Six Approaches to Enhancing the Efficiency of Dynamic Time Warping: A Study Using Nearest Neighbour Classification

Thesis submitted in accordance with the requirements of
the University of Liverpool for the degree of Doctor in Philosophy by

Mohammed Alshehri

Monday 10th January, 2022

Dedication

To my dearest parents, my beloved wife and children, and faithful brothers and sisters.

Acknowledgements

Alhamdulillah, praise be to Allah Almighty for His graces and guidance.

With delight and gratefulness, I would like to take this opportunity to convey my gratitude to all who supported and encouraged me during my PhD journey. My completion of this project could not have been accomplished without their support.

I would like to express my deepest gratitude and most profound thanks to my primary supervisor, Professor. Frans Coenen, for his continuous support and encouragement. His guidance and extensive scientific knowledge helped me throughout the time of pursuing my PhD. I have been privileged and pleased to work under his supervision.

I would like also to express my warm thanks to my second supervisor, Dr. Keith Dures who has provided me with insightful comments at various stages of my research. My sincere thanks also go to my advisors Dr. Valentina Tamma and Dr. Xiaowei Huang for their constructive feedback and suggestions.

I am also very grateful to my friends and colleagues at the Department of Computer Science who were always helpful and provided me with their assistance whenever necessary. I wish to extend my special thanks to all staff members of the Department of Computer Science who also facilitated the work of my PhD since the first day.

My profound gratitude is also expressed to my parents, sibling and all my family for the unfailing love and emotional support. I would especially like to express a heartfelt thanks to my wife, Maha, and my children, Abdullah and Kayan, for standing beside me during this challenging time.

Finally, last but by no means least, I would also like to show my appreciation to the Saudi government for the financial and medical support during my PhD journey, I also extend my gratitude to King Khalid University and the Saudi Cultural Bureau in London for their generous sponsorship that helped me to complete my postgraduate studies.

Abstract

The work presented in this thesis is directed at investigating approaches whereby Dynamic Time Warping (DTW) can be applied in a more effective and efficient manner. To act as a focus for the work, kNN classification was considered. The main research question to be answered was ***"How can the process of dynamic time warping be applied so that time series can be more effectively and efficiently compared?"*** (than as when applied in its standard form).

The main contributions of the thesis are six approaches for reducing the computational complexity of DTW process: (i) Sub-Sequence-Based DTW approach (SSBDTW), (ii) Fuzzy Sub-Sequence-Based DTW approach (FSSBDTW), (iii) Candidate Reduction Based on Euclidean Distance approach (CRBED), (iv) Candidate Reduction Based on Lower Bounding approach (CRBLB), (v) Exact Discriminator-Based DTW approach (EDBDTW) and (vi) Distance Profile -Based DTW approach (DPBDTW). The SSBDTW approach was proposed to split time series into equal sub-sequences and then apply DTW. The FSSBDTW approach was an improvement on SSBDTW. Instead of splitting time series into equal segments, FSSBDTW split time series into non-equal sub-sequences by identifying the best splitting point. The CRBED approach was used as a filtering technique to find the most similar time series by applying Euclidean Distance (ED) as a similarity measurements with k-Nearest Neighbor classification, and then applying FSSBDTW for the final classification. The CRBLB approach is similar to the CRBED approach; however, instead of using ED, CRBLB used a Lower Bounding method to identify the most similar candidates. The FSSBDTW approach was then applied to the identified candidates. The EDBDTW approach was directed at reshaping time series in an input data set to a new form by only considering the most similar subsequence (with the same indexes) as a discriminator of a class. FSSBDTW was then applied to the discriminators for classification purposes. The DPBDTW approach was an alternative reshaping approach, the distinction between the EDBDTW and DPBDTW approaches was that DPBDTW used Matrix Profile technique to reshape the time series in an input data set to a new form based on distance profiles. FSSBDTW was then applied to this new form. Note that the last four approaches use FSSBDTW as a final stage for classification.

The approaches were comprehensively analyzed and evaluated using fifteen time series data sets taken from the UEA and UCR Time Series Classification repository. Two benchmarks (Standard DTW and Sakoe–Chiba Band DTW) were considered in the evaluation analysis. The evaluation was conducted to compare the proposed approaches with the benchmarks in terms of run time, accuracy and F1–score. Ten Cross–Validation was adopted to validate the performance of the proposed approaches. The evaluation demonstrated that all the proposed approaches provide efficiency and effectiveness gains for enhancing the DTW process compare to the benchmarks. More specifically, the proposed CRBED approach was found to report the best improvements over the other approaches.

Contents

Illustrations

List of Figures

List of Tables

Notations

The following abbreviations are used throughout this thesis:

DTW	Dynamic Time Warping similarity measurement
ED	Euclidean Distance similarity measurement
SSBDTW	Sub-Sequence-Based Dynamic Time Warping
FSSBDTW	Fuzzy Sub-Sequence-Based Dynamic Time Warping
CRBED	Candidate Reduction Based on Euclidean Distance
CRBLB	Candidate Reduction Based on Lower Bounding
EDBDTW	Exact Discriminator-Based Dynamic Time Warping
DPBDTW	Distance Profile-Based Dynamic Time Warping
LB–Keogh	Lower Bounding method by Keogh
UEA	University of East Anglia
UCR	University of California Riverside
k**NN**	k-Nearest Neighbour classification algorithm
DT	Decision Tree classification algorithm
SVM	Support Vector Machine classification algorithm
DL	Deep Learning
LSTM	Long short-Term Memory
RNN	Recurrent Neural Network

Chapter 1

Introduction

1.1 Overview

Over recent years there has been a substantial increase in the amount of data that institutions and commercial enterprises collect. This has largely been as a consequence of technical advances. The data collected takes many forms; one such form is temporal data, specifically time series data [10, 89]. A time series is a set of observations recorded over time, these observations form a numerical sequence of values. Examples of time series are electrocardiogram (ECG) data [85], daily stocks prices [27] or daily temperatures [22]. Many types of data that are not true time series can be transformed into time series, including DNA, speech, body movement and shapes [87].

Given our increasing ability to collect large amounts of time series data there is a corresponding increasing desire to apply the tools and techniques of machine learning to this data. At a high level, the types of machine learning that can be applied to time series data are no different to the types of machine learning that can be applied to other forms of data. We typically want to find patterns and/or build models of various kinds that reflect the data. The distinctions are: (1) that the number of values in a time series is typically much greater than that found in a standard tabular data record, and (2) that we want to consider each time series (record) in its entirety, or at least substantial sub-sequences of a parent time series, as opposed to considering individual values as in the case of (say) the generation of neural networks with respect to traditional tabular data formats. Consequently, the application of the tools and techniques of machine learning to time series typically entail many comparisons of pairs of time series or time series sub-sequences. Given the number of comparisons, and the length of the time series concerned (in comparison with tabular data records), the complexity of time series analysis algorithms is very much influenced by the number of comparisons undertaken. For example, time series classification is typically conducted using a k Nearest Neighbour (kNN) classification model, with $k = 1$, where the model simply comprises a "bank" of time series with associated class labels. Given a new time series to be classified this is compared to every time series in the bank and the new time series labelled with the label for the most similar time series in the bank. The complexity, using "big O notation", is thus $O(n)$ where n is the number of time series in the bank.

There are two fundamental ways in which the similarity between two time series S_1 and S_2 can be calculated: (1) Euclidean Distance (ED) calculation and (2) Dynamic Time Warping (DTW). The first requires linear alignment hence both time series must be of equal length. DTW matches time series sequences by "warping" them in a nonlinear fashion (hence the name). Therefore, the time series under consideration do not need to be of the same length [11, 72, 100]. DTW coupled with kNN has been used with

respect to time series classification over many decades [92]. Many research studies directed at time series classification have demonstrated that DTW outperforms ED similarity measurement in terms of accuracy (although not in run time) [65, 87, 100, 109]. DTW is thus a popular choice because of its associated accuracy and because it supports comparison of time series of different length. Example applications include speech recognition, robotics, finance, medicine, image processing, music processing, and gesture recognition. There are many more.

However, DTW has a significant disadvantage, compared to ED measurement, which is its quadratic complexity. Given two time series, $S_1 = [p_1, p_2, \ldots, p_x]$ and $S_2 = [q_1, q_2, \ldots, q_y]$, where x and y are the lengths of S_1 and S_2 respectively; DTW operates by creating a $x \times y$ "distance matrix" and finding the minimum "warping path" from the points at either end of the leading diagonal. The computational complexity of one application of DTW is therefore given by $O(x \times y)$, while that of ED is given by $O(x)$ or $O(y)$ (recall that ED requires that $x = y$). Note therefore that the computational complexity of DTW increases quadratically as the time series size increases, while the complexity of ED increases linearly as the time series size increases.

1.2 Motivation

From the foregoing, the main motivation for the work presented in this thesis is the time complexity associated with the application of DTW. This therefore serves to limit its application in terms of the size of the time series data sets that it can be applied to. The overriding objective of the work presented in this thesis is thus to reduce this complexity without adversely affecting the accuracy of its application. The fundamental idea is to make the use of DTW both effective and efficient with respect to time series data. The work presented is therefore motivated by the observation that, as noted above, DTW has a quadratic complexity with respect to the size of the time series being compared. In other words, the complexity of DTW increases as the time series considered get longer. The intuition underpinning the work presented in this thesis is that DTW complexity can be reduced by using novel mechanisms not considered previously in the literature.

1.3 Research Question

From the foregoing, DTW is a well-established time/point series similarity checking mechanism which has been used effectively. It has been adopted in many domains, such as; speech recognition [86], time series data mining [15, 34], music analysis [79] and pattern recognition [121]. It is also worth noting that the DTW concept has similarities with Levenshtein Distance calculation; also referred to as Edit Distance, used for measuring the similarity between two strings. However, in this case, the values used are the number of deletions, insertions or substitutions required to transform the first string into the second.

The overriding research question that this thesis seeks to address is therefore:

How can the process of dynamic time warping be applied so that time series can be more effectively and efficiently compared?"

To provide an answer to this research question, a number of subsidiary research questions were considered:

1. Given that DTW operates using a distance matrix what are the most appropriate mechanisms, in addition to those described in the literature, for limiting the size of this matrix?
2. Is there any advantage to be gained, in terms of efficiency, from segmenting the distance matrix into a set of sub-matrices?
3. Is it possible to utilise knowledge of a given application domain to limit the DTW processing required. For example, in the case of kNN classification, using early abandonment with respect to time series that are clearly not going to be very similar to the query time series?
4. Is it possible to only consider sub-sequences that are repeated in instances (records) of the same class, so that DTW can be applied only to such sub-sequences instead of the entire time series?
5. Is it possible to identify a mechanisms for transforming time series from their original form and apply DTW to this new form?
6. Assuming that the answers to the above entail the use of parameters of various kinds, how can these parameters be optimised?

1.4 Research Methodology

The research methodology, designed to provide answers to the above research questions and to address the central motivation of this thesis, is presented in this section. To act as a focus for the research it was decided to concentrate on time series classification using kNN with $k = 1$, because: (i) classification is a common time series analysis application and (ii) kNN with $k = 1$ was, and remains, the most frequently used time series classification model [65, 87, 100, 109].

The start point for the research was to generate a baseline "Standard" DTW classification mechanism and analyse its operation. The Standard DTW implementation was evaluated with respect to fifteen time series data sets taken from the UEA and UCR (University of East Anglia and University of California Riverside) Time Series Classification Repository [14]. The evaluation metrics used were accuracy, F1-Score and run time. The same evaluation metrics were used with respect to all evaluations reported in the thesis. The complexity of the Standard DTW was considered using big O notation.

From the literature, a frequently adopted technique for reducing the time complexity of DTW is to use what is known as a "warping window"; a mechanism for reducing the overall search space within the distance matrix. There are a number of mechanisms whereby the warping window idea can be realised; one of these, the Sakoe-Chiba Band, was selected for further evaluation. The idea was to use the results from this evaluation as a further benchmark with which to evaluate and compare the approaches proposed later in the thesis. in terms of run time, accuracy, and F1 score.

Once the two benchmark approaches had been established the intention was to investigate a series of alternatives whereby the complexity of DTW could be reduced without adversely affecting accuracy. A number of alternatives were considered:

1. **Vertical Segmentation**: Tabular data can be segmented either horizontally by grouping sequences of rows (records), or vertically by grouping sequences of columns. The first idea to be considered was vertical segmentation splitting a time series into sub-sequences and processing those sub-sequences individually, thus reducing the overall size of the DTW distance matrix.

2. **Reducing The Number of Candidates**: Reducing the number of DTW comparisons required, so that the DTW process does not have to be applied to the entire data set. Only the time series which are likely to be good matches will thus be considered in the DTW process.

3. **Time Series Transformation**: Instead of applying the DTW to entire time series, time series could be transformed to a new form (shorter than the original), to which DTW could be applied more efficiently.

