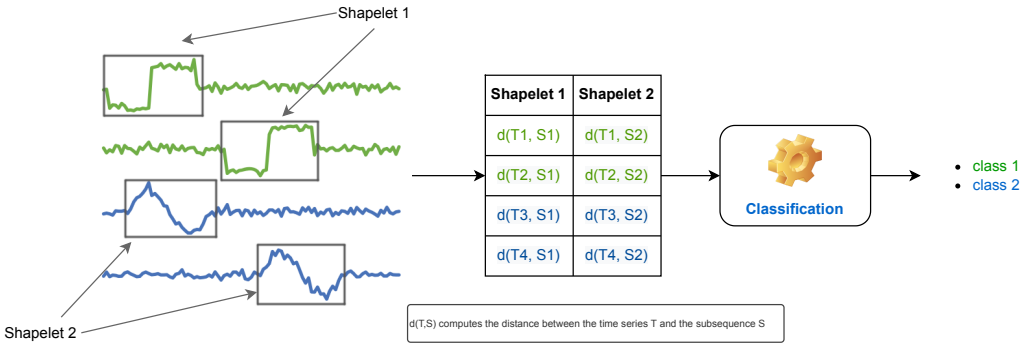


Figure 1.12: Overview of time series classification based on shapelets

ingful to domain expert: It is an explainable-by-design approach.

The first shapelet method is the Shapelet Decision Tree (`SDT`) [Ye & Keogh 2009b]. By discretizing the time series using SAX [Lin *et al.* 2007], the Fast Shapelet (`FS`) [Rakthanmanon & Keogh 2013] method has been proposed to reduce `SDT`'s time complexity from $O(n^2m^4)$ to $O(nm^2)$ while achieving a similar accuracy. Unlike `SDT` and `FS` which extract shapelets while performing the classification, Shapelet Transform Classifier (`STC`) isolated the extraction phase from the classification phase, allowing the usage of any supervised classifier [Hills *et al.* 2014]. Some authors proposed to improve the time complexity by reducing the space of shapelet candidates. This is achieved by evaluating a random subset of shapelet candidates [Renard *et al.* 2015, Wistuba *et al.* 2015]. Although these methods can find competitive shapelets, they are less accurate in general.

Instead of extracting shapelets from the training dataset, other methods start with a random set of shapelets, then modify them iteratively using optimization techniques such as gradient descent in order to minimize the final classification error. These methods are Learning time series Shapelet (`LS`) [Grabocka *et al.* 2014], `ELIS` and `ELIS++` [Zhang *et al.* 2021], and `XCM` [Wang *et al.* 2020]. These approaches are generally faster, but it is not guaranteed that the learned shapelets will be meaningful to domain experts.

Subsequence

Recently, a new approach able to learn shapelets has been proposed, this approach is XEM [Fauvel *et al.* 2022]. Instead of trying to find the most important subsequences before hand of classification, XEM uses every subsequences from the dataset then let the classifier learns the most discriminative ones during its training process. The used classifier is the Local Cascade Ensemble (LCE), a tree-based classifier that combines bagging (like in Random Forest) and Boosting (like in eXtreme Gradient Boosting trees) in order to reduce variance and bias. Since there is no guarantee that the most important subsequences learned by XEM will be shapelets, we find it justified to put this method in a dedicated category. Similar to shapelet-based

methods, XEM is explainable-by-design by extracting the most important region of the time series.

Dictionary

Shapelet, interval and whole series approaches work in the time domain as they perform classification based directly on the observed values of the time series. In some cases, this could be ineffective. The dataset shown on Figure 1.13 illustrates a situation where these approaches may not be suitable. This is because the time series are made of similar subsequences with different frequencies depending on the time series class.

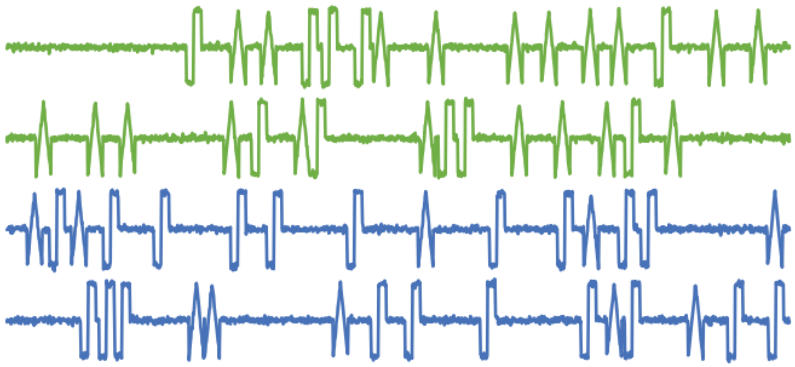


Figure 1.13: Illustration of a dictionary dataset [Lines *et al.* 2018]

This kind of dataset are generally classified in tree step: discretization, features extraction and finally classification. Discretizing a time series consists of converting it to a string or word using a finite set of symbols. The obtained string is an approximation of the original time series and is generally of a smaller length, therefore, leads to a kind of dimensionality reduction and accelerate computation. The most used discretization techniques in time series classification are the Symbolic Fourier Approximation (SFA [Schäfer & Högqvist 2012]) and the Symbolic Aggregate approXimation (SAX [Lin *et al.* 2007]). We illustrate the discretization using the example of SAX on Figure 1.14.

The next step after discretization is feature extraction from the obtained discretized time series. The Bag of SFA Symbol (`BOSS` [Schäfer 2015]) and the Word ExtrAction for time SEries cLassication (`WEASEL` [Schäfer & Leser 2017a]) methods use histograms as features. The last step is the actual classification, where any supervised classifier can be used. Additionally to `BOSS` and `WEASEL`, other dictionary methods are `MUSE` [Schäfer & Leser 2017b] which extends `WEASEL` to multivariate time series classification, the Temporal Dictionary Ensemble (`TDE` [Middlehurst *et al.* 2020b]) which uses ensemble techniques in order to improve the classification performance, and `TEASER` [Schäfer & Leser 2020] which classifies a time series as soon as possible given its very first values.

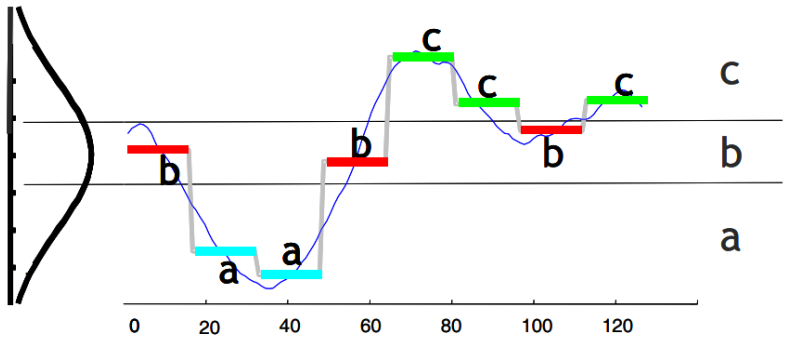


Figure 1.14: SAX illustration [Lin *et al.* 2007]. The string representing the time series is **baabccbc**.

Spectral

Datasets like the one shown on Figure 1.15 cannot be classified using feature extracted in the time domain as the time series look too similar regardless of the class. This kind of datasets are preponderant in the field of speech analysis.

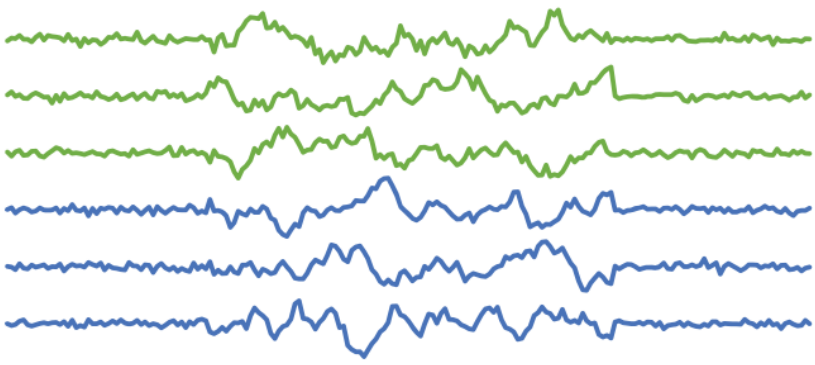


Figure 1.15: Illustration of spectral dataset [Lines *et al.* 2018]

For these datasets, going from the time domain to the frequency domain generally leads to a more effective classification [Bagnall *et al.* 2012]. The idea is to extract features from the high frequencies of the time series using Fourier transform, then classify time series based on their respective Fourier coefficients using any supervised classifier. Besides, Fourier coefficients, other transforms are autocorrelation (including partial autocorrelation), power spectrum and autoregressive features. The effectiveness of using Fourier coefficients, autocorrelation and power spectrum has been demonstrated in [Bagnall *et al.* 2012]. [Corduas & Piccolo 2008] used different supervised classifiers on autoregressive features in order to perform time series clustering and classification.

Deep learning approaches

Whole series, shapelet, interval, dictionary and spectral approaches perform classification using handcrafted features. Deep learning has been proved in diverse domains (image, speech, video, text analysis) to be able to automatically extract not only handcrafted features, but also relevant features that were unexpected by humans from the data. For this reason, some authors explored deep learning effectiveness in time series classification. In 2017, three different deep learning architecture has been applied on 44 time series datasets from the UCR & UEA [Anthony Bagnall & Keogh 2018] and the obtained results has been compared to the state-of-the-art time series classification method at that time [Wang *et al.* 2017]. The compared architectures were the multi-layer perceptron (`MLP`), the fully connected convolutional neural network (`FCN`) and the residual neural network (`ResNet`). `FCN` achieved the best performance, followed by `ResNet`. This work shown that deep learning is effective for time series classification, with the great advantage that there is no need to handcraft features.

With the development of new time series classification methods, a new benchmark of deep learning methods for time series classification has been proposed by [Fawaz *et al.* 2019b]. This study includes recent advances in time series classification, but also many methods that were not considered in [Wang *et al.* 2017]. Furthermore, with the availability of more open time series datasets, this review also considered more diverse application domains. This time, `ResNet` significantly outperformed the others deep learning methods, followed by the `FCN`. Additionally, the state-of-the-art handcrafted features method at this time, `HIVE-COTE` [Lines *et al.* 2018], achieved better classification performance compared to `ResNet`. `HIVE-COTE` will be described in the next section. Subsequently, the deep learning method `InceptionTime` has been proposed [Fawaz *et al.* 2020]. It is an ensemble of convolutional neural network models that include residual connections as in `ResNet`.

The state-of-the-art deep learning method for time series classification is `ROCKET` [Dempster *et al.* 2020]. It is a special kind of convolutional neural network as it uses random filters. In fact, instead of learning convolutional kernels by optimization as it is generally done in deep learning, `ROCKET` uses random kernels sampled from a uniform distribution. As there is no learning, this method is extremely fast. Additionally, `ROCKET` uses a new feature called the proportion of positive values (PPV) computed after applying the convolutions. PPV allows `ROCKET` to achieve comparable performance to state-of-the-art methods. An almost deterministic variant of `ROCKET` has been developed recently under the name `MiniRocket` [Dempster *et al.* 2021]. Another variant of `ROCKET` is `MultiROKET` which extract features from the first order derivative of the raw time series in addition to features extracted on the raw time series [Tan *et al.* 2022]. In addition to PPV, `MultiROCKET` considers the Mean of Positive Values (MPV), the Mean of Indices of Positive Values (MIPV) and the Longest Stretch of Positive Values (LSPV).

Hybrid approaches

Some time series datasets simultaneously contains whole series, shapelet, interval, dictionary and spectral features. In this situation, using a single type of feature to perform the classification leads to poor performance. Moreover, it would be beneficial to have a method than could perform well on any dataset. Hybrid approaches has been proposed to tackle these challenges. The main idea is to extract every type of features, then use a strategy to merge the predictions from each feature type to obtain the final prediction. For instance, the Random Interval Spectral Ensemble (`RISE` [Lines *et al.* 2018, Flynn *et al.* 2019]) combines features from the frequency and interval domains, and uses ensemble techniques to reduce the variance. `RISE` is significantly more accurate than any other spectral method.

The Hierarchical Vote Collective of Transformation-Based Ensembles (`HIVE-COTE`) [Lines *et al.* 2018], inspired from the collective of transformation-based ensembles (`COTE`) is the state-of-the-art method for time series classification. It combined 35 time series classifiers grouped in 5 modules: 1 shapelet module, 1 whole series module, 1 dictionary module, 1 interval module and 1 spectral module. Each module is made of ensemble classifiers. The final classification is obtained by a majority vote. The second version of `HIVE-COTE` has been developed exploiting the recent advances in time series classification [Middlehurst *et al.* 2021]. This version is made of four modules of ensemble classifiers whose base learners are: `STC`, `TDE`, ROCKET and `CIF`.

The most important limitation of `HIVE-COTE` is the computational time which is in the order of $O(n^2m^4)$ for a dataset of n time series of length m. This limitation is solved in practice by searching shapelet for a given amount of time only. Although this works well in practice, there is no guarantee that some interesting shapelets will not be missed by the algorithm. The Time Series Combination of Heterogeneous and Integrated Embedding Forest (`TS-CHIEF` [Shifaz *et al.* 2020]) combines whole series, interval and dictionary features in a forest of trees fashion in order to reduce the variance, increase the classification performance, while reducing the computational time. The authors did not include shapelet features because they are computationally expensive. `TS-CHIEF` achieves performance comparable to the first `HIVE-COTE` version while being much more scalable. However, the second `HIVE-COTE` version is significantly more accurate than `TS-CHIEF`.

Summary of time series classification approaches

It can be realized by looking carefully at the existing time series classification approaches that they follow the same pattern to perform classification. In fact, each approach works in three main steps: features extraction, feature transformation, and finally classification. This process is summarized in Figure 1.16 and is described as follow: Given a dataset of time series with their class labels, a feature extractor is used to extract relevant features (shapelets, intervals, words, artificial neural network (ANN) weighs, etc), then the input dataset is transform with respect to the

extracted features (shapelet transformation, interval transformation, histogram, forward pass for ANN, etc), and finally, a supervised classifier is trained on the obtained tabular dataset. The feature transformation is performed using the function $v(T, f)$ which computes the values of the feature f for the the time series T. This function is the distance function in shapelet and whole series approaches, it is the statistic functions (mean, standard deviation, median, etc) for interval approaches, it is the count for dictionary approaches, it is the correlation for spectral approaches, and it is the forward pass for deep learning approaches.

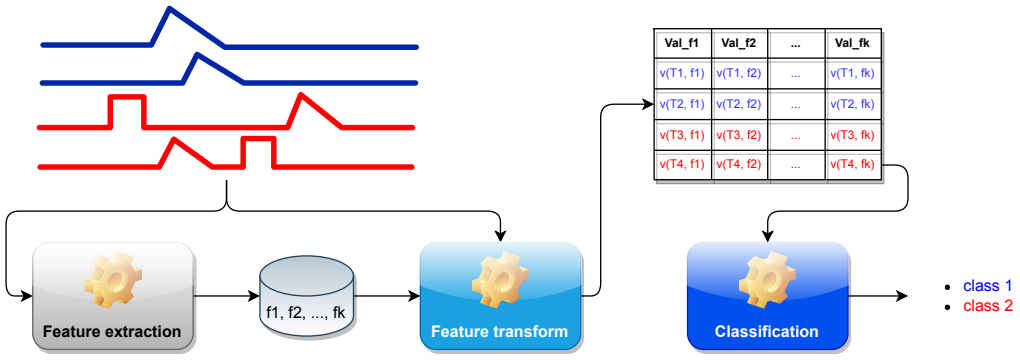


Figure 1.16: Overview of existing time classification approaches

Whole series approaches are particular as they do not perform feature extraction. However, they can also be designed following the general process depicted in Figure 1.16. Concretely, it can be considered that the feature extraction process returns the whole dataset (in other words, the identity function is applied), then the feature transformation replaces each time series with the vector of its distances to the remaining time series. Finally, the classification is performed.

We synthesized the existing methods for time series classification in Table 1.1. The first two columns respectively indicate the category and the features that are used for classification. The last column gives some state-of-the-art methods in the corresponding category. The third column indicates the type of explainability, which is either *absent* when the method is not explainable, *by design* when the method is explainable right after training without using any additional tool, and *post hoc* when the method is explainable after the training by using additional techniques such as `LIME` [Ribeiro *et al.* 2016] and `SHAP` [Lundberg & Lee 2017]. Note that although post hoc explainability methods such as `LIME` and `SHAP` are model-agnostic, and thus are applicable to any method, there is no theoretical guarantee that the result will be meaningful to the end-users.

1.2.2 Uncertain time series classification approaches

Existing uncertain time series classification (uTSC) approaches are inspired from regular time series classification (TSC) approaches. The general idea is to take uncertainty into account in an existing TSC approach by the means of some adap-

Table 1.1: Summary of the current time series classification state-of-the-art (SOTA)

Category	Features	Explainability	SOTA methods
Whole series	time series	absent	1NN-DTW EE FastEE
Subsequence-based	subsequences	by design	XEM
Shapelet-based	shapelets	by design	LS STC ELIS++
Interval-based	intervals	post hoc	TSF STSF CIF
Dictionary	words	post hoc	BOSS WEASEL TDE
Deep Learning	Artificial neural network weights	post hoc	InceptionTime ResNet
Hybrid	combination of some or all of the above features	post hoc	RISE HIVE-COTE TS-CHIEF

tations. Despite the fact that any measure is uncertain (either for epistemic, or for aleatoric reasons), researchers have not focused on uTSC as much as they did for TSC.

The few methods that exist for uncertain time series classification work in the same way: combining an uncertain similarity measure with the 1-Nearest Neighbor (1-NN) classifier. The development has been therefore focused on the building of similarity measures for uncertain time series.

The uncertain similarity measure MUNICH has been proposed by [Aßfalg *et al.* 2009] and can be used when the uncertainty is represented by a set of possible observations at each time step. It is used to compute the probability that the similarity between two uncertain time series is below a user-defined threshold. Therefore, MUNICH does not actually compute the uncertain similarity and has never been used in a classification context. The uncertain similarity measure PROUD [Yeh *et al.* 2009] also computes the probability of the similarity being below a threshold, but unlike MUNICH, uses PDF representation of uncertain time series.

The uncertain similarity distance DUST [Sarangi & Murthy 2010] has been been proposed as a generalized notion of distance between uncertain time series to overcome the limitation of PROUD and MUNICH. It makes fewer assumptions on the uncertainty compared to PROUD, is computationally less expensive compared to

MUNICH and degenerates to the Euclidean distance when the uncertainty is very small. Moreover, it does not compute a probability on the similarity, but the similarity itself. DUST assumes that the imprecision in the time series follows a normal distribution. N_DUST is a DUST variant that assumes the best estimate to be normally distributed while U_DUST considers uniform distribution.

Definition 1.8 (DUST). *The DUST similarity is defined as follows:*

$$U\_DUST(T_1, T_2) = \sqrt{\sum_{1}^{l} \left(\frac{\hat{t_{1,i}} - \hat{t_{2,i}}}{2\sigma_i} \right)^2} \tag{1.7}$$

$$N\_DUST(T_1, T_2) = \sqrt{\sum_{1}^{l} \left(\frac{\hat{t_{1,i}} - \hat{t_{2,i}}}{2\sigma_i(1+\sigma_i^2)} \right)^2} \tag{1.8}$$

where σ_i is the uncertainty at time step i.

Since DUST requires the compared values to have the same uncertainty, we consider $\sigma_i = \max(\delta t_{1,i}, \delta t_{2,i})$ in this work.

About 10 years after DUST, the uncertain similarity measure FOTS has been proposed [Siyou Fotso *et al.* 2020]. Compared to DUST, MUNICH and PROUD, FOTS does not explicitly model the uncertainty, but assumes the time series are noisy and there is no information nor assumptions made on that noise. FOTS uses Eigenvalues decomposition to keep only the most important components of the uTS and to reduce the noise.

Definition 1.9 (FOTS). *The FOTS similarity is defined as follows:*

$$FOTS(T_1, T_2) = \sqrt{\sum_{i=1}^{l} \sum_{j=1}^{k} (U_1 - U_2)_{ij}^2} \tag{1.9}$$

where U_1 and U_2 are the k first Eigenvector matrices of the local auto-covariance matrices of T_1 and T_2 respectively.

*The **local auto-covariance** at timestamp t of a time series T is computed using M sliding windows of size w as follows:*

$$\Gamma_t(T, w, M) = \sum_{\tau=t-M+1}^{t} T_{\tau,w} \otimes T_{\tau,w} \tag{1.10}$$

where $T_{\tau,w}$ is the subsequence of length w from T and which starts at timestamp τ. $\otimes$ is the outer product operator.

DUST, MUNICH, PROUD have been compared against each other on some time series analysis tasks such as querying and classification [Dallachiesa *et al.* 2012]. In the same work, the authors proposed the uncertain moving average (UMA), a moving average strategy for uncertain time series.

We have noted four main limitations in the existing works on uncertain time series:

1. First, existing similarity measures give the similarity between two uncertain time series as an *exact* similarity without any uncertainty information. Since the compared objects are uncertain, it is natural to expect the similarity between them to have some uncertainty. Elsewhere, since the existing measures do not give the uncertainty on the similarity, the classifier cannot be aware that the input data are uncertain. Likewise, the result of applying UMA is provided without the associated uncertainty,

2. second, except FOTS which has been proved to be more robust for clustering uncertain time series than Euclidean distance, the other uncertain measures have not been proved to be more effective compared to any exact distance that ignores uncertainty. This opens a question: are the existing uncertain similarity measures really necessary/useful?

3. third, the existing uncertain similarity measures have never been applied on real uncertain time series datasets, but always on exact time series on which random uncertainty have been added,

4. Finally, the synthetic uncertain time series data on which the existing methods have been tested are never provided, making it difficult to reproduce or verify the published results.

The Uncertain Euclidean Distance (UED) that we are proposing in Section 2.2 is a solution for the first aforementioned limitations. This thesis mitigates the second limitation by giving a comparison of DUST, which is to our knowledge the state-of-the-art uncertain measure, to Euclidean distance. In addition, we also consider FOTS and UED in this comparison. We apply these measures on synthetic datasets, but also on PLAsTiCC, an astronomical dataset of time series with real uncertainty [Allam Jr *et al.* 2018]. Finally, we share the source code, datasets and results of our experiment on a public repository to encourage reproducibility and re-usability.

As mentioned earlier, the existing methods are based on the combination of a similarity measure with a 1-NN classifier and it has been shown that this approach is significantly less effective than approaches that extract local and/or global features on which classification is then performed [Bagnall *et al.* 2017]. The UST and uSAST methods, which we respectively propose in Chapter 2 and Chapter 4 are based on shapelet features which are known to be very competitive for *exact* time series classification when tested on the UCR archive [Dau *et al.* 2019]. Beside giving accurate classification, shapelet features provide an inherent and easy explanation of the predictions [Hills *et al.* 2014].

Figure 1.17 summarizes existing uncertain time series classification methods. Because the existing uncertain similarity measures do not provide the similarity uncertainty, the brutal disappearance of the uncertainty at the very beginning of the classification process can be observed, misleading the remaining part of the process to wrongly believes that the input data is certain.

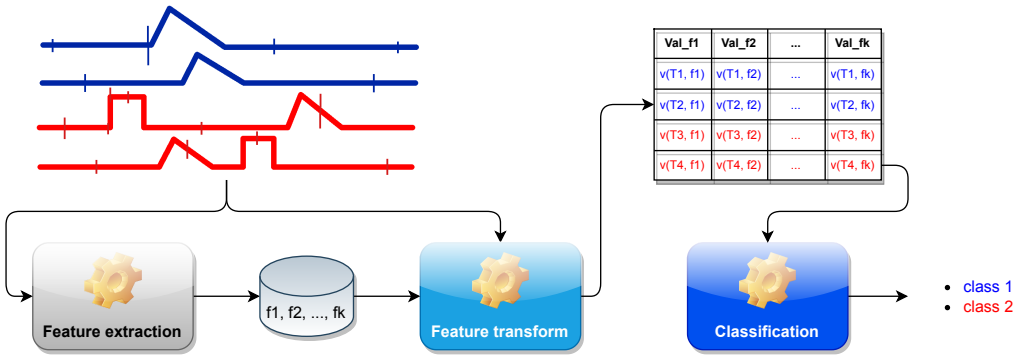


Figure 1.17: Overview of existing uncertain time classification approaches

1.3 Our proposed general approach for uTSC

Given the limitations that we noticed in the state-of-the-art, we argue that uncertainty should be handled throughout the whole classification process. Doing so will increase the reliability of uTSC methods, but also its usability by non expert users. Our proposal follows the same steps as existing uTSC approaches, but with the difference that the extracted features are accompanied with their respective uncertainties. Similarly, the dataset obtained after the transformation step also have associated uncertainty, and finally, the classification is performed while taking uncertainty into account. Hence, the uncertainty on the predicted labels can also be quantified. This process is summarized in Figure 1.18.

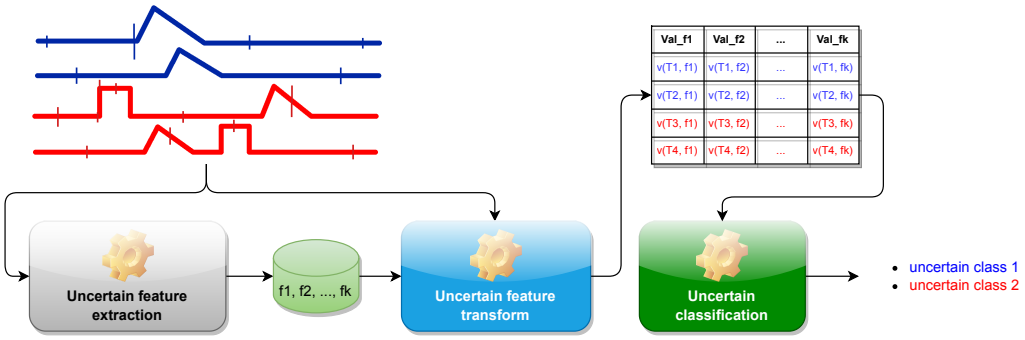


Figure 1.18: Overview of the new proposed uncertain time classification approach

1.4 Conclusion

In this chapter we defined the background notions that are going to be used throughout all this work. In addition, we presented a detailed review of the existing methods for certain and uncertain time series classification. We identified the scientific locks in the existing methods, and proposed a novel design of uncertain time series classification which takes uncertainty into account from the beginning to the end of the

process.

In the next chapter, we follow our proposed design by propagating uncertainty in the euclidean distance in order to obtain the uncertain euclidean distance (UED) which allows us not only to have the similarity, but also the quantification of the uncertainty on that similarity. Subsequently, UED is used to perform uTSC classification efficiently.

Chapter 2

Uncertain Time Series Classification With Shapelet Transform

Time series classification is a task that aims at classifying chronological data. It is used in a diverse range of domains such as meteorology, medicine and physics. In the last decade, many algorithms have been built to perform this task with very appreciable accuracy. However, applications where time series have uncertainty has been under-explored. Using uncertainty propagation techniques, we propose a new uncertain dissimilarity measure based on Euclidean distance. We then propose the uncertain shapelet transform algorithm for the classification of uncertain time series. The experiments we conducted on state-of-the-art datasets show the effectiveness of our contribution.

Contents

2.1 Introduction 31
2.2 UED: a new uncertain dissimilarity measure 32
2.3 UST: The uncertain shapelet transform classification 35
2.3.1 Definition of concepts 35
2.3.2 Uncertain shapelet transform classification 37
2.3.3 UST time complexity 39
2.4 Experiments 40
2.4.1 Compared models 40
2.4.2 Datasets 41
2.4.3 Results 42
2.5 Conclusion 45

2.1 Introduction

The last decade has been characterized by the availability of measurements in a large and variate set of domains such as meteorology, astronomy, medicine and object tracking. Generally, these measurements are represented as time series

[Dallachiesa *et al.* 2012], that means a sequence of data ordered in time. Time series classification is used in many applications such as astronomy, land cover classification and human activity recognition. Meanwhile, there has been an increase of the number of methods for time series classification [Shifaz *et al.* 2020, Bagnall *et al.* 2017]. However, to the best of our knowledge, these methods do not take data uncertainty into account. Any measurement is subject to uncertainty that can be due to the environment, the mean of measurement, privacy constraints and other factors. Furthermore, even if uncertainty can be reduced, it cannot be eliminated [Taylor 1996]. In some applications, uncertainty cannot be neglected and has to be explicitly handled [Sarangi & Murthy 2010].

Shapelet based methods are one of the best approaches that have been developed for time series classification. A shapelet is a subseries that is representative for a class of time series. These methods are especially appreciated for their interpretability, their robustness and their classification speed [Ye & Keogh 2009b].

Almost every time series classification methods are built by coupling a similarity measure with a supervised classifier. We follow this pattern in this chapter to build an uncertain time series classifier. We are not aware of any existing method in the literature for the classification of uncertain time series.

Our contribution is as follows, we first propose an uncertain dissimilarity measure based on Euclidean distance. Secondly we use it to build the uncertain shapelet transform algorithm, which is the shapelet transform algorithm adapted to the classification of time series with available uncertainty information.

The rest of this chapter is organized as follows: In Section 2.2, we present a new uncertain dissimilarity measure called UED, and in Section 2.3, we present the uncertain shapelet transform algorithm (UST). Section 2.4 is about experiments and Section 2.5 finally concludes this chapter.