1.5 Contribution

A number of contributions are presented in this thesis with respect to machine learning in general, and time series classification in particular. The main contributions of this thesis are a number of approaches for reducing the complexity of DTW while maintaining effectiveness, as follows:

1. Sub-Sequence-Based DTW (SSBDTW): A first benchmark data segmentation approach founded.

2. Fuzzy Sub-Sequence-Based DTW (FSSBDTW): A refinement of the SSBDTW approach.

3. Candidate Reduction Based on Euclidean Distance (CRBED): Use of the concept of Euclidean Distance similarity measurement to reduce the number of applications of DTW.

4. Candidate Reduction Based on Lower Bounding (CRBLB): Use of the concept of lower-bounding reduction to reduce the number of applications of DTW.

5. Exact Discriminator-Based DTW (EDBDTW): An approach directed at finding a sub-sequences, called discriminators, based on a class, so they can be used to classify a new time series instead of using the entire time series.

6. Distance Profile-Based (DPBDTW): An approach influenced by the concept of the "Matrix Profile" idea, where time series are transformed to a new form, namely a distance profile format.

The main advantages of the above can be itemised as follows:

1. Reducing the time complexity of the Dynamic Time Warping (DTW) process, so that the run time is reduced compared to Standard DTW and warping window mechanisms such as the Sakoe-Chiba Band mechanism.

2. Efficiency gains with respect to both short and long time series. Although the efficiency advantages were more relevant with respect to longer time series.

3. Effectiveness gains in that (interestingly), the reported evaluations also indicated enhancement in the classification effectiveness, measured using accuracy and F1-score, with respect to both long and short time series data sets. In some cases accuracy reached 100%.

4. Unlike most work in the literature, which is directed at improving DTW performance in only one direction, either reducing the number of points or the number of records to be considered, the mechanisms presented in this thesis were directed at both.

1.6 Publications

A number of academic papers, presented in this thesis, have been published as follows:

1. *Alshehri, M., Coenen, F. and Dures, K. (2019). Sub-Sequence-Based Dynamic Time Warping. Proc 11th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2019), Volume 1: Knowledge Discovery and Information Retrieval (KDIR'19), pp274-281.*

   This paper proposed the initial idea of SSBDTW. Given two time series S_1 and S_2 these are divided into s sub-sequences so that we have $S1 = [U_{1_1}, U_{1_2}, ...U_{1_s}]$ and $S2 = [U_{2_1}, U_{2_2}, ...U2_s]$. DTW is then applied to each sub-sequence paring U_{1_i}, U_{2_j} where $i = j$. The final minimum warping distance arrived at was the accumulated warping distance for all sub-sequences after s application of DTW. The operation of the proposed SSBDTW was compared with the operation of Standard) DTW. More detail concerning SSBDTW presented In Chapter 4.

2. *Alshehri, M., Coenen, F. and Dures, K. (2019). Effective Sub-Sequence-Based Dynamic Time Warping. Accepted for publication at AI-2019, Thirty-ninth SGAI International Conference on Artificial Intelligence.*

   This paper presented a number of variations on SSBDTW by presenting and evaluating a sequence of mechanisms to enhanced the splitting process. It was conjectured that a rigid definition of s, number of sub-sequences resulted from the splitting process, might not result in the best segmentation. It was found that good points at which to cut (fragment) two time series to be compared was where they meet (same value), or at least at points where the distance between values of corresponding pairs of points was at a minimum. Thus a degree of fuzziness should be included to derive the best fragmentation. More detail concerning the mechanism presented in this paper are included in Chapter 4.

3. *Mohammed Alshehri. M., Coenen, F. and Keith Dures (2020). Candidates Reduction Coupled with Enhanced Sub-Sequence-Based Dynamic Time Warping. Accepted for publication at the BCS SGAI AI conference, AI2020.*

   This paper presented one of the proposed candidate reduction mechanisms, presented later in this thesis in Chapter 5, directed at limiting the number of DTW process that needed to be applied when conducting kNN time series classification. A two phase approach were proposed: (i) use of Euclidean Distance (ED) as the distance measure to find the most similar candidates given that ED measurement is significantly cheaper than DTW, and (ii) processing of the identified candidates using FSSBDTW (one of the splitting mechanisms proposed in the thesis). More detail will be presented in Chapter 5.

4. *Alshehri, M., Coenen, F. and Dures, K. (2021). Motif-based Classification using Enhanced Sub-Sequence-Based Dynamic Time Warping. In Proceedings of the 10th International Conference on Data Science, Technology and Applications - DATA.*

   This paper presented two approaches to reformatting time series to a new form (i) based on a discriminator approach and (ii) based on a distance profile approach. Both approaches changed the time series in a data set to a shorter form. The new form can then be processed using FSSBDTW. More detail will be presented in Chapter 6.

1.7 Thesis Structure

The rest of the thesis is organized as follows:

Chapter 2: Background and Related Work.
The background and the related work to the work presented in this thesis are reviewed in this chapter. The chapter starts with a review of Dynamic Time Warping similarity measurement. This is followed with a review of the related work with respect to the time series analysis approaches and mechanisms considered throughout the work presented in the thesis.

Chapter 3: Evaluation Data sets and Benchmark Approach.
In this chapter, a detailed description is given of the fifteen time series data sets used with respect to the evaluation reported on in this thesis. In addition, an analysis of the two time series classification benchmarks, used later in this thesis for comparing alternative proposed approaches, is presented: (i) Standard DTW and (ii) Sakoe-Chiba Band DTW (S-C Band DTW). These are evaluated using the fifteen evaluation time series data sets introduced in the previous chapter

Chapter 4: Sub-Sequence-Based Dynamic Time Warping Approaches.
Two alternative, segment-based DTW approaches are considered in this chapter: (i) Sub-Sequence-Based DTW (SSBDTW) and (ii) Fuzzy Sub-Sequence-Based DTW (FSSBDTW). The latter featuring a number of improvements over the first. Both adopted the idea of vertically fragmenting the input data. A comparative evaluation is included in the chapter.

Chapter 5: Candidate Reduction Based in Euclidean distance and Lower Bounding techniques.
In this chapter, the idea of "candidate reduction" is investigated in the context of FSSBDTW. The idea is to limit the number of applications of DTW by reducing the number of records that need to be considered. In other words, so that DTW only needs to be applied to the most similar records. Two candidate reduction techniques, incorporated into FSSBDTW, are proposed: (i) Candidate Reduction Based on Euclidean Distance (CRBED) and (ii) Candidate Reduction Based on Lower Bounding (CRBLB). A comprehensive comparative evaluation is included in the chapter.

Chapter 6: Time Series Data transformation approaches.
The work presented in this chapter is founded on the concept of data transformation with respect FSSBDTW. The main idea is to transform the time series in a data set to a new form (shorter than the original) to which FSSBDTW can then be applied. Two transformation-based techniques are presented: (i) Exact Discriminator-Based DTW (EDBDTW) and (ii) Distance Profile-Based (DPBDTW). The chapter is concluded with a comparative analysis of the two techniques.

Chapter 7: Conclusion and Future Work.
This final chapter concludes the work presented in this thesis. The chapter summarises the main findings of the work with respect to the research question, and subsidiary questions presented in this chapter. The chapter also proposes a number of recommendations regarding possible directions for future work.

1.8 Summary

This introductory chapter has presented the background and motivation for the work presented in this thesis, together with the research question and subsidiary research questions that the thesis seeks to address. The chapter also included a review of the adopted research methodology, and listed the contributions made by the thesis and the publications produced as the work progressed. The chapter was concluded with a review of the structure of the remainder of the thesis. The following chapter will provide a literature reviews covering related work to that presented in this thesis.

Chapter 2

Literature Review

2.1 Introduction

As noted in Chapter 1, the work conducted in this thesis is directed at speeding up Dynamic Time Warping (DTW) similarity measurement. To provide context for the work, the selected focus was time series classification using the kNN algorithm, an algorithm frequently used with respect to time series classification and which features significant amounts of similarity measurement, and hence was deemed to provide an excellent vehicle for the proposed study [50, 91, 106, 117]. This chapter, therefore, commences by presenting the necessary background concerning time series classification (Section 2.2) and then goes on to consider time series similarity measurement (Section 2.3) categorising these as either: (i) lock-step measures and (ii) elastic measures. Section 2.3.1 gives a detailed description of the Dynamic Time Warping (DTW) algorithm. Section 2.3.2 presents a comprehensive review of previously proposed approaches, from the literature, directed at speeding up DTW using the concept warping windows, either: (i) predefined or (ii) learnt. Local, as opposed to global, warping windows are also considered. Section 2.3.3 discusses limiting the number of DTW calculations rather than limiting the size of the distance matrix as in the case of warping window techniques. A number of techniques are listed of which candidate reduction is the most significant, Candidate reduction is thus considered in further detail in Section 2.3.4. The chapter is concluded with a discussion of mechanisms used to reformat time series data in Section 2.3.5. A summary of the work presented in this chapter is then given in Section 2.3.6

2.2 Time Series Classification

Time series classification is broadly the process of labelling a previously unseen time series with a class label c taken from a predefined set of classes $C = \{c_1, c_2, \dots\}$. In other words, given an unlabelled time series, the classification task is to label the time series [7, 125]. The classification is conducted using a model constructed from a pre-labelled set of examples, often referred to as training data or simply the training set [42, 45]. Hence we categorise the process of generating a classification model as a form of supervised learning (the opposite is unsupervised learning, but unsupervised learning is not relevant to this thesis) [7, 35]. The objective of classification model generation is to build a model that maximises classification accuracy, this is achieved by focusing on features or patterns within the training data that serve to distinguish between classes [7]. There are a range of mechanisms available that can be used to generate time series classification models; frequently referenced examples include: (i) Decision Trees [16], (ii) Support Vector Machine [25], (iii) Deep Learning [37] and (iv) k-Nearest Neighbour [28].

These are considered in some more detail in the following four sub-sections, Sub-sections 2.2.1 to 2.2.4.

2.2.1 Decision Tree (DT)

The oldest form of time series classification model generation mechanisms is, arguably, the Decision Tree (DT) algorithm. The main idea of this classification model is to divide the training data in a hierarchical manner to form a tree structure. The root and body nodes are designed to direct the classification resolution process to a leaf node that hold a class value [7, 94]. Further details concerning DT algorithms can be found in [105]. The challenge when generating DTs is defining the splitting criteria at the root and body nodes, this can be particularly challenging with respect to time series data. Example where decision trees have been used for time series classification can be found in [21, 84]. The drawbacks of using DT can be listed as follows: (i) unlike categorical data, DT does not perform effectively when considering large continuous numerical data (ii) with respect to some data sets calculating the splitting criteria can be complex resulting along model training times, and (iii) a small change in the data tends to cause instability because it can cause a need for change in the tree structure [103, 107].

2.2.2 Support Vector Machines (SVM)

Support Vector Machine (SVM) is a widely used time series classification algorithm [53, 105]. SVM uses positive and negative discrimination to classify data to perform binary classification; two classes, a positive class and a negative class [19]. In linear cases, the SVM model "maps" data using a high-dimensional feature space. SVM separates between classes by using a linear hyperplane which has margins equally distanced between the classes and the hyperplane [7]. In non-linear cases, a Kernel function, which separates between negative and positive classes using a nonlinear boundary, is utilized [7]. Further details concerning the SVM algorithm can be found in [19]. In the context of time series classification SVMs have been used in time series forecasting [63] and used with DTW to classify normal and pre-seizure electroencephalograms as in [57]. The disadvantages of using SVMs can be listed as follows: (i) SVMs do not perform well with overlapped classes, (ii) sufficient generalization performance is required for selecting appropriate hyperparameters, and (iii) SVMs are slow, compared to kNN models, with respect to large time series because they require a large amount of time to process. [2, 25].

2.2.3 Deep Learning (DL)

Deep learning, facilitated by the computer power that is currently available, has revolutionised the field of machine learning. The take up with respect to time series classification has been less pronounced. In deep learning, Long Short-Term Memory (LSTM), an artificial Recurrent Neural Network (RNN) architecture, is used typically [39, 119]. It is used for processing entire sequences of data, such as video or speech data, as well as two dimensional data such as image data. LSTM consist of 4 units: (i) a cell, (ii) an input gate, (iii) an output gate and (iv) a forget gate; where the cell remembers values and the three gates regulate the flow of information [97]. In the context of time series data, LSTM networks are very useful for classification, processing and prediction [52, 131]. The main challenge of deep learning is the amount of training data that is required, compared to DT and SVM models, which renders it inappropriate for many time series

classification applications. For this reason, LSTM has not be considered for comparison purposes in this thesis.

2.2.4 k-Nearsest Neighbors (kNN)

The most commonly used time series classification method is k Nearest Neighbour (kNN) classification [113]. The idea is to use a set of pre-labelled examples as a data bank (a data repository), D, comprised of r examples each associated with a class c taken from C. A new time series to be labelled is then compared with every time series in D and the labels associated with the k most similar time series used to label the new time series. Where $k > 1$ there is a possibility of conflict, in which case a conflict resolution mechanism, such as voting, is required. When $k = 1$ this issue does not arise hence 1NN is the most popular form of kNN classification. Further detail concerning the kNN algorithm can be found in [103]. Given its popularity, and its ease of implementation and use, 1NN was adopted with respect to the evaluations presented in this thesis. Any of the other classification algorithms listed above could have been used for the comparative evaluation; however, 1NN allows for straight-forward complexity calculation. A further reason for adopting 1NN is that there is evidence to suggest that 1NN outperforms other time series classification algorithms [65, 100, 109].

2.3 Similarity Measures for Time Series

In time series analysis, finding the similarity between time series is a significant task. This is specifically the case with respect to time series classification and clustering that require many comparisons [112]. Selecting the most appropriate similarity measurement depends on the nature of the data [1, 87]. Exact matching between two time series is rare; however, looking for approximate matches is a common requirement [8, 41]. This can best be illustrated by considering a 1NN classification scenario, Figure 2.1 shows a database D holding four time series $\{S_1, S_2, S_3, S_4\}$ and a new time series (the query time series) to be labelled according to the labelling in D. The best match is time series S_3 and this label is used to label the new time series. From the figure, it can be seen that this is a "closest" match and not an "exact" match.

Finding the similarity between time series can be categorized according to the requirements of the application under consideration. Three categories of approach can be identified:

1. **Auto-correlation-based approach**: Measuring the similarity of a time series and a lagged version of itself [58]. For example, measuring the relationship between current values of a time series and its previous values to identify (say) a change in trend [78].

2. **Correlation-based approach**: Measuring the relationship between two time series based on time. For instance, whether two time series change in a similar way as they progress in relation to one another [74].

3. **Shape-based approach**: Measuring the similarity based on shape. For example, how a time series is similar to another in its shape [78].

The last two can also be distinguished by categorising them as dynamic and static approaches. The last, the Shape-based approach, is the most frequently encountered and therefore used with respect to the work presented in this thesis.

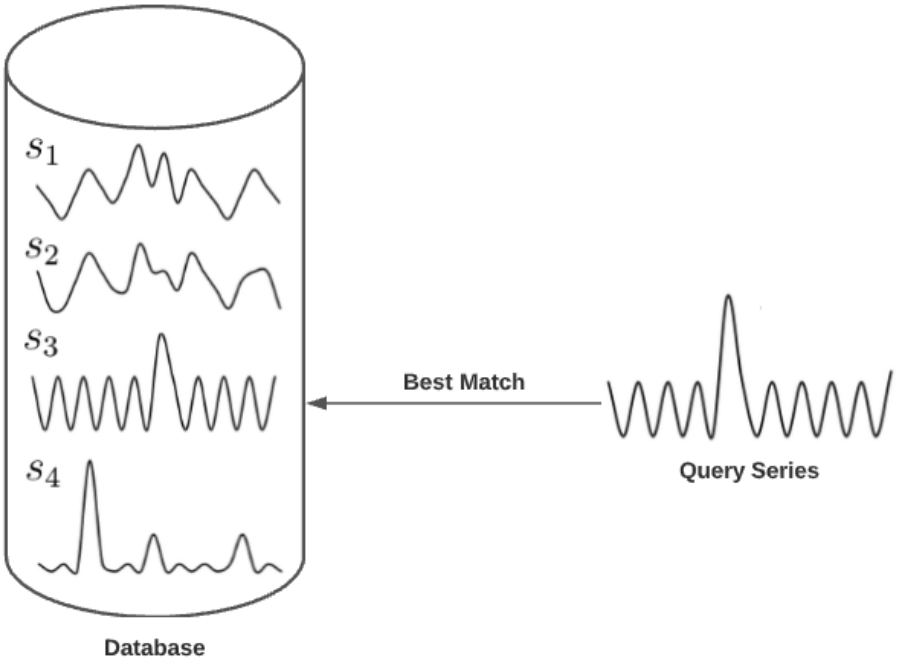

FIGURE 2.1: Time Series Similarity [8].

From the literature, distance measures used to determine the similarity between two time series, $S_1 = [p_1, p_2, \ldots, p_x]$ and $S_2 = [q_1, q_2, \ldots, q_y]$ (where p_i and q_j are individual values in the time series, and $x \in \mathbb{N}$ and $y \in \mathbb{N}$ are the time series lengths), can be categorized (at a high level) in terms of the nature of the measure used: (i) Lock-step Measures, and (ii) Elastic Measures. Each is considered in further detail in the following two sub-sections.

Lock-step Measure

The lock-step measure is used to calculate the distance between points in a one-to-one manner. Given two time series $S_1 = [p_1, p_2, \ldots, p_x]$ and $S_2 = [q_1, q_2, \ldots p_x]$, both of length $x \in \mathbb{N}$, the distance will be calculated between each point of S_1 with the corresponding point in S_2 [118]. A common example of such a measure is Euclidean Distance (ED), where similarity d_E is measured as shown in Equation 2.1, the square root of the sum of the squares of the differences between corresponding points in the two time series [12, 65].

An alternative might be Manhattan distance [26]. Figure 2.2 provides an illustration of the Lock-step measure approach.

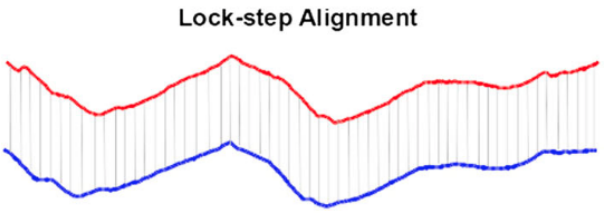

FIGURE 2.2: Lock-step Measure Example.

Lock-step measures offer the advantage of comparative computation efficiency (compared to Elastic measures) because they operate in a linear manner. Therefore, the

complexity of calculating the similarity between two time series, using "Big O" notation, can be expressed as $O(x)$, where x is the length of the time series [121]. However, Lock-step measures have a weakness in terms of accuracy with respect to finding the similarity between two time series based on shape, this weakness is because of the one-to-one mapping between the points in the two time series. In other words, local time-shifting, where the time series shape can hint at similarity, cannot be captured [8]. An additional disadvantage is that lock-step measures only work with time series of the same length [12, 87].

$$d_E = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{2.1}$$

Elastic Measure

Elastic measures provide more flexibility compared to Lock-step measures. The flexibility is derived from the approach that elastic measures adopt, which is a one-to-many or many-to-one mapping between points in a pair of given time series S_1 and S_2 [98]. Therefore, local time shifting can be captured simply [8]. Elastic measures, therefore, provide greater accuracy with respect to finding the similarity between time series. Figure 2.3 illustrates the mapping approach adopted by Elastic measures.

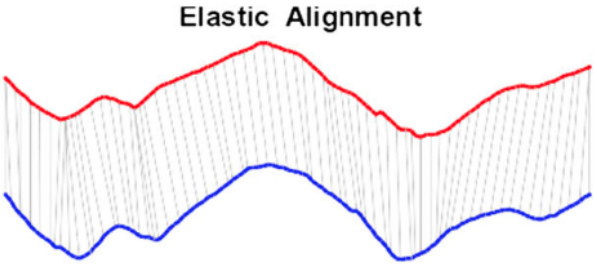

FIGURE 2.3: Elastic Measure Example.

However, elastic measures have a disadvantage, in comparison with Lock-step measures, in terms of their time complexity since they are quadratic in nature [40, 48]. Therefore, the complexity of calculating the similarity between a time series $S_1 = [p_1, p_2, \ldots, p_x]$ of length $x \in \mathbb{N}$ and a time series $S_2 = [q_1, q_2, \ldots, q_y]$ of length $y \in \mathbb{N}$, and again using "Big O" notation, can be expressed as: $O(x \times y)$, or if $x = y$, $O(x^2)$ [77].

The most frequently encountered example of Elastic measure is the warping distance measure which is derived from the Dynamic Time Warping (DTW) process [76, 93]. The work presented in this thesis is therefore directed at addressing the time complexity of DTW. Thus, further details regarding DTW are given in the following sub-section, Sub-section 2.3.1.

2.3.1 Dynamic Time Warping

DTW is founded on the idea of identifying an optimal alignment between two time series which may be of different lengths [99, 109]. Unlike Lock-step similarity measurement, DTW matches time series sequences by "warping" them in a nonlinear fashion (hence the name). DTW was first proposed in the context of speech recognition for the comparison of speech patterns [92]. Subsequently, it has been used for many other applications, for example, music analysis [32], image processing [17], medicine analysis [49], gesture recognition [111] and weather forecasting [67].

The DTW process can best be described by considering two time series $S_1 = [p_1, p_2, \ldots, p_x]$ and $S_2 = [q_1, q_2, \ldots, q_y]$, where x and y are the lengths of the two series respectively and $x, y \in \mathbb{N}$. The first step is to construct a "distance matrix" M of size $x \times y$ where the value held at each cell $m_{i,j} \in M$ is the summation of $d_{i,j}$ and the minimum cumulative distance value held at the three "previous" cells to $m_{i,j}$ [81], where $d_{i,j}$ is a Euclidean distance between the corresponding points, see Equation 2.2; and the calculation of $m_{i,j}$ utilizes Equation 2.3. At the end of the process, the minimum warping distance (wd) will be held at $m_{x,y}$.

$$d_{i,j} = \sqrt{(p_i - q_j)^2} \tag{2.2}$$

$$m_{i,j} = d_{i,j} + min\{m_{i-1,j}, m_{i,j-1}, m_{i-1,j-1}\} \tag{2.3}$$

An alternative to Euclidean distance might be absolute value distance calculation.

The distance matrix M is used to determine a minimum warping distance wd, which is then used as a similarity measure. The warping distance is a function of the minimum warping path, WP, from cell $m_{0,0}$ to cell $m_{x,y}$. A minimum warping path is thus a sequence of cell locations, $WP = [w_1, w_2, \ldots]$ in the matrix M, that minimises the warping distance. Given two time series $s_1 = [p_1, p_2, \ldots, p_x]$ of length $x \in \mathbb{N}$ and $s_2 = [q_1, q_2, \ldots, q_y]$ of length $y \in \mathbb{N}$, and using "Big O" notation, the complexity of DTW can be expressed as: $O(x \times y)$, or if $x = y$ $O(x^2)$. Thus DTW becomes computationally expensive when x and/or y are large [109]. This quadratic complexity therefore renders DTW to be impractical with respect to many application domains.

From the foregoing, it can be seen that the operation of DTW is such that it meets the following conditions in its warping path [100]:

1. **Boundary condition**: The path starts from the location $m_{(0,0)}$ and ends at the location $m_{(x,y)}$.
2. **Monotonic condition**: The warping path will stay the same or increase. Both the i and j indexes never decrease.
3. **Continuity condition**: The path continues one step at a time. Both i and j can only increase by 1 on each step along the path. Therefore, the follow-on location is either $m_{i-1,j}, m_{i,j-1}$, or $m_{i-1,j-1}$.

The above is illustrated in Figure 2.4 [79]. Given two time series $S_1 = 1, 2, 3, 4, 5, 6, 7, 8, 9$ and $S_2 = 1, 2, 3, 4, 5, 6, 7$. The figure comprises four sub-figures as follows: (a) warping path meets all the conditions, (b) warping path where the first condition is ignored, (c) warping path where the second condition is avoided, and (d) warping path where the third condition is discarded.

The basic DTW process is illustrated in Figure 2.5. The figure shows the distance matrix M assuming two time series, $S_1 = [1, 2, 2, 3, 2, \ 1, 1, 0, 1, 0, 3, 2, 4, 2, 0]$ and $S_2 = [1, 2, 4, 3, 3, 0, 3, 3, 1, 2, 1, 1, 3, 4, 2]$. The minimum warping path is shown by the red line. The final warping distance arrived at is highlighted using a green box.

2.3.2 Warping Window Approaches

Given the above, the use of DTW, despite its claimed accuracy, can be expensive, especially when having large time series collections and/or very long time series. One approach that has been proposed to address this issue is to define a "warping window" [36]. The idea is founded on the observation that, given a classification or clustering

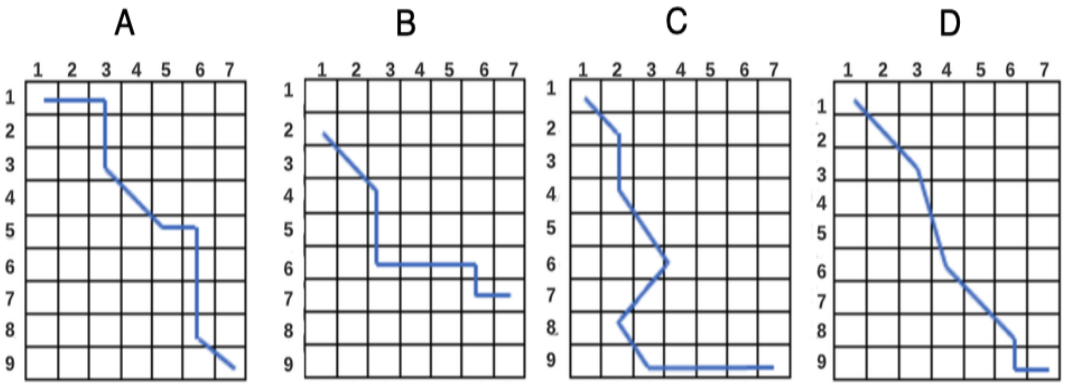

FIGURE 2.4: Warping Path Conditions of Dynamic Time Warping.