2.2 UED: a new uncertain dissimilarity measure

As stated by [Taylor 1996], uncertainty is different from error since it cannot be eliminated, but it can be reduced up to a certain magnitude. Regardless of the measurement method, there is always an uncertainty and uncertain measures cannot be compared with a 100% reliability: the result of the comparison of uncertain values should also be uncertain.

From now on, we consider only PDF-based representation of uncertain values. Let x be an uncertain value, we have $x = \hat{x} \pm \delta x$, the exact value of x follows a probability distribution and lies in the interval $[\hat{x} - \delta x, \hat{x} + \delta x]$. $\hat{x}$ is the best guess of the exact value of x. Let y be another uncertain value, any mathematical operator applied on x and y produces a new uncertain value. We have the following

uncertainty propagation properties [Taylor 1996]:

$$\begin{aligned} z = x + y &= (\hat{x} + \hat{y}) \pm (\delta x + \delta y) \\ z = x - y &= (\hat{x} - \hat{y}) \pm (\delta x + \delta y) \\ z = x \times y &= (\hat{x} \times \hat{y}) \pm (\frac{\delta x}{|\hat{x}|} + \frac{\delta y}{|\hat{y}|}) \times (|\hat{x} \times \hat{y}|) \\ z = \frac{x}{y} &= \frac{\hat{x}}{\hat{y}} \pm (\frac{\delta x}{|\hat{x}|} + \frac{\delta y}{|\hat{y}|}) \times \frac{|\hat{x}|}{|\hat{y}|} \end{aligned}$$

Euclidean distance (ED) is widely used in the literature to measure the dissimilarity between time series. It is particularly used in shapelet-based approaches [Ye & Keogh 2009b, Hills *et al.* 2014, Bagnall *et al.* 2017]. Using the uncertainty propagation properties, an uncertain dissimilarity measure based on ED can be computed for two uncertain time series T_1 and T_2 by propagating uncertainty in the ED formula. We name the obtained measure UED for Uncertain Euclidean Distance, and it is defined as follows:

$$UED(T_1, T_2) = \sum_{i=1}^{n} (\hat{t_{1i}} - \hat{t_{2i}})^2 \pm 2 \sum_{i=1}^{n} |\hat{t_{1i}} - \hat{t_{2i}}| \times (\delta t_{1i} + \delta t_{2i}) \tag{2.1}$$

where $\hat{T_i}$ is the time series of the best guesses of T_i.

The output of UED is an uncertain measure representing the similarity between the two uncertain time series given as inputs. In order to use UED to classify time series, especially with a shapelet algorithm, an ordering relation for the set of uncertain measures is needed. We propose three ways to compare uncertain measures: the first one is the simpler one and is based on confidence, the second one is a stochastic order and the last one is an interval number ordering.

Simple ordering for uncertain measures

This ordering is based on two simple properties. Let x and y be two PDF-based uncertain measures, the first property is the property of equality and states that two uncertain measures are equal if their best guesses and their uncertainties are equals.

$$x = y \iff \hat{x} = \hat{y} \wedge \delta x = \delta y \tag{2.2}$$

The property of inferiority is the second one and states that the uncertain measure x is smaller than the uncertain measure y if and only if the best guess of x is smaller than the best guess of y. In the case where x and y have the same best guesses, the smaller is the one with the smallest uncertainty.

$$x < y \iff (\hat{x} < \hat{y}) \vee ((\hat{x} = \hat{y}) \wedge (\delta x < \delta y)) \tag{2.3}$$

Unlike the property of equality which is straightforward, the property of inferiority need some explanations. Unfortunately, we don't have a mathematical justification of this property but it is guided by two points: firstly we are in some way confident

about the best guess since it must have been given by an expert, and secondly we are more confident with smaller uncertainties.

Of course, these properties do not always give a correct ordering; in fact, if $x = 2 \pm 0.5$ and $y = 2 \pm 0.1$ then the inferiority property says that $y < x$. Now, if there is an oracle able to compute the exact value of any uncertain measure, it might says that $x = 1.8$ and $y = 2$, and thus invalidating our ordering. This observation also holds for the properties of equality.

Stochastic ordering of uncertain measures

An uncertain measure can be considered as a random variable with mean equals to the best guess and standard deviation equals to the uncertainty. Given this consideration, a stochastic order can be defined on the set of uncertain measures. A random variable X is *stochastically less than or equal to* (noted $\leq_{st}$) another random variable Y if and only if $Pr[X > t] \leq Pr[Y > t]\ \forall t \in \mathbb{I}$, where $\mathbb{I}$ is the union of the domains of X and Y [Marshall *et al.* 2010]. The stochastic order can be rewritten and developed as follows:

$$\begin{aligned} X \leq_{st} Y &\iff Pr[X > t] \leq Pr[Y > t]\ \forall t \in \mathbb{I} \\ &\iff 1 - Pr[X > t] > 1 - P[Y > t]\ \forall t \in \mathbb{I} \\ &\iff Pr[X \leq t] > Pr[Y \leq t]\ \forall t \in \mathbb{I} \\ &\iff CDF_X(t) > CDF_Y(t)\ \forall t \in \mathbb{I} \end{aligned} \tag{2.4}$$

$CDF_X(t)$ is the cumulative distribution function of the random variable X evaluated at t. Because the cardinality of $\mathbb{I}$ is infinite, we discretized it as being the set of the following values:

$$\min(\mathbb{I}) + i \times \frac{\max(\mathbb{I}) - \min(\mathbb{I})}{k} \tag{2.5}$$
$$0 \leq i \leq k \text{ and } k \text{ is a whole number to be defined.}$$

Unlike the simple ordering which is a total order, the stochastic ordering is a partial order. That means, the relation *stochastically less than or equal to* is not defined for any two random variables as the condition may not hold for every t in $\mathbb{I}$, and thus, sorting some uncertain measures using the stochastic ordering is impossible. This is clearly a limitation, but we did not find a total stochastic ordering in the literature.

Interval numbers ordering

Definition 2.1 (Interval number). *An interval number i_n is a number represented as an interval, that is $i_n = [i_n^l, i_n^u]$, where i_n^l and i_n^u are respectively the lowest and highest possible values of the number.*

A PDF-based uncertain value is by definition an interval number, enhanced with a probability distribution in that interval. Given two uncertain values x and y, the

interval number-based ordering can be estimated using the following probability [Xu & Da 2002, Yue 2011]:

$$Pr[x \geq y] = \max(1 - \max(\frac{\hat{y} + \delta y - \hat{x} + \delta x}{2\delta x + 2\delta y})) \tag{2.6}$$

Unlike the stochastic ordering, the simple ordering and the interval number-based ordering do not exploit the uncertainty distribution, nor the best guess given by the expert. The simple ordering required the best guess to be known, and this is not always the case in practice. If the compared uncertain values do not overlap at all, that is they do not have some possible exact values in common, all the three ordering give the same order.

Now that we know how to sort uncertain measures, let us see how to use UED to classify uncertain time series.

2.3 UST: The uncertain shapelet transform classification

In this section, we describe how to classify uncertain time series using shapelets. Uncertain observations are represented using the probability density function model (or simply PDF model). We start by defining the concepts that are used in our algorithm, then we describe the algorithm itself.

2.3.1 Definition of concepts

Definition 2.2 (Uncertain subsequence). *An uncertain subsequence S of an uncertain time series T is a series of l consecutive uncertain values in T.*

$$\begin{aligned} S = \hat{S} \pm \delta S = (\hat{t}_{i+1} \pm \delta t_{i+1}, ..., \hat{t}_{i+l} \pm \delta t_{i+l}) \\ 1 \leq i \leq m - l,\ 1 \leq l \leq m, m = |T| \end{aligned} \tag{2.7}$$

Definition 2.3 (Distance). *The distance between a subsequence S of length l and a time series of length m is defined as follows:*

$$Dist(S, T) = \min_{P \in T^l} dist(S, P),$$

, where $T^l = \{(t_i, t_{i+1}, ..., t_{i+l}) |\ 1 \leq i \leq m - l + 1\}$

The $dist(\cdot, \cdot)$ function in Definition 2.3 could be any distance metric. In practice the Euclidean Distance (ED) and the Dynamic Time Warping (DTW) are generally used. The definition is also applicable between uTS and uncertain subsequence by ignoring the uncertainty or by taking it into account using an uncertain distance, namely the UED distance.

Let $D = \{(T_i, c_i) | 1 \leq i \leq n\}$ be a dataset of n time series T_i (respectively uncertain time series) with their class labels c_i taken from a discrete finite set C such that the cardinality of C is much less than n. We can define the notions of *separator* and *shapelet* for this dataset.

Definition 2.4 (Separator). *A separator (respectively uncertain separator) is a pair of a subsequence S (respectively uncertain subsequence) and a threshold ε that divide the dataset in two groups D_L and D_R such that:*

$$D_L = \{(T_i, c_i) | Dist(S, T_i) < \varepsilon, 1 \leq i \leq n\}$$
$$D_R = \{(T_i, c_i) | Dist(S, T_i) \geq \varepsilon, 1 \leq i \leq n\}$$

Definition 2.5 (Shapelet). *Given a dataset $D = \{(T_1, c_1), (T_2, c_2), ..., (T_n, c_n)\}$ of time series with their class labels c_i taken from a finite set of classes C, a shapelet $S^\star$ is a separator that maximizes the information gain.*

$$S^\star = \arg\max_{S \in W} IG(D, S), \text{ with } W \text{ being the set of all subsequences in } D. \quad (2.8)$$

Definition 2.6 (Information gain (IG)). *Let D be a time series dataset and S a shapelet. Let $D_L = \{T \in D \mid dist(T, S) \leq \varepsilon\}$ and $D_R = \{T \in D \mid dist(T, S) > \varepsilon\}$, then*

$$IG(D, S) = \max_{\varepsilon \in SP \subset \mathbb{R}} \left(H(D) - \frac{|D_L|}{|D|} H(D_L) - \frac{|D_R|}{|D|} H(D_R) \right), \text{ with } H(D) = -\sum_{c \in C} p_c \log p_c \quad (2.9)$$

$H(\cdot)$ is the entropy, p_c is the probability of having the class c in the dataset D, C is the set of classes in D, and SP is the set of possible split points.

Shapelets have been introduced as primitives for time series classification by [Ye & Keogh 2009b]. The authors proposed a shapelet based decision tree in which each node is a subsequence and the time series arriving at a node are split in two groups such that one group contains data that are similar to the subsequence at that node, and the other group is the set of data that are not similar to the subsequence. Figure 2.1 illustrates a node in the proposed decision tree. The blue time series contain the subsequence in the node (i.e they are similar to the subsequence at the node), so they follow the branch labeled with *yes*. The red time series does not contain the subsequence in the node (i.e they are not similar to the subsequence at the node), therefore the follow the branch labeled with *no*.

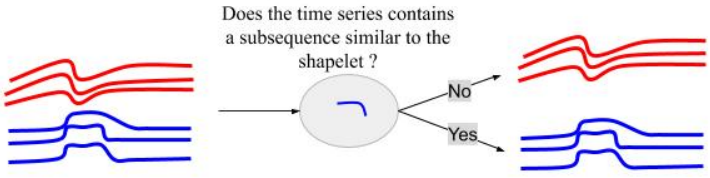


Figure 2.1: An illustration of a node in a shapelet decision tree for a binary time series classification

Training a shapelet based decision tree consists of learning the best subsequence (i.e shapelet) to use at each node, and for each shapelet the best similarity threshold.

This training is done in a top-down approach as in a classical decision tree using the information gain (IG) at each node to find the best split.

2.3.2 Uncertain shapelet transform classification

Our algorithm for uncertain time series classification is an extension of the shapelet transform algorithm [Hills *et al.* 2014].

Given a dataset D of uncertain time series, the first step is to select the top k best uncertain shapelets from the dataset. This step is achieved using the procedure described by Algo. 1, which takes as input, the dataset D, the maximum number of uncertain shapelets to be extracted k, the minimum and the maximum length of an uncertain shapelet MIN and MAX. This algorithm uses three subprocedures:

- $GenCand(T, MIN, MAX)$ which generates every possible uncertain shapelet candidates from the input uncertain time series T. These candidates are uncertain subsequences of T, with length at least MIN and at most MAX.
- $AssessCand(cands, D)$ which computes the quality of each candidate in the list of candidates $cands$. The quality of a candidate is the information gain it produces when used as a separator for the dataset.
- $ExtractBest(C, Q, k)$ which takes the list of uncertain shapelet candidates C, their associated qualities Q and returns first k uncertain shapelets with highest qualities.

In summary, Algo. 1 generates every uncertain subsequences of length at least MIN and at most MAX from the dataset, assesses the quality of each one by computing the information gain obtained when it is used as a separator for the dataset and finally returns the k subsequences that produce the highest information gain. The parameters MIN and MAX should be optimized to reduce the execution time of the algorithm. With the knowledge of the domain, the length of a typical shapelet can be estimated and used to set MIN and MAX in order to reduce the number of shapelet candidates. By default MIN is set to 3 and MAX is set to $m - 1$, where m is the length of the time series.

The next step after the top-k uncertain shapelets selection is the uncertain shapelet transformation. This step is done using Algo. 2, which takes as input the dataset D, the set of the top-k uncertain shapelets S and the number of uncertain shapelets k. For each uncertain time series in the dataset, its uncertain feature vector of length k is computed using UED. The i^{th} element of the vector is the UED between the uncertain time series and the uncertain shapelet i. Because the uncertainties add up during the transformation, the uncertain feature vectors are such that the scale of the best guesses is smaller than the scale of the uncertainties. It is very important to have everything on the same scale. The second for loop of Algo. 2 performs the standard normalization of the transformed dataset. We use $\hat{D}_{:j}$ to represent the list of the best guesses of uncertain dissimilarities between every uncertain time series and the j^{th} uncertain shapelet, and $\delta D_{:j}$ is the list of the

Algorithm 1: Top-K Uncertain Shapelet Selection

Input: D, k, MIN, MAX

```
begin
    C ← ∅; Q ← ∅
    for i ← 1, n do
        cands ← GenCand(T_i, MIN, MAX)
        qualities ← AssessCand(cands, D)
        C ← C + cands
        Q ← Q + qualities
    end
    S ← ExtractBest(C, Q, k)
    return S // Top k uncertain shapelets
end
```

corresponding uncertainties. The scaled and transformed dataset is finally returned by the algorithm.

The third and last step is the effective classification. A supervised classifier is trained on the uncertain transformed dataset, such that, given the feature vector of an unseen uncertain time series, it can predict its class label. Since the uncertainty have been propagated, the training process can be aware of uncertainty by taking it as part of the input. More specifically, best guesses are features and uncertainties are features of best guesses, and thus are meta-features. There exists many supervised classifiers in the literature for the classification of uncertain tabular data [Li *et al.* 2020a, Aggarwal & Yu 2009]. We have decision tree-based methods [Tsang *et al.* 2009, Qin *et al.* 2009], SVM-based methods [Bi & Zhang 2005, Yang & Gunn 2007, Li *et al.* 2020b] and Naive Bayes-based methods [Qin *et al.* 2011, Qin *et al.* 2010]. Since the transformed data is an uncertain tabular data, uncertain supervised classifiers can be used for the classification step. Furthermore, any supervised classifier can be used as soon as the transformed dataset is formatted in a way that is accepted by that classifier.

If instead of UED, one of the existing metrics from the state of the art (DUST, MUNICH, PROUD or FOTS) is used, the classifier would not be able to learn while being aware of uncertainty in the input since the output of these metrics are apparently 100% reliable; most importantly, it would not be possible to take advantage of an uncertain classifier.

Fig. 2.2 gives an overview of the classification process. During the training step, top-k uncertain shapelets are selected and an uncertain supervised model (illustrated here by a decision tree for simplicity) is trained on the uncertain transformed dataset. During the test step, the uncertain shapelets extracted during the training step are used to transform the test set, and the trained model is used to predict the class labels of the test set according to the result of the transformation. We call this model UST for Uncertain Shapelet Transform classification.

Algorithm 2: Uncertain Shapelet Transformation

Input: D, S, k

begin
  for $i \leftarrow 1, n$ **do**
    $temp \leftarrow \emptyset$ **for** $j \leftarrow 1, k$ **do**
      $temp_j \leftarrow UED(T_i, S_j)$
    end
    $D_i \leftarrow temp_j$
  end
  for $i \leftarrow 1, n$ **do**
    for $j \leftarrow 1, k$ **do**
      $best \leftarrow \frac{\hat{D_{ij}} - mean(\hat{D_{:j}})}{std(\hat{D_{:j}})}$
      $delta \leftarrow \frac{\delta D_{ij} - mean(\delta D_{:j})}{std(\delta D_{:j})}$
      $D_{ij} = best \pm delta$
    end
  end
  return D `// The transformed dataset`
end

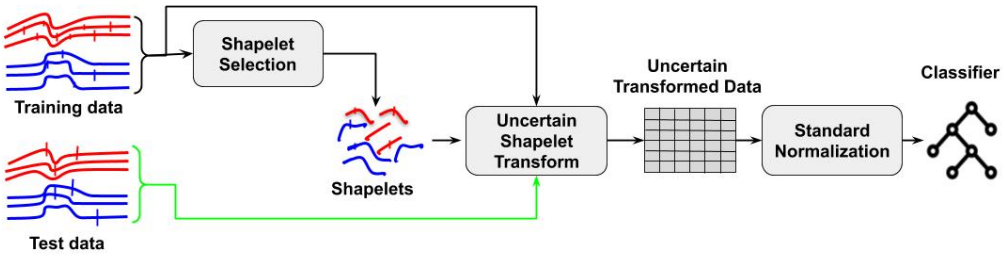


Figure 2.2: Uncertain time series classification process

2.3.3 UST time complexity

Since the number of shapelet candidates of length l in the dataset is bounded to $n \times (m - l + 1)$, the subprocedures used by the Uncertain Shapelet Transform algorithm do not last forever; the rest of the algorithm are for loops, and thus the UST algorithm ends. The algorithm also evaluates all possible shapelet candidates and keep only the k best ones. In term of time complexity, it is the same time complexity as the original shapelet transform (ST) which is $O(n^2m^4)$. In fact the differences are the dissimilarity computation and the transformation process. UST uses UED to compute the dissimilarity, while ST uses the Euclidean distance (ED). The time complexity of UED is $O(n)+O(n)$ which is asymptotically equal to $O(n)$, the time complexity of ED. The scaling of the transformed dataset does not change the complexity order of the transformation process as well.

2.4 Experiments

In this section, we experimentally assess our model and compare it with the state of the art. The comparison criterion is the model classification accuracy as it is always done in the literature [Bagnall *et al.* 2017, Hills *et al.* 2014, Siyou Fotso *et al.* 2020, Dallachiesa *et al.* 2012]. Since the output of our model is the probability distribution over the set of classes, we take the most probable class as the predicted class and use it to compute the model accuracy.

2.4.1 Compared models

We have compared different models which are different configurations of the UST model. In particular, our models are built regarding the following attributes: uncertain similarity, ordering strategy, and supervised classifier.

Uncertain similarity

This is how the dissimilarity between uncertain subsequences is computed by UST. This attribute has five possible values which are ED, UED, FOTS, U_DUST, and N_DUST.

We set the parameter of FOTS following the specifications in its original paper [Siyou Fotso *et al.* 2020]. Hence, the number of windows (m) is equal to the length of a window (w) which is equal to half the length of the time series. The time index (t) used to compute the auto-covariance matrices is equal to half the length of the time series, and finally the number of eigenvectors (k) is set to 4.

Ordering strategy

This is the method used to sort uncertain measures, that is simple, stochastic or interval ordering. When measures are not uncertain (when using FOTS or DUST), we use the natural order.

For the stochastic ordering we consider an uncertain measure x to be normally distributed. Given this assumption, the following cumulative distribution function can be used

$$CDF_X(t) = \frac{1}{2}(1 + erf(\frac{t - \hat{x}}{\delta x\sqrt{2}})) \tag{2.10}$$

where $erf(\cdot)$ is the Gauss error function. To discretize I (using Eq. 2.5), we fixed the value of k to 100. Larger values of k lead to best approximation of I, however slow the classification process. We tried several values of k, but $k = 100$ worked better. We have also used a relaxed version of the stochastic ordering: given two random variables X and Y, we have $X \leq_{st} Y$ if the number of values t in I such that $CDF_X(t) > CDF_Y(t)$ is greater than the number of values t in I such that $CDF_X(t) \leq CDF_Y(t)$.

For the interval ordering, we say that $x \leq y$ if $Pr[x \leq y] > 0.5$

Supervised classifier

This is the model used to classify the transformed dataset in the last step of UST. We used the classical Gaussian Naive Bayes (GNB) and the Uncertain Gaussian Naive Bayes (UGNB) models. We implemented UGNB following [Qin *et al.* 2011]. We chosen these classifiers for their simplicity in order to evaluate UED and the importance of propagating uncertainty followed by the use of a classifier that takes uncertainty into account during its training phase.

For each model, the parameter MIN and MAX are set to 3 and $m-1$ respectively, where m is the length of the uncertain time series in the dataset being processed. Because of the high time complexity of the algorithm, we have used a time contract to limit the execution time of each model. After the evaluation of an uncertain shapelet candidate, the next candidate is evaluated only if there is time remaining in the contract; otherwise the shapelet search is ended. Because FOTS is more time consuming than ED, UED and DUST, we set FOTS's time contract 12 times higher above the time limit of other measures.

Tab. 2.1 gives a summary of the different models that are evaluated and compared throughout our experiments. In order to apply the model UST(UED, GNB),

Table 2.1: Summary of the models that are compared in our experiments.

Name	Measure	Ordering	Classifier	Time contract
ST	ED	Natural	GNB	10 minutes
UST(DUST_NORMAL)	DUST NORMAL	Natural	GNB	10 minutes
UST(DUST_UNIFORM)	DUST UNIFORM	Natural	GNB	10 minutes
UST(FOTS)	FOTS	Natural	GNB	120 minutes
UST(UED, GNB)	UED	Simple, Stochastic and Interval	GNB	10 minutes
UST(UED, UGNB)	UED	Simple, Stochastic and Interval	UGNB	10 minutes

each uncertain feature vector is flatten such that the first half contains the best guesses and the second half contains the uncertainty deviation. This is required because the Gaussian naive bayes (GNB) classifier does not take uncertainty into account.

2.4.2 Datasets

We used datasets from the well known UCR repository [Dau *et al.* 2019]. Instead of running our experiment on the whole repository, we use only datasets on which shapelet approaches are known to work well. According to [Bagnall *et al.* 2017], shapelet approaches are more suitable for electric device, ECG, sensor and simulated datasets. Tab. 2.2 gives a summary of the 15 shapelet datasets on which we conducted our experiments. The first column is the name of the dataset, the second is the number of instances in the training/test set, the third is the length of time series and the fourth and last column is the number of different classes in the dataset. Each dataset is already split into the training and the test sets on the repository.

Table 2.2: Datasets Description

Datasets	train/test	length	#classes
BME	30/150	129	3
CBF	30/900	129	3
Chinatown	20/345	25	2
ECGFiveDays	23/861	137	2
ECG200	100/100	96	2
ItalyPowerDemand	67/1029	24	2
Plane	105/105	145	7
ShapeletSim	20/180	500	2
SmoothSubspace	150/150	16	3
SonyAIBORobotSurface1	20/601	70	2
SonyAIBORobotSurface2	27/953	65	2
SyntheticControl	300/300	60	6
Trace	100/100	275	5
TwoLeadECG	23/1139	83	2
UMD	36/144	151	3

Since the datasets in this repository are without uncertainty, we manually add uncertainty in the datasets listed in Tab. 2.2. Given a dataset, the standard deviation σ_i of each timestep is computed. For each time series in the dataset, the added uncertainty for the observation at timestep i follows a normal distribution with mean 0 and standard deviation $c \times \sigma$, where σ is randomly chosen from a normal distribution with mean 0 and standard deviation σ_i . We used different values of c ranging from 0.1 to 2. The uncertainty result for an instance from the Chinatown dataset is shown by Fig. 2.3 for $c = 0.6$. The orange line is the original time series, and the blue one is the obtained uncertain time series. Sometimes, the original time series does not cross the uncertainty interval (vertical red bars), these cases are there to represent situations where the uncertainty has not been well estimated, maybe because the expert has been too optimistic. Situations where the expert had been too pessimistic are represented by very large uncertainty bars. During the training phase, original time series are not used, only the uncertain time series are used.

We implemented UST in the Python programming language, and we used the open source package sktime [Löning *et al.* 2019]. The code and the data used for our experiment are publicly available[1].

2.4.3 Results

For each of the models we compared, we have recorded the obtained accuracy, the training duration and the testing duration. These values are recorded for each level of uncertainty.

[1] https://github.com/frankl1/ustc/releases/tag/litsa

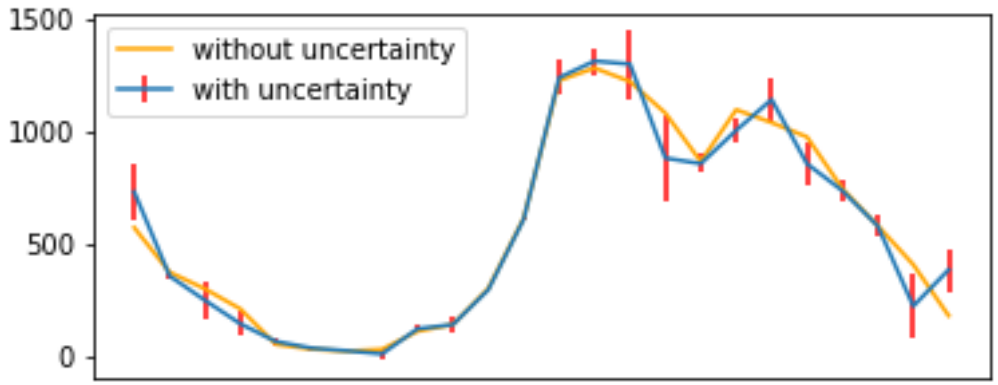


Figure 2.3: Illustration of uncertainty for an instance from the Chinatown dataset. the uncertainty level is $c = 0.6$

Uncertain ordering analysis

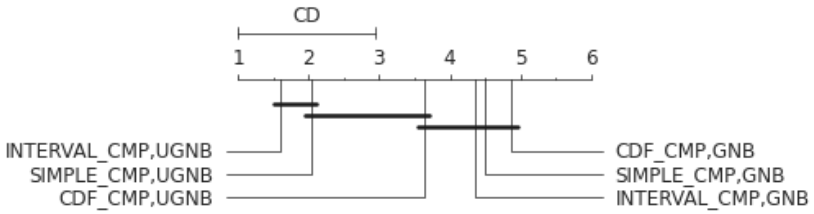
Let's analyze how the uncertain ordering used affects the accuracy of UED-based models. Fig. 2.4 shows the critical difference diagram to exhibit any significant difference between our ordering strategies. The ordering strategy does not affect the model accuracy too much, this is normal given the fact that the three ordering could be different only if the uncertain measures to be sort overlap. However the simple ordering (called SIMPLE_CMP on the plots) and the interval ordering (called INTERVAL_CMP on the plots) are better than the stochastic ordering (called CDF_CMP on the plots).

Accuracy analysis

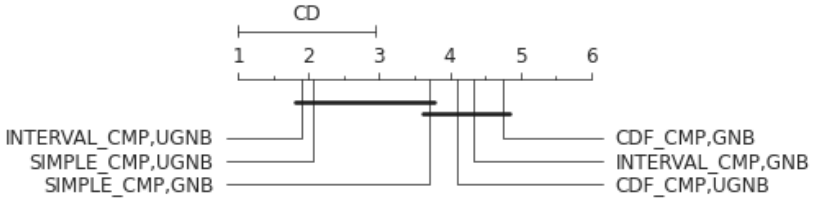
In this analysis, UST(UED, GNB) and UST(UED, UGNB) use the interval ordering only. UED-based models are better than the others. They are even better when the uncertain naive bayes is used as classifier, that is UST(UED, UGNB). We observe also that the accuracy of each model decreases when the uncertainty level increases; however the accuracy of UED-based models decreases more slowly than the accuracy of DUST and FOTS-based models: UED-based models are more robust to uncertainty changes, and even more when an uncertain classifier is used.

The critical difference diagrams on Fig. 2.5 show how different the compared models are for some levels of uncertainty. Whatever the level of uncertainty is, propagating the uncertainty gives models that are significantly more accurate than not using uncertainty propagation.

The model ST is the classical shapelet approach, in which the Euclidean distance is used. The classifier used is the Gaussian naive bayes. In this model only the best guesses are used, and there is no uncertainty handling at all. For low uncertainty levels, this model is the second best model, behind UST(UED, UGNB).



(a) Uncertainty level $c = 0.4$



(b) Uncertainty level $c = 1.2$

Figure 2.4: Critical difference diagrams of UED-based models regarding the ordering strategy for some levels of uncertainty

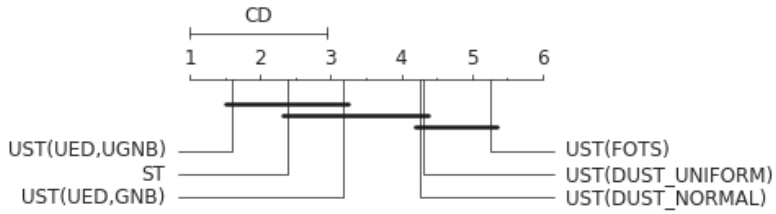
When the uncertainty increases, ST becomes among the worst models, exhibiting the importance of uncertainty handling in time series classification. For instance, on the dataset SmoothSubspace and the uncertainty level of $c = 0.4$, ST achieves 47% of accuracy, UST(UED, GNB) achieves 49%, while UST(UED, UGNB) achieves 83%.

Training duration

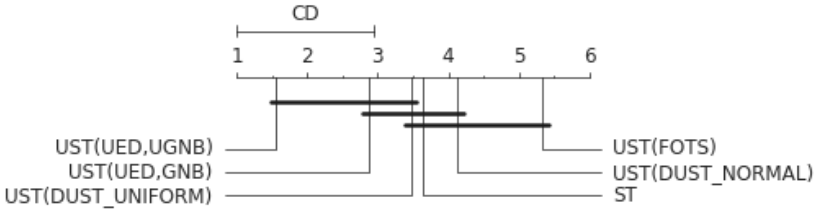
In term of training duration, the UST model that uses FOTS dissimilarity takes in average 128 ± 9 minutes to finish. When using DUST, ED or UED, UST takes in average 10 ± 0.35 minutes to finish. Hence, FOTS-based models are much slower compared to DUST and UED-based models. This is because FOTS has a quadratic time complexity (due to eigenvector decomposition) while ED, DUST and UED have a linear time complexity. Hence the FOTS-based model requires more time to learn.

Overall discussion

To build the uncertain shapelet transform classification algorithm, we have handled uncertainty in each step of a typical shapelet algorithm. During the first step where shapelets are selected, UED is used to select shapelets while taking uncertainty into account; when two shapelets has the same information gain, the one that is less uncertain is preferred. During the shapelet transform step, the uncertainty is propagated so that the transformed dataset contains uncertainty information. To really take advantage of the propagated uncertainty, a classifier that has been built for uncertain data is used in the last step. Without the use of an uncertain classifier,



(a) Uncertainty level $c = 0.1$



(b) Uncertainty level $c = 0.8$

Figure 2.5: Critical difference diagrams of models for some levels of uncertainty

all the previous steps of uncertainty management are not really worthy.

Uncertainty is unpredictable, and because we are dealing with it, it is difficult to identify in which uncertain situation our approach will work well. For this reason, we use different levels of uncertainty in our experiment, expecting to cover at most possible situations as we can. The uncertainty levels from $c = 1$ to $c = 2$ are more likely to be extreme and may not be found in a real application, but it is important to see how the model's behavior as the uncertainty becomes too large.

We manually added uncertainty in our datasets. Applying our model on a real uncertain dataset will strengthen our contribution. Nevertheless, by using different levels of uncertainty in our datasets, we expect to cover any real situation.

The uncertain classifier we used is the uncertain naive Bayes. We choose it for its simplicity. There are other uncertain classifiers [Li *et al.* 2020a, Aggarwal & Yu 2009], and they can be used in UST, but we did not try them because our goal was to show how important it is to correctly handle uncertainty in the context of uncertain time series classification. We highly recommend to try other uncertain classifiers when in real application.

Finally, the time contract we set during our experiments limits in some ways the discovery of more, and why not better uncertain shapelets. In fact, maybe new uncertain shapelets might have been discovered with a larger time contract.

2.5 Conclusion

The goal of this chapter was to classify uncertain time series using the shapelet transform approach. To achieve this goal, we use uncertainty propagation techniques to define an uncertain dissimilarity measure called UED. Then we adapt the well known shapelet algorithm to the context of uncertain time series using

UED and propose the uncertain shapelet transform algorithm (UST). We have run experiments on state of the art datasets. The results show that propagating uncertainty during the shapelet transformation and then using an uncertain classifier lead to a more accurate model for uncertain time series classification. The idea of uncertainty propagation can be used with any dissimilarity measure, and any uncertain supervised classifier can be used in the classification phase.

Similarly to shapelet transform, UST's high computation time for identifying shapelets is one the of the greatest limitations of the approach. In the next chapter, we will describe a novel design of shapelet-based classification that overcomes this limitation while keeping at least the same level of accuracy.

Key points

- We proposed the Uncertain Euclidean Distance (UED) which provides uncertain similarities between uncertain time series.
- We used UED to take uncertainty into account in the shapelet-based classification and proposed the Uncertain Shapelet Transform (UST).
- We shown that using UED leads to better classification than using existing uncertain measures in the state-of-the-art.

Communications

- Michael F. Mbouopda, Engelbert Mephu Nguifo. *Classification des Séries Temporelles Incertaines Par Transformation Shapelet.* In Conférence Nationale en Intelligence Artificielle (CNIA), pp.14-21, Jun. 2020.
- Michael F. Mbouopda, Engelbert Mephu Nguifo. *Classification of Uncertain Time Series by Propagating Uncertainty in Shapelet Transform.* In ECML/PKDD Workshop on Uncertainty in Machine Learning (WUML), pp.1-12, Sep. 2020.
- Michael F. Mbouopda, and Engelbert Mephu Nguifo. *Uncertain time series classification with shapelet transform.* In International Conference on Data Mining Workshops (ICDMW), pp. 259-266, Nov. 2020.

Chapter 3

Scalable and Accurate Subsequence Transform for Time Series Classification

Time series classification using phase-independent subsequences called shapelets is one of the best approaches in the state of the art. This approach is especially characterized by its interpretable property and its fast prediction time. However, given a dataset of n time series of length at most m, learning shapelets requires a computation time of $O(n^2m^4)$ which is too high for practical datasets. In this chapter, we exploit the fact that shapelets are shared by the members of the same class to propose the SAST (Scalable and Accurate Subsequence Transform) algorithm which has a time complexity of $O(nm^3)$. SAST is accurate, interpretable and does not learn redundant shapelets. The experiments we conducted on the UCR archive datasets shown that SAST is more accurate than the state of the art Shapelet Transform algorithm while being significantly more scalable.

Contents

3.1 Introduction 48
3.2 SAST: Scalable and Accurate Subsequence Transform . . . 49
3.2.1 Reducing the number of shapelet candidates 49
3.2.2 Identify shapelets using feature importance analysis 53
3.2.3 Time series classification with SAST 54
3.2.4 SAST time complexity 55
3.2.5 Ensemble of SAST models 56
3.3 Experiments 56
3.3.1 Accuracy 57
3.3.2 Scalability 65
3.3.3 Interpretability 67
3.4 Conclusion 70

3.1 Introduction

Time series classification with shapelets is accurate, robust to noise and interpretable [Ye & Keogh 2009b]. In particular, the shapelet transform algorithm is known to be among the most effective when tested on the UEA & UCR archive [Bagnall *et al.* 2017]. Shapelet have also been proved to be effective in time series clustering [Siyou Fotso *et al.* 2020], showing how useful shapelets are. The interpretability of a shapelet method is obtained by visualizing the subsequences that triggered the class label of a given instance. Since the introduction of time series classification using shapelets, one of the major limitations of the developed algorithms is their time complexity. In fact, the state-of-the-art time complexity of shapelet based methods is n^2m^4 where n is the number of time series in the dataset and m is the length of the longest time series. This high time complexity is due to the large number of shapelet candidates that need to be evaluated in order to find the top best shapelets.

A human brain is able to recognize a lot of variations of an object after seeing a single variant. For instance, we are able to recognize any model of car after seeing one of them, we can recognize many species of dog if we have ever seen a dog. This ability is called *core object recognition* [DiCarlo *et al.* 2012]. Inspired by this amazing behavior of our brain, we claim that a shapelet model should be able to recognize any variant of a shapelet if it knows one or a few number of its variants. Simply defined, a shapelet is a pattern that is shared by the time series that belong to the same class. Therefore, any single instance of a class should contain all the shapelet or at least a variant of each shapelet for that class. Guided by this observation, we propose the Scalable and Accurate Subsequence Transform (SAST) algorithm, a time series classification algorithm that is accurate, scalable and whose predictions are interpretable.

Existing shapelet based methods use the whole dataset to generate shapelet candidates, then use information gain to select the top best shapelets before doing the classification using a supervised classifier. We claim that it is not necessary to generate the shapelet candidates from the whole dataset, only one or few instances per class is enough. We also claim that pruning shapelet candidates without taking into account the classifier can lead to inaccurate classification. We propose the SAST model to support our claims ; it uses only a single instance per class in order to generate shapelet candidates. Furthermore, shapelet candidates are not assessed beforehand of classification. The supervised classifier automatically identifies the top best shapelets during its training phase. The key points of our contribution are the following:

- We introduce the *core shapelet recognition* task which aims to recognize any variant of a shapelet from one or few variants of that shapelet. We claim that time series classification by shapelets is a core shapelet recognition task and therefore the size of the shapelet space is considerably reduced without losing crucial information.

- We propose the SAST method, which successfully performs the core shapelet recognition task in order to accurately classify time series. SAST is also more scalable than the state of the art shapelet methods. In particular, SAST took 1 second to classify the Chinatown dataset with an accuracy of 96%, while the state of the art shapelet based algorithm STC took 51 seconds and achieved an accuracy of 97% on the same computer. Furthermore our proposed method can successfully classify some datasets on which STC fails.

The rest of this chapter is organized as follows: In Section 3.2 we describe our proposed method SAST, which is inspired by the core object recognition capability of human brain. In Section 3.3, we assess SAST on various datasets and compare it to state of the art shapelet and non-shapelet based methods. Section 3.4 summarizes this work and presents future direction.

3.2 SAST: Scalable and Accurate Subsequence Transform

In time series classification, a shapelet is ideally a pattern that is shared by every instances of the same class, and that instances of other classes do not have, they are called discriminative patterns or subsequences. The number of patterns in a dataset of n time series of length m is $O(nm^2)$, and state of the art shapelet algorithms evaluate each of them by computing their information gain for a set of similarity thresholds before keeping the patterns and their corresponding similarity thresholds that give the highest information gain. Reducing the number of patterns to be assessed will make shapelet models faster to train. In this section we propose a way to reduce the number of shapelet candidates. Then we show that there is no need to select the top best shapelets beforehand. Finally we present a novel method for shapelet based time series classification.

3.2.1 Reducing the number of shapelet candidates

Human brain effortlessly performs *core object recognition*, the ability to recognize objects despite substantial appearance variations [DiCarlo *et al.* 2012]. This gives human the capability to recognize a vast number of objects that have the same name just by seeing a few of them. [Heeger 2014] used Figure 3.1 in his lecture notes on Perception to illustrate the notion of invariance in recognition. This figure shows different ducks. Some are in water while others are not, some ducks are photographs and other are drawings. Furthermore, the ducks have different sizes, colors, etc. Despite all these variabilities, a human brain that has already seen a duck is able to recognize that each object on this figure is a duck.

A shapelet is a pattern, a shape that is "common" to time series that have the same class label. By "common", we do not mean that these time series have exactly that shapelet, but they have a pattern that is very similar to the shapelet. Any pattern that is similar to a shapelet can be considered as a variant of that shapelet.

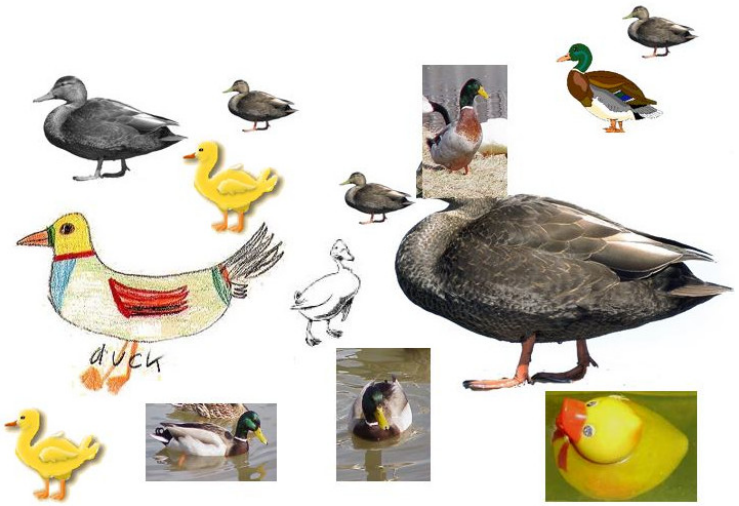


Figure 3.1: Illustration of invariance in recognition.

Therefore, we introduce the *core shapelet recognition* task, which goal is to recognize any variant of a shapelet by just seeing one or very few number of its variants. We argue that time series classification based on shapelets is a core shapelet recognition task and that it should be solved using very few shapelet candidates than it has been done since the introduction of shapelets by [Ye & Keogh 2009b]. Hence, instead of generating shapelet candidates from the whole dataset, we propose to use only one or a few number of instances per class. In fact, the members of a class should contain common patterns or at least different variants of the same pattern. In this way, the learning algorithm must focus on one (or a small number) variant of each shapelet candidate to learn how to classify a time series. We acknowledge that the more variants the model learns from, the more accurate the model would be. However this can also cause the model to overfit, especially when the training data is not enough representative of the testing data.

On Figure 3.2, we have three randomly selected instances of the Chinatown dataset from the UEA & UCR archive [Anthony Bagnall & Keogh 2018]. This dataset has two classes of data. The left example is from class 1 and the second example (in the middle) is from class 2. It is easy to observe that the instance from class 1 starts by a deep valley, while the instance of class 2 does not. One reason that can be considered in order to classify the instance on the right in class 2 is that it does not start by a valley. Hence, observing only one instance per class can be enough to discover discriminative patterns and successfully perform classification of new instances.

Figure 3.3 shows 4 randomly selected instances for each class. Instances of the same class are superposed in order to expose global patterns. The figure emphasizes the previous observation that class 1 contains instances that start by a deep valley while class 2 are instances that are more flat at the beginning.

Based on this observation, we propose the following statement:

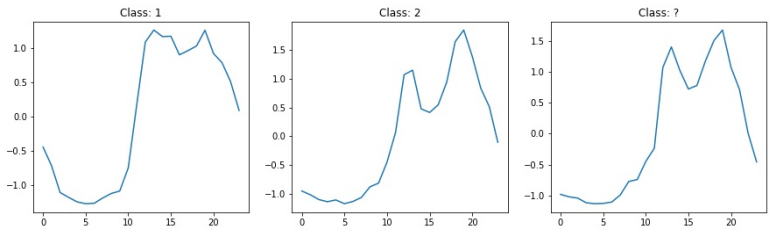


Figure 3.2: Three randomly selected instances from the Chinatown dataset. The instance on the right is probably from class 2 since it does not start with a valley

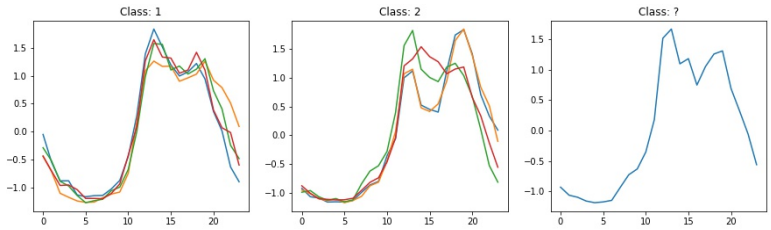


Figure 3.3: Some randomly selected instances from the Chinatown dataset. The instance on the right is definitely from class 2

Proposition 3.2.1. *Let $D = \{(T_1, c_1), (T_2, c_2), ..., (T_n, c_n)\}$ be a dataset of time series. Let D_c be any subset of D that contains at most k ($k \geq 1$) instances from each class. If classes in D can be distinguished using shapelets, then for any shapelet shp of D, there exists a time series in D_c that is similar to shp.*

Proof. Let's assume that classes in D are distinguishable using shapelets and that there exists a shapelet *shp* for the dataset D that is not similar to any time series in the set D_c. Since D_c contains at least a time series of each class in D, any shapelet for the dataset D must be similar to at least one time series in D_c. It follows from there that assuming *shp* to be a shapelet is wrong. Therefore the statement is true. □

From the previous proposition, any shapelet *shp* of D is always similar to a pattern in D_c. Therefore, a shapelet algorithm that generated shapelet candidates from D_c can achieve the same accuracy as if D was used. We run the shapelet transform algorithm (STC) [Hills *et al.* 2014] on the Chinatown dataset and plotted the top 5 shapelets that have been selected for each class on Figure 3.4. The shapelets on the first row clearly identify the valley at the beginning of time series in class 1. Although they are coming from different time series, they are very similar in shape. Likewise, the shapelets on the last row identify the flat starting of instances in class 2. Generating shapelet candidates from the whole dataset makes STC learns different variants of the same patterns. The variations are in terms of starting position, length and shape.

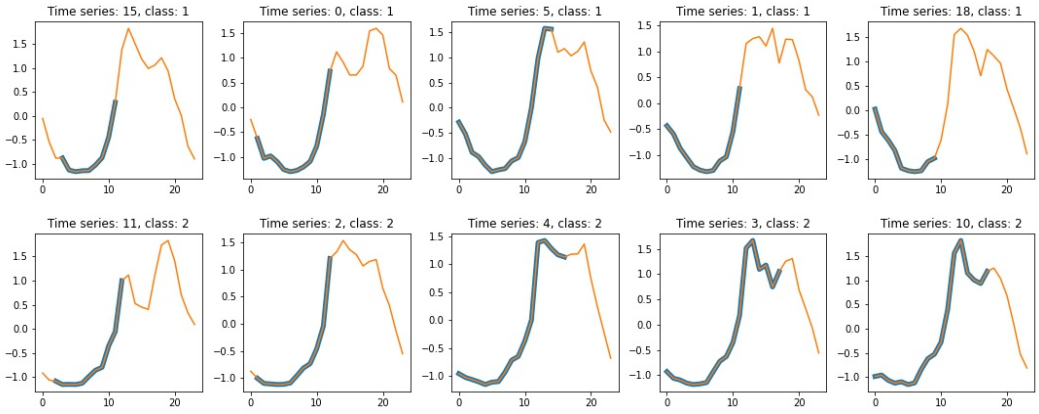


Figure 3.4: Top 5 shapelets extracted for each class of the Chinatown dataset by the shapelet transform algorithm.

By applying proposition 3.2.1 in the STC algorithm, we introduce STC-k, a variation of the STC algorithm that uses at most k time series from each class to generate the shapelet candidates. Hence, for a dataset with c classes, the number of shapelet candidates to be evaluated in STC-k is $O(ckm^2)$ unlike STC in which $O(nm^2)$ need to be evaluated. Algorithm 3 is an outline of the STC-k algorithm. The only difference with STC is that the size of the shapelet space can be controlled by the parameter k. For a more detailed description of the algorithm, the reader should refer to the original STC's paper [Hills *et al.* 2014].

By default, the $length\_list$ parameter is the set $\{3, 4, ..., m\}$, where m is the length of the time series in the dataset. min_ig is set to 0.05. In practice, two other parameters are used in STC: the maximum number of shapelets to keep per class and the time contract. The number of shapelets to keep per class is by default set to 200. The time contract is the maximum time allocated to the algorithm to search shapelets on the given dataset. [Middlehurst *et al.* 2020a] stated that in one hour of searching per dataset, the result is not significantly worse than the full search. Following the implementation of STC in the sktime library [Löning *et al.* 2019], our implementation of Algorithm 3 handles the time contract and the maximum number of shapelets to keep per class.

We use the Chinatown dataset from the UCR archive [Anthony Bagnall & Keogh 2018] as a toy dataset to assess STC-1 (that is STC-k with $k = 1$). STC-1 took about 10 seconds to classify the Chinatown dataset with an accuracy of 96% (average over five runs), while the original STC algorithm took 51 seconds and gave an accuracy of 97% on the same computer. Therefore, the STC-1 algorithm is about 5 times faster and achieves almost the same accuracy as the original algorithm. The extracted shapelets are shown in Figure 3.5. Different variants of the same shapelet are not learned anymore. For this dataset, exactly one shapelet has been selected for each class. As we will show in Section 3.3, STC-k is significantly less accurate than STC, even when k is equal

to 75% of the number of time series in each class.

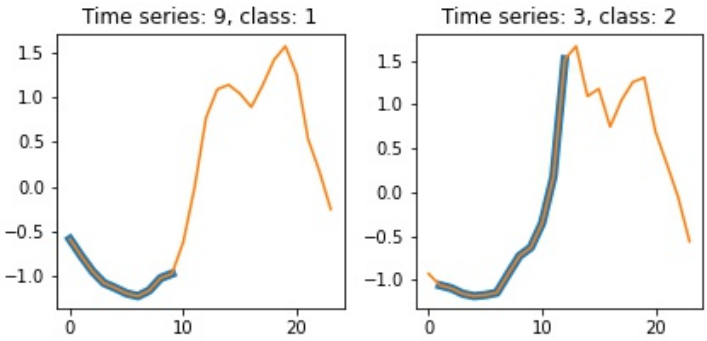


Figure 3.5: Shapelets extracted by STC on the Chinatown dataset using a single randomly selected instance per class to generate shapelet candidates.

3.2.2 Identify shapelets using feature importance analysis

When time series classification using shapelets was introduced, shapelets were learned while building a decision tree [Ye & Keogh 2009b]. Later, shapelet transform (STC) [Hills *et al.* 2014] has been proposed to allow the use of any supervised classifier. The algorithm proceeds by finding the top best shapelets, then transforms the dataset using the found shapelets and finally trains a classifier on the transform dataset [Hills *et al.* 2014, Bostrom & Bagnall 2017, Karlsson *et al.* 2016]. Therefore, there are three main steps: feature extraction where best shapelets are selected, dataset transformation where each time series is replaced by a vector of its distance to the selected shapelets and finally training where a classifier is trained on the transformed dataset.

We propose to remove the feature extraction step and use every shapelet candidates to transform the dataset. After training the classifier on the transformed dataset, a post hoc method for model explanation can be used to find the most important features. The importance of a feature represents how much that feature is correlated to the target variable [Dash & Liu 1997, Molnar 2020].

Proposition 3.2.2. *Let $D = \{(T_1, c_1), (T_2, c_2), ..., (T_n, c_n)\}$ be a dataset of time series, and S the set of all subsequences in D. Let $D_f = \{(x_1, c_1), (x_2, c_2), ..., (x_n, c_n)\}$ be a dataset such that $x_i = [x_{i:1}, x_{i:2}, ..., x_{i:|S|}]$, where $x_{i:j} = dist(T_i, S_j)$. If the j^{th} feature is an important feature given by the analysis of feature importance for the dataset D_f, then S_j is a shapelet for the dataset D.*

Proof. Let's suppose the j^{th} feature is an important feature, and that S_j is not a shapelet for the dataset. By definition 2.5, not being a shapelet means that the information gain of S_j is not high enough, and whether a time series T is similar or not to S_j does not give any clue about the class of T. Therefore, knowing $dist(T, S_j)$ doesn't help to classify T. In other words, the j^{th} feature is not correlated to

the target variable. Hence, it cannot be an important feature. This proves the statement. □

The importance of a feature in a tree based algorithm determines how much it reduces the variance of the data compared to the parent node [Dash & Liu 1997, Molnar 2020]. This corresponds exactly to the definition of a shapelet (see Definition 2.5). In a linear model, the absolute value of the weight of an important feature will be greater than the one of a less important feature [Molnar 2020]. Classifiers such as decision trees and linear models are said to be inherently interpretable since a post hoc analysis is not required to interpret their predictions. More generally, when a classifier is fitted, a post hoc explainer can be used to find most important features [Murdoch *et al.* 2019] in order to interpret predictions. Two examples of these post hoc explainers are LIME [Ribeiro *et al.* 2016] and SmoothGrad [Smilkov *et al.* 2017] for saliency maps. More methods can be found in the review of [Samek *et al.* 2020]. Hence, selecting shapelets beforehand of classification using information gain can be skipped, since the classifier can automatically learn the top best shapelets during its training iterations and feature analysis can be used after training to get the learned shapelets.

3.2.3 Time series classification with SAST

Time series classification with SAST (Scalable and Accurate Subsequence Transform) is designed with respect to Proposition 3.2.1 and Proposition 3.2.2. A visual view of the the method is shown on Figure 3.6. There are two main blocks:

- *The classification block:* this block is actually the SAST algorithm and begins with the random selection of reference time series from which subsequences are then generated. Thereafter, the dataset is transformed by replacing each time series with a vector of its distances to each subsequence. Finally a supervised classifier (illustrated here by a decision tree) is trained on the transformed dataset.

- *The interpretability block:* The role of this block is to explain the SAST algorithm by identifying shapelet candidates associated with the most important features learned by the classifier. For inherently interpretable classifiers such as decision trees, the importance of each feature is computed while fitting the classifier. For other classifiers, eventually not inherently interpretable, an existing post hoc explainer such as LIME [Ribeiro *et al.* 2016] can be used to find the importance of each feature.

A pseudo code of the SAST algorithm is given by Algorithm 4. SAST takes as input the time series dataset D, the number k of instances to randomly select from each class in order to create the shapelet candidates, the list of lengths to use to generate shapelet candidates, and finally the supervised classifier C that is going to be trained on the transformed dataset.

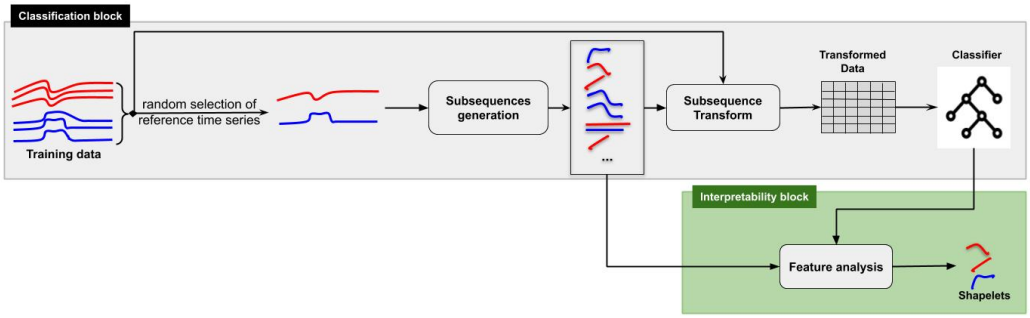


Figure 3.6: Overview of the SAST method

SAST starts by randomly select k instances per class from the dataset (line 2). By default, k is set to one. We call the selected instances *reference time series*. The next step is the generation of every subsequences of length in $length\_list$ from the reference time series (line 3). The dataset transformation is performed from line 4 to 11. Here, the similarity between each time series in the dataset and each shapelet candidate is computed. The classifier taken as input is then trained on the transformed dataset (line 12). The algorithm returns the trained classifier, the shapelet candidates that have been generated.

After the training is done, the class labels of a test dataset can be predicted in two steps: firstly the dataset is transformed using the shapelet candidates that have been generated during training, and finally the trained classifier is used to predict the class labels of the transformed test dataset.

Since the number of time series in the dataset and the number of subsequences in a time series are limited, Algorithm 4 always terminates. According to Proposition 3.2.2, the returned classifier should have learned to classify time series regarding features that are related shapelets.

3.2.4 SAST time complexity

Each step of the SAST algorithm runs in a finite amount of time, therefore the algorithm always terminates. Selecting k reference time series is done in $O(c)$ time complexity, c is the number of classes in the dataset. There are $m - l + 1$ subsequences of length l in a time series of length m. The total number of subsequences for a time series is $\frac{m(m+1)}{2}$. Since there are kc reference time series in a dataset with c classes, generating all shapelet candidates is done in $O(kcm^2)$. The transformation step requires $O(nm^2)$ distance computations, each of which requires $O(l)$ (l is the length of the subsequence) point wise operations. As the maximum subsequence length is m, the time complexity of the transformation step is $O(nm^3)$. Therefore, to total time complexity of SAST is $O(c) + O(kcm^2) + O(nm^3) + O(classifier)$, where $O(classifier)$ is the time complexity of the classifier used. The overall asymptotic time complexity of the SAST algorithm is therefore $O(nm^3) + O(classifier)$. SAST is much faster than the state of the art shapelet transform algorithm (STC)

[Hills *et al.* 2014] which time complexity is $O(n^2m^4) + O(classifier)$.

3.2.5 Ensemble of SAST models

SAST accuracy is highly dependent on the randomly selected reference series. If a reference time series is noisy or not representative of its class, then it could be difficult for SAST to learn the best shapelets for the dataset. Furthermore, the random selection of reference time series could lead to a variance in performance. We use Bagging [Breiman 1996] to leverage these possible issues and we call the obtained model SASTEnsemble (or SASTEN in reduced form). SASTEN is obtained by ensembling r SAST models. Each individual model in the ensemble uses randomly selected reference time series and may also have different parameters, especially the parameters controlling the length of shapelet candidates (that is $length\_list$ in Algorithm 4). The final prediction is obtained by averaging the predictions of every SAST models in the ensemble.

The time complexity of SASTEN is r times the time complexity of SAST if run sequentially. But this can be reduced using parallelization. SASTEN uses r times more memory than a regular SAST.

3.3 Experiments

We have implemented STC-k, SAST and SASTEN in Python. Our implementation is based on the scikit-learn machine learning library [Pedregosa *et al.* 2011]. We have also followed scikit-learn design principles so that our models are compatible with any scikit-learn pipeline. We have used the implementation of STC (Shapelet Transform Classifier) from the sktime library [Löning *et al.* 2019]. The source code of our experiments and all the results we discuss here are publicly available here [1].