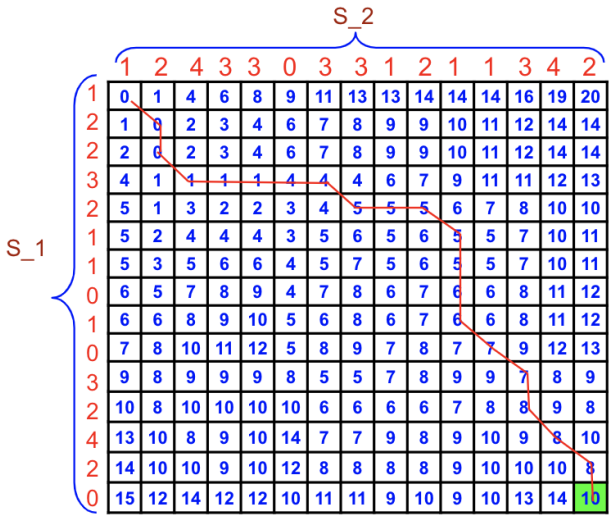

FIGURE 2.5: Dynamic Time Warping Matrix Example.

scenario, we are looking for best matches and those will occur when we have a warping path that is on, or is close to, the leading diagonal in M; hence any effort calculating cell values towards the ends of the antidiagonal are wasted [30]. The warping window is thus the collection of cells $\lambda \subseteq M$ for which values should be calculated, a default value is used otherwise [31]. In other words, the warping window idea places constraints on the matrix area to be considered when calculating a minimum warping path [88]. The question then is how to define the warping window λ. Two broad categories of technique can be identified:

1. **Predefined**: Techniques where the nature of the warping window is predefined using one or more parameters (global constraints).

2. **Learnt**: Techniques where the nature of the warping window is learnt using training data.

Both are discussed in further detail below. The use of a warping window λ thus defines a constrained area inside the matrix M for which cell values need to be calculated. In addition, it prevents any pathological alignment by forcing the warping path to remain inside the constrained warping window area.

Both the predefined and learnt warping window techniques define a global warping window to be used in all cases given a particular application. An alternative is to define a

bespoke warping window for each pair of time series to be considered. This is considered in further detail at the end of this Sub-section.

The computational complexity of DTW is dependent on the size of the distance matrix M. Warping window approaches reduce the overall size of M. The speed up gained using warping window approaches is therefore in the proportion to the reduction in the size of M. Of course this should not be achieved at the expense of accuracy, which should remain the overriding consideration. The speed up is in the order of:

$$O(|M| - |M'|) \tag{2.4}$$

where M is the size of the distance matrix before the application of a warping window approach and $|M'|$ is the size of the distance matrix after the application of a warping window approach.

Predefinition Warping Window Techniques

Predefined warping windows use one or more global constraints to define their shape. The simplest mechanism for expressing a warping window is to define a band, of width ℓ, stretching from $m_{0,0}$ to $m_{x,y}$ (given two time series $S_1 = [p_1, p_2, \ldots, p_x]$ and $S_2 = [q_1, q_2, \ldots, q_y]$). From the literature, the most well-documented example of this approach is the Sakoe-Chiba band [43, 44, 92], originally introduced and used by the speech analysis community. In [92] it was suggested that that the value for ℓ defining the band width should be set to 10% of the time series length. However, it has been demonstrated that, with respect to some data sets, better accuracy can be obtained using a different value for ℓ than 10% of the time series length [69]. A disadvantage of the Sakoe-Chiba band idea is therefore the need to identify the best value for ℓ.

An alternative to using a warping window in the shape of a band is to use a parallelogram thus avoiding unnecessary calculation at the start and end of the warping path. The best-known example of this is the Itakura parallelogram where the warping window λ is defined by two slope constraints [55, 96]. A disadvantage is therefore the need to identify values for the slope constraints.

The Sakoe-Chiba band and the Itakura parallelogram are illustrated in Figure 2.6. Both offer the advantage that they are simple to implement [43]. Both also feature the disadvantage that, for the same parameter settings, the size of the warping window is directly related to the size of the time series to be compared. Therefore, given a very long time series, where the warping window idea should be of particular relevance, the warping window can be very large [69].

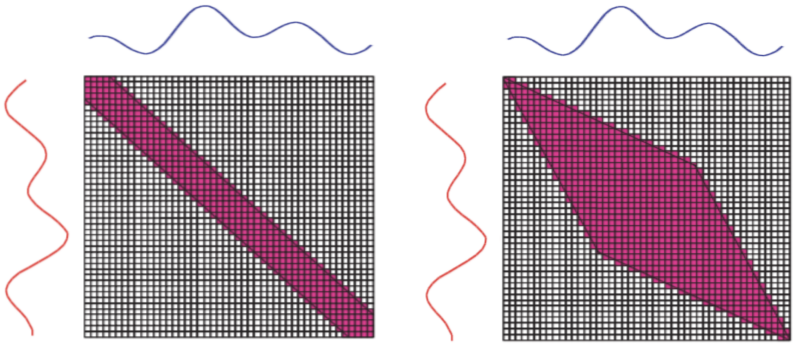

FIGURE 2.6: Left: The Sakoe-Chiba band, Right: The Itakura Parallelogram [81].

Learnt Warping Window Techniques

The predefinition of a warping window requires the user to, more-or-less, guess at the required definition of the window; users thus tend to err on the side of caution. A more accurate way of defining the warping window is to use a machine learning approach, although this requires training data. The idea of learning the nature of the warping window was first proposed in [81] in the context of time series classification. The idea here was to produce an arbitrarily shaped window. This was defined by considering each class in the training set in turn and identifying the minimum warping path for each pair of time series subscribing to that class. The collected warping paths for each class then defined a "warping sub-window". These sub-windows were then merged to define a global warping window. The approach is illustrated in Figure 2.7 where the training set features three classes (red, blue and green) whose associated warping sub-windows are merged to form a global window. A disadvantage of learnt warping window techniques is the computational overhead associated with the training process; this can be substantial, especially given a large number of time series and/or a large number of classes increases. The advantage is that the derived warping window is tuned to a particular application as defined by a given set of time series.

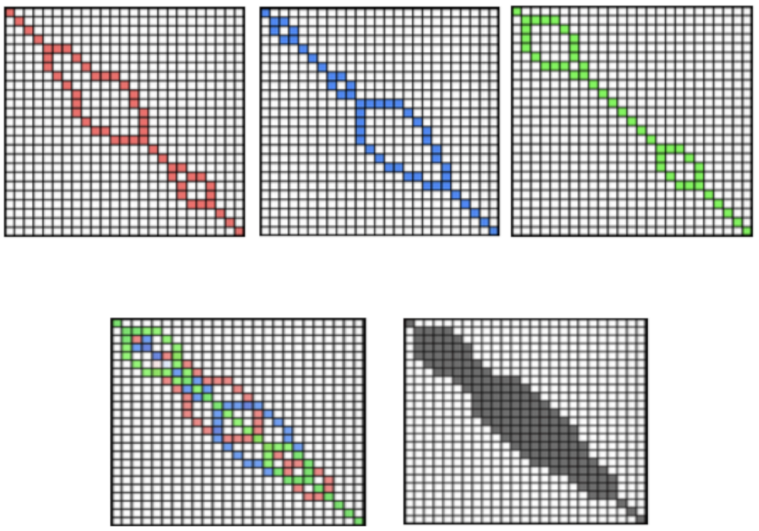

FIGURE 2.7: Warping Window learning example using three individual classes (red, blue, green), gray is the combined global window. [81].

Local Warping Windows Techniques

The warping window idea, described in the foregoing section, is used to define an area of the DTW matrix where calculations do not need to be made, hence speeding up the DTW process [126]. The warping window once calculated is applied globally to all time series to be compared. Using kNN classification this will be the time series held in the kNN bank D and a query time series Q ($D = \{T_1, T_2, \dots\}$. An alternative is to consider the warping window in terms of a particular pair of time series to be compared. In other words tailoring the warping window with respect to each comparison. One example can be found in [99] where Silva et al. proposed a method to speed up DTW known as PrunedDTW. The fundamental idea was to place upper bounds on the calculation process. The distances along the prime diagonal, from $m_{0,0}$ to $m_{x,y}$, were first calculated using the squared Euclidean distance. These distances were considered

to be "upper bounds". Then for each point in the diagonal the distances along each row were calculated moving away from the diagonal, in row and column order, until a distance greater than the current upper bound is reached, further cells are then outside of the warping window. Interestingly, in [99], an example is given where parts of the leading diagonal are outside of the warping window.

2.3.3 Limiting DTW Calculation

In the context of limiting the number of DTW comparisons with respect to kNN time series classification where a new time series to be classified is compared to a "bank" of time series, Rakthanmanon et al. [87] reported on four different techniques for achieving this:

1. Early abandonment.
2. Reordering.
3. Candidate reduction using bounding.
4. Candidate reduction using cascaded bounding.

The idea of underpinning early abandonment is to stop the warping path calculation if the *wd* value so far is equal to or larger than the best so far; otherwise, the new value is the best so far. The second promotes the idea of reordering the time series in the bank so that the time series that are likely to be the most similar to the new time series are tested first so that the early abandonment process will result in less calculation than if the time series were not ordered in this way. One way of ordering time series is according to Euclidean distance similarity (much cheaper than DTW calculation). The third considered pruning time series from the bank that was unlikely to be a close match. One way of doing this is by using the lower bounding technique proposed in [59], the so called LB_Keogh technique. This operates by creating an envelop lower bound and upper bound) around the new time series, and the distance between the bounds and times series from the bank will be calculated. Where the calculated value exceeds a given threshold the associated time series is discounted. The fourth idea was directed at using a "cascading lower bound" where different lower bounds are considered to identify the bound most suitable for the data set in question. The idea of candidate reduction was incorporate into two of the proposed approaches presented in this thesis, the Candidate Reduction Based on Euclidean Distance (CRBED) and the Candidate Reduction Based on Lower Bounding (CRBLB) approaches. Candidate reduction is therefore discussed in further detail in the following sub-section.

Time complexity savings resulting from early abandonment, and reordering coupled with early abandonment, are difficult to calculate. Much depends on the nature of the data set to be considered. However, in the worse case performance will be the same as the baseline DTW.

2.3.4 Candidate Reduction Techniques

From the previous sub-section, candidate Reduction is a technique used to reduce the number of DTW comparisons required when using kNN classification [51, 71]. The idea is to prune the set of time series to be considered so that the DTW process does not need to be applied on all the recorders in the kNN "bank"; only the k most similar time series to the query time series Q will thus be considered for classification purposes. The

most commonly adopted technique for candidate reduction is to use some form of Lower Bounding [64, 114, 115]. Thus the kNN time series classification becomes a two-step process for each comparison between Q and a time series T in the kNN bank:

1. Determine the lower bound for the set of time series in the bank and retain the lowest K time series.
2. Apply the DTW process to the retained time series to obtain a classification with respect to the query time series.

In the remainder of this sub-section, a number of lower bounding methods are considered, namely: (i) LB_Yi lower bound, (ii) LB_Kim lower bound and (iii) LB_Keogh lower bound.

The computational speed-up attained by candidate reduction techniques is proportional to the number of comparisons that do not have to be considered, given a classification scenario. The speed up is therefore in the order of:

$$O(r - n) \tag{2.5}$$

where r is the number of records in D and n is the number of records in D after candidate reduction has been applied.

Candidate Reduction Using LB_Yi Lower Bound

Yi et al. introduced a lower bounding method for DTW known as LB_Yi lower bounding [130] to be used in the context of kNN time series classification. Given a time series $T = \{t_1, t_2, \ldots t_x\}$ held in the kNN bank and a query time series Q, the LB_Yi value is calculated as shown in Equation 2.6 where: max and min are the maximum and minimum values in Q respectively [121]. This is done for all T in the the kNN bank and only the time series with the lowest LB_Yi value will have DTW applied. Figure 2.8 illustrates the calculation of the LB_Yi lower bound.

$$LB\_Yi = \sum_{i=0}^{i=x} abs((t_i \in T : t_i > max) – max + \sum_{i=0}^{i=x} abs((t_i \in T : t_i < min) – min \tag{2.6}$$

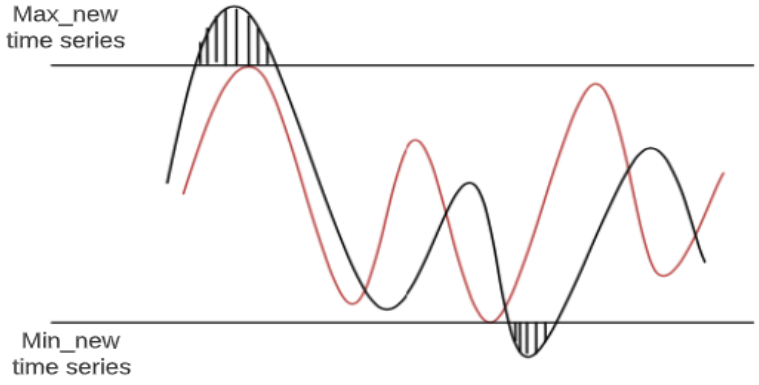

FIGURE 2.8: LB_Yi Lower Bound (red is the query (new) time series) [80].