In all our experiments, the number of reference time series per class (that is the parameter k in Algorithm 4) is always set to one. The supervised classifier used in STC-k, STC and SAST is the Ridge classifier with Leave-One-Out (LOO) cross validation. This classifier is available in the scikit-learn library. The LOO cross validation is used to find the best regularization parameter among 10 log spaced values ranging from -3 to 3 (these values are inspired from [Dempster *et al.* 2020]). The other parameters are left to their default values and are not fine tuned.

We have also used the Random Forest classifier in SAST. For this classifier all features are evaluated at each node to find the best split and a split is selected if the impurity decreases by about 0.05, the minimal information gain for a shapelet like in STC. Although it is generally better to evaluate only a subset of the feature space in Random Forest in order to reduce the correlation between the trees, we have not followed this guideline in our work because we want the model to always select the best possible split (that is the best shapelet). However, each tree in the

[1] https://github.com/frankl1/sast/tree/master

ensemble is trained on a random subset of the training set. This classifier is also available in the scikit-learn library.

We make use of the Wilcoxon significance test with a p-value of 0.05 to compare our models. We give the result of this test as a critical difference diagram on which models that are not significantly different from each other are linked with a bold line. The code used for this test and to draw critical difference diagrams is from [Fawaz *et al.* 2019b].

Table 3.1 describes the models that we use in our experiments.

We experiment using 72 randomly selected datasets from the UEA & UCR repository [Anthony Bagnall & Keogh 2018]. The datasets in the repository are different in terms of series length, number of series, number of classes and application domain. For each dataset, the repository provides a training set and a test set. Since searching shapelets for one hour is not significantly worse than the full search on the UEA & UCR archive [Middlehurst *et al.* 2020a], we used a time contract of one hour for each STC-k models as well as for STC.

3.3.1 Accuracy

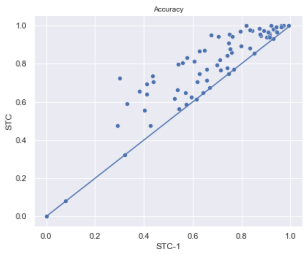
In this subsection, we compare the models in terms of accuracy and we use scatter plots and critical difference diagrams to summarized the results. However, the exact accuracy of SAST, STC and STC-k, which are the core models of this work are given in Table A.2.

3.3.1.1 STC-k results

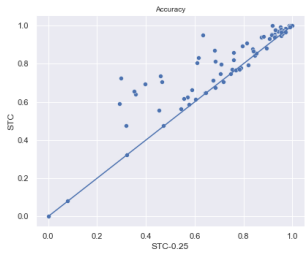
We have evaluated STC-k on 72 datasets with different value of the parameters k. We have considered STC-1, STC-0.25, STC-0.5, STC-0.75 and STC. These models are described in Table 3.1. Figure 3.7 shows pairwise comparisons of these model accuracies on the test set of each dataset.

STC is better than any STC-k on almost every datasets. This is because an STC-k model does not search the whole shapelet space, and therefore the shapelets obtained using the minimum information gain are not good enough to classify the dataset. The critical difference diagram on Figure 3.8 shows that STC-0.75 is not significantly more accurate than STC-0.5, which is significantly more accurate than STC-0.25, which is in turn significantly more accurate than STC-1. Therefore, STC-k accuracy increases with the value of the parameter k. All STC-k models are considerably less accurate than STC.

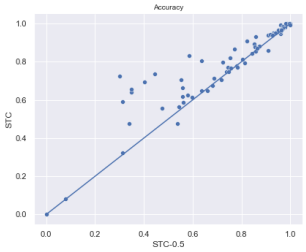
We have observed that, STC generally fails at classifying datasets that have few time series in the training set. In particular, STC failed to find shapelets on the Fungi datasets. This dataset has 18 classes with one instance per class in the training set. In this particular case, STC is exactly the same as STC-1.



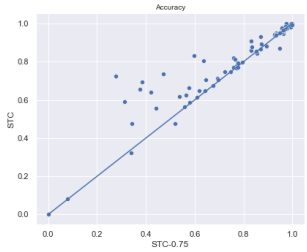
(a) STC (69 wins) vs STC-1 (1 win), 2 draws



(b) STC (58 wins) vs STC-0.25 (12 wins), 2 draws



(c) STC (55 wins) vs STC-0.5 (12 wins), 5 draws



(d) STC (50 wins) vs STC-0.75 (16 wins), 6 draws

Figure 3.7: Comparison of STC-k to STC in terms of accuracy

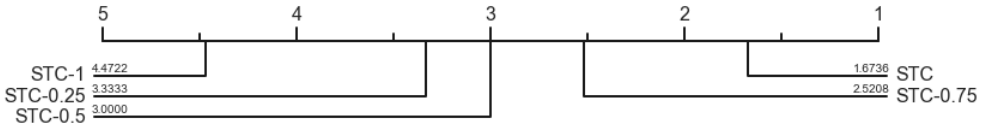


Figure 3.8: Critical difference diagram between STC and STC-k

3.3.1.2 SAST model results

Before comparing SAST to STC, let's see how SAST and ensemble of SAST are compared to each other in terms of accuracy. Figure 3.10 shows a pairwise comparison of the SAST based models described in Table 3.1 on the 39 datasets marked with a star in Table A.2. The first thing to note is that SAST-Ridge is generally more accurate than SAST-RF on our datasets (Figure 3.10a). There are many parameters in RF that can be optimized in order to improve SAST-RF, but we did not perform parameter tuning in this work and we consider SAST-Ridge as the best model for our experiment. This is why we use SAST-Ridge as the default SAST model and as the pivot in our comparison.

We tried several *length_list* for the approximated SAST model, and we are presenting here only the four that achieved the best accuracy on our datasets. The critical difference diagram between these four models is given in Figure 3.9. There is no significant difference between the models, however the model using $length\_list = \{7, 11, 15\}$ is the best of all. When not clearly precised in the

rest of this chapter, SAST-Ridge-A is the approximated SAST-Ridge model with $length\_list = \{7, 11, 15\}$.

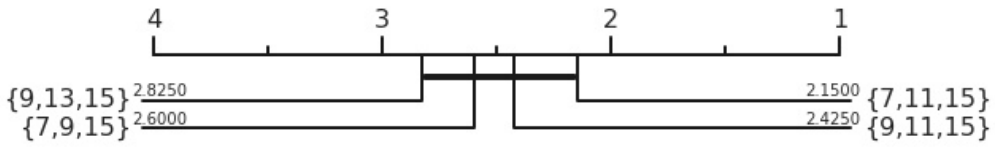
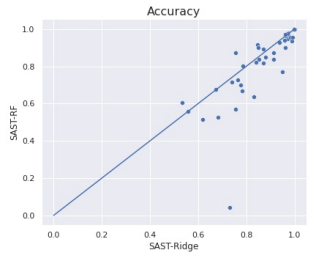
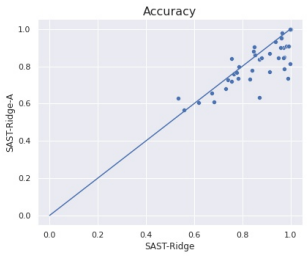


Figure 3.9: Critical difference diagram between approximated SAST models

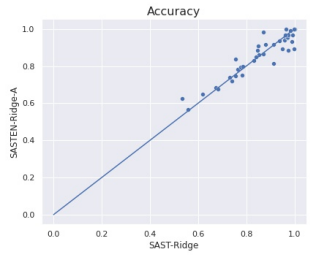
The approximated SAST-Ridge is less accurate than SAST-Ridge in general (Figure 3.10b). However it is important to note that the approximated model wins on 9 datasets among 30. Therefore, knowing a prior about possible shapelet lengths can be used to train the model faster and without losing accuracy. Furthermore, ensembling approximated SAST models, each one focusing on different shapelet lengths is a possible way to improve accuracy while decreasing the computation time. In fact, SASTEN-Ridge-A is more accurate than SAST-Ridge on 20 datasets and less accurate on 18 (Figure 3.10c).



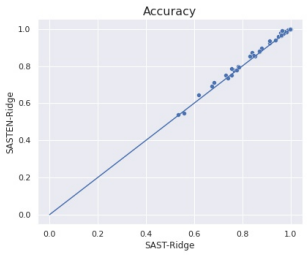
(a) SAST-Ridge (25 wins) vs SAST-RF (12 wins), 2 draws



(b) SAST-Ridge (29 wins) vs SAST-Ridge-A (9 wins), 1 draw



(c) SAST-Ridge (18 wins) vs SASTEN-Ridge-A (20 wins), 1 draw



(d) SAST-Ridge (4 wins) vs SASTEN-Ridge (33 wins), 2 draws

Figure 3.10: Pairwise comparison of model accuracies

Figure 3.10d reveals that ensembling SAST-Ridge models improves accuracy on almost every dataset. But the improvement is slight, because even though the

reference time series are chosen randomly, SAST-Ridge has very low variance in accuracy over multiple runs. We think that this capability comes from the fact that the model uses only one variant of each shapelet to learn the decision boundaries.

The critical difference diagram between SAST models is shown in Figure 3.11. SASTEN-Ridge is the best of all, follows by SASTEN-Ridge-A which is not significantly less accurate. SAST-Ridge is the third best model and is not significantly worse than SASTEN-Ridge-A, but is considerably less accurate than SASTEN-Ridge. SAST-Ridge-A and SAST-RF are significantly less accurate.

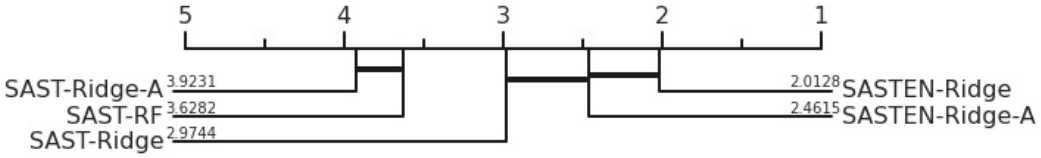


Figure 3.11: Critical difference diagram between SAST models

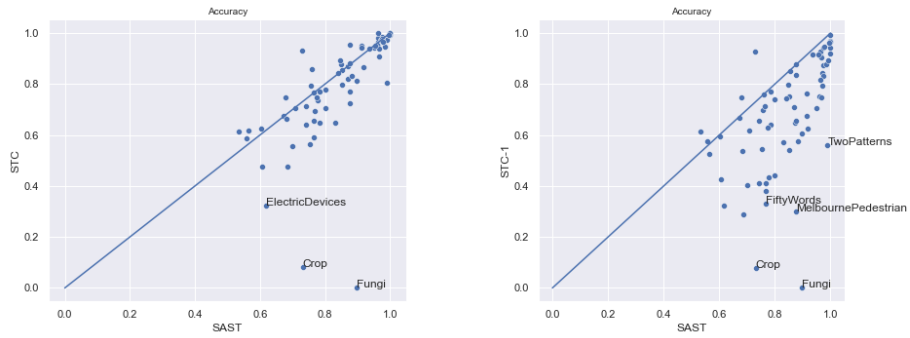
Although SASTEN-Ridge and SASTEN-Ridge-A are more accurate than SAST-Ridge, we believe that the accuracy gain does not worth the computation time overhead required by SASTEN-Ridge or the engineering work required to find the appropriate *length_list* to use in SASTEN-Ridge-A. Therefore, in the rest of this work , we consider only the SAST-Ridge model and call it SAST for simplicity.

3.3.1.3 SAST vs STC

We now compare SAST (i.e SAST-Ridge) to STC, the state of the art shapelet method to our knowledge. This experiment is performed on the same 72 datasets and a pairwise comparison of SAST, STC and STC-1 is shown of Figure 3.12. SAST is more accurate than STC on 43 datasets, worse on 27 and there are two draws. STC-1 is more accurate than SAST on only 5 datasets among 72, although the only difference between these two models is that the Ridge classifier in SAST is trained using the whole shapelet space while only a subset of the shapelet space is used in STC-1. STC-1, STC and SAST respectively achieve an average accuracy of 0.68 ± 0.21, 0.79 ± 0.20 and 0.84 ± 0.12 on the 72 datasets. The standard deviation of STC and STC-1 models is higher due to the zero score obtained on one dataset (Fungi).

There are datasets on which STC and STC-1 hardly achieve 50% accuracy, while SAST performs significantly better. This is the case for the datasets Crop, ElectricDevices and Fungi. These datasets contain respectively 24, 7 and 18 classes. It is difficult to find a subsequence in these datasets that is present in one class and not in the others. A subsequence is generally shared among multiple classes, and therefore is not highly discriminative in terms of information gain by itself. Subsequences need to be combined in other to differentiate classes, and since all the subsequences are available in SAST, this combination is automatically learned by the classifier. Elsewhere SAST achieves 90% accuracy on the dataset Fungi, while

STC and STC-1 fails to find any shapelet on it. These results confirm our though that pruning shapelet candidates, without taking into account the classifier can lead to very inaccurate classification.



(a) SAST (43 wins) vs STC (27 wins), 2 draws

(b) SAST (66 wins) vs STC-1 (5 wins), 1 draw

Figure 3.12: Pairwise comparison of SAST, STC and STC-1

The critical difference diagram on Figure 3.13 reveals that SAST is generally more accurate that STC, but the difference is not highly significant.

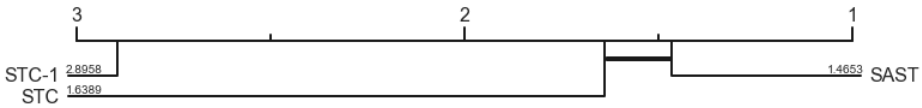
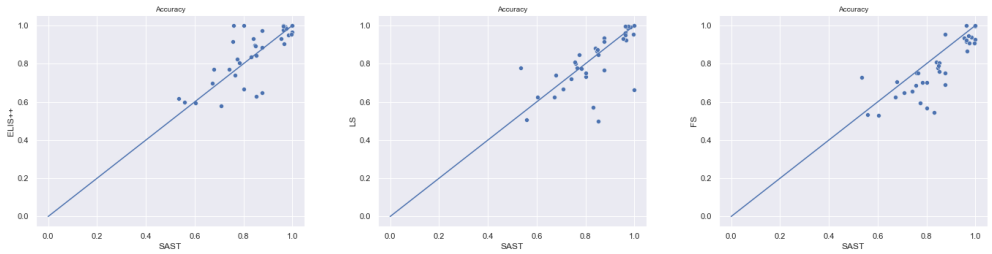


Figure 3.13: Critical difference diagram between SAST, STC and STC-1

3.3.1.4 SAST vs other shapelet methods

We compare our proposal to Fast Shapelet or FS [Rakthanmanon & Keogh 2013]. We also compare our proposal to methods that learn shapelets, namely Learning time series Shapelets or LS [Grabocka *et al.* 2014] and ELIS++ [Zhang *et al.* 2021]. The accuracy of ELIS++, FS and LS are taken from the ELIS++ paper and we considered the same 35 datasets they used (marked with a *plus* sign in Table A.2). The average accuracies of these models on the 35 datasets are 0.78±0.14, 0.81±0.14, 0.83 ± 0.13 and 0.85 ± 0.14 for FS, LS, SAST and ELIS++ respectively.

Figure 3.14 shows a pairwise comparison of these method and the critical different diagram on Figure 3.15 shows how significant is each model compared to others in terms of accuracy. LS, ELIS++ and SAST are not significantly different in terms of accuracy, however they outperform FS. It is important to note that LS and ELIS++ do not select shapelets from the training set, but learn them through an optimization process. Therefore the shapelet space is unlimited, the learned shapelets are unpredictable as well as the time required for convergence. Furthermore, finding the hyper-parameters and the appropriate shapelet initialization for



(a) SAST (12 wins) vs ELIS++ (22 wins), 1 draw

(b) SAST (17 wins) vs LS (16 wins), 2 draws

(c) SAST (30 wins) vs FS (4 wins), 1 draw

Figure 3.14: Pairwise comparison of SAST, ELIS++, LS and FS

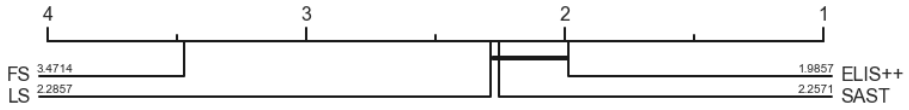


Figure 3.15: Critical difference diagram between SAST, LS, FS and ELIS++

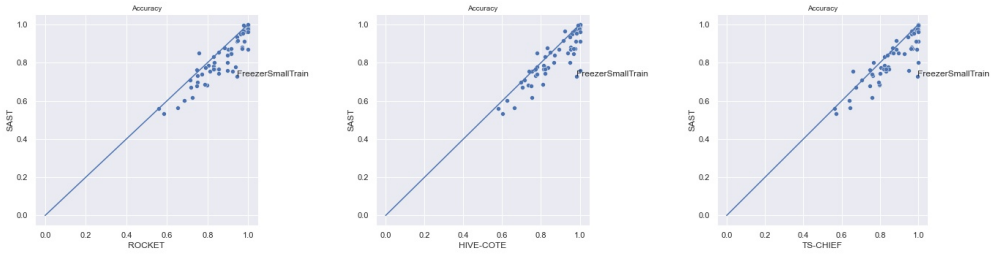
these models is challenging. Without no optimization, SAST can achieve accuracies that are not significantly worse than ELIS++ accuracies and that are slightly better that LS accuracies.

3.3.1.5 SAST vs other types of methods

We also compare SAST to other state of the art algorithms that are not necessarily based on shapelets, nor on one type of features. ROCKET [Dempster *et al.* 2020], HIVE-COTE [Lines *et al.* 2018] and TS-CHIEF [Shifaz *et al.* 2020] are to our knowledge the most accurate methods for time series classification. The results of these models are taken from the UEA & UCR repository [Anthony Bagnall & Keogh 2018]. Among the 72 datasets on which we have SAST results, ROCKET, HIVE-COTE and TS-CHIEF do not have results for the datasets DodgerLoopDay, DodgerLoopGame, DodgerLoopWeekend, Fungi and Melbourne-Pedestrian; so we excluded these 5 datasets from this comparison. Elsewhere, we believe that the comparison we are doing here is not fair since these methods are not based on only shapelet features. However, considering the *no free lunch theorem* [Wolpert & Macready 1997], SAST could outperform these models on some datasets and the goal of this experiment is to see how SAST stands w.r.t to these methods that are based on combination of features.

Although our model uses only shapelet features, it manages to outperform ROCKET on 5 among the 67 with 4 draws (Figure 3.16a). Elsewhere, SAST respectively outperforms HIVE-COTE and TS-CHIEF on 10 and 9 datasets among the 67 with 4 and 3 draws. Since SAST can perform better than HIVE-COTE on some datasets, replacing the shapelet module in HIVE-COTE with a SAST based model

could increase HIVE-COTE accuracy and could reduce its time complexity since the shapelet module is the most time consuming one in HIVE-COTE. When TS-CHIEF was proposed, their authors decided not to exploit shapelet features because of their computation time. With the core shapelet recognition task we introduce in this work, we believe that shapelet feature can be added in TS-CHIEF a low cost and that this could increase the accuracy of this model .



(a) SAST (5 wins) vs ROCKET (58 wins), 4 draws

(b) SAST (10 wins) vs HIVE-COTE (53 wins), 4 draws

(c) SAST (9 wins) vs TS-CHIEF (55 wins), 3 draws

Figure 3.16: SAST vs SOTA

The Wilcoxon statistical test failed to reject the null hypothesis with a p-value of 0.05, meaning that these four models are not significantly different on the considered 67 datasets. In fact, SAST, ROCKET, HIVE-COTE and TS-CHIEF respectively achieve an average accuracy of 0.84 ± 0.12, 0.88 ± 0.11, $.88 \pm 0.11$ and 0.88 ± 0.12. These average scores clearly show that SAST is comparable to ROCKET and HIVE-COTE in terms of accuracy, and in addition SAST is more interpretable as it is a shapelet based method [Ye & Keogh 2009b, Bagnall *et al.* 2017].

3.3.1.6 Model accuracies per dataset type

The datasets on the UEA & UCR archive are categorized in problem types. Among the 72 datasets we have experimented on, there is 1 electric device problem, 4 ECG problems, 1 High Resolution Melt (HRM) problem, 25 image problems, 9 motion recognition problems, 1 power consumption problem, 16 sensor reading problems, 7 simulated dataset problems, 6 spectrograph problems and 2 traffic problems.

We would like to see the method that is more appropriate for each problem type. However, be careful drawing too much conclusions because the number of datasets per problem type is relatively small to be representative. We compute these statistics among three groups of methods as in the previous subsections: the first group is SAST, STC-1 and STC; the second group is SAST, ELIS++, FS and LS; and the last group is SAST, ROCKET, TS-CHIEF and HIVE-COTE. For each group and for each problem type, the percentage of times each method achieves the highest accuracy is computed. These statistics are shown as stacked bar plots with problem types on the x-axis and the number of times the highest accuracy is achieved on the y-axis. Above each bar, the number of datasets in the corresponding problem type

is displayed. Since more than one model can achieve the highest accuracy on the same dataset, summing the percentage in a bar could be greater than 100% and the value above a bar can be less than the bar height.

Figure 3.17 shows the percentage of times SAST, STC-1 and STC achieve the highest accuracy per problem type. STC-1 achieves the highest accuracy on the image dataset MiddlePhalanxOutlineAgeGroup and on the sensor dataset Earthquake. STC is the only method that achieves the highest accuracy for ECG and Power. Elsewhere STC seems more appropriate for simulated datasets. SAST tends to be generally the best choice for electric device, HRM, image, motion recognition, sensor and is always the best for spectrograph problems compared to STC approaches.

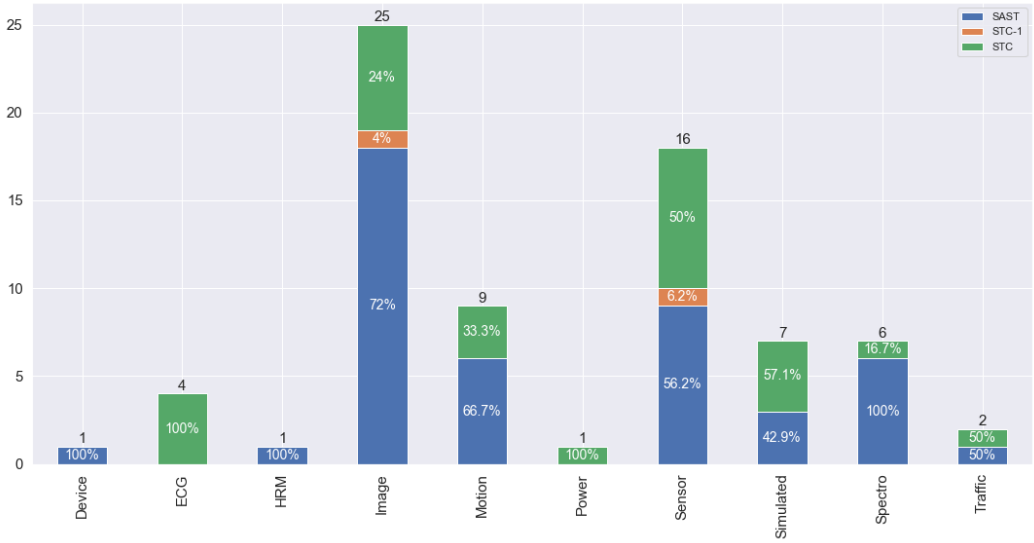


Figure 3.17: SAST, STC-1 and STC percentage of wins per dataset type

When comparing SAST to other shapelet methods (ELIS++, FS and LS), we can see on Figure 3.18 that SAST always achieves the highest accuracy on spectrograph problems and is therefore a good choice for this problem type. Elsewhere it achieves the highest accuracy on more than 25% of image and sensor datasets. ELIS++ is more suitable for ECG, image, motion, and sensor problem types. LS is a good choice for simulated datasets.

Finally, Figure 3.19 reveals that ROCKET, TS-CHIEF and HIVE-COTE win on more datasets than SAST, but with a relatively small difference in accuracy. ROCKET seems to be the most promising method for ECG, motion, sensor, simulated , spectrograph and traffic datasets while HIVE-COTE is a good choice for image and power datasets. TS-CHIEF is a fair option for device.

Although SAST achieves the highest accuracy than ROCKET, HIVE-COTE and TS-CHIEF on some datasets, it sometimes obtains the same average accuracy as these methods. In fact, Table 3.2 gives the mean and standard deviation of each model accuracy per dataset type. We can see that SAST achieves the same average

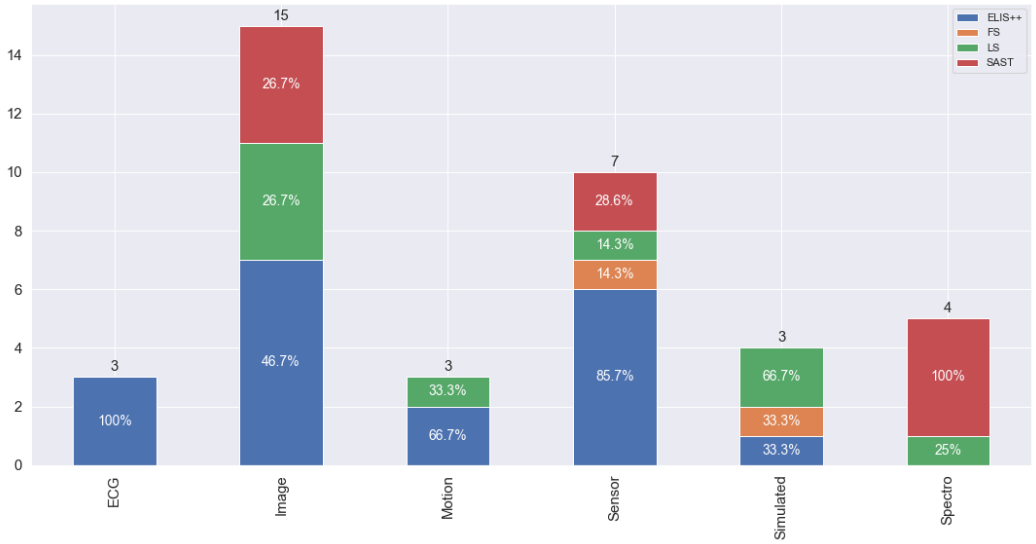


Figure 3.18: SAST, ELIS++, LS and FS percentage of wins per dataset type considering the 35 datasets used in ELIS++ paper.

accuracy as the state of the art methods on spectrograph and is on average relatively closed on many other data types, except device (but there is only one dataset of that type). This results emphasize the fact that SAST can achieve accuracy equal to or closed to the state of the art method accuracy while offering easier interpretability.

3.3.2 Scalability

The scalability of SAST based models and STC is assessed regarding two criteria: the time series length and the number of time series in the dataset. In this experiment, the time contract is not used for STC, and therefore the full search is performed. Elsewhere, the training set and the test set are exactly the same. For each model, the time taken to fit the model on the training set and then predict the test set is recorded.

3.3.2.1 Time series length

Here we use the dataset HouseTwenty from the UEA & UCR repository [Anthony Bagnall & Keogh 2018]. It is a binary dataset of electricity usage in houses. The training set has 34 time series and of length 3000 each. We vary the series length starting at 32 and only the first time steps up to the current length are used to train our models. More precisely, we consider the HouseTwenty dataset with time series truncated at length 2^5, 2^6, 2^7 and finally 2^8. The running time of each model is given in Figure 3.20a.

For each of the four models, the running time increases with the length of time series in the dataset. However, SAST models are much more scalable than STC, and SASTEN-A is the most scalable of all, since it uses a fixed number of shapelet

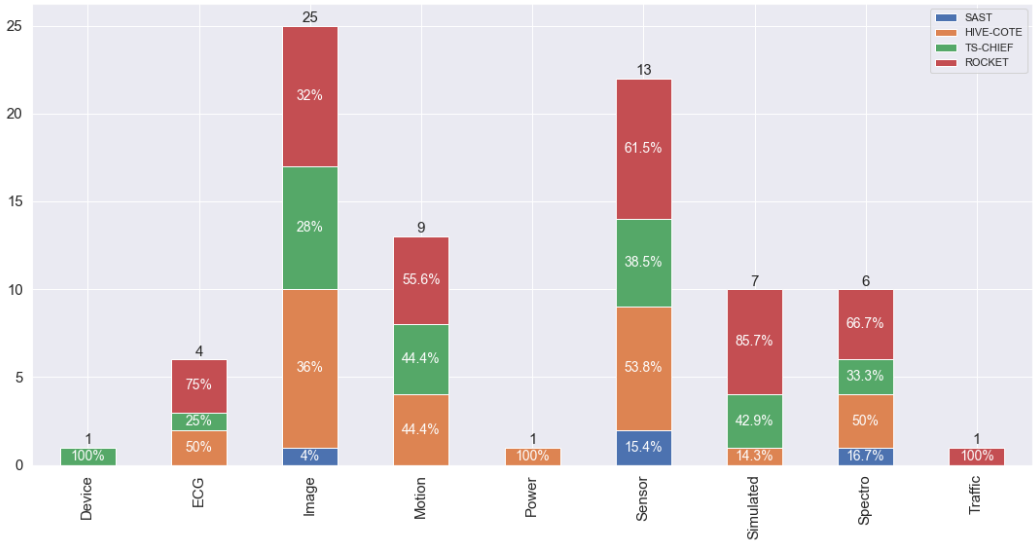
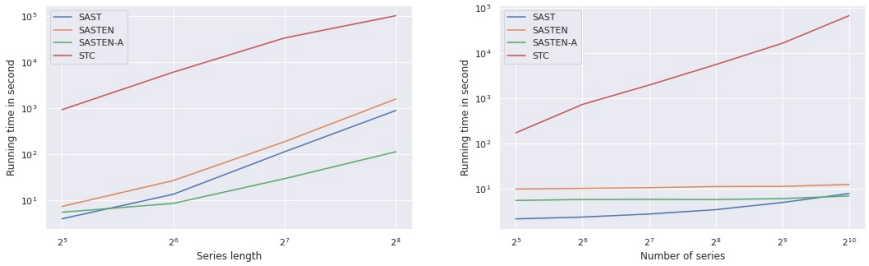


Figure 3.19: SAST, HIVE-COTE, TS-CHIEF and ROCKET percentage of wins per dataset type



(a) Regarding time series length (b) Regarding the number of time series

Figure 3.20: Running time (in second) of each model

candidates whatever the length of time series. For SASTEN-A, increasing the length of the time series only increases the computation time of the similarity between time series and shapelet candidates. More specifically, STC takes about 1 hour and 40 minutes to train on a dataset of 34 time series of length 64, while SAST, SASTEN and SASTEN-A take about 13 seconds, 27 seconds and 8 seconds respectively. For the same number of time series but now of length 256, STC takes a bit more than a day, while SAST, SASTEN and SASTEN-A take about 14 minutes, 26 minutes and 2 minute respectively. Therefore, even our slowest method SASTEN is 55 times faster than STC. SASTEN-A and SAST are respectively 1440 times and 102 times faster than STC.