Candidate Reduction Using LB_Kim Lower Bound

Kim et al. presented a DTW lower bounding method known as LB_Kim lower bounding [61]. The calculation of LB_Kim depended on eight values: (i) the values at the first index of T and Q, (ii) the value at the last index of T and Q, (iii) the maximum value in both T and Q and (iv) the minimum value in both T and Q, [80]. The sum of the four differences between these values was then the value for the LB_Kim lower bound. Only the time series on the kNN bank with the lowest LB_Kim value will have DW applied. Figure 2.9 illustrates the calculation of the LB_Kim lower bound.

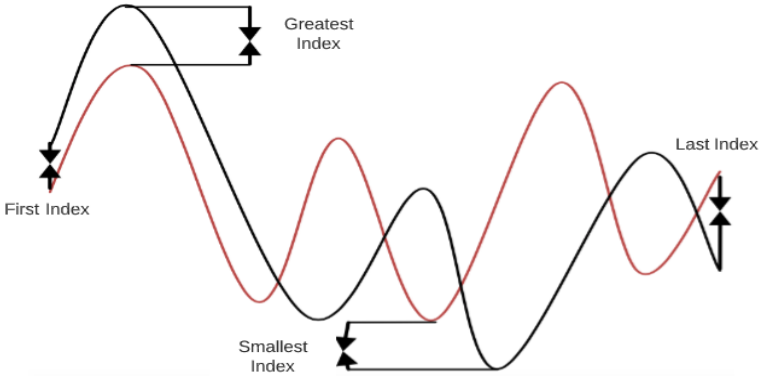

FIGURE 2.9: LB_Kim Lower Bound (red is the query (new) time series) [80].

Candidate Reduction Using LB_Keogh Lower Bound

Keogh et al. proposed a DTW lower bounding method known as LB_Keogh lower bounding [59]. The process of calculating LB_Keogh starts with defining an envelop or warping window (such as the Sakoe-Chiba Band [92] or Itakura Parallelogram [55]) of the form considered previously in Sub-section 2.3.2 that cover a new time series. The LB_Keogh value is then the square root of the sum of the squared difference between the upper and lower edge of the warping window for T [80]. Figure 2.10 illustrates the LB_Keogh lower bound calculation process.

LB_Keogh was used with respect to the Candidate Reduction Based on Lower Bounding (CRBLB) approach presented in Chapter 5. This was adopted because, in general, it performs well for most time series data sets [60, 110].

From the foregoing, the lower bounding methods described were all directed at reducing the number of records that need to be processed using DTW, so that there was no need to apply DTW to the majority of records in the kNN bank [80]. Even though lower bounding methods improved the performance of DTW in term of run time, the accuracy was not maintained, with respect to some data sets, when DTW was applied without lower bounding [80]. The reason for this lower accuracy performance was that some lower bounding methods used dimensionality reduction mechanisms to reduce the size of the time series to be considered.

2.3.5 Motif Discovery Mechanisms

A motif is a frequently occurring sub-sequence (pattern) in a given time series. The intuition is that, in the context of time series classification, motifs will be good indicators of class because they are frequently occurring [3, 6]. However, finding motifs is a computationally expensive task [70]. Research and experiments have been conducted using different algorithms directed at motif discovery, both exact and approximate [47]. This

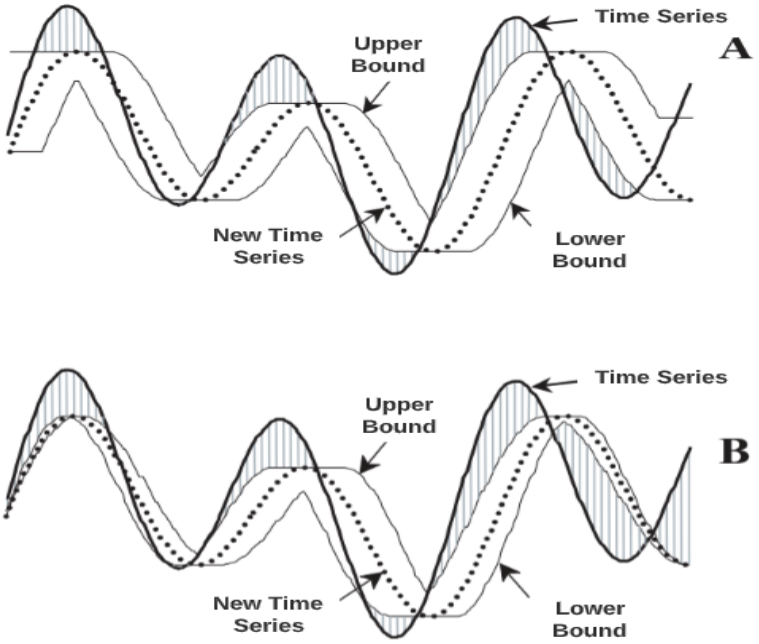

FIGURE 2.10: LB_Keogh Lower Bound, **A** when using Sakoe-Chiba Band and **B** when using Itakura Parallelogram [59].

section presents a review of a number of different motif discovery algorithms. Namely the MK Algorithm (Sub-section 2.3.5) and the Matrix Profile technique (Sub-section 2.3.5). The significance is that the idea of motifs, and especially, the context of matrix profiles influenced the proposed Distance Profile Based DTW (DPBDTW) approach presented in Chapter 6 of the thesis.

The expect of using motifs is to reduce the overall size of the time series to be considered. The speed up will be in the order of:

$$O(|M'|) \tag{2.7}$$

where $|M'|$ is the reduced size of the distance matrix as a result of using motifs instead of whole records.

MK Algorithm

The MK algorithm, named after Mueen-Keogh (MK) who first proposed the algorithm, is a sampling-based motif discovery algorithm. It is claimed to be the first algorithm that finds exact motifs in long time series [70]. The algorithm starts by randomly selecting a sub-sequence (usually the first points of the time series) of a predetermined length n [24]. Then, the similarity distance between the selected sub-sequence and other sub-sequences is obtained by sliding the sub-sequence along with the original time series and calculating the Euclidean distance between the selected sub-sequence and all other sub-sequences of length w excluding trivial comparisons [23, 47]. Finally, the lowest distance represents the motif which in turn is considered a good discriminator of a class.

Matrix Profile

Matrix profiles, as first introduced in [128], are used to find motifs and discords within time series. A Matrix Profile has two main components: (i) a distance profile and (ii) a profile index. The distance profile is constructed using a sliding window technique and

holds similarity values. The profile index holds indexes to sub-sequences referenced in the distance profile [73, 75, 132]. Similarity is measured using Euclidean Distance. Only one parameter is used, the length n, the sliding window size. The profiles are used to identify "nearest-neighbours"', most similar time series sub-sequences, and consequently motifs. Figure 2.11 gives an example of the matrix profile generated from an original time series.

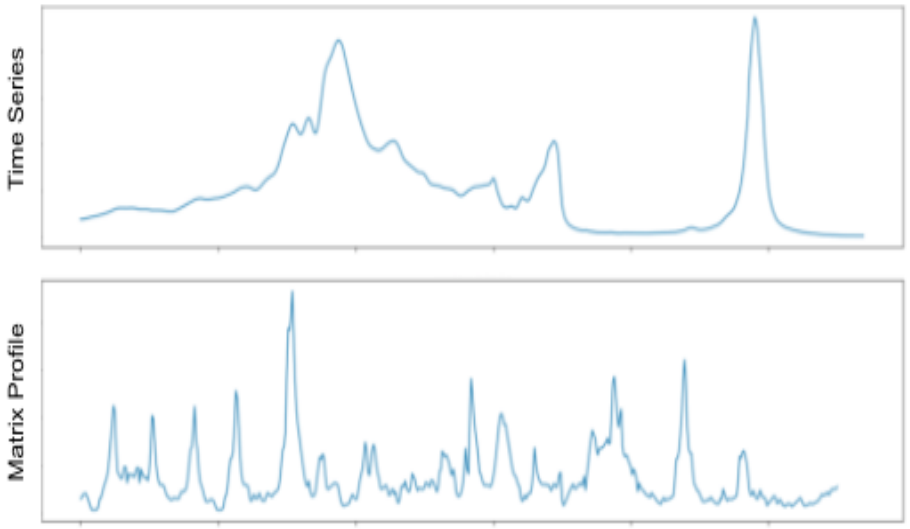

FIGURE 2.11: Matrix Profile generation. Top: the original time series. Bottom: the resulting Matrix Profile.

2.3.6 Summary

The chapter has presented the necessary background to the work presented in this thesis, and the related work that intersects with the work presented in this thesis. The chapter started with the background to time series classification including consideration of a number of classification approaches: (i) Decision Trees, (ii) Support Vector Machines and (iii) k-Nearest Neighbour. This was followed by a discussion of time series similarity measurement approaches, namely: (i) lock-step measures and (ii) elastic measures. A detailed description of the Dynamic Time Warping (DTW) process was then presented. It was noted that the main challenge of DTW is its computational complexity. The chapter also presented some previous work directed at addressing the DTW complexity starting with the idea of warping windows: (i) predefined warping windows, (ii) learnt warping windows and (iii) localised windows. An alternative to addressing the computational complexity of DTW is to adopt a lower bounding mechanism, three were considered: (i) LB_Yi Lower Bound, (ii) LB_Kim Lower Bound and (iii) LB_Keogh Lower Bound. The Chapter was concluded with consideration of a number of motif discovery mechanisms, some of which were incorporated in the approaches presented later in this thesis. These mechanisms including: (i) the MK Algorithm and (ii) the Matrix Profile technique. The following chapter introduces two benchmarks DTW approaches that were used for the comparative evaluation of the approaches proposed later in this thesis: (i) Standard DTW and (ii) DTW coupled with the Sakoe-Chiba band.

Chapter 3

Standard DTW and Sakoe-Chiba DTW

3.1 Introduction

This chapter presents an analysis of two established DTW approaches: (i) Standard DTW and (ii) Standard DTW coupled with the Sakoe-Chiba Band (S-C Band DTW) to enhance the run time performance of DTW. The motivations for the analysis of standard DTW were two-fold:

1. To obtain a deep understanding of the operation of DTW and its computational complexity.
2. To establish a benchmark with which the various alternatives presented later in this thesis could be compared. Any proposed alternative, to be worthwhile, would need to outperform Standard DTW with respect to at least one criteria while providing a compatible performance with respect to any other criteria. The two main criteria considered were efficiency and effectiveness.

The motivations for the analysis of S-C Band DTW were:

1. To obtain a deep understanding of the operation of a warping window approach to addressing the computational complexity of DTW. From the literature, various solutions have been proposed to mitigate against the complexity of DTW [80, 99, 109]. One such method, as noted in Chapter 2 and one of the most frequently sighted solutions, is the Sakoe-Chiba band [92], see for example [55, 81].
2. As in the case for the motivation for analysing the Standard DTW approach, to establish a benchmark with which the various alternatives presented later in this thesis could be compared.

For the analysis presented in this chapter, time series classification was considered with respect to kNN; and experiments were conducted using fifteen time series data sets taken from the UEA and UCR (the University of East Anglia and the University of California Riverside) Time Series Classification Repository; a well-documented training/test time series data set source [14, 29, 108]. The same fifteen data sets were used throughout the work presented in this thesis. The Selection of the data sets was based on size (number of records and length of time series) and number of classes. Further detail concerning the data sets will be presented in Sub-section 3.2.

A classification scenario was considered because this allowed measurement of the effectiveness of the individual DTW approaches considered in this chapter and later chapters. The classification was conducted using KNN classification because this entailed a significant amount of time series comparison. For the evaluation $k = 1$ was used because, from the literature, this was the most frequently adopted [13, 82, 109, 120]; $k = 1$ also provides the advantage that it does not require any conflict resolution as in the case where $k > 1$.

The rest of this chapter is organized as follows. Section 3.2 presents a brief review of the evaluation data sets. Section 3.3 presents the benchmark standard DTW approach, including (i) Standard DTW operation, (ii) a worked example, (iii) pseudo code, (iv) theoretical complexity calculation, (v) evaluation and (vi) actual complexity calculation. Section 3.4 presents the Sakoe-Chiba (S-B) DTW approach, also including: (i) S-C Band DTW operation, (ii) a worked example, (iii) pseudo code, (iv) the theoretical complexity, (v) evaluation and (vi) actual complexity calculation. A comparison between the two approaches is given in Section 3.5. The chapter is concluded with a short summary in Section 3.6.

3.2 Data sets

This section presents a brief overview of the data sets used for the work presented in this thesis. Each data set D comprised a set of records $\{r_1, r_2, \ldots, r_r\}$. Each record r_i in turn comprised x points, $[p_1, p_2, \ldots, p_x]$. Note that, for each example data set considered x was constant, this is frequently the case in practice. In total, fifteen data sets were downloaded from the UEA and UCR repository. These were selected so that a mix of data sets was obtained in terms of number of points, number of records, number of classes and the nature (type) of the data sets. An overview of the fifteen data sets is given in Table 3.1. Column one gives the ID number for the data set, this is used later in this chapter and in subsequent chapters. Column two gives the name of the data set. Column three (x) gives the length of a single time series. Column four gives the number of records (r) (instances). Column five, $x \times r$, gives an indication of the overall size of each data set. The "Type" of the data set describes the nature of the data set. The terminology used to describe time series type is that used with respect to the UEA and UCR repository. The "Sensor" data type indicates data recorded by a sensor such as electric power signal sensor, the "Motion" data type refers to data recorded by body motion, the "HAR" data type describes data recorded by hand movement recognition, the "Spectro" type refers to data collected using a spectrograph, the "Image" type describes data that was collected by boundary extraction and/or colour image segmentation, and the "Simulated" type refers to data generated using some form of simulation. For more detail about the data sets see [14].

3.3 Standard DTW Approach

In this section, the standard DTW benchmark approach is presented and evaluated. The section is organised as follows. Sub-section 3.3.1 describes the operation of standard DTW time series similarity measurement. To aid this understanding Sub-section 3.3.2 presents a worked example. The pseudo code for the Standard DTW approach is presented in Sub-section 3.3.3. Sub-section 3.3.4 then discusses the theoretical computational complexity of Standard DTW using "Big O" notation. Sub-section 3.3.5 presents a practical evaluation of Standard DTW, followed by the derivation of a complexity equation in Sub-section 3.3.6 based on the work reported in the preceding two

TABLE 3.1: Time series data sets used for evaluation purposes

ID No.	**Data set Name**	**Length (x)**	**Records (r)**	**Size ($x \times r$)**	**No. of Classes**	**Data set Type**
1.	SmoothSubspace	15	300	4500	3	Simulated
2.	ItalyPowerDemand	24	1096	26304	2	Sensor
3.	Libras	45	360	16200	15	HAR
4.	SyntheticControl	60	600	36000	10	Simulated
5.	GunPoint	150	200	30000	2	Motion
6.	OliveOil	570	60	34200	4	Spectro
7.	Trace	275	200	55000	4	Sensor
8.	ToeSegmentation2	343	166	56938	2	Motion
9.	Car	577	120	69240	4	Sensor
10.	Lightning2	637	121	77077	2	Sensor
11.	ShapeletSim	500	200	100000	2	Simulated
12.	DiatomSizeReduction	345	322	111090	4	Image
13.	Adiac	176	781	137781	37	Image
14.	HouseTwenty	2000	159	318000	2	Image
15.	PenDigits	8	10992	87936	10	Motion

sub-sections. As noted earlier DTW coupled with kNN was used for the evaluation because this is the most commonly used combination as evidenced by the previous work presented in Chapter 2.

3.3.1 Operation of Standard DTW

The operation of Standard DTW was described previously in sub-section 2.3.1; however, for completeness, it is presented again here. The fundamental idea of DTW is to identify the optimal alignment between two time series which may be of different lengths [54, 92, 95]. DTW finds the minimum warping distance (wd) between two time series assuming a nonlinear alignment. The process of DTW can be described as follows. Given two time series $S_1 = [p_1, p_2, \ldots, p_x]$ and $S_2 = [q_1, q_2, \ldots, q_y]$, where x and y are the lengths of S_1 and S_2 respectively, a distance matrix M of size $x \times y$ will be generated. Each value held at each cell $m_{i,j} \in M$ is derived by applying a distance calculation to the points $p_i \in S_1$ and $q_j \in S_2$ using Equation 3.1, where $d_{i,j}$ is the Euclidean distance calculated as shown in Equation 3.2; an alternative could be the absolute difference between the value p_i and q_j as in Equation 3.3. The distance value assigned to $m_{i,j}$ is then the summation of $d_{i,j}$ and the minimum cumulative distance value held at of the three "previous" elements to $m_{i,j}$. At the end of the process, the minimum warping distance (wd) will be held at $m_{x,y}$ [81].

$$m_{i,j} = d_{i,j} + min\{m_{i-1,j}, m_{i,j-1}, m_{i-1,j-1}\} \tag{3.1}$$

$$d_{i,j} = \sqrt{(p_i - q_j)^2} \tag{3.2}$$

$$d_{i,j} = |p_i - q_j| \tag{3.3}$$

Given the above, the minimum wd for a pair of time series can be interpreted as a similarity measure between the two time series. Note that if $wd = 0$ the two time series in question will be identical.

3.3.2 Standard DTW Worked Example

A worked example of Standard DTW was presented sub-section 2.3.1; the same worked example is considered here, but in greater detail. The basic DTW process is illustrated in Figure 3.1 (2.5 in sub-section 2.3.1). The figure shows the distance matrix M assuming two time series:

$$S_1 = [1, 2, 2, 3, 2, 1, 1, 0, 1, 0, 3, 2, 4, 2, 0]$$

and

$$S_2 = [1, 2, 4, 3, 3, 0, 3, 3, 1, 2, 1, 1, 3, 4, 2]$$

The value held at $m_{0,0}$ is always 0. The values held at $m_{1,0}$, $m_{0,1}$ and $m_{1,1}$ are then calculated as follows:

$$m_{1,0} = d_{1,0} + m_{0,0}$$

$$m_{0,1} = d_{0,1} + m_{0,0}$$

$$m_{1,1} = d_{1,1} + min\{m_{0,1}, m_{1,0}, m_{0,0}\}$$

and so on.

The resulting minimum warping path is as shown by the red line in Figure 3.1. In the figure the final warping distance arrived at is at $m_{x-1,y-1}$ (highlighted in green).

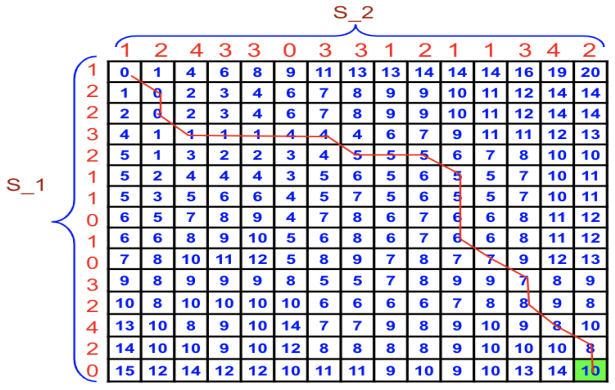

FIGURE 3.1: Dynamic Time Warping Matrix Worked Example repeated of Figure 2.5.

3.3.3 Standard DTW Algorithm

This pseudo code for the Standard DTW algorithm is given in Algorithm 1. The input, line 1, is two time series $S_1 = [p_1, p_2, \ldots, p_x]$ and $S_2 = [q_1, q_2, \ldots, q_y]$, where x and y are the lengths of S_1 and S_2 respectively. The first step, line 2, is to define the $x \times y$ distance matrix M. Next, line 3, $m_{0,0}$, the start of the warping path, is set to the value 0. Then, lines 4 to 9, the value of each cell $m_{i,j} \in M$, where $i > 0$ and $j > 0$ (this excluding $m_{0,0}$), is calculated as the Euclidean distance between the corresponding

point $p_i \in S_1$ and $p_j \in S_2$ (absolute value calculation would be an alternative), to which is added the minimum value from the three "previous" cells ($m_{i-1,j}$, $m_{i-1,j-1}$ and $m_{i,j-1}$). At the end of the process, line 10, the minimum wd value, held at $m_{x-1,y-1}$, will be returned. Two time series are identical if wd equates to zero. As the value of wd increases, the similarity reduces. A minimum warping path is thus a sequence of cell locations, $WP = [wp1, wp2, ...]$ in the matrix M, that serves to minimise the warping distance.

Algorithm 1 Standard Dynamic Time Warping (Standard DTW)

```
1: input S1,S2
2: M = [x, y]                                   ▷ Generate a matrix x × y
3: DTW[0, 0] = 0
4: for i = 0 to x do
5:     for j = 0 to y do
6:         dist = d(S1[i], S2[j])               ▷ Calculate dist using Euclidean distance
7:         DTW[i, j] = dist+minimum(DTW[i−1, j], DTW[i, j−1], DTW[i−1, j−1]
8:     end for
9: end for
10: returnDTW[x, y]
```

3.3.4 Theoretical Computational Complexity of Standard DTW

The theoretical time complexity of Standard DTW is considered in this section by considering a kNN time series classification scenario coupled with Ten Cross Validation (TCV). The complexity of Standard DTW, $DTW_{complexityStand}$, applied to two time series, S_1 and S_2 is given by:

$$DTW_{complexityStand} = O(x \times y) \tag{3.4}$$

where x and y are the lengths of S_1 and S_2 respectively.

For the evaluation data sets used in this thesis, all the time series were of the same length (with respect to each data set), as they frequently are in time series classification applications, the above thus simplifies to:

$$DTW_{complexityStand} = O(x^2) \tag{3.5}$$

Using kNN a previously unseen time series is compared with a "data bank" D of time series whose class labels are known; the label for the previously unseen time series is then derived from the k most similar time series in the bank. As noted in the introduction to this chapter, the most commonly used value for k in time series analysis, using DTW, is $k = 1$, thus we have 1NN. If we have a data repository D with r examples the time complexity to classify a single record using 1NN is given by:

$$O\left(r \times DTW_{complexityStand}\right) \tag{3.6}$$

If there are t new time series to be classified ($t > 1$) the complexity is given by:

$$O\left(r \times DTW_{complexityStand} \times t\right) \tag{3.7}$$

In the case of cross-validation, a well-established statistical technique for evaluating classification models, the complexity becomes:

$$O\left(r \times DTW_{complexityStand} \times t \times numFolds\right) \tag{3.8}$$

With respect to the evaluation presented in the following sub-section Ten Cross–Validation (TCV) was adopted [89]. When using ten cross validation the data set D is split into ten folds, in which case $numFolds = 10$, $r = \frac{9 \times |D|}{10}$ and $t = \frac{|D|}{10}$. Thus the complexity becomes:

$$O\left(\frac{9 \times |D|}{10} \times DTW_{complexityStand} \times \frac{|D|}{10} \times 10\right) \tag{3.9}$$

Which simplifies to:

$$O\left(\frac{9 \times |D|^2}{10} \times DTW_{complexityStand}\right) \tag{3.10}$$

3.3.5 Evaluation of Standard DTW

In this section, a practical analysis of the operation of Standard DTW is presented in term of 1NN classification. The analysis was conducted using the fifteen time series data sets presented earlier. Thus, a total of fifteen experiments were run, one for each data set. For each experiment TCV, as described above (see also [18, 89, 123]) was adopted. For the experiments, a desktop computer was used, the details of which were as follows: Apple M1 chip with 8 core CPU, 8 core GPU, 16 core Neural Engine, 16GB unified memory, and 512GB SSD storage. The same set–up as used with respect to further evaluations reported on later in this thesis. The metrics collected comprised run time (seconds), accuracy and F1-score; the last two were derived from a confusion matrix [33, 46, 90]. These were collected with respect to each "fold" of the TCV and averages calculated together with associated standard deviations so that a measure of the "spread" of the results could be obtained.

For the evaluation the class to be assigned to a previously unseen time series was taken from a set of classes $C = \{c_1, c_2, \dots\}$. The number of classes in C is usually small, less than ten often, only two [66, 104]. Where $|C| = 2$ this is referred to as a binary classification, where $|C| > 2$ this is referred to a multi-class classification [46, 122]. The assumption was that only one class label can be attached to a time series.

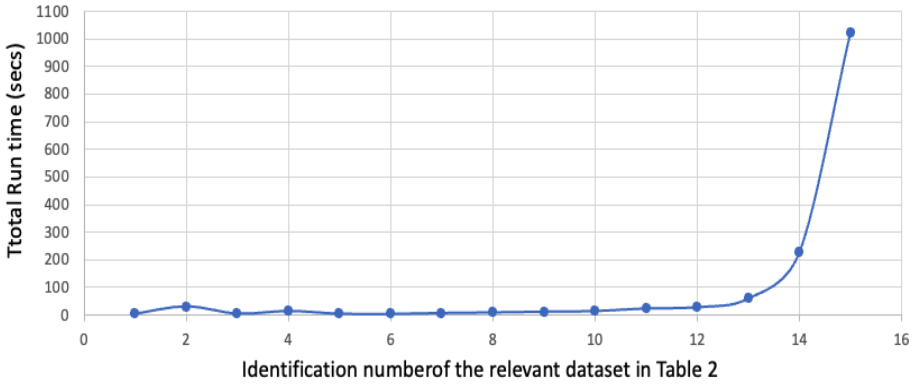

FIGURE 3.2: Run time results (seconds) using: 1NN classification applied to the fifteen time series evaluation data sets, Standard DTW and TCV.