3.3.2.2 Training set size

The Chinatown dataset is used here. It is a binary dataset with time series of length 24. There are 20 instances in the training set and we use random oversampling to create bigger versions of this dataset. Figure 3.20b shows the running time of each model.

The running time of each model increases nearly linearly with the number of time series in the dataset. STC running time starts higher and increases much faster compared to other models. This is not surprising since the training time of shapelet methods is extremely related to the number of shapelet candidates, and the number of shapelet candidates in STC increases with the number of time series while the number of shapelet candidates in a SAST model increases with the number of classes. More precisely, STC takes about 12 minutes on a dataset of 64 time series of length 24, while SAST takes only 2 seconds, SASTEN requires 10 seconds and SASTEN-A needs about 6 seconds. For a dataset with 1024 time series of length 24, SASTEN, SAST and SASTEN-A are respectively about 5000 times, 8000 times and 9000 times faster than STC.

3.3.3 Interpretability

The predictions of a SAST model trained on a dataset are explained by identifying and visualizing the shapelets that have been learned for that dataset. This is how the explanation of shapelet methods is given in the litterature [Ye & Keogh 2009b, Wang *et al.* 2020]. This is done using feature importance analysis (see Proposition 3.2.2). Each feature is related to a shapelet candidate extracted from a time series whose class label is known. Shapelet candidates related to the most important features are the top best shapelets. We say that any shapelet candidate is from the class of the time series from which it has been extracted. Therefore, the class label of a time series can be interpreted by looking at the class labels of the shapelet candidates to which it is the most similar. Let us interpret the predictions of SAST-RF and SAST-Ridge trained on the Chinatown dataset. We consider this dataset because it has only two classes and time series of length 24, it is therefore easy to visualize this dataset. However, what we are doing here is applicable to any dataset.

Since SAST-RF uses a tree based classifier, information gain is used as feature importance. With SAST-Ridge, the importance of feature is given by the absolute value of the corresponding learned weight. Although feature importance is computed differently for both models, we show that their predictions are interpretable in the same manner.

Figure 3.21 and 3.22 show the top 5 best shapelets plotted on the reference time series for the Chinatown dataset with respect to SAST-Ridge and SAST-RF respectively. The top rows of the figures are the reference time series selected from class 1, while the second rows are the reference time series selected from class 2. A perfect match between a shapelet candidate and a reference time series means that the shapelet has been extracted from that reference time series. Hence, the top 5

best shapelets learned by SAST-Ridge are from class 1. The second of the top 5 best shapelets learned by SAST-RF is from class 1, while the four others are from class 2.

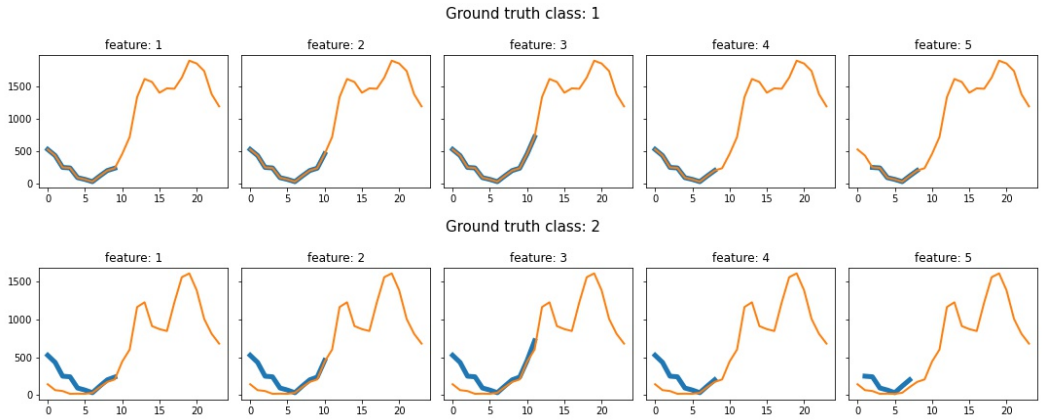


Figure 3.21: Top 5 shapelets learned by SAST-Ridge on Chinatown.

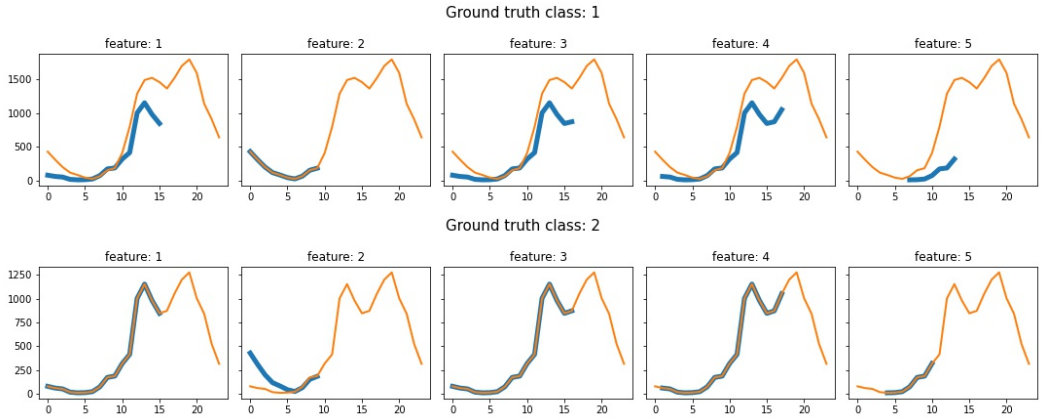


Figure 3.22: Top 5 shapelets learned by SAST-RF on Chinatown.

In order to predict the class label of a test time series, SAST identifies the most important features similar to the time series. In other words, SAST checks if the time series contains subsequences that are similar to the most important features. Figure 3.23 shows the matches between the top 5 most important features learned by SAST-Ridge and two randomly selected test time series. We can note that the model correctly predicts the class labels. Since the top 5 shapelets learned by SAST-Ridge are from class 1, there are near perfect matches with the test instance from class 1 (see Figure 3.23 top). A near perfect match between a subsequence and shapelet candidate means that the subsequence is a variant of that shapelet candidate. No good match is found with the test instance from class 2 (see Figure 3.23 bottom). Therefore, we have an explanation (i.e the most important features that triggered the predicted class label) of why the first instance is predicted as coming from class

1, while the second one is predicted as coming from class 2.

The same analysis is shown for SAST-RF in Figure 3.24. Like SAST-Ridge, SAST-RF also predicted the class labels correctly. The first test time series has a near perfect match with the second top best shapelet candidate (see Figure 3.24 top) which is a shapelet candidate of class 1. The other top best shapelet candidates, which are all from class 2 do not match with the first time series. This explains why the predicted class label for the first time series is class 1 and not class 2. The first, third, fourth and fifth top best shapelet candidates, which are all from class 2 have near perfect matches with the second time series, while the second top best shapelet candidate, which is from class 1 does not match (see Figure 3.24 bottom). Hence, we can interpret why the class label of the second instance is predicted as class 2 and not class 1.

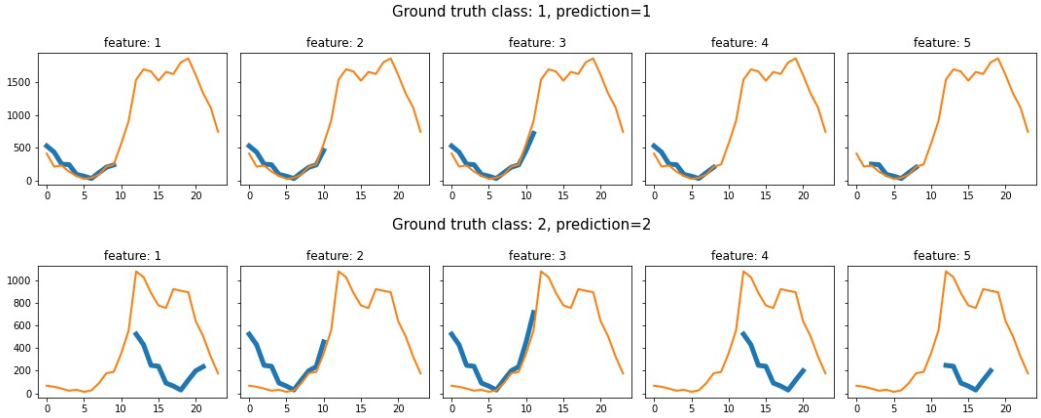


Figure 3.23: Explanation of SAST-Ridge predictions on two random test instances

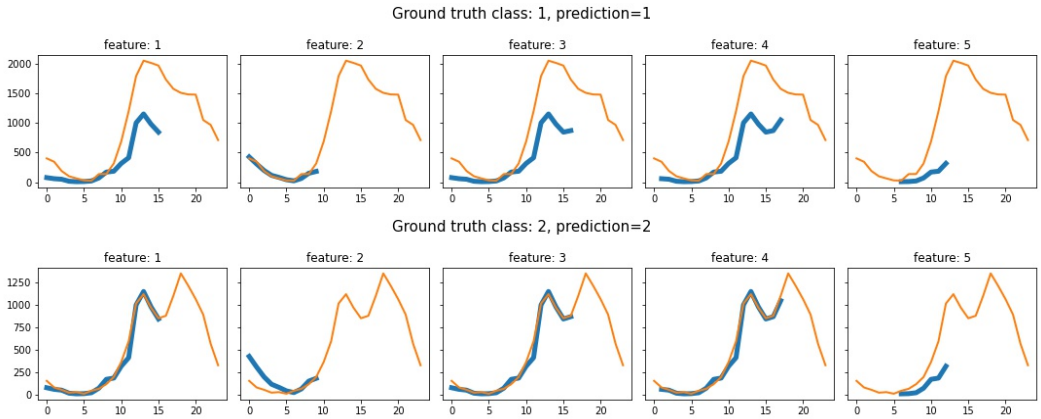


Figure 3.24: Explanation of SAST-RF predictions on two test instances

Therefore, we experimentally proved in this Section that SAST-RF and SAST-Ridge automatically learn to put more attention on the subsequences that are shapelets for any given dataset. More generally a SAST model automatically learns

to put more attention on shapelet candidates that are actually shapelets during its training. It also automatically learns to put less attention on the shapelet candidates that are not shapelets. We also note that the top best shapelets learned by SAST-Ridge and SAST-RF on the Chinatown dataset are the same as the one selected by STC (see Figure 3.4).

3.4 Conclusion

In this work, we shown that the number of shapelet candidates in a shapelet algorithm can be reduced considerably without losing accuracy. We also shown that it is not always necessary to learn shapelets beforehand of classification. We introduced the Scalable and Accurate Subsequence Transform (SAST) algorithm which is interpretable, accurate and a more scalable alternative to the Shapelet Transform algorithm. Furthermore, SAST is comparable in terms of accuracy to the state of the art methods ROCKET, HIVE-COTE and TS-CHIEF, especially for the spectrograph dataset type, while offering easier interpretability. Our experiments revealed that a good trade-off between accuracy and scalability can be found by ensembling different SAST models, each one focusing on different length of shapelet candidates. We have also introduced the *core shapelet recognition* task which consists of learning a shapelets model using only few variants of each shapelet candidate. SAST achieves this task accurately and we hope future shapelet methods will follow the design we introduced.

We plan to do many improvements on the SAST algorithm in the future. Particularly, distance computation could be speed up using lower bounding and early abandon techniques. Different variants of the same shapelet can be present in the same time series, therefore similar subsequences can be pruned in order to further reduce the number of shapelet candidates. We are also planing to explore how core shapelet recognition can be applied in TS-CHIEF in order to take shapelet features into account.

Although we focused on time series classification in this chapter, we hope that the same idea can be used in the near future to increase the scalability of shapelet-based time series clustering [Siyou Fotso *et al.* 2020].

In the next chapter, we will improve SAST by removing duplicate subsequences and extend it to uncertain time series using uncertainty propagation.

Key points

- We introduced the *core shapelet recognition* task, consisting of building a ML model able to recognize any shapelet by seeing one or a few number of its variants.
- We proposed the Scalable and Accurate Subsequence Transform (SAST), a novel design of subsequence-based time series classification approach which is many magnitudes more scalable than shapelet transform while being more accurate.
- We demonstrated SAST's effectiveness on respectively 72 and 8 state-of-the-art datasets and methods.
- We demonstrated that SAST is an interpretable-by-design method.

Communications

- Michael F. Mbouopda, Engelbert Mephu Nguifo. *Scalable and Accurate Subsequence Transform for Time Series Classification.* In Conférence sur l'Apprentissage automatique (CAp), pp 1-27, Jun. 2021.
- Michael F. Mbouopda, Engelbert Mephu Nguifo. *Scalable and Accurate Subsequence Transform for Time Series Classification.* Pattern Recognition, pp 1-35, submitted in May 2021.

Algorithm 3: ShapeletTransformK

Input: $D = \{(T_1, c_1), (T_2, c_2), ..., (T_n, c_n)\}$: the training dataset, k: the number of instances to use per class: the list of subsequence lengths, C: the classifier to use, *length_list*: the list of subsequence lengths, *min_ig*: the minimum information gain to consider a subsequence as shapelet

```
1  begin
       /* randomly select k instances per class from the dataset  */
2      D_c ← randomlySelectInstancesPerClass(D, k)
       /* generate every subsequence of length in length_list from D_c
          */
3      S ← generateShapeletCandidates(D_c, length_list)
       /* compute the information gain of each subsequence and
          return the one that have at least the required
          information gain                                        */
4      S ← extractShapelet(S, D, min_ig)
       /* transformed the dataset using every patterns in S       */
5      D_f ← ∅
6      for i ← 1 to n do
7          x_i ← []
8          for j ← 1 to |S| do
9              x_i[j] ← dist(T_i, S_j)
10         end
11         D_f ← D_f ∪ {(x_i, c_i)}
12     end
       /* train the classifier on the transformed dataset         */
13     clf ← trainClassifier(C, D_f)
14     return (clf, S) ;     // the trained classifier and the shapelet
        candidates
15 end
```

Algorithm 4: ScalableAndAccurateSubsequenceTransform

Input: $D = \{(T_1, c_1), (T_2, c_2), ..., (T_n, c_n)\}$, k: the number of instances to use per class, *length_list*: the list of subsequence lengths, C: the classifier to use

begin
  /* randomly select k instances per class from the dataset */
  $D_c \leftarrow randomlySelectInstancesPerClass(D, k)$
  /* generate every patterns of length in *length_list* from D_c */
  $S \leftarrow generateShapeletCandidates(D_c, length\_list)$
  /* transformed the dataset using every patterns in S */
  $D_f \leftarrow \emptyset$
  for $i \leftarrow 1$ **to** n **do**
    $x_i \leftarrow []$
    for $j \leftarrow 1$ **to** $|S|$ **do**
      $x_i[j] \leftarrow dist(T_i, S_j)$
    end
    $D_f \leftarrow D_f \cup \{(x_i, c_i)\}$
  end
  /* train the classifier on the transformed dataset */
  $clf \leftarrow trainClassifier(C, D_f)$
  return *(clf, S)* ; // the trained classifier and the shapelet candidates
end

Table 3.1: List of models used in our experiments

Name	**classifier**	**length_list**	**Description**
STC-1	Ridge classifier	$\{3, 4, .., m\}$	STC-k with $k = 1$, meaning that shapelets are selected from a randomly selected time series per class
STC-0.25	Ridge classifier	$\{3, 4, .., m\}$	STC-k that select shapelets from 25% of time series per each class randomly selected
STC-0.5	Ridge classifier	$\{3, 4, .., m\}$	STC-k that select shapelets from 50% of time series per each class randomly selected
STC-0.75	Ridge classifier	$\{3, 4, .., m\}$	STC-k that select shapelets from 75% of time series per each class randomly selected
STC	Ridge classifier	$\{3, 4, .., m\}$	STC-k that select shapelets from every time series in the dataset
SAST-RF	Random Forest	$\{3, 4, .., m\}$	SAST model using Random Forest classifier
SAST-Ridge	Ridge classifier	$\{3, 4, .., m\}$	SAST model using Ridge classifier with LOO
SAST-Ridge-A	Ridge classifier	$\{9, 13, 15\}$, $\{7, 11, 15\}$, $\{7, 9, 15\}$ or $\{9, 11, 15\}$	Approximated SAST-Ridge, that is a SAST-Ridge which considers only some subsequence lengths
SASTEN-Ridge	Ridge classifier	-	Ensemble of 3 SAST-Ridge
SASTEN-Ridge-A	Ridge classifier	-	Ensemble of 3 Approximated SAST-Ridge with *length_list* $\{3, 4, .., 9\}$, $\{10, 11, ..., 16\}$, and $\{17, 18, ..., 23\}$ respectively

Table 3.2: Average accuracy of each model per problem type

	HIVE-COTE	ROCKET	SAST	TS-CHIEF	Number of datasets
Device	0.75 ± 0.0	0.73 ± 0.0	0.62 ± 0.0	$\mathbf{0.76 \pm 0.0}$	1
ECG	0.95 ± 0.06	$\mathbf{0.96 \pm 0.05}$	0.93 ± 0.07	0.94 ± 0.07	4
Image	0.82 ± 0.12	0.82 ± 0.12	0.78 ± 0.12	$\mathbf{0.83 \pm 0.12}$	25
Motion	$\mathbf{0.93 \pm 0.09}$	$\mathbf{0.93 \pm 0.07}$	0.88 ± 0.1	$\mathbf{0.93 \pm 0.09}$	9
Power	$\mathbf{1.0 \pm 0.0}$	0.94 ± 0.0	0.91 ± 0.0	0.99 ± 0.0	1
Sensor	0.89 ± 0.12	$\mathbf{0.9 \pm 0.11}$	0.85 ± 0.14	0.89 ± 0.13	13
Simulated	0.99 ± 0.02	$\mathbf{1.0 \pm 0.01}$	0.95 ± 0.04	$\mathbf{1.0 \pm 0.01}$	7
Spectro	$\mathbf{0.87 \pm 0.11}$	$\mathbf{0.87 \pm 0.12}$	$\mathbf{0.87 \pm 0.11}$	$\mathbf{0.87 \pm 0.11}$	6
Traffic	$\mathbf{0.98 \pm 0.0}$	$\mathbf{0.98 \pm 0.0}$	0.96 ± 0.0	0.97 ± 0.0	1
Average	0.88 ± 0.11	0.88 ± 0.11	0.84 ± 0.12	0.88 ± 0.12	67

Chapter 4

Explainable Classification of Astronomical Uncertain Time Series

Exploring the expansion history of the universe, understanding its evolutionary stages, and predicting its future evolution are important goals in astrophysics. Today, machine learning tools are used to help achieving these goals by analyzing transient sources, which are modeled as uncertain time series. In this chapter, we propose an uncertainty-aware subsequence based model which achieves a classification comparable to that of state-of-the-art methods. Unlike conformal learning which estimates model uncertainty on predictions, our method takes data uncertainty as additional input. Moreover, our approach is explainable-by-design, giving domain experts the ability to inspect the model and explain its predictions. The explainability of the proposed method has also the potential to inspire new developments in theoretical astrophysics modeling by suggesting important subsequences which depict details of light curve shapes.

Contents

4.1 Introduction **75**
4.2 Uncertain Subsequence Transform Classification **77**
4.3 Experiment **79**
4.3.1 The PLAsTiCC dataset 79
4.3.2 Results et discussion 80
4.4 Conclusion and future directions **87**

4.1 Introduction

Uncertain time series are preponderant in transient astrophysics. Astronomical objects whose brightness varies with time (a.k.a transients) are primarily characterized by the presence or absence of specific chemical elements found in their spectra. This data taking process (called spectroscopy) is very time-consuming and requires very good observation conditions to be performed. Moreover, since transients are objects

which appear in the sky for a limited period of time then disappear forever, there is a small time-window of opportunities when such measurements can be taken.

Alternatively, we can also associate different classes of astronomical transients to the respective shape of their light curves (brightness variation as a function of time). In this case, we need to repeatedly measure the brightness of the source in a relatively broad region of the wavelength spectrum. This process, called photometry, is less expensive and imposes more manageable constraints on observation conditions. However, measurements are more prone to uncertainties (due to moonlight, twilight, clouds, etc) in the flux determination and the distinction between light curves from different classes is subtle, resulting in less accurate classification. Nevertheless, since there is not enough spectroscopic resources to provide definite label for all photometric observed objects, being able to effectively analyze uncertain photometric light curves means that a wider range of the universe can be quickly understood and at a lower cost.

The Vera C. Rubin Observatory[1] is a ground-based observatory, currently under construction in Chile, whose goal is to conduct the 10-year Legacy Survey of Space and Time (LSST) in order to produce the deepest and widest images of the universe. The observatory is expected to start producing data in early 2024, and in order to prepare the community for the arrival of its data, one important data challenge was put in place: the Photometric LSST Astronomical Time-Series Classification Challenge or simply PLAsTiCC [Allam Jr *et al.* 2018]. The goal was to identify machine learning models able to classify 14 types of transients in simulated data, represented by uncertain time series, or light curves. The ultimate goal behind the challenge was to understand which methods are expected to perform better in LSST-like data, thus preparing the community to the arrival of its data and help understanding the universe's expansion history. Therefore, using interpretable approaches was very important. However, contributors focused on minimizing the classification loss by employing techniques such as mixture of classifiers and data augmentation [Hložek *et al.* 2020] while neglecting explainability. In this chapter, we address this problem with explainability in mind.

We consider two approaches to classify uTS in an explainable manner : the first one ignores uncertainty and uses only the best estimates, while the second one takes uncertainty into account. Ignoring uncertainty makes the task a *regular* time series classification task, allowing the usage of Shapelet Transform Classification or simply STC [Hills *et al.* 2014], an effective and explainable *regular* time series classification algorithm. This model failed to find a valid shapelet on PLAsTiCC, and therefore could not perform the classification task. We performed extensive hyper-parameter tuning tests, but the result was the same. We also tried to take uncertainty into account by using the Uncertain Shapelet Transform algorithm, but as expected, this method also failed since it is an extension of STC for uncertain time series.

In this chapter, we propose the Uncertain Scalable and Accurate Subsequence Transform (or uSAST for short) method which is able to achieve an F1-score of

[1] https://lsst.org/

70% while providing faithful explanation similarly to STC. The rest of this chapter is organized as follows: we start by describing the uSAST method (Section 4.2). Then, we detail our experiments and the obtained results Section 4.3 before concluding this Chapter in Section 4.4.

4.2 Uncertain Subsequence Transform Classification

In this section, we describe a new uncertain time series classification method based on uncertainty propagation as in UST and subsequence transform as in SAST. In fact, uncertainty propagation is an effective approach to analyze uncertain data [Gruber *et al.* 2020, Liu *et al.* 2021] and particularly uncertain time series. By using a single random instance from each class, SAST is more scalable and at least as accurate as STC while keeping STC interpretability capabilities.

Given a time series dataset, SAST follows four steps: *i)*, one instance is randomly selected from each class: these are called reference time series; *ii)* a set containing every subsequences from the selected time series is created; *iii)* each instance in the dataset is replaced by the vector of its distances to each subsequence obtained in the second step; *iv)* a supervised classifier is trained on the transform dataset.

Performing classification following the SAST steps could be inefficient because of the redundancy in the set of subsequences obtained at the second step. The redundancy is particularly high for small length subsequences and in datasets such as electrocardiogram (ECG) and PLAsTiCC, in which repetitive patterns occur very often. Furthermore, the third step is based on the application of Definition 2.3 using the Euclidean distance and, therefore, only the most similar subsequence is considered; however, taking into account the number of occurrences of the best match is important in some contexts. To overcome these limitations, we define the notion of ε-*similarity* as follows:

Definition 4.1 (ε-similarity). *Two subsequences (respectively uncertain subsequences) S_1 and S_2 of same length l are ε-similar if the distance between them is less than or equal to a user-defined threshold $\varepsilon \geq 0$.*

$$\varepsilon\text{-}similar(S_1, S_2) = \begin{cases} True, & \text{if } dist(S_1, S_2) \leq \varepsilon \\ False, & otherwise \end{cases}$$

Theorem 4.2.1. *The ε-similar relationship is not transitive.*

Proof. Let X, Y, and Z be three subsequences of same length l such that ε-similar$(X, Y) = True$ and ε-similar$(Y, Z) = True$. Let us assume that the transitivity property is verified, that is ε-similar$(X, Z) = True$. A counterexample is built by considering X, Y, and Z as points in a high dimensional space ($\mathbb{R}^l$) such that $dist(X, Y) = dist(Y, Z) = \varepsilon$, and $XY \perp XZ$. The following derivation proves the theorem:

$$\begin{aligned} dist(X,Z) &= \sqrt{dist(X,Y)^2 + dist(Y,Z)^2} \\ &= \sqrt{\varepsilon^2 + \varepsilon^2} \\ &= \varepsilon\sqrt{2} \\ &> \varepsilon \\ &\implies \varepsilon\text{-similar}(X,Z) = False \end{aligned}$$

□

Using Definition 4.1, we can reduce redundancies and count subsequence frequencies in SAST. The updated SAST method, hereafter SAST+, is detailed in Algorithm 5.

Algorithm 5: SAST+

Input: $D = \{(T_1, c_1), (T_2, c_2), ..., (T_n, c_n)\}$, k: the number of instances to use per class, $length\_list$: the list of subsequence lengths, C: the classifier to use, ε: ε-similarity parameter.

```
begin Randomly select k instances per class from the dataset
    D_c ← randomlySelectInstancesPerClass(D, k)
    /* Generate every patterns of length in length_list from D_c, using
       ε to remove similar patterns                                      */
    S ← generateSubsequences(D_c, length_list, ε)
    D_f ← ∅
    for i ← 1 to n do
        /* Transformed the dataset using every patterns in S            */
        x_i ← []
        for j ← 1 to |S| do
            /* The procedure distAndCount(T_i, S_j, ε) returns Dist(T_i, S_j)
               and the number of occurrences of the subsequence S_j in
               T_i                                                       */
            x_i[j], x_i[j + |S|] ← distAndCount(T_i, S_j, ε)
        end
        D_f ← D_f ∪ {(x_i, c_i)}
    end
    clf ← trainClassifier(C, D_f) /* Train the classifier on the
       transformed dataset                                               */
    Result: (clf, S) /* The trained classifier and the subsequences     */
end
```

The time complexity of the SAST method is $O(N_c) + O(kN_cm^2) + O(nm^3) + O(classifier)$, where N_c is the number of classes, n the number of time series, m the length of the time series and k the number of reference time series per class. In practice, it is not necessary to have k greater than one. Removing re-

dundancies in SAST is done only once (during the training phase) with a theoretical time complexity of $O(km^4)$; counting frequencies is done while computing the distance in a constant time. Therefore, the SAST+ time complexity is $O(N_c) + O(kN_cm^2) + O(nm^3) + O(classifier) + O(km^4)$ which is asymptotically equivalent to $O(classifier)+O(km^4)$. Removing redundancies makes SAST+ much faster than SAST during inference.

Similarly to the Uncertain Shapelet Transform, the uncertain SAST+ (uSAST+) is obtained by using UED as the distance metric in Algorithm 5; allowing uncertainties to be propagated to the classifier which then uses these uncertainties to learn robust decision boundaries.

4.3 Experiment

4.3.1 The PLAsTiCC dataset

As far as we know, existing methods published on uTS classification have never been evaluated on real uncertain time series datasets, but solely on simulated datasets. The corresponding simulated datasets have never been made publicly accessible neither for reproducibility reasons, nor for facilitating research on uTS. In this work, we evaluate our method on a realistic publicly available uncertain time series dataset from the astrophysics domain.

The Photometric LSST Astronomical Time-Series Classification Challenge (PLAsTiCC) dataset contains uncertain time series representing the brightness evolution of astronomical transients including supernovae, kilonovae, active galactic nuclei and eclipsing binary systems [Allam Jr *et al.* 2018], among others. The uncertainty in this dataset is modeled by the probability density-based model. Therefore, for each measurement, the astrophysicists provides a best estimate and the maximal possible deviation from that estimate. Each object is represented as a multivariate uncertain time series of 6 dimensions named *u, g, r, i, z, y*, each corresponding to a particular broadband wavelength filter. After the challenge was finished, the organizers made available an updated version of the data through Zenodo [PLASTICC Team and PLASTICC Modelers 2019] with some bug fixes and the classification answers for both the training and test sets. In this work, we demonstrate our method using only uncertain time series from the training set, but the methodology is general enough to be extended to the test set. There are 7848 transients in the dataset, grouped in 14 different classes, and the number of objects in the classes are highly imbalanced. More specifically, the most underpopulated class has only 0.3% of objects, whereas the most populated one contains 29% of the objects. Furthermore, the dataset contains a lot of missing observations. We handled this with the help of astrophysicists who suggested to fill missing data using a rolling average with a window of length 5. Missing values and corresponding error bars are replaced by the mean and standard deviation of the window. This procedure translated the original dataset into a homogeneously sampled uncertain time series.

The preprocessed dataset is made public[2].

Our implementation uses the Python programming language and is based on the Scikit-learn machine learning library [Pedregosa *et al.* 2011] and the Sktime time series dedicated machine learning library [Löning *et al.* 2019]. The experiment is run on a computing node equipped with 1 Gb of RAM and an AMD EPIC 7452 processor containing 64 logical cores of 2.35 GHz frequency. The source code of our experiments and all the results we discuss in this chapter are publicly available on GitHub[3].

4.3.2 Results et discussion

Since PLAsTiCC is a multivariate uncertain time series dataset, the subsequence transformation is performed on each dimension independently. The transformations from each dimension are then concatenated together to build a large matrix which is subsequently fed to the supervised classifier. We used 80% of the data for training and the remaining is used for testing.

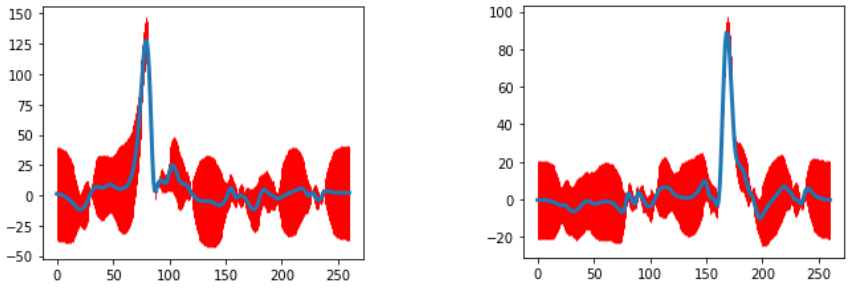
4.3.2.1 Shapelet-based methods results:

Shapelet-based classification is a special case of subsequence-based classification which consider only shapelets as relevant subsequences. We considered two shapelet-based methods STC [Hills *et al.* 2014] and UST for their interpretability. For both methods, we kept every parameters to their default values except the *minimum information gain* parameter which is the threshold used to decide if a separator is a valid shapelet. We tried different values for this parameter without success, none of these methods were able to find a single valid shapelet in the dataset. Since feature extraction was not successful, classification was not possible. This result is due to the dataset being highly imbalanced and the uncertain time series from different classes being too similar in shape. The same dimension of two randomly selected samples from two different classes is shown on Figure 4.1. The left figure which is a Supernova Type Ia-x (SNIax) looks like a left-shifted version of the right figure which is a Supernova Type Ia-91bg (SNIa-91bg). SNIax and SNIa-91bg are known to be difficult to distinguish by astrophysicists. This observation holds, with different magnitude, for other classes in the PLAsTiCC dataset and therefore, any shapelet-based methods might struggle to find shapelets in this dataset.

4.3.2.2 SAST-based methods results:

For this experiment we considered different SAST+ configurations in order to measure the effect of taking uncertainty into account, dropping duplicates and counting the number of occurrences of patterns (i.e patterns frequency). We named configurations that ignore uncertainty as SAST<X> and those which take uncertainty into account as uSAST<X>, where <X> is either : *i)* an empty string to specify

[2]Cleaned dataset: https://drive.uca.fr/f/f0741be3fb77402f8e82/
[3]Source code: https://anonymous.4open.science/r/usast-FBC0/



(a) Supernova Type Ia-x (b) Supernova Type Ia-91bg

Figure 4.1: Two supernova from PLAsTiCC. They look similar in terms of shapes although they are from distinct classes.

that duplicate subsequences are not removed and the patterns frequency is ignored; *ii)* the character *d*, meaning that duplicate patterns are removed; *iii)* the string *dc*, meaning that duplicate patterns are removed and the frequency of patterns is taken into account.

We use three different supervised classifiers, namely Random Forest (RF), eXtreme Gradient Boosting (XGBoost) and the Ridge regression with Leave-One-Out cross-validation (RidgeCV). The cross-validation procedure is used to find the best regularization parameter. We set the minimum and maximum subsequence lengths to 20 and 60 respectively, with a step of 10. Compared to a step of 1, a step of 10 reduces the chance of having similar subsequences while reducing the number of subsequences to be used. We observed that the classification performance is better with this setup as can be seen in Appendix B. The ε-similarity is computed with $\varepsilon = 0.25$ as experiments shown that too much relevant subsequences are discarded with higher values. The parameters of the classifiers are left to their default values, except for the regularization parameter in RidgeCV which is selected using cross-validation. As the reference time series are chosen randomly, we run each experiment 3 times and we report the average precision, recall, F1 score, cross entropy loss and the time taken for training and inference (in hours). As PLAsTiCC is an imbalanced multiclass dataset, we use a weighted average to compute the precision, recall and F1 score; the weights being the percentage of each class in the dataset. Table 4.1 shows the result using the XGBoost classifier only as it has led to the best classification performance. However, detailed results are available in Appendix B.

Table 4.1: Results on PLAsTiCC averaged over 3 runs.

	Precision	Recall	F1 score	LogLoss	Time (h)
uSAST	0.72 ± 0.01	0.72 ± 0.00	0.69 ± 0.01	0.96 ± 0.01	51.03 ± 0.12
uSASTd	$\mathbf{0.72 \pm 0.00}$	$\mathbf{0.73 \pm 0.00}$	$\mathbf{0.70 \pm 0.01}$	$\mathbf{0.97 \pm 0.01}$	$\mathbf{43.49 \pm 0.27}$
uSASTdc	0.71 ± 0.01	0.72 ± 0.01	0.69 ± 0.01	0.96 ± 0.01	43.52 ± 0.72

The first observation is that any variant of our proposed method is able to achieve

around 70% precision, recall and F1 score, unlike shapelet-based methods which completely failed on the PLAsTiCC dataset. This result corroborates with the claim that pruning subsequences before the effective classification could sometimes lead to poor performance. Dropping duplicates, counting patterns frequency or doing both does not have significant impact on the classification performance. However, dropping duplicate makes the models faster. In particular, uSASTd is about 12 hours faster than uSAST. Counting pattern frequency does not add a computation overhead because it is done while computing the distance in $O(1)$ time.

Choosing the right subsequence lengths to considered is challenging and assessing all possible values is computationally expensive; However, domain knowledge could guide in setting this parameter as it is application-dependent.

PLAsTiCC contains objects that are either galactic or extra-galactic, and whose light curves were obtained following a Deep Drilling Fields (DDF) or Wide Fast Deep (WFD) observation strategy. Extra-galactic objects are further away than galactic ones, they are fainter and more difficult to be observed. DDF light curves contain more frequent observation points than WFD ones. Thus, DDF light curves provide a more certain determination of the time series properties than their WFD counterparts which have more uncertainties. Table 4.2 gives the performances of the model uSASTd regarding if the objects are galactic or not, DDF or WFD. The model is considerably better at classifying galactic objects than extra-galactic ones, and a little better at classifying DDF objects than WFD ones. While the model achieves an F1 score of 94% for galactic objects in DDF, it achieves an F1 score of only 67% for extra-galactic objects in WFD. This is directly related to the astrophysical nature of galactic objects. These are, in general, variables whose brightness go through many cycles within the 3 years covered by our data. On the other hand, extragalactic objects are dominated by transients, consisting of only 1 region of signal which never repeats, thus rendering a smaller quantity of information encoded in its time series.

Table 4.2: uSASTd performance regarding if the object are galactic or extra-galactic, are from the DDF or WFD.

		Galactic	Extra-galactic	Both
DDF	Precision	0.96	0.73	0.77
	Recall	0.94	0.75	0.79
	F1 score	0.94	0.71	0.76
WDF	Precision	0.94	0.67	0.71
	Recall	0.84	0.64	0.71
	F1 score	0.87	0.61	0.67
Both	Precision	0.94	0.68	0.72
	Recall	0.86	0.67	0.73
	F1 score	0.88	0.64	0.70

The data set includes 6 classes with overall similar behavior (42, 52, 62, 67, 90, 95). Among these, astronomers are specially interested in type 90 (SNIa), which is used as distance indicator in cosmological analysis [Ishida 2019]. Reporting our results as a binary problem with class 90 against all others, we achieve 85% precision, 81% recall and 82% F1 score. Therefore, our method is able to correctly classify a

high proportion of SNIa despite its similar behavior to other classes.

4.3.2.3 Ablation study:

Here, we study the impact of taking uncertainty into account. In particular, we compare the results obtained when uncertainty is ignored (Table 4.3) to the results obtained when uncertainty is taken into account (Table 4.1).

Table 4.3: Results on PLAsTiCC averaged over 3 runs when uncertainty is ignored.

	Precision	Recall	F1 score	LogLoss	Time (h)
SAST	0.65 ± 0.01	0.67 ± 0.00	0.63 ± 0.00	1.16 ± 0.01	16.41 ± 0.52
SASTd	0.66 ± 0.02	0.68 ± 0.00	0.64 ± 0.00	1.14 ± 0.00	12.79 ± 0.84
SASTdc	0.66 ± 0.01	0.68 ± 0.00	0.64 ± 0.01	1.14 ± 0.01	12.99 ± 0.30

Taking uncertainty into account increases the classification performance in terms of precision, recall, F1 score and cross entropy loss. In fact, from SASTd to uSASTd, there is a gain of 6% in precision, 5% in recall, 6% in F1 score. It can also be seen that the model is more confident on its predictions as the loss has decreased. However, this gain in performance requires almost four times more computation.

4.3.2.4 Comparison to SOTA:

In this subsection, we compare our proposed method to the state-of-the-art multivariate time series classification methods ROCKET [Dempster *et al.* 2020], MUSE [Schäfer & Leser 2017b] and XEM [Fauvel *et al.* 2022] which have been shown to be among the most accurate methods for this task [Ruiz *et al.* 2021]. Results are shown in Table 4.4.

Table 4.4: uSASTd vs SOTA results.

	Precision	Recall	F1 score	Time (h)
uSASTd	0.72 ± 0.00	0.73 ± 0.00	0.70 ± 0.01	43.49 ± 0.27
MUSE	0.71 ± 0.01	0.73 ± 0.01	0.71 ± 0.01	3.36 ± 0.04
ROCKET	0.77 ± 0.00	0.77 ± 0.00	0.75 ± 0.00	0.05 ± 00
XEM	0.69 ± 0.01	0.71 ± 0.00	0.69 ± 0.00	12.24 ± 0.46

The classification performance of our method is comparable to those of the SOTA methods. In particular, uSASTd achieves better precision, recall and F1 score compared to XEM on PLAsTiCC. uSASTd and MUSE have similar classification performance. ROCKET achieves the best classification performance. SOTA methods are faster than our proposal. Except for XEM which is explainable-by-design, SOTA

methods are not explainable. In fact, ROCKET uses the proportion of positive values obtained after applying random convolutions. MUSE uses bag of words obtained after applying some transformations to the time series. These features have no particular meaning for domain experts. Our method does not have this limitation, as it is based on features that are intelligible to domain experts.

4.3.2.5 Explainability:

One of the best properties of subsequence-based classification is its interpretability. The explanation could be done either locally, when it concerns only a single instance, or globally when it concerns the whole model. In any case, this is generally done by inspecting the model in order to extract the most discriminative subsequences [Ye & Keogh 2009a]. These subsequences could also be found using a post-hoc method such as LIME [Ribeiro *et al.* 2016] or SHAP [Lundberg & Lee 2017], but since our approach is explainable-by-design, inspecting the model is sufficient. More specifically, since the classifier used in our model is tree-based, the information gain can be used as a measure of the discriminative power of the subsequences similarly to what is done in shapelet-based methods. The local explainability of our method is obtained by inspecting the subsequence on which the model focused the most in order to make the prediction for a single instance, these are the subsequences which led to the highest information gain (see Definitions 2.4 and 2.5). Figure 4.2 shows local explanations for a Supernova Type Ia (SNIa) and a Core-collapse Supernova Type II-P (SNII-P) correctly classified by the model. The "P" in the denomination of the latter references the plateau phase observed in its time-series just after maximum brightness. This feature is clearly shown in the bottom panel of Figure (4.2). This confirms that our model focuses on the relevant regions and dimensions of the time series to make the classification. Being able to correctly learn the dimension's relevance is crucial as the discriminative subsequence may appear only in a subset of the dimensions. Furthermore, the location of the discriminative subsequence may not be the same on every dimension. In PLAsTiCC in fact, depending how far is the object, the light may be visible only on some wavelengths (i.e. dimension). Due to the accelerated expansion of the universe, objects which are further away are also moving with a higher velocity. Thus, there is a Doppler effect in the observed light which shifts it to higher wavelengths. Thus, closer (galactic) objects will generally have higher signals in lower wavelengths than further away (extragalactic) ones. Our method perfectly captures the Doppler effect unlike XEM which cannot identify from which dimensions the discriminative subsequences is located.

A global explanation is obtained by building a subsequence-based profile of each of the class. The top 20 most discriminative subsequences from the uSASTd model are shown in Figure 4.3. Subsequences that are from the same class label are plotted with the same color, its rank, its class label and its type are given at the top of its corresponding plot. The type is either *Value* if the discriminative power comes from the value itself or *Uncertainty* if the discriminative powers comes from the

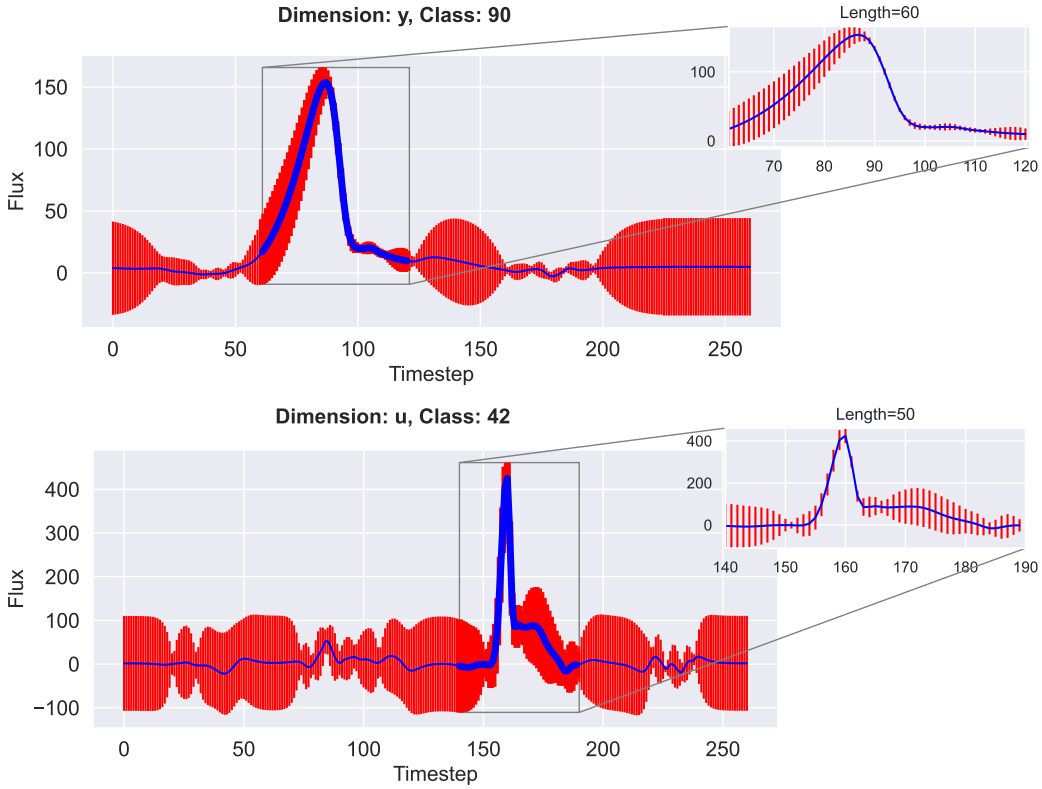


Figure 4.2: Local explainability of a Supernova Type Ia (top) and a Core-collapse Supernova Type II (bottom)

uncertainty. The dimension from which the subsequences are coming from are also given on the figure.

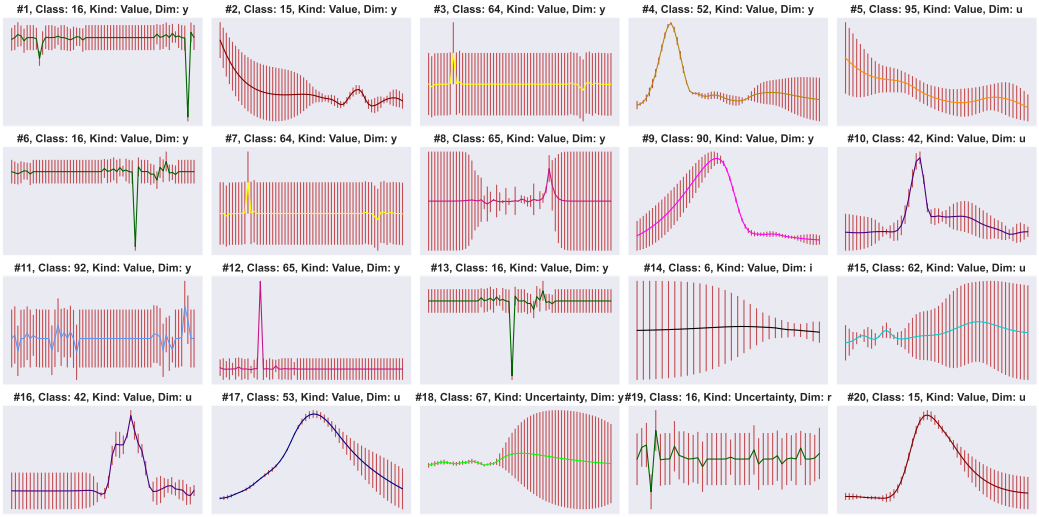


Figure 4.3: The top 20 most discriminative subsequences in the PLAsTiCC dataset

It is observed that the discriminative power is generally due to the value, but sometimes it is due to the uncertainty (for example subsequences #18 and #19). Seeing that some subsequences are important because of their uncertainty emphasizes the fact that taking uncertainty into account is important and improves the classification performance. There are also some subsequences that are too similar despite the fact that duplicate subsequences have been dropped; for instance, the subsequences #3 and #7. This is because the similarity between subsequences is computed using the Uncertain Euclidean Distance (UED) which considers the subsequences to be perfectly aligned. This problem can be resolved by using an elastic distance such as the DTW distance at the cost of more computational time since such distances generally have at least quadratic time complexity while UED is linear. From the domain knowledge point of view, these discriminative subsequences are able to grasp the important shapes commonly associated with their respective class of astronomical transients. Subsequences #1 and #6 were taken from class 16 (eclipsing binary) and clearly show the expected light curve from a well measured binary system where one star eclipses the other exactly in the line of sight, thus leading to a decrease in brightness. Subsequence #19 is also associated to the eclipsing binary class, but in this case the signal is less clear, corresponding to an object which is further away – thus leading to low signal and large uncertainties. We also call attention to the supernova-like behavior exhibited by subsequences #4 and #9 – one single burst events whose brightness are only visible for weeks to months. The fact that such characteristic behaviors are easily spotted in the list of most important subsequences certifies that our final classification results are in line with the expert definition of such classes and hence, shows that our model is safe and trustworthy. Moreover, further investigations of a more extensive list of impor-

tant subsequences have the potential to reveal unexpected time series shapes and promote the development of more detail theoretical models for such astrophysical sources.

4.4 Conclusion and future directions

The classification of time series with available uncertainty measures is an under-explored and challenging task. In this work, we proposed an approach to perform this task with a global F1 score of 70%, without using techniques such as data augmentation nor oversampling. The explainability of the proposed approach allows domain experts to not only understand individual predictions, but also to characterized each class by a set of subsequences with high discriminative power, which can then be used to perform other important tasks in astrophysics such as novel astronomical transients detection and anomaly detection. The ablation study shown the positive impact of taking uncertainty into account. A limitation of the approach is the time complexity, which could be considerably high for datasets with relatively long uncertain time series. A future direction would consist of further reducing the number of subsequences to be used and optimizing the computation time of the method. Another future direction would consist of finding a better way of managing uncertainty during the classification step in order to improve the performances. Nevertheless, the results presented in this work illustrate how our approach is effective in identifying meaningful subsequences which, beyond the classification performance, can provide important information to the expert. The approach is flexible enough to be applied to other scientific domains where uncertain time series are the common, thus enabling future advances in multiple subject areas.

Key points

- We reduced SAST's computational time and proposed the uncertain SAST (uSAST), an extension of SAST to uncertain time series classification.
- We applied uSAST to a realistic uncertain time series dataset and demonstrated its classification effectiveness as well as explainability.

Communications

- Michael F. Mbouopda, Emille E. O. Ishida, Engelbert Mephu Nguifo, Emmanuel Gangler. *Explainable Classification of Astronomical Uncertain Time Series*. HAL preprint, pp 1-8. 2022.

Chapter 5

General conclusion and future directions

In this chapter, we summarize our contributions, discuss the limitations of this work and give some future directions. We also give the list of publications that we did throughout this thesis.

Contents

5.1 Conclusion 89
5.1.1 Main contributions 90
5.1.2 Scientific valorization 90
5.1.3 Open sourced codes and data 90
5.2 Limitations and perspectives 91
5.3 Recommendations 92
5.4 List of publications 92
5.4.1 First-authored publications 92
5.4.2 Non first-authored publications 93

5.1 Conclusion

In this thesis, we addressed the task of classifying uncertain time series. The uncertainty in the data makes this task more challenging than a regular time series classification task. We defined what are uncertainty and uncertain time series, then we shown that the current state-of-the-art on the classification of uncertain time series is limited and difficult to adopt by end-users because of a brutal disappearance of the uncertainty in the classification process. This work advances the state-of-the-art of uncertain time series classification by proposing explainable, robust and efficient methods. Specifically, we proposed a novel general design of uncertain time series classification which propagates uncertainty throughout the whole classification process. The propagation is achieved using uncertainty propagation techniques, widely used in physics.

5.1.1 Main contributions

From the novel design, we derived the Uncertain Shapelet Transform (UST) which is a shapelet-based, hence explainable method for the classification of uncertain time series. UST internally uses the Uncertain Euclidean Distance (UED) that we proposed as a novel similarity measure for uncertain time series with the ability to give not only the similarity, but also the uncertainty on that similarity. We demonstrated the performance of UST on synthetic datasets.

Given the high time complexity of existing shapelet-based methods (including UST), we proposed the Scalable And Accurate Subsequence Transform (SAST), a subsequence-based method to significantly reduce the computation time without losing accuracy, nor interpretability. Concretely, while the state-of-the-art shapelet-based method STC's time complexity is $O(n^2m^4)$, SAST's time complexity is $O(nm^3)$, where n and m are respectively the number and the length of time series. This correspond to a reduction by a factor of nm. We also shown that SAST is competitive with state-of-the-art methods for regular time series classification HIVE-COTE and ROCKET. We have also demonstrated SAST explainability on the Chinatown dataset from the UCR & UEA archive.

Finally we proposed uSAST, an extension of SAST to uncertain time series classification. We assessed uSAST not only on simulated datasets, but also on a real uncertain time series dataset from the astrophysics domain named PLAsTiCC. As far as we know, this is the first open-sourced experiment on real uncertain data. uSAST shown good classification performance on PLAsTiCC as well as great local and global explanations.

5.1.2 Scientific valorization

We valorized and shared our contributions at national and international conferences. In particular, the first UST version has been accepted and presented at the French national conference on artificial intelligence (CNIA 2020) and at the Workshop on Uncertainty in Machine Learning (WUML 2020) which was hosted by the European Conference on Machine Learning (ECML/PKDD 2020). Later on, the final UST version has been presented at the workshop on Large-Scale Industrial Time Series Analysis (LITSA 2021) hosted by the IEEE International Conference on Data Mining (IEEE ICDM 2021).

SAST has been accepted for a long presentation at the french conference on machine learning (CAp 2021) and is currently under review at the Elsevier journal Pattern Recognition. A working paper on uSAST is available on the open archive HAL.

5.1.3 Open sourced codes and data

We open sourced the codes and data used throughout this thesis in order to facilitate research and applications in uncertain time series classification in particular, and in uncertain time series analysis in general. Access links are given below:

- UST: https://github.com/frankl1/ustc
- SAST: https://github.com/frankl1/sast
- uSAST: https://github.com/frankl1/usast

5.2 Limitations and perspectives

We proposed to use uncertainty propagation to take uncertainty into account in uncertain time series classification. We shown that this approach lead to models that are more robust and accurate, but also more natural to users and therefore, more trustable. Both UST and uSAST have three steps: feature selection, feature transformation and classification. We successfully took uncertainty into account in the three steps, but we strongly believe that uncertainty handling could be improved in the classification step. In fact, we used regular supervised classifiers (Random Forest, XGBoost, etc) for this step, these classifiers consider uncertainties as regular features although they should be considered as *meta features*. This step would be more effective if the classifier used was aware of uncertainties and differentiated them from regular features. Few works exist in the literature to achieve that, namely the Decision Tree for Uncertain data [Qin *et al.* 2009, Qin *et al.* 2011]. Therefore, it would be interesting to explore how the work started in this thesis could be improved using uncertain supervised classifiers.

Given that preprocessing generally makes learning easier, it would be legitimate to ask why we did not mention it in this work. In fact, preprocessing is generally application and data-dependent, and therefore not straightforward to be integrated in an end-to-end approach. Instead, we wanted our models to have good performance on the raw data. Nevertheless, it would be a good future direction to see if the performance of our models could be improved by applying some preprocessing of the time series first. One of this techniques could be the uncertain moving average (UMA) and the uncertain exponential moving average (UEMA) [Dallachiesa *et al.* 2012].

In order to ensure the explainability of our methods, we used only the shapelet features, which are acknowledge in the literature for their natural explainability. However, it has been shown in the literature that time series classification is more effective when different features (shapelet, interval, word, etc) are combined together. Hence, another future direction will be to take uncertainty into account in interval, dictionary, spectral, hybrid and deep learning methods for time series classification.

We modeled uncertainty in this work using two values which are the best estimate (or guess) and the standard deviation from that estimate, however, there exist other representations: random sets, possibility distributions, probability intervals, etc [Destercke *et al.* 2008] which have particular properties. It is worthy to analyze how these representation could be used with time series data. For instance, using imprecise probability modeling will ease the adaptation of existing works to time series data [Destercke & Couso 2015, Carranza Alarcon & Destercke 2019].

In this work, we considered the case were input time series were uncertain while the labels were certain. In their current form, our proposed methods cannot be applied in scenarios where the labels are also uncertain. Given that this scenarios is likely to arise in practice, it is important to adapt or extend UST and uSAST to this case. A possible way of tackling this issue is to use belief function theory as proposed in [Quost & Denœux 2009]. Furthermore, it would be interesting to evaluate the confidence of our methods on their predictions using conformal learning techniques. We expect our methods' confidence to be correlated with the uncertainty level, that is, the more uncertain is the data, the less confident our model would be. Combining conformal learning with the explainability of our methods will increase their trustability and adoption by end-users.

The last future direction, but not the least is to explore other time series analysis tasks. In particular, it would be worthy to see how our proposed uncertain similarity measure UED is compared to FOTS for the task of uncertain time series clustering. Beyond that, assessing the performance of uncertainty propagation in the tasks of uncertain time series anomaly detection and forecasting would be good steps to advance the state-of-the-art of uncertain time series analysis in general.

5.3 Recommendations

Uncertain time series classification and more generally the classification of uncertain data is challenging, yet under-explored. We shown in this work that uncertainty is not necessarily a problem, but an additional input that can be used to improve the classification performance and the decision boundaries' robustness. We also hope that uncertainty quantification will be integrated in the data collection process so that the final data is released with the associated uncertainty.

5.4 List of publications

During this thesis, we made many publications on the topic of uncertain time series classification, but also on uncertain time series clustering, anomaly detection in dataset stream, and time series forecasting.

5.4.1 First-authored publications

- Michael F. Mbouopda, Engelbert Mephu Nguifo. *Classification des Séries Temporelles Incertaines Par Transformation Shapelet.* In Conférence Nationale en Intelligence Artificielle (CNIA), pp.14-21, Jun. 2020.

- Michael F. Mbouopda, Engelbert Mephu Nguifo. *Classification of Uncertain Time Series by Propagating Uncertainty in Shapelet Transform.* In ECML/PKDD Workshop on Uncertainty in Machine Learning (WUML), pp.1-12, Sep. 2020.