The obtained results are presented in Table 3.2. Considering run time first, from the table, it is clear that the size of the data set affects the time complexity of the

TABLE 3.2: Run time (Secs), Accuracy and F1-Score using: 1NN classification applied to the fifteen time series evaluation data sets, Standard DTW and TCV.

ID #	Dataset Name	Standard DTW		
		Run time (Secs)	**Accuracy** (SD)	**F1-Score** (SD)
1	**SmoothSubspace**	4.36	91.00 (0.04)	0.91 (0.04)
2	**ItalyPowerDemand**	33.97	95.61 (0.02)	0.95 (0.02)
3	**Libras**	5.04	58.89 (0.10)	0.60 (0.11)
4	**SyntheticControl**	18.10	98.50 (0.01)	0.98 (0.01)
5	**GunPoint**	6.84	94.50 (0.05)	0.94 (0.05)
6	**OliveOil**	8.15	86.67 (0.15)	0.86 (0.16)
7	**Trace**	17.30	99.00 (0.03)	0.99 (0.03)
8	**ToeSegmentation2**	22.52	88.46 (0.09)	0.88 (0.10)
9	**Car**	31.12	80.83 (0.07)	0.80 (0.09)
10	**Lightning2**	38.15	87.76 (0.09)	0.87 (0.08)
11	**ShapeletSim**	64.03	82.00 (0.10)	0.81 (0.11)
12	**DiatomSizeReduction**	72.29	99.38 (0.01)	0.99 (0.01)
13	**Adiac**	127.74	65.30 (0.04)	0.62 (0.04)
14	**HouseTwenty**	653.64	95.00 (0.05)	0.95 (0.05)
15	**PenDigits**	1569.87	85.47 (0.01)	0.85 (0.01)

classification. This is emphasised in Figure 3.2 which shows a graph of the Standard DTW recorded run times. In the figure, the x-axis gives the identification number of the relevant data set (see Table 3.1) and the y-axis the run time in seconds. Recall that the data sets were ordered according to their perceived complexity.

The accuracy and F1 results given in Table 3.2 will be used later in this thesis for comparison purposes. Recall that the work presented in this thesis is directed at improving the run time without adversely affecting effectiveness (accuracy and F1 score).

3.3.6 Standard DTW Run time Equation

In this section, an equation for determining the actual complexity (run time) of Standard DTW, in the context of 1NN and using TCV, is presented. Equation 3.10 gave the theoretical complexity of Standard DTW using "Big O" notation. Using this equation we can derive an equation of the actual run time, in seconds, as follows:

$$runtime = \frac{9 \times |D|^2}{10} \times x^2 \times z \tag{3.11}$$

TABLE 3.3: Recorded run time, z value and calculated run time for Standard DTW.

ID.	Data set	Recorded Run time (secs.)	Constant z (secs.)	Calculated Run time (secs.)
1.	SmoothSubspace	4.36	0.0000023920	4.24
2.	ItalyPowerDemand	33.97	0.0000005455	18.97
3.	Libras	5.04	0.0000002134	7.20
4.	SyntheticControl	18.10	0.0000001552	35.54
5.	GunPoint	6.84	0.0000000844	5.91
6.	OliveOil	8.15	0.0000000774	7.68
7.	Trace	17.30	0.0000000635	19.85
8.	ToeSegmentation2	22.52	0.0000000772	21.28
9.	Car	31.12	0.0000000721	31.46
10.	Lightning2	38.15	0.0000000714	38.99
11.	DiatomSizeReduction	65.63	0.0000000711	65.63
12.	ShapeletSim	64.03	0.0000000651	81.00
13.	Adiac	127.74	0.0000000751	124.01
14.	HouseTwenty	653.64	0.0000000718	663.69
15.	PenDigits	1569.87	0.0000002260	1617.40

where z is a constant that represents the time to process a single cell in the matrix M. Regardless of the data set under consideration. Given any run time from Table 3.3, the value of z can be calculated as follow:

$$z = \frac{10 \times runtime}{9 \times |D|^2 \times x^2} \tag{3.12}$$

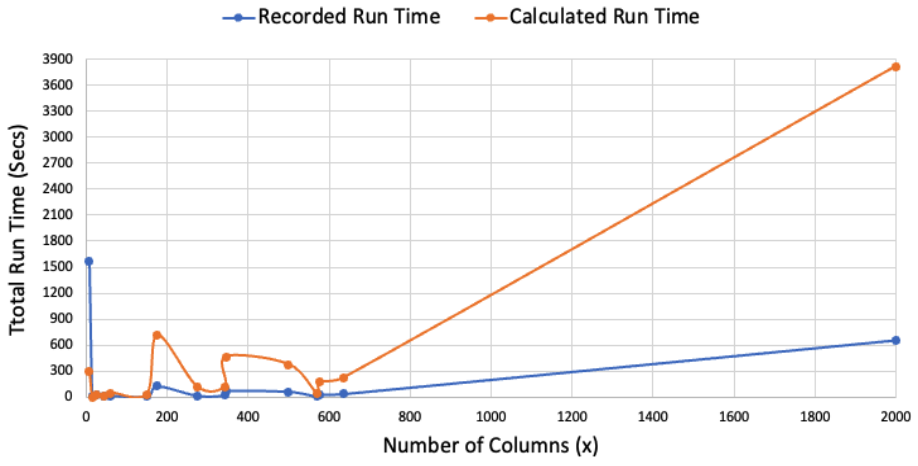

FIGURE 3.3: Recorded and calculated run times (seconds) for Standard DTW. using $\bar{z} = 0.0000004194$, with respect to the fifteen evaluation data sets ordered according to time series length x.

The calculated z value for each data set is listed in Table 3.3. A global z value can be calculated by simply averaging the calculated z values to give $\bar{z}$. In this case $\bar{z} = 0.0000004194$. Figure 3.3 shows a plot of the recorded run times and the calculated run times obtained using $\bar{z} = 0.0000004194$ with the data sets ordered according to length x of the time series (number of points) rather than the ordering given in Table 3.2. Figure 3.4 gives a more detailed view of the beginning of the graph given in Figure 3.3. Inspection of the graphs indicates that the recorded and calculated run times are

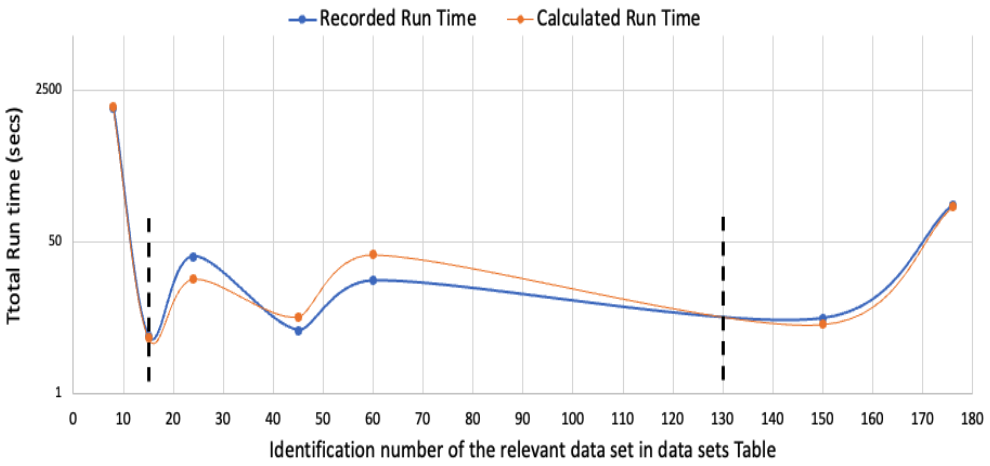

FIGURE 3.4: Detail of start of graph given in Figure 3.3.

not always aligned as well as we would like; especially for longer time series. It is suggested that this is because shorter time series have less of a DTW overhead than longer time series. Consequently, when calculating $\bar{z}$ the value of x should be taken into consideration. Inspection of the graphs given in Figures 3.3 and 3.4, and further analysis of the run times, indicated changes of behaviour at $x = 15$ points and $x = 130$ points. We will refer to time series that feature less than 15 points as *short time series*, time series that feature between 15 and 130 points as *medium time series* and time series that feature more than 130 points as *long time series*. It was therefore proposed that $\bar{z}$ be calculated by averaging small, medium and large time series z values individually. Thus:

1. For short time series where $x < 15$ points, as in the case of the *PenDigits* and *SommthSubspace* data sets, $\bar{z} = 0.0000023240$.
2. For medium time series where $15 \leq x < 130$ points, as in the case of the *ItalyPowerDemand*, *Libras*, and *SyntheticControl* data sets, $\bar{z} = 0.0000003047$.
3. For long time series where $x \geq 130$ points, as in the case of the largest ten data sets used in this thesis, $\bar{z} = 0.0000000729$.

Figure 3.5 plots the calculated run times, derived using the above process, against the recorded run times from Table 3.2. From the Figure, it can be seen that the calculated run times and the recorded run times are now more aligned. Thus it can be concluded that a better fit can be obtained if we take the time series length into account when calculating $\bar{z}$.

3.4 Sakoe-Chiba Band DTW Benchmark

In thiss section, the Sakoe-Chiba (S-C) Band DTW benchmark approach is presented; the second of the two DTW benchmark approaches considered in this chapter. The section is structured in a similar manner to the previous section. The section commences, Sub-section 3.4.1, with an overview of the Sakoe-Chiba Band warping window idea. A worked example is given in Sub-section 3.4.2, followed by the pseudo code for the S-C Band DTW approach in Sub-section 3.4.3. The associated theoretical complexity is then discussed in Sub-section 3.4.4. A practical analysis of the S-C Band DTW approach,

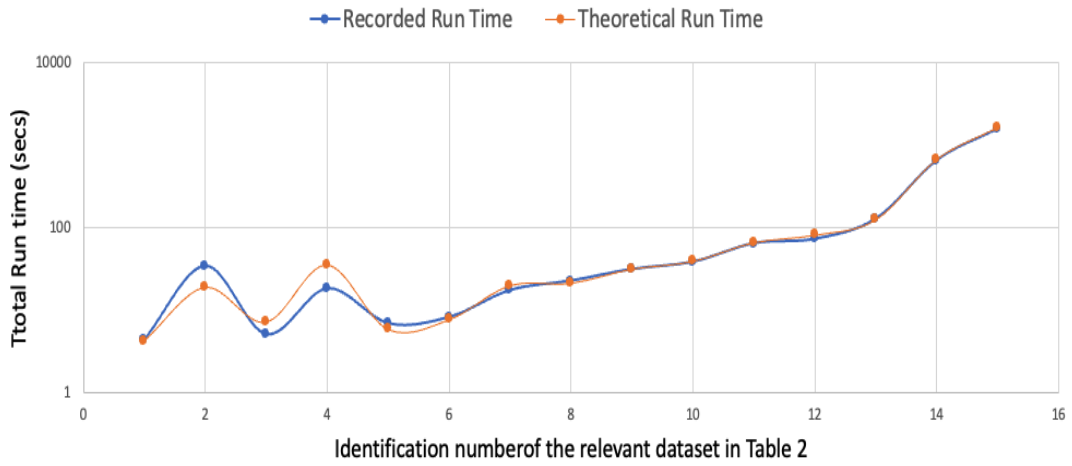

FIGURE 3.5: Recorded and calculated run times (seconds) for Standard DTW using $\bar{z}$ calculated according to time series length, with respect to the fifteen evaluation data sets.

using the fifteen time series data sets used earlier with respect to the evaluation of the Standard DTW approach, is presented in Sub-section 3.4.5; and the derivation of a runtime equation in Sub-section 3.5. Again a DTW and kNN combination was used for the evaluation because of its popularity.

3.4.1 Operation Sakoe-Chiba Band DTW

The Sakoe-Chiba Band (S-C Band) [92], as noted in sub-section 2.3.2, is a well-known technique used to address the complexity of DTW by applying a predefined warping window. The technique first gained popularity in the speech recognition community [43, 55, 83]. The idea is to define a warping window λ that establishes a constrained area inside the matrix M for which cell values need to be calculated; cells outside of this area are ignored. The use of a warping window, therefore, serves to limit the number of calculations needed to generate M. The intuition is that when undertaking 1NN (or similar) the "best fit" warping path, the warping path with the lowest wd value, will be located on, or close to, the leading diagonal in M. In addition, it is also argued that the use of warping windows prevents any pathological alignment by forcing the warping path to remain inside the constrained warping window area [43, 83, 92].

The S-C Band is the simplest approach for predefining a warping window λ, and hence selected for the second benchmark used in this thesis. The idea is to define a band stretching from $m_{0,0}$ to $m_{x,y}$ (given two time series $s_1 = [p_1, p_2, \ldots, p_x]$ and $s_2 = [q_1, q_2, \ldots, q_y]$). The band is demarcated in terms of a parameter ℓ, a percentage of the number of points in the x- or y-coordinate directions. In [81, 92] it was suggested that the value for ℓ should be set to 10% of the time series length; $\ell = 10\%$ was therefore adopted with respect to the work presented in this thesis. The operation of S-C Band DTW is similar to Standard DTW; however, focused on the area within the band.