- Michael F. Mbouopda, and Engelbert Mephu Nguifo. *Uncertain time series classification with shapelet transform.* In International Conference on Data Mining Workshops (ICDMW), pp. 259-266, Nov. 2020.
- Michael F. Mbouopda, Engelbert Mephu Nguifo. *Scalable and Accurate Subsequence Transform for Time Series Classification.* In Conférence sur l'Apprentissage automatique (CAp), pp 1-27, Jun. 2021.
- Michael F. Mbouopda, Engelbert Mephu Nguifo. *Scalable and Accurate Subsequence Transform for Time Series Classification.* HAL preprint. pp 1-35. 2021. <hal-03087686v2> (under review at Pattern Recognitio since May 2021).
- Michael F. Mbouopda, Emille E. O. Ishida, Engelbert Mephu Nguifo, Emmanuel Gangler. *Explainable Classification of Astronomical Uncertain Time Series.* HAL preprint. pp 1-8. 2022. <hal-03790123>
- Michael F. Mbouopda. *Etude de la prédiction du niveau de la nappe phréatique à l'aide de modèles neuronaux convolutif, récurrent et résiduel.* Revue des Nouvelles Technologies de l'Information, Extraction et Gestion des Connaissances (EGC), pp. 1-8, 2022.
- Michael F. Mbouopda, Thomas Guyet, Nicolas Labroche and Abel Heuriot. *Local vs global forecasting methods for groundwater level prediction.* In ECML/PKDD Workshop on Advanced Analytics and Learning on Temporal Data, pp. 1-16, 2022 (accepted for publication).

5.4.2 Non first-authored publications

- Anne M. S. Ngo Bibinbe, **Michael F. Mbouopda**, Gertrude R. Mbiadou Saleu and Engelbert Mephu Nguifo. *A survey on unsupervised learning algorithms for detecting abnormal points in streaming data.* In International Join Conference on Neural Networks (IJCNN), pp 1-8, 2022.
- Michael Dinzinger, **Michael F. Mbouopda** and Engelbert Mephu Nguifo. *Experimental study of similarity measures for clustering uncertain time series. International Federation of Classification Societies (IFCS)*, p 205, 2022 (extended abstract).
- Anne M. S. Ngo Bibinbe, Abdoul J. D. Mahamadou, **Micheal F. Mbouopda**, and Engelbert Mephu Nguifo. *DragStream: An Anomaly And Concept Drift Detector In Univariate Data Streams.* In International Conference on Data Mining Workshop (ICDMW), pp 1-10, 2022 (accepted for publication).

Appendices

Appendix A

Appendix for chapter 2

Contents

A.1 Models performance on each dataset 97
A.2 Scalability of SAST, SASTEN and SASTEN-A regarding the dataset size 99

A.1 Models performance on each dataset

Table A.2 gives the average accuracy obtained by each STC-k models, STC and SAST on the 72 datasets listed in Table A.1. For each dataset, each model is run 5 times and the accuracy mean and standard deviation on the test set are recorded. The last row in the table gives the mean and standard deviation of each method accuracy on the datasets.

Table A.2: Accuracy of models on 72 UEA & UCR datasets (average over 5 runs). The numbers are rounded at 2 decimals.

	STC-1	STC-0.25	STC-0.5	STC-0.75	STC	SAST
1	0.29 ± 0.06	0.32 ± 0.03	0.34 ± 0.04	0.34 ± 0.02	0.48 ± 0.04	$\mathbf{0.68 \pm 0.0}$
2	0.63 ± 0.06	0.71 ± 0.08	0.74 ± 0.03	0.72 ± 0.04	0.75 ± 0.03	$\mathbf{0.77 \pm 0.02}$
3	0.65 ± 0.1	0.68 ± 0.05	0.86 ± 0.05	$\mathbf{0.95 \pm 0.02}$	0.87 ± 0.03	0.87 ± 0.02
4	0.44 ± 0.05	0.47 ± 0.09	0.55 ± 0.09	0.65 ± 0.06	0.71 ± 0.09	$\mathbf{0.8 \pm 0.02}$
5	0.74 ± 0.06	0.79 ± 0.09	0.78 ± 0.09	0.77 ± 0.07	0.78 ± 0.05	$\mathbf{0.8 \pm 0.03}$
6	0.76 ± 0.1	0.76 ± 0.1	$\mathbf{0.91 \pm 0.04}$	0.83 ± 0.14	0.86 ± 0.1	0.76 ± 0.1
7	0.88 ± 0.09	0.95 ± 0.01	0.96 ± 0.01	0.96 ± 0.01	0.95 ± 0.01	$\mathbf{0.98 \pm 0.01}$
8	0.66 ± 0.04	0.75 ± 0.02	0.74 ± 0.02	0.76 ± 0.03	0.77 ± 0.06	$\mathbf{0.88 \pm 0.01}$
9	0.91 ± 0.08	0.95 ± 0.02	0.95 ± 0.03	0.96 ± 0.01	$\mathbf{0.97 \pm 0.01}$	0.96 ± 0.01
10	0.54 ± 0.02	0.54 ± 0.01	0.54 ± 0.01	0.56 ± 0.0	0.56 ± 0.0	$\mathbf{0.75 \pm 0.04}$
11	0.96 ± 0.03	0.99 ± 0.02	0.99 ± 0.03	$\mathbf{1.0 \pm 0.0}$	$\mathbf{1.0 \pm 0.0}$	$\mathbf{1.0 \pm 0.0}$
12	0.38 ± 0.04	0.35 ± 0.03	0.35 ± 0.01	0.38 ± 0.03	0.66 ± 0.02	$\mathbf{0.77 \pm 0.01}$
13	0.41 ± 0.04	0.36 ± 0.04	0.35 ± 0.03	0.42 ± 0.05	0.64 ± 0.02	$\mathbf{0.74 \pm 0.01}$
14	0.41 ± 0.06	0.4 ± 0.06	0.4 ± 0.03	0.38 ± 0.03	0.69 ± 0.01	$\mathbf{0.77 \pm 0.01}$
15	0.08 ± 0.0	0.08 ± 0.0	0.08 ± 0.0	0.08 ± 0.0	0.08 ± 0.0	$\mathbf{0.73 \pm 0.0}$
16	0.9 ± 0.03	0.87 ± 0.06	0.91 ± 0.04	0.93 ± 0.02	0.94 ± 0.04	$\mathbf{0.97 \pm 0.0}$

Continued on next page

Table A.2: Accuracy of models on 72 UEA & UCR datasets (average over 5 runs). The numbers are rounded at 2 decimals.

	STC-1	STC-0.25	STC-0.5	STC-0.75	STC	SAST
17	0.71 ± 0.02	**0.77 ± 0.01**	0.76 ± 0.01	**0.77 ± 0.01**	**0.77 ± 0.01**	0.76 ± 0.02
18	0.66 ± 0.01	0.68 ± 0.02	0.69 ± 0.01	0.69 ± 0.03	0.71 ± 0.01	**0.74 ± 0.01**
19	0.67 ± 0.01	**0.68 ± 0.01**	**0.68 ± 0.01**	**0.68 ± 0.01**	**0.68 ± 0.0**	0.67 ± 0.02
20	0.43 ± 0.03	0.47 ± 0.02	0.54 ± 0.02	0.52 ± 0.04	0.47 ± 0.04	**0.61 ± 0.04**
21	0.61 ± 0.09	0.68 ± 0.11	0.8 ± 0.02	0.77 ± 0.03	0.81 ± 0.03	**0.9 ± 0.02**
22	0.94 ± 0.05	0.97 ± 0.0	0.96 ± 0.0	0.97 ± 0.01	0.97 ± 0.01	**0.98 ± 0.01**
23	0.74 ± 0.07	**0.85 ± 0.02**	0.84 ± 0.01	**0.85 ± 0.01**	0.84 ± 0.0	0.84 ± 0.03
24	0.91 ± 0.01	0.93 ± 0.01	0.93 ± 0.01	0.93 ± 0.01	**0.94 ± 0.0**	**0.94 ± 0.0**
25	0.92 ± 0.05	**1.0 ± 0.0**	**1.0 ± 0.0**	**1.0 ± 0.0**	**1.0 ± 0.0**	**1.0 ± 0.0**
26	**0.75 ± 0.0**	**0.75 ± 0.0**	**0.75 ± 0.0**	**0.75 ± 0.0**	**0.75 ± 0.0**	0.68 ± 0.04
27	0.32 ± 0.04	0.32 ± 0.07	0.31 ± 0.05	0.34 ± 0.05	0.32 ± 0.03	**0.62 ± 0.01**
28	0.43 ± 0.04	0.46 ± 0.03	0.44 ± 0.04	0.47 ± 0.03	0.74 ± 0.01	**0.78 ± 0.01**
29	0.96 ± 0.06	0.95 ± 0.03	0.96 ± 0.06	0.99 ± 0.01	0.99 ± 0.01	**1.0 ± 0.01**
30	0.71 ± 0.04	0.88 ± 0.01	0.92 ± 0.01	0.93 ± 0.01	0.94 ± 0.0	**0.95 ± 0.0**
31	0.33 ± 0.06	0.29 ± 0.03	0.31 ± 0.02	0.31 ± 0.03	0.59 ± 0.01	**0.77 ± 0.0**
32	0.95 ± 0.09	0.97 ± 0.03	0.96 ± 0.03	0.96 ± 0.03	0.97 ± 0.01	**0.98 ± 0.01**
33	**0.93 ± 0.1**	0.91 ± 0.13	0.86 ± 0.17	0.87 ± 0.08	**0.93 ± 0.03**	0.73 ± 0.01
34	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	**0.9 ± 0.0**
35	0.79 ± 0.05	0.94 ± 0.06	**0.97 ± 0.02**	0.96 ± 0.03	**0.97 ± 0.02**	**0.97 ± 0.02**
36	0.84 ± 0.03	0.96 ± 0.01	0.97 ± 0.0	**0.98 ± 0.0**	0.97 ± 0.0	0.97 ± 0.0
37	0.89 ± 0.05	0.97 ± 0.01	0.97 ± 0.01	0.98 ± 0.01	0.97 ± 0.01	**0.99 ± 0.01**
38	0.75 ± 0.03	0.93 ± 0.02	0.95 ± 0.01	**0.96 ± 0.01**	0.95 ± 0.01	**0.96 ± 0.02**
39	0.62 ± 0.06	**0.72 ± 0.02**	**0.72 ± 0.04**	0.7 ± 0.01	0.7 ± 0.02	0.71 ± 0.03
40	0.59 ± 0.07	0.57 ± 0.04	0.58 ± 0.02	0.57 ± 0.06	**0.62 ± 0.03**	0.6 ± 0.04
41	0.53 ± 0.02	0.56 ± 0.01	0.56 ± 0.01	0.54 ± 0.01	**0.62 ± 0.0**	0.56 ± 0.01
42	0.91 ± 0.06	**0.96 ± 0.0**	**0.96 ± 0.01**	**0.96 ± 0.0**	**0.96 ± 0.0**	**0.96 ± 0.01**
43	0.63 ± 0.1	0.84 ± 0.04	0.77 ± 0.02	0.87 ± 0.03	0.87 ± 0.06	**0.92 ± 0.02**
44	0.54 ± 0.02	0.59 ± 0.03	0.56 ± 0.01	0.58 ± 0.02	0.66 ± 0.01	**0.68 ± 0.01**
45	0.3 ± 0.04	0.3 ± 0.07	0.3 ± 0.03	0.28 ± 0.01	0.72 ± 0.02	**0.87 ± 0.0**
46	**0.62 ± 0.02**	0.6 ± 0.01	0.6 ± 0.02	0.61 ± 0.02	0.61 ± 0.02	0.53 ± 0.02
47	0.57 ± 0.0	0.65 ± 0.05	0.66 ± 0.04	0.62 ± 0.04	0.65 ± 0.06	**0.83 ± 0.01**
48	0.57 ± 0.02	0.58 ± 0.02	0.56 ± 0.01	0.58 ± 0.01	**0.59 ± 0.02**	0.56 ± 0.02
49	0.8 ± 0.06	0.79 ± 0.05	0.85 ± 0.02	0.87 ± 0.02	**0.89 ± 0.01**	0.85 ± 0.03
50	0.64 ± 0.01	0.64 ± 0.01	0.64 ± 0.0	0.64 ± 0.0	0.65 ± 0.01	**0.78 ± 0.01**
51	0.97 ± 0.02	0.99 ± 0.0	**1.0 ± 0.0**	**1.0 ± 0.0**	**1.0 ± 0.0**	**1.0 ± 0.0**
52	0.76 ± 0.03	0.93 ± 0.02	0.92 ± 0.02	0.93 ± 0.02	**0.94 ± 0.02**	0.91 ± 0.02
53	0.85 ± 0.01	0.85 ± 0.01	**0.86 ± 0.01**	0.85 ± 0.01	**0.86 ± 0.01**	0.85 ± 0.0
54	0.71 ± 0.03	0.76 ± 0.03	0.75 ± 0.04	0.76 ± 0.01	0.82 ± 0.02	**0.87 ± 0.01**
55	0.77 ± 0.02	**0.78 ± 0.01**	**0.78 ± 0.01**	**0.78 ± 0.0**	0.77 ± 0.01	**0.78 ± 0.01**
56	0.82 ± 0.15	0.92 ± 0.06	0.98 ± 0.01	0.98 ± 0.02	**1.0 ± 0.0**	0.96 ± 0.01
57	0.68 ± 0.05	0.91 ± 0.01	0.94 ± 0.01	**0.95 ± 0.01**	**0.95 ± 0.01**	0.91 ± 0.02
58	0.7 ± 0.14	**0.82 ± 0.03**	0.81 ± 0.02	0.78 ± 0.05	0.79 ± 0.04	0.76 ± 0.05
59	0.75 ± 0.05	0.84 ± 0.03	0.86 ± 0.05	0.83 ± 0.03	**0.88 ± 0.01**	0.85 ± 0.04

Continued on next page

Table A.2: Accuracy of models on 72 UEA & UCR datasets (average over 5 runs). The numbers are rounded at 2 decimals.

	STC-1	STC-0.25	STC-0.5	STC-0.75	STC	SAST
60	0.75 ± 0.05	0.81 ± 0.02	0.82 ± 0.03	0.83 ± 0.02	0.91 ± 0.03	$\mathbf{0.97 \pm 0.01}$
61	0.58 ± 0.05	0.61 ± 0.02	0.59 ± 0.07	0.6 ± 0.09	0.83 ± 0.02	$\mathbf{0.88 \pm 0.02}$
62	0.91 ± 0.05	0.63 ± 0.07	0.93 ± 0.02	0.93 ± 0.01	$\mathbf{0.95 \pm 0.01}$	$\mathbf{0.95 \pm 0.0}$
63	0.87 ± 0.02	0.97 ± 0.01	0.97 ± 0.01	0.97 ± 0.01	$\mathbf{0.98 \pm 0.0}$	$\mathbf{0.98 \pm 0.0}$
64	0.88 ± 0.04	0.91 ± 0.02	0.94 ± 0.01	$\mathbf{0.95 \pm 0.0}$	$\mathbf{0.95 \pm 0.0}$	0.88 ± 0.04
65	0.84 ± 0.08	$\mathbf{0.9 \pm 0.04}$	0.87 ± 0.02	0.89 ± 0.02	0.88 ± 0.03	0.88 ± 0.03
66	0.94 ± 0.07	0.99 ± 0.01	$\mathbf{1.0 \pm 0.0}$	$\mathbf{1.0 \pm 0.0}$	0.99 ± 0.0	$\mathbf{1.0 \pm 0.0}$
67	0.93 ± 0.04	0.95 ± 0.02	0.98 ± 0.03	$\mathbf{0.99 \pm 0.01}$	0.98 ± 0.01	0.96 ± 0.03
68	0.56 ± 0.07	0.61 ± 0.05	0.64 ± 0.07	0.64 ± 0.03	0.81 ± 0.03	$\mathbf{0.99 \pm 0.0}$
69	0.83 ± 0.07	0.93 ± 0.03	0.97 ± 0.03	0.96 ± 0.03	$\mathbf{0.98 \pm 0.01}$	$\mathbf{0.98 \pm 0.01}$
70	0.99 ± 0.01	$\mathbf{1.0 \pm 0.0}$	$\mathbf{1.0 \pm 0.01}$	$\mathbf{1.0 \pm 0.0}$	$\mathbf{1.0 \pm 0.0}$	$\mathbf{1.0 \pm 0.0}$
71	0.54 ± 0.06	0.71 ± 0.1	0.72 ± 0.07	0.8 ± 0.04	0.8 ± 0.06	$\mathbf{0.85 \pm 0.06}$
72	0.4 ± 0.01	0.45 ± 0.02	0.47 ± 0.02	0.44 ± 0.01	0.56 ± 0.01	$\mathbf{0.7 \pm 0.01}$

A.2 Scalability of SAST, SASTEN and SASTEN-A regarding the dataset size

Figure A.1 shows a zoom on Figure 3.20b. We can now clearly see that SASTEN is slower than SAST and SASTEN-A whatever the number of time series in the dataset. SASTEN-A running time is quite linear because the shapelet space is constant and only the transform time increases with the number of series.

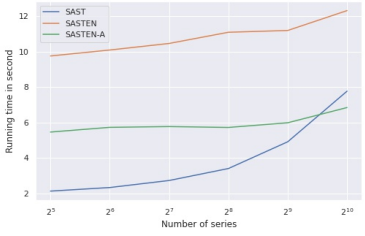


Figure A.1: Running time in seconds of SAST, SASTEN and SASTEN-A regarding the number of time series

Table A.1: Dataset identifiers, names, and types. Datasets marked with a star are those used for the experiment which results are presented in Section 3.3.1.2 and datasets marked with a *plus* sign are those used for the experiment which results are given in Section 3.3.1.4.

Id	Name	Type	Id	Name	Type
1	Adiac	Image	37	GunPointMaleVersusFemale*	Motion
2	ArrowHead+	Image	38	GunPointOldVersusYoung*	Motion
3	BME*	Simulated	39	Ham+	Spectro
4	Beef+	Spectro	40	Herring+	Image
5	BeetleFly+	Image	41	InsectWingbeatSound	Sensor
6	BirdChicken+	Image	42	ItalyPowerDemand*+	Sensor
7	CBF*+	Simulated	43	Meat	Spectro
8	Car+	Sensor	44	MedicalImages*	Image
9	Chinatown*	Traffic	45	MelbournePedestrian	Traffic
10	ChlorineConcentration*	Sensor	46	MiddlePhalanxOutlineAgeGroup*+	Image
11	Coffee+	Spectro	47	MiddlePhalanxOutlineCorrect*+	Image
12	CricketX	Motion	48	MiddlePhalanxTW*+	Image
13	CricketY	Motion	49	MoteStrain*+	Sensor
14	CricketZ	Motion	50	PhalangesOutlinesCorrect*	Image
15	Crop*	Image	51	Plane*+	Sensor
16	DiatomSizeReduction+	Image	52	PowerCons*	Power
17	DistalPhalanxOutlineAgeGroup*+	Image	53	ProximalPhalanxOutlineAgeGroup*+	Image
18	DistalPhalanxOutlineCorrect*+	Image	54	ProximalPhalanxOutlineCorrect*	Image
19	DistalPhalanxTW*+	Image	55	ProximalPhalanxTW*+	Image
20	DodgerLoopDay	Sensor	56	ShapeletSim+	Simulated
21	DodgerLoopGame	Sensor	57	SmoothSubspace*	Simulated
22	DodgerLoopWeekend	Sensor	58	SonyAIBORobotSurface1*+	Sensor
23	ECG200*+	ECG	59	SonyAIBORobotSurface2*+	Sensor
24	ECG5000*	ECG	60	Strawberry	Spectro
25	ECGFiveDays*+	ECG	61	SwedishLeaf*	Image
26	Earthquakes+	Sensor	62	Symbols+	Image
27	ElectricDevices*	Device	63	SyntheticControl*+	Simulated
28	FaceAll*	Image	64	ToeSegmentation1+	Motion
29	FaceFour+	Image	65	ToeSegmentation2+	Motion
30	FacesUCR*	Image	66	Trace	Sensor
31	FiftyWords	Image	67	TwoLeadECG*+	ECG
32	FreezerRegularTrain	Sensor	68	TwoPatterns*	Simulated
33	FreezerSmallTrain	Sensor	69	UMD*	Simulated
34	Fungi	HRM	70	Wafer*	Sensor
35	GunPoint*+	Motion	71	Wine+	Spectro
36	GunPointAgeSpan*	Motion	72	WordSynonyms	Image

Appendix B

Appendix for chapter 3

Contents

B.1 SAST-based methods results 101
B.2 uSAST-based methods results 102
B.3 Results of state-of-the-art models 103

B.1 SAST-based methods results

Table B.1: SAST<X> models results on the PLAsTiCC dataset average over 3 runs. The stared row is the run that achieved the lowest cross entropy loss.

	Classifier	Precision	Recall	F1 score	LogLoss	Time (h)
SAST	RF	0.59 ± 0.04	0.61 ± 0.00	0.56 ± 0.01	1.29 ± 0.05	16.41 ± 0.52
	RidgeCV	0.56 ± 0.00	0.57 ± 0.00	0.55 ± 0.00	2.22 ± 0.01	16.41 ± 0.52
	XGBoost	0.65 ± 0.01	0.67 ± 0.00	0.63 ± 0.00	1.16 ± 0.01	16.41 ± 0.52
	XGBoost*	0.65	0.67	0.64	1.15	16.23
SASTd	RF	0.58 ± 0.05	0.60 ± 0.00	0.54 ± 0.00	1.33 ± 0.08	12.79 ± 0.84
	RidgeCV	0.56 ± 0.01	0.58 ± 0.01	0.55 ± 0.01	2.20 ± 0.00	12.79 ± 0.84
	XGBoost	0.66 ± 0.02	0.68 ± 0.00	0.64 ± 0.00	1.14 ± 0.00	12.79 ± 0.841
	XGBoost*	0.66	0.68	0.64	1.13	11.84
SASTdc	RF	0.58 ± 0.03	0.61 ± 0.00	0.55 ± 0.00	1.33 ± 0.04	12.99 ± 0.30
	RidgeCV	0.52 ± 0.02	0.54 ± 0.01	0.52 ± 0.01	2.25 ± 0.00	12.99 ± 0.30
	XGBoost	0.66 ± 0.01	0.68 ± 0.00	0.64 ± 0.01	1.14 ± 0.01	12.99 ± 0.30
	XGBoost*	0.65	0.68	0.64	1.14	13.04

B.2 uSAST-based methods results

Table B.2: uSAST<X> models results on the PLAsTiCC dataset average over 3 runs. The row with the star is the run that achieved the lowest cross entropy loss. Subsequence length: from 16 to 32 with a step of 1.

	Classifier	Precision	Recall	F1 score	LogLoss	Time (h)
uSAST	RF	0.62 ± 0.01	0.66 ± 0.01	0.60 ± 0.01	1.20 ± 0.03	53.61 ± 0.10
	RidgeCV	0.33 ± 0.02	0.34 ± 0.02	0.33 ± 0.02	2.98 ± 0.11	53.61 ± 0.10
	XGBoost	0.69 ± 0.00	0.70 ± 0.00	0.66 ± 0.00	1.04 ± 0.01	53.61 ± 0.10
	XGBoost*	0.69	0.70	0.66	1.04	53.56
uSASTd	RF	0.61 ± 0.03	0.64 ± 0.00	0.58 ± 0.00	1.20 ± 0.04	41.58 ± 0.47
	RidgeCV	0.35 ± 0.01	0.37 ± 0.01	0.35 ± 0.01	2.90 ± 0.06	41.58 ± 0.47
	XGBoost	0.70 ± 0.01	0.70 ± 0.00	0.67 ± 0.00	1.06 ± 0.02	41.58 ± 0.47
	XGBoost*	0.71	0.70	0.67	1.04	41.78
uSASTdc	RF	0.60 ± 0.00	0.64 ± 0.01	0.58 ± 0.00	1.21 ± 0.00	40.45 ± 1.00
	RidgeCV	0.37 ± 0.04	0.38 ± 0.03	0.37 ± 0.03	2.78 ± 0.07	40.45 ± 1.00
	XGBoost	0.69 ± 0.01	0.70 ± 0.00	0.66 ± 0.00	1.04 ± 0.02	40.45 ± 1.00
	XGBoost*	0.68	0.70	0.66	1.03	41.16

Table B.3: uSAST<X> models results on the PLAsTiCC dataset average over 3 runs. The row with the star is the run that achieved the lowest cross entropy loss. Subsequence length: from 20 to 60 with a step of 10.

	Classifier	Precision	Recall	F1 score	LogLoss	Time (h)
uSAST	RF	0.64 ± 0.01	0.68 ± 0.01	0.62 ± 0.01	1.09 ± 0.01	51.03 ± 0.12
	Ridge	0.39 ± 0.01	0.40 ± 0.01	0.39 ± 0.01	2.82 ± 0.08	51.03 ± 0.12
	XGBoost	0.72 ± 0.01	0.72 ± 0.00	0.69 ± 0.01	0.96 ± 0.01	51.03 ± 0.12
	XGBoost*	0.71	0.72	0.69	0.95	51.17
uSASTd	RF	0.65 ± 0.02	0.67 ± 0.01	0.62 ± 0.01	1.12 ± 0.04	43.49 ± 0.27
	Ridge	0.39 ± 0.02	0.41 ± 0.02	0.40 ± 0.02	2.77 ± 0.07	43.49 ± 0.27
	XGBoost	0.72 ± 0.00	0.73 ± 0.00	0.70 ± 0.01	0.97 ± 0.01	43.49 ± 0.27
	XGBoost*	0.72	0.73	0.69	0.95	43.18
uSASTdc	RF	0.67 ± 0.03	0.67 ± 0.00	0.62 ± 0.00	1.11 ± 0.00	43.52 ± 0.72
	Ridge	0.39 ± 0.02	0.40 ± 0.02	0.39 ± 0.02	2.79 ± 0.02	43.52 ± 0.72
	XGBoost	0.71 ± 0.01	0.72 ± 0.01	0.69 ± 0.01	0.96 ± 0.01	43.52 ± 0.72
	XGBoost*	0.70	0.72	0.68	0.95	44.26

B.3 Results of state-of-the-art models

Table B.4: Performance of state-of-the-art models average over 3 runs. MUSE uses a linear classifier. The row with the star is the run that achieved the lowest cross entropy loss

	Classifier	Precision	Recall	F1 score	LogLoss	Time (h)
MUSE	-	0.71 ± 0.01	0.73 ± 0.01	0.71 ± 0.01	1.78 ± 0.03	3.36 ± 0.04
ROCKET	RF	0.75 ± 0.01	0.75 ± 0.00	0.72 ± 0.00	0.94 ± 0.03	0.05 ± 0.00
	Ridge	0.71 ± 0.01	0.71 ± 0.01	0.70 ± 0.01	2.07 ± 0.00	0.05 ± 0.00
	XGBoost	0.77 ± 0.00	0.77 ± 0.00	0.75 ± 0.00	0.82 ± 0.01	0.05 ± 00
	XGBoost*	0.78	0.77	0.75	0.81	0.05

Bibliography

[Abdar *et al.* 2021] Moloud Abdar, Farhad Pourpanah, Sadiq Hussain, Dana Rezazadegan, Li Liu, Mohammad Ghavamzadeh, Paul Fieguth, Xiaochun Cao, Abbas Khosravi, U Rajendra Acharya*et al. A review of uncertainty quantification in deep learning: Techniques, applications and challenges.* Information Fusion, vol. 76, pages 243–297, 2021. (Cited on page 1.)

[Aggarwal & Yu 2009] C. C. Aggarwal and P. S. Yu. *A Survey of Uncertain Data Algorithms and Applications.* IEEE Transactions on Knowledge and Data Engineering, vol. 21, no. 5, pages 609–623, 2009. (Cited on pages 38 and 45.)

[Allam Jr *et al.* 2018] Tarek Allam Jr*et al. The photometric LSST astronomical time-series classification challenge (PLAsTiCC): Data set.* arXiv:1810.00001, 2018. (Cited on pages 12, 27, 76 and 79.)

[Anthony Bagnall & Keogh 2018] William Vickers Anthony Bagnall Jason Lines and Eamonn Keogh. *The UEA & UCR Time Series Classification Repository.* www.timeseriesclassification.com, 2018. [Online, Accessed 19-October-2020]. (Cited on pages 22, 50, 52, 57, 62 and 65.)

[Aßfalg *et al.* 2009] Johannes Aßfalg, Hans-Peter Kriegel, Peer Kröger and Matthias Renz. *Probabilistic similarity search for uncertain time series.* In International Conference on Scientific and Statistical Database Management, pages 435–443, 2009. (Cited on page 25.)

[Audibert *et al.* 2020] Julien Audibert, Pietro Michiardi, Frédéric Guyard, Sébastien Marti and Maria A Zuluaga. *USAD: UnSupervised Anomaly Detection on Multivariate Time Series.* In ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, volume 20, pages 3395–3404, 2020. (Cited on page 12.)

[Bagnall *et al.* 2012] Anthony Bagnall, Luke Davis, Jon Hills and Jason Lines. *Transformation based ensembles for time series classification.* In Proceedings of the 2012 SIAM international conference on data mining, pages 307–318. SIAM, 2012. (Cited on page 21.)

[Bagnall *et al.* 2017] 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, vol. 31, no. 3, pages 606–660, 2017. (Cited on pages 12, 14, 15, 27, 32, 33, 40, 41, 48 and 63.)

[Bi & Zhang 2005] Jinbo Bi and Tong Zhang. *Support vector classification with input data uncertainty.* In Advances in Neural Information Processing Systems, pages 161–168, 2005. (Cited on page 38.)

[Boone 2019] Kyle Boone. *Avocado: Photometric classification of astronomical transients with gaussian process augmentation.* The Astronomical Journal, vol. 158, no. 6, page 257, 2019. (Cited on page 1.)

[Bostrom & Bagnall 2017] Aaron Bostrom and Anthony Bagnall. *Binary shapelet transform for multiclass time series classification.* In Transactions on Large-Scale Data-and Knowledge-Centered Systems XXXII, pages 24–46. Springer, 2017. (Cited on page 53.)

[Breiman 1996] Leo Breiman. *Bagging predictors.* Machine Learning, vol. 24, no. 2, pages 123–140, 1996. (Cited on page 56.)

[BRGM] BRGM. *Piezometric level in the city of Charbonnières-les-Varennes in the French department Puy-de-Dôme.* https://hubeau.eaufrance.fr/sites/default/files/api/demo/piezo/piezo.htm. [Onlince, Accessed 20-April-2022]. (Cited on page 8.)

[Cabello *et al.* 2020] Nestor Cabello, Elham Naghizade, Jianzhong Qi and Lars Kulik. *Fast and Accurate Time Series Classification Through Supervised Interval Search.* In international Conference on Data Mining, pages 948–953. IEEE, 2020. (Cited on page 17.)

[Carranza Alarcon & Destercke 2019] Yonatan Carlos Carranza Alarcon and Sébastien Destercke. *Imprecise Gaussian Discriminant Classification.* Pattern Recognition, vol. 112, page 107739, June 2019. (Cited on page 91.)

[Corduas & Piccolo 2008] Marcella Corduas and Domenico Piccolo. *Time series clustering and classification by the autoregressive metric.* Computational statistics & data analysis, vol. 52, no. 4, pages 1860–1872, 2008. (Cited on page 21.)

[Dallachiesa *et al.* 2012] Michele Dallachiesa, Besmira Nushi, Katsiaryna Mirylenka and Themis Palpanas. *Uncertain time-series similarity: Return to the basics.* Proceedings of the VLDB Endowment, vol. 5, no. 11, pages 1662–1673, 2012. (Cited on pages 3, 10, 26, 32, 40 and 91.)

[DARPA] DARPA. *Explainable Articial Intelligence (XAI).* https://www.darpa.mil/program/explainable-artificial-intelligence. [Online, Accessed 23-September-2022]. (Cited on page 2.)

[Dash & Liu 1997] Manoranjan Dash and Huan Liu. *Feature selection for classification.* Intelligent Data Analysis, vol. 1, no. 3, pages 131–156, 1997. (Cited on pages 53 and 54.)

[Dau *et al.* 2019] 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, vol. 6, no. 6, pages 1293–1305, 2019. (Cited on pages 27 and 41.)

[Dempster *et al.* 2020] 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, vol. 5, pages 1454–1495, 2020. (Cited on pages 22, 56, 62 and 83.)

[Dempster *et al.* 2021] Angus Dempster, Daniel F Schmidt and Geoffrey I Webb. *MiniRocket: A Very Fast (Almost) Deterministic Transform for Time Series Classification.* In ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, page 10. ACM, 2021. (Cited on page 22.)

[Deng *et al.* 2013] Houtao Deng, George Runger, Eugene Tuv and Martyanov Vladimir. *A time series forest for classification and feature extraction.* Information Sciences, vol. 239, pages 142–153, 2013. (Cited on page 17.)

[Destercke & Couso 2015] Sébastien Destercke and Ines Couso. *Ranking of fuzzy intervals seen through the imprecise probabilistic lens.* Fuzzy Sets and Systems, vol. 278, pages 20–39, 2015. (Cited on page 91.)

[Destercke *et al.* 2008] Sébastien Destercke, Didier Dubois and Eric Chojnacki. *Unifying practical uncertainty representations–I: Generalized p-boxes.* International Journal of Approximate Reasoning, vol. 49, no. 3, pages 649–663, 2008. (Cited on page 91.)

[Destercke 2022] Sébastien Destercke. *Uncertain data in learning: challenges and opportunities.* In Symposium on Conformal and Probabilistic Prediction with Applications, pages 322–332. PMLR, 2022. (Cited on page 10.)

[DiCarlo *et al.* 2012] James J DiCarlo, Davide Zoccolan and Nicole C Rust. *How does the brain solve visual object recognition?* Neuron, vol. 73, no. 3, pages 415–434, 2012. (Cited on pages 48 and 49.)

[Ding *et al.* 2008] 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.* In Proceedings of the VLDB Endowment, volume 1, pages 1542–1552, 2008. (Cited on page 12.)

[Fauvel *et al.* 2022] Kevin Fauvel, Elisa Fromont, Véronique Masson, Philippe Faverdin and Alexandre Termier. *XEM: An explainable-by-design ensemble method for multivariate time series classification.* DMKD, vol. 36, no. 3, pages 917–957, 2022. (Cited on pages 19 and 83.)

[Fawaz *et al.* 2019a] H Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar and P Muller. *Adversarial attacks on deep neural networks for time series classification.* arXiv preprint arXiv:1903.07054, 2019. (Cited on page 2.)

[Fawaz *et al.* 2019b] 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, vol. 33, no. 4, pages 917–963, 2019. (Cited on pages 22 and 57.)

[Fawaz *et al.* 2020] 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. *Inceptiontime: Finding alexnet for time series classification.* Data Mining and Knowledge Discovery, vol. 34, no. 6, pages 1936–1962, 2020. (Cited on page 22.)

[Flynn *et al.* 2019] 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. (Cited on page 23.)

[Grabocka *et al.* 2014] Josif Grabocka, Nicolas Schilling, Martin Wistuba and Lars Schmidt-Thieme. *Learning time-series shapelets.* In Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 392–401, 2014. (Cited on pages 19 and 61.)

[Gruber *et al.* 2020] M Gruber, T Dorst, A Schütze, S Eichstädt and C Elster. Advanced mathematical and computational tools in metrology and testing xii, chapter Discrete wavelet transform on uncertain data: Efficient online implementation for practical applications. World Scientific, 2020. (Cited on page 77.)

[Hamooni & Mueen 2014] Hossein Hamooni and Abdullah Mueen. *Dual-domain hierarchical classification of phonetic time series.* In 2014 IEEE International Conference on Data Mining, pages 160–169. IEEE, 2014. (Cited on page 14.)

[Heeger 2014] David Heeger. *Center for Neural Science, lecture notes: Perception (undergraduate).* http://www.cns.nyu.edu/~david/courses/perception/lecturenotes/recognition/recognition.html, 2002-2014. [Online; Accessed 19-May-2021]. (Cited on page 49.)

[Hernandez & Brown 2020] Danny Hernandez and Tom B Brown. *Measuring the algorithmic efficiency of neural networks.* arXiv preprint arXiv:2005.04305, 2020. (Cited on page 2.)

[Herrmann & Webb 2021] Matthieu Herrmann and Geoffrey I Webb. *Early abandoning and pruning for elastic distances including dynamic time warping.* Data Mining and Knowledge Discovery, vol. 35, no. 6, pages 2577–2601, 2021. (Cited on page 16.)

[Hewamalage *et al.* 2022] Hansika Hewamalage, Christoph Bergmeir and Kasun Bandara. *Global models for time series forecasting: A Simulation study.* Pattern Recognition, vol. 124, page 108441, 2022. (Cited on page 12.)

[Hills *et al.* 2014] Jon Hills, Jason Lines, Edgaras Baranauskas, James Mapp and Anthony Bagnall. *Classification of time series by shapelet transformation.* Data Mining and Knowledge Discovery, vol. 28, pages 851–881, 2014. (Cited on pages 19, 27, 33, 37, 40, 51, 52, 53, 56, 76 and 80.)

[Hložek *et al.* 2020] R. Hložek*et al.* *Results of the Photometric LSST Astronomical Time-series Classification Challenge (PLAsTiCC).* arXiv:2012.12392v1, 2020. (Cited on page 76.)

[Ishida 2019] Emille E. O. Ishida. *Machine learning and the future of supernova cosmology.* Nature Astronomy, vol. 3, pages 680–682, 2019. (Cited on page 82.)

[Karlsson *et al.* 2016] Isak Karlsson, Panagiotis Papapetrou and Henrik Boström. *Generalized random shapelet forests.* Data Mining and Knowledge Discovery, vol. 30, no. 5, pages 1053–1085, 2016. (Cited on page 53.)

[Kauffman *et al.* 2022] James M Kauffman, Jeanmarie Badar, Andrew L Wiley, Dimitris Anastasiou and Jennifer Koran. *Uncertainty in Education: Policy Implications.* Journal of Education, page 00220574221090283, 2022. (Cited on page 2.)

[Le Nguyen *et al.* 2019] Thach Le Nguyen, Severin Gsponer, Iulia Ilie, Martin O'Reilly and Georgiana Ifrim. *Interpretable time series classification using linear models and multi-resolution multi-domain symbolic representations.* Data Mining and Knowledge Discovery, vol. 33, no. 4, pages 1183–1222, 2019. (Cited on page 14.)

[Leoni *et al.* 2021] Marco Leoni, Emille E. O. Ishida, Julien Peloton and Anais Möller. *Fink: early supernovae Ia classification using active learning.* arXiv:2111.11438v1, 2021. (Cited on page 1.)

[Lewandowski *et al.* 2010] Michal Lewandowski, Jesus Martinez-del Rincon, Dimitrios Makris and Jean-Christophe Nebel. *Temporal Extension of Laplacian Eigenmaps for Unsupervised Dimensionality Reduction of Time Series.* In 20th International Conference on Pattern Recognition, pages 161–164, 2010. (Cited on page 12.)

[Li *et al.* 2020a] Lingli Li, Hongzhi Wang, Jianzhong Li and Hong Gao. *A survey of uncertain data management.* Frontiers of Computer Science, vol. 14, no. 1, pages 162–190, Feb 2020. (Cited on pages 38 and 45.)

[Li *et al.* 2020b] Xue Li, Bo Du, Chang Xu, Yipeng Zhang, Lefei Zhang and Dacheng Tao. *Robust learning with imperfect privileged information.* Artificial Intelligence, vol. 282, page 103246, 2020. (Cited on page 38.)

[Lin *et al.* 2007] Jessica Lin, Eamonn Keogh, Li Wei and Stefano Lonardi. *Experiencing SAX: A novel symbolic representation of time series.* Data Mining

and Knowledge Discovery, vol. 15, pages 107–144, 2007. (Cited on pages ix, 12, 19, 20 and 21.)

[Lines *et al.* 2018] Jason Lines, Sarah Taylor, Anthony Bagnall and A Bagnall. *Time Series Classification with HIVE-COTE: The Hierarchical Vote Collective of Transformation-Based Ensembles.* ACM Transactions on Knowledge Discovery from Data, vol. 12, page 52, 2018. (Cited on pages ix, 16, 17, 18, 20, 21, 22, 23 and 62.)

[Liu *et al.* 2021] Haibo Liu, Ming Chen, Chong Du, Jiachang Tang, Chunming Fu and Guilin She. *A copula-based uncertainty propagation method for structures with correlated parametric p-boxes.* International Journal of Approximate Reasoning, 2021. (Cited on page 77.)

[Löning *et al.* 2019] Markus Löning, Anthony Bagnall, Sajaysurya Ganesh, Viktor Kazakov, Jason Lines and Franz J Király. *Sktime: A Unified Interface for Machine Learning with Time Series.* In Workshop on Systems for ML at NeurIPS 2019, 2019. (Cited on pages 42, 52, 56 and 80.)

[Lopez Conde & Twinn 2019] Maria Lopez Conde and Ian Twinn. *How Artificial Intelligence is Making Transport Safer, Cleaner, More Reliable and Efficient in Emerging Markets.* EMCompass;Note 75, pages 1–8, 2019. (Cited on page 1.)

[Lu *et al.* 2022] Yue Lu, Renjie Wu, Abdullah Mueen, Maria A Zuluaga and Eamonn Keogh. *Matrix Profile XXIV: Scaling Time Series Anomaly Detection to Trillions of Datapoints and Ultra-fast Arriving Data Streams.* In 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pages 1173–1182, 2022. (Cited on page 2.)

[Lubba *et al.* 2019] 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, vol. 33, no. 6, pages 1821–1852, 2019. (Cited on page 17.)

[Lundberg & Lee 2017] Scott M. Lundberg and Su-In Lee. *A Unified Approach to Interpreting Model Predictions.* In 31st International Conference on Neural Information Processing Systems, NIPS'17, page 4768–4777. Curran Associates Inc., 2017. (Cited on pages 24 and 84.)

[Ma & Perkins 2003] J. Ma and S. Perkins. *Time-series novelty detection using one-class support vector machines.* In Proceedings of the International Joint Conference on Neural Networks, volume 3, pages 1741–1745, 2003. (Cited on page 12.)

[Marshall *et al.* 2010] Albert W. Marshall, Ingram Olkin and Barry C. Arnold. Stochastic Ordering, chapter 17, pages 693–756. Springer New York, New York, NY, 2010. (Cited on page 34.)

[May 2021] Mike May. *Eight ways machine learning is assisting medicine.* Nat Med, vol. 27, no. 1, pages 2–3, 2021. (Cited on page 1.)

[Mazzi *et al.* 2019] Gian Luigi Mazzi, James Mitchell and Florabela Carausu. Data uncertainties: Their sources and consequences. Publications Office of the European Union, 2019. (Cited on page 1.)

[Mbouopda *et al.* 2022] Michael Franklin Mbouopda, Thomas Guyet, Nicolas Labroche and Abel Heuriot. *Local vs global forecasting methods for groundwater level prediction.* In ECML/PKDD Workshop on Advanced Analytics and Learning on Temporal Data, pages 1–16. Springer, 2022. *accepted for publication.* (Cited on page 12.)

[Mbouopda 2022] Michael Franklin Mbouopda. *Etude de la prédiction du niveau de la nappe phréatique à l'aide de modèles neuronaux convolutif, récurrent et résiduel.* Revue des Nouvelles Technologies de l'Information, vol. Extraction et Gestion des Connaissances, RNTI-E-38, pages 313–320, 2022. (Cited on page 12.)

[Middlehurst *et al.* 2020a] Matthew Middlehurst, James Large and Anthony Bagnall. *The Canonical Interval Forest (CIF) Classifier for Time Series Classification.* In BigData, pages 188–195. IEEE, 2020. (Cited on pages 17, 52 and 57.)

[Middlehurst *et al.* 2020b] Matthew Middlehurst, James Large, Gavin Cawley and Anthony Bagnall. *The Temporal Dictionary Ensemble (TDE) Classifier for Time Series Classification.* In Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pages 660–676, 2020. (Cited on page 20.)

[Middlehurst *et al.* 2021] Matthew Middlehurst, James Large, Michael Flynn, Jason Lines, Aaron Bostrom, Anthony Bagnall, Gustavo Batista and Anthony Bagnall. *HIVE-COTE 2.0: a new meta ensemble for time series classification.* Machine Learning, vol. 110, pages 3211–3243, 2021. (Cited on page 23.)

[Miller *et al.* 2022] Elishiah Miller, Zane MacFarlane, Seth Martin, Nilanjan Banerjee and Ting Zhu. *Radar-based monitoring system for medication tampering using data augmentation and multivariate time series classification.* Smart Health, vol. 23, 2022. (Cited on page 1.)

[Möller & de Boissière 2020] Anais Möller and Thibault de Boissière. *SuperNNova: an open-source framework for Bayesian, neural network-based supernova classification.* Monthly Notices of the Royal Astronomical Society, vol. 491, no. 3, pages 4277–4293, 2020. (Cited on page 14.)

[Molnar 2020] Christoph Molnar. Interpretable machine learning. Lulu. com, 2020. (Cited on pages 53 and 54.)

[Moss 2018] Adam Moss. *Improved Photometric Classification of Supernovae using Deep Learning.* arXiv:1810.06441, 2018. (Cited on page 14.)

[Murdoch *et al.* 2019] W. James Murdoch, Chandan Singh, Karl Kumbier, Reza Abbasi-Asl and Bin Yu. *Definitions, methods, and applications in interpretable machine learning.* Proceedings of the National Academy of Sciences, vol. 116, no. 44, page 22071–22080, Oct 2019. (Cited on page 54.)

[Nakamura *et al.* 2020] Takaaki Nakamura, Makoto Imamura, Ryan Mercer and Eamonn Keogh. *MERLIN: Parameter-free discovery of arbitrary length anomalies in massive time series archives.* In IEEE International Conference on Data Mining, pages 1190–1195, 2020. (Cited on page 12.)

[OLUD 2022] OLUD. *Workshop on Online Learning from Uncertain Data Streams.* https://sites.google.com/view/olud/home, July 2022. [Online, Accessed 22-September-2022]. (Cited on page 3.)

[Our World in Data 2022] Our World in Data. *Daily new confirmed COVID-19 cases per million people.* https://ourworldindata.org/explorers/coronavirus-data-explorer, 2022. [Online, Accessed 20-April-2022]. (Cited on page 8.)

[Patterson *et al.* 2022] David Patterson, Joseph Gonzalez, Urs Hölzle, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David R So, Maud Texier and Jeff Dean. *The carbon footprint of machine learning training will plateau, then shrink.* Computer, vol. 55, no. 7, pages 18–28, 2022. (Cited on page 2.)

[Pedregosa *et al.* 2011] 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, vol. 12, pages 2825–2830, 2011. (Cited on pages 56 and 80.)

[PLASTICC Team and PLASTICC Modelers 2019] PLASTICC Team and PLASTICC Modelers. *Unblinded Data for PLAsTiCC Classification Challenge*, Jan 2019. https://doi.org/10.5281/zenodo.2539456. (Cited on page 79.)

[Qin *et al.* 2009] Biao Qin, Yuni Xia and Fang Li. *DTU: a decision tree for uncertain data.* In Pacific-Asia conference on Knowledge Discovery and Data Mining, pages 4–15, 2009. (Cited on pages 38 and 91.)

[Qin *et al.* 2010] Biao Qin, Yuni Xia and Fang Li. *A Bayesian Classifier for Uncertain Data.* In Proceedings of the 2010 ACM Symposium on Applied Computing, SAC '10, page 1010–1014. Association for Computing Machinery, 2010. (Cited on page 38.)

[Qin *et al.* 2011] Biao Qin, Yuni Xia, Shan Wang and Xiaoyong Du. *A novel Bayesian classification for uncertain data.* Knowledge-Based Systems, vol. 24, no. 8, pages 1151–1158, 2011. (Cited on pages 38, 41 and 91.)

[Quost & Denœux 2009] Benjamin Quost and Thierry Denœux. *Learning from data with uncertain labels by boosting credal classifiers.* In ACM SIGKDD Workshop on Knowledge Discovery From Uncertain Data, pages 38–47, 2009. (Cited on page 92.)

[Rakthanmanon & Keogh 2013] Thanawin Rakthanmanon and Eamonn Keogh. *Fast shapelets: A scalable algorithm for discovering time series shapelets.* In SIAM International Conference on Data Mining, pages 668–676. SIAM, 2013. (Cited on pages 19 and 61.)

[Ratanamahatana & Keogh 2004] 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. (Cited on page 16.)

[Renard *et al.* 2015] Xavier Renard, Maria Rifqi, Walid Erray and Marcin Detyniecki. *Random-shapelet: an algorithm for fast shapelet discovery.* In 2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA), pages 1–10, 2015. (Cited on page 19.)

[Ribeiro *et al.* 2016] Marco Tulio Ribeiro, Sameer Singh and Carlos Guestrin. *"Why should I trust you?" Explaining the predictions of any classifier.* In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 1135–1144, 2016. (Cited on pages 2, 24, 54 and 84.)

[Ruiz *et al.* 2021] Alejandro Pasos Ruiz, Michael Flynn, James Large, Matthew Middlehurst and Anthony Bagnall. *The great multivariate time series classification bake off: a review and experimental evaluation of recent algorithmic advances.* DMKD, vol. 35, no. 2, pages 401–449, 2021. (Cited on page 83.)

[Sakoe & Chiba 1978] Hiroaki Sakoe and Seibi Chiba. *Dynamic programming algorithm optimization for spoken word recognition.* IEEE transactions on acoustics, speech, and signal processing, vol. 26, no. 1, pages 43–49, 1978. (Cited on pages 15 and 16.)

[Samek *et al.* 2020] Wojciech Samek, Grégoire Montavon, Sebastian Lapuschkin, Christopher J. Anders and Klaus-Robert Müller. *Toward Interpretable Machine Learning: Transparent Deep Neural Networks and Beyond*, 2020. (Cited on page 54.)

[Sarangi & Murthy 2010] Smruti R Sarangi and Karin Murthy. *DUST: a generalized notion of similarity between uncertain time series.* In Proceedings of the 16th

ACM SIGKDD international conference on Knowledge Discovery and Data Mining, pages 383–392, 2010. (Cited on pages 25 and 32.)

[Schäfer & Leser 2017a] 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. (Cited on page 20.)

[Schäfer & Leser 2017b] Patrick Schäfer and Ulf Leser. *Multivariate time series classification with WEASEL+ MUSE.* arXiv:1711.11343, 2017. (Cited on pages 20 and 83.)

[Schäfer & Leser 2020] Patrick Schäfer and Ulf Leser. *TEASER: early and accurate time series classification.* Data Mining and Knowledge Discovery, vol. 34, no. 5, pages 1336–1362, 2020. (Cited on page 20.)

[Schäfer 2015] Patrick Schäfer. *The BOSS is concerned with time series classification in the presence of noise.* Data Mining and Knowledge Discovery, vol. 29, no. 6, pages 1505–1530, 2015. (Cited on page 20.)

[Schäfer & Högqvist 2012] Patrick Schäfer and Mikael Högqvist. *SFA: A Symbolic Fourier Approximation and Index for Similarity Search in High Dimensional Datasets.* In 15th International Conference on Extending Database Technology, pages 516–527, 2012. (Cited on page 20.)

[Shifaz *et al.* 2020] Ahmed Shifaz, Charlotte Pelletier, François Petitjean and Geoffrey I Webb. *TS-CHIEF: a scalable and accurate forest algorithm for time series classification.* Data Mining and Knowledge Discovery, vol. 34, pages 742–775, 2020. (Cited on pages 23, 32 and 62.)

[SIPTA] SIPTA. *The Society for Imprecise Probabilities: Theory and Applications.* https://sipta.org/. [Online; Accessed 22-September-2022]. (Cited on page 3.)

[Siyou Fotso *et al.* 2020] Vanel Steve Siyou Fotso, Engelbert Mephu Nguifo and Philippe Vaslin. *Frobenius correlation based u-shapelets discovery for time series clustering.* Pattern Recognition, page 107301, 2020. (Cited on pages 10, 11, 12, 26, 40, 48 and 70.)

[Slingo & Palmer 2011] Julia Slingo and Tim Palmer. *Uncertainty in weather and climate prediction.* Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, vol. 369, no. 1956, pages 4751–4767, 2011. (Cited on page 2.)

[Sliwa *et al.* 2020] Benjamin Sliwa, Nico Piatkowski and Christian Wietfeld. *LIMITS: Lightweight machine learning for IoT systems with resource limitations.* In IEEE International Conference on Communications (ICC), pages 1–7. IEEE, 2020. (Cited on page 2.)

[Smilkov *et al.* 2017] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas and Martin Wattenberg. *SmoothGrad: removing noise by adding noise*, 2017. (Cited on page 54.)

[Stanton & Roelich 2021] Muriel C Bonjean Stanton and Katy Roelich. *Decision making under deep uncertainties: A review of the applicability of methods in practice.* Technological Forecasting and Social Change, vol. 171, page 120939, 2021. (Cited on page 3.)

[Tan *et al.* 2020] Chang Wei Tan, François Petitjean and Geoffrey I. Webb. *FastEE: Fast Ensembles of Elastic Distances for time series classification.* Data Mining and Knowledge Discovery, vol. 34, pages 231–272, 2020. (Cited on page 16.)

[Tan *et al.* 2021] Chang Wei Tan, Christoph Bergmeir, François Petitjean and Geoffrey I. Webb. *Time series extrinsic regression: Predicting numeric values from time series data.* Data Mining and Knowledge Discovery, vol. 35, pages 1032–1060, 2021. (Cited on page 12.)

[Tan *et al.* 2022] Chang Wei Tan, Angus Dempster, Christoph Bergmeir and Geoffrey I Webb. *MultiRocket: Multiple pooling operators and transformations for fast and effective time series classification.* Data Mining and Knowledge Discovery, pages 1–24, 2022. (Cited on page 22.)

[Taylor 1996] John R. Taylor. An introduction to error analysis: The study of uncertainties in physical measurements. University Science Books, 2 édition, 1996. (Cited on pages 3, 8, 32 and 33.)

[Tsang *et al.* 2009] Smith Tsang, Ben Kao, Kevin Y Yip, Wai-Shing Ho and Sau Dan Lee. *Decision trees for uncertain data.* IEEE Transactions on Knowledge and Data Engineering, vol. 23, no. 1, pages 64–78, 2009. (Cited on page 38.)

[Ulanova *et al.* 2015] Liudmila Ulanova, Nurjahan Begum and Eamonn Keogh. *Scalable clustering of time series with U-shapelets.* In SIAM International Conference on Data Mining, pages 900–908, 2015. (Cited on page 12.)

[Wang *et al.* 2017] 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, 2017. (Cited on page 22.)

[Wang *et al.* 2020] Yichang Wang, Rémi Emonet, Elisa Fromont, Simon Malinowski and Romain Tavenard. *Adversarial Regularization for Explainable-by-Design Time Series Classification.* In 2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI), pages 1079–1087, 2020. (Cited on pages 19 and 67.)

[Wistuba *et al.* 2015] Martin Wistuba, Josif Grabocka and Lars Schmidt-Thieme. *Ultra-fast shapelets for time series classification.* arXiv:1503.05018, 2015. (Cited on page 19.)

[Wolpert & Macready 1997] David H Wolpert and William G Macready. *No free lunch theorems for optimization.* IEEE Transactions on Evolutionary Computation, vol. 1, no. 1, pages 67–82, 1997. (Cited on page 62.)

[WUML 2020] WUML. *ECML/PKDD 2020 Tutorial and Workshop on Uncertainty in Machine Learning.* https://sites.google.com/view/wuml-2020/, September 2020. [Online; Accesses 22-September-2022]. (Cited on page 3.)

[Xu & Da 2002] Z. S. Xu and Q. L. Da. *The uncertain OWA operator.* International Journal of Intelligent Systems, vol. 17, no. 6, pages 569–575, 2002. (Cited on page 35.)

[Yagoubi *et al.* 2018] Djamel-Edine Yagoubi, Reza Akbarinia, Florent Masseglia and Themis Palpanas. *Massively distributed time series indexing and querying.* IEEE Transactions on Knowledge and Data Engineering, vol. 32, no. 1, pages 108–120, 2018. (Cited on page 12.)

[Yang & Gunn 2007] Jianqiang Yang and Steve Gunn. *Iterative Constraints in Support Vector Classification with Uncertain Information.* Constraint-based Mining and Learning, page 49, 2007. (Cited on page 38.)

[Yang *et al.* 2020] Yaoqing Yang, Rajiv Khanna, Yaodong Yu, Amir Gholami, Kurt Keutzer, Joseph E Gonzalez, Kannan Ramchandran and Michael W Mahoney. *Boundary thickness and robustness in learning models.* Advances in Neural Information Processing Systems, vol. 33, pages 6223–6234, 2020. (Cited on page 2.)

[Ye & Keogh 2009a] Lexiang Ye and Eamonn Keogh. *Time series shapelets.* In ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, page 947. ACM, 2009. (Cited on page 84.)

[Ye & Keogh 2009b] Lexiang Ye and Eamonn Keogh. *Time series shapelets: a new primitive for data mining.* In 15th ACM SIGKDD international conference on Knowledge Discovery and Data Mining, pages 947–956, 2009. (Cited on pages 12, 17, 19, 32, 33, 36, 48, 50, 53, 63 and 67.)

[Yeh *et al.* 2009] Mi-Yen Yeh, Kun-Lung Wu, Philip S Yu and Ming-Syan Chen. *PROUD: a probabilistic approach to processing similarity queries over uncertain data streams.* In Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology, pages 684–695, 2009. (Cited on page 25.)

[Yeh *et al.* 2018] 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, vol. 32, pages 83–123, 2018. (Cited on page 12.)

[Yue 2011] Zhongliang Yue. *An extended TOPSIS for determining weights of decision makers with interval numbers.* Knowledge-Based Systems, vol. 24, no. 1, pages 146–153, 2011. (Cited on page 35.)

[Zhang *et al.* 2021] Hanbo Zhang, Peng Wang, Zicheng Fang, Zeyu Wang and Wei Wang. *ELIS++: a shapelet learning approach for accurate and efficient time series classification.* World Wide Web, vol. 24, no. 2, pages 511–539, 2021. (Cited on pages 19 and 61.)

[Zhao *et al.* 2016] Yannan Zhao, Yuan Li, Lifen Zhang and Qiuliang Wang. *Groundwater level prediction of landslide based on classification and regression tree.* Geodesy and Geodynamics, vol. 7, no. 5, pages 348–355, 2016. (Cited on page 14.)

[Zheng *et al.* 2021] Yunhan Zheng, Shenhao Wang and Jinhua Zhao. *Equality of opportunity in travel behavior prediction with deep neural networks and discrete choice models.* Transportation Research Part C: Emerging Technologies, vol. 132, pages 1–25, 2021. (Cited on page 1.)