For completeness, as noted in sub-section 2.3.2 an alternative to using a warping window in the shape of a band is to use a parallelogram thus avoiding unnecessary calculation at the start and end of the warping path. The best–known example of this is the Itakura parallelogram where the warping window λ is defined by two slope constraints [55]. The distinction between the Sakoe-Chiba band and the Itakura parallelogram was illustrated in Figure 2.6 in the previous Chapter, but for convenience is given again in Figure 3.6.

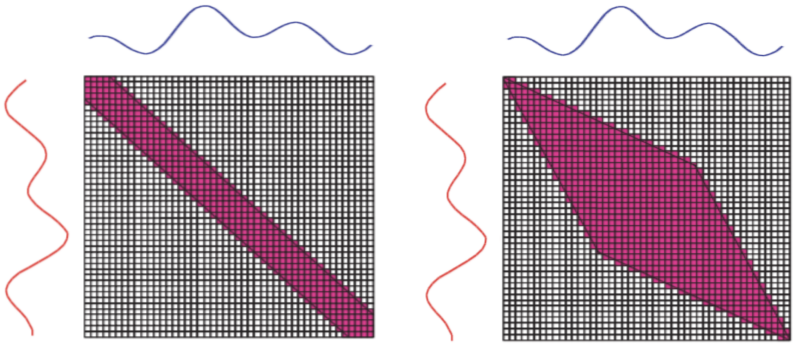

FIGURE 3.6: Left: The Sakoe-Chiba band, Right: The Itakura Parallelogram [81] (repeat of Figure 2.6).

3.4.2 Sakoe-Chiba Band DTW Example

A worked example of the Sakoe-Chiba Band DTW process is presented in this sub-section using the same two time series also used for illustrative purposes in Sub-section 3.3.2 with respect to the Standard DTW approach:

$$S_1 = [1, 2, 2, 3, 2, 1, 1, 0, 1, 0, 3, 2, 4, 2, 0]$$

$$S_2 = [1, 2, 4, 3, 3, 0, 3, 3, 1, 2, 1, 1, 3, 4, 2]$$

The operation of the Sakoe-Chiba Band DTW process is shown in Figure 3.7. The warping window λ is represented by the coloured cells using $\ell = 26.667\%$ (which happens to equate to $15 \times 0.26667 = 4$ points in the x and y coordinate directions). The minimum warping path is shown by the red line. As in the previous worked example, the final warping distance arrived at is highlighted using a green box.

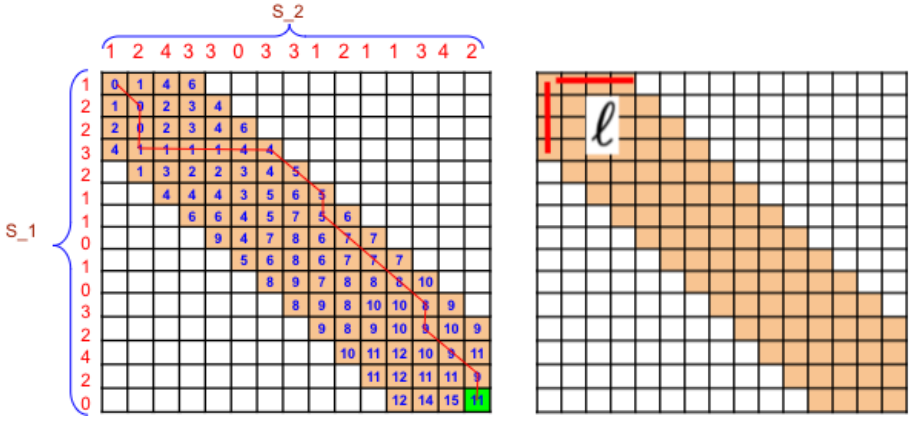

FIGURE 3.7: Left: S-C Band DTW worked example, Right: S-C Band Warping Window λ defined using the parameter ℓ (straight red line).

3.4.3 Sakoe-Chiba Band DTW Algorithm

Sakoe-Chiba (S-C) Band algorithm is a well-known algorithm; first gaining popularity within the speech recognition community [43, 55, 83]. S-C Band algorithm is similar to DTW algorithm presented in Algorithm 1; however, the main difference is that now the

cells $m_{i,j} \in M$ to be considered are confined to indexes $x \times \ell$ and $y \times \ell$ (recall that ℓ is a percentage).

The pseudo code for the S-C Band DTW approach is given in Algorithm 2. As before, the input (line 1) is two time series $S_1 = [p_1, p_2, \ldots, p_x]$ and $S_2 = [q_1, q_2, \ldots, q_y]$, and parameter ℓ. The first step (line 2) is to generate a distance Matrix M with size $x \times y$ where x and y are the lengths of S_1 and S_2 respectively. Next (lines 3 to 7), the cells in the distance matrix $m_{i,j}$ are initialised to infinity. Next, (line 8) $m_{0,0}$ is set to zero. Then, lines 9 to 14, the value of each cell, inside the warping window λ, excluding $m_{0,0}$, is calculated using the Euclidean distance of the corresponding points $p_i \in S_1$ and $p_j \in S_2$ (again absolute value calculation would be an alternative), to which is added the minimum value from the three "previous" cells ($m_{i,j}$ ($m_{i-1,j}$, $m_{i-1,j-1}$ or $m_{i,j-1}$) [10]. At the end of the process (line 15), the minimum wd held at $m_{x,y}$ will be returned. As before, two time series are identical if wd equates to zero. As the value of wd increases, the similarity reduces. Again, the minimum warping path is the sequence of cell locations, $WP = [wp_1, wp_2, ...]$ in the matrix M, that minimises the warping distance.

Algorithm 2 Sakoe-Chiba Band Dynamic Time Warping (S-C Band DTW)

```
1: input S1,S2,ℓ
2: M = [x, y]                                   ▷ Generate a matrix x × y
3: for ∀p_i ∈ S1 do
4:     for ∀q_i ∈ S2 do
5:         DTW[i, j] = infinity
6:     end for
7: end for
8: DTW[0, 0] = 0
9: for i = 0 to x × ℓ do
10:    for j = 0 to y × ℓ do
11:        dist = d(S1[i], S2[j])        ▷ Calculate the distance using Euclidean distance
12:        DTW[i, j] = dist+minimum(DTW[i−1, j], DTW[i, j−1], DTW[i−1, j−1]
13:    end for
14: end for
15: return DTW[x, y]
```

3.4.4 Theoretical Computational Complexity of Sakoe-Chiba Band DTW

The run time complexity in terms of "Big O" notation, given two time series, S_1 and S_2, when using S-C Band DTW, is given by:

$$DTW_{complexitySCbandDTW} = O(x \times y \times \ell) \tag{3.13}$$

where x and y are the lengths of S_1 and S_2 respectively, and ℓ is the Sakoe-Chiba parameter used to define the size of the warping window λ in the x and y coordinate directions. For the data sets used for evaluation purposes in this thesis (see Section 3.2), all the time series were of the same length, hence $x = y$. As suggested in [81, 92], $\ell = 0.1 \times x$ was adopted, with respect to the work presented in this thesis. The above thus simplifies to:

$$DTW_{complexitySCbandDTW} = O\left(\frac{x^2 \times 10}{100}\right) = O\left(\frac{x^2}{10}\right) \tag{3.14}$$

Inserting the above into Equation 3.10 we get the following equation to determine the theoretical run time complexity associated with the application of the 1NN classification model [13, 92, 109] to a data set D when using TCV.

$$O\left(\frac{9 \times |D|^2}{10} \times DTW_{complexitySCbandDTW}\right) \tag{3.15}$$

or:

$$O\left(\frac{9 \times |D|^2 \times x^2}{100}\right) \tag{3.16}$$

3.4.5 Evaluation Sakoe-Chiba Band DTW

An operational analysis of S-C Band DTW is presented in this sub-section. The evaluation was conducted using 1NN classification, and the fifteen selected datasets from the UEA and UCR Time Series Classification repository presented in Section 3.2, in the same manner as presented above with respect to the Standard DTW approach. As noted earlier $\ell = 10\%$ was used as recommended in [81]. The evaluation metrics used were again run time (seconds), accuracy and F1 score. The results are presented in Table 3.4. Note that, these results will be used later in this thesis for comparison purposes. The runtime results included in Table 3.4 are given in graph form in Figure 3.8. In the figure, the x-axis records the identification number of the relevant data set (see Table 3.1) and the y-axis the run time in seconds. A comparison with Standard DTW is presented in the following section.

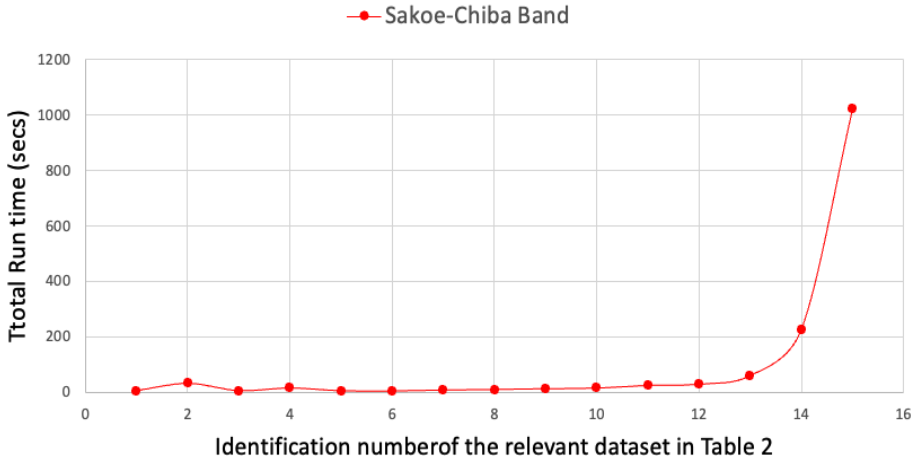

FIGURE 3.8: Run time results (seconds) using: 1NN classification applied to the fifteen time series evaluation data sets, S-C Band DTW and TCV.

Sakoe-Chiba Band Run time Equation

This section presents the derivation of an equation for the complexity (run time) of the S-C Band DTW approach using the theoretical complexity equation given in Subsection 3.4.4 and the evaluation results given in Sub-section 3.4.5. Using the theoretical complexity equation given in Equation 3.16, the run time using 1NN and TCV, for the S-C Band DTW approach, can be expressed as follows:

$$O\left(\frac{9 \times |D|^2 \times x^2}{100} \times z\right) \tag{3.17}$$

TABLE 3.4: Runtime (Secs), Accuracy and F1-Score using: 1NN classification applied to the fifteen time series evaluation data sets, S-C Band DTW and TCV.

ID #	Dataset Name	Sakoe - Chiba Band		
		Run time (Secs)	**Accuracy** (SD)	**F1-Score** (SD)
1	**SmoothSubspace**	4.32	95.00 (0.02)	0.95 (0.02)
2	**ItalyPowerDemand**	30.83	95.70 (0.02)	0.95 (0.02)
3	**Libras**	4.46	63.06 (0.10)	0.60 (0.11)
4	**SyntheticControl**	14.03	98.50 (0.01)	0.98 (0.01)
5	**GunPoint**	3.89	97.50 (0.03)	0.97 (0.03)
6	**OliveOil**	3.39	86.67 (0.15)	0.86 (0.16)
7	**Trace**	6.65	99.00 (0.03)	0.99 (0.03)
8	**ToeSegmentation2**	8.70	92.68 (0.07)	0.92 (0.7)
9	**Car**	11.38	81.67 (0.07)	0.81 (0.08)
10	**Lightning2**	13.52	87.76 (0.09)	0.87 (0.08)
11	**ShapeletSim**	23.34	82.00 (0.10)	0.81 (0.11)
12	**DiatomSizeReduction**	27.61	99.69 (0.01)	0.99 (0.01)
13	**Adiac**	58.89	65.30 (0.04)	0.62 (0.04)
14	**HouseTwenty**	226.56	93.00 (0.08)	0.93 (0.08)
15	**PenDigits**	1023.42	87.62 (0.01)	0.87 (0.01)

where z is again a constant that represents the time to process a single point. Regardless of the data set under consideration, the value of z should be constant and, we would anticipate this to be the same, or at least similar, to when using Standard DTW. Given the known run times from Table 3.4 the value of z can be calculated as follow:

$$z = \frac{runtime \times 100}{9 \times |C|^2 \times x^2} \tag{3.18}$$

Using this equation, the z values for each data set are listed in Table 3.5. Using the same mechanism for determining the average z value ($\bar{z}$) according to whether a time series is defined as short, medium or long time series, as described with respect to the Standard DTW approach, gives:

1. For short time series where $x < 15$, $\bar{z} = 0.0000019205$.
2. For medium time series where $15 \leq x \leq 130$, $\bar{z} = 0.0000002681$.
3. For long time series where $x > 130$, $\bar{z} = 0.0000000296$.

Note that these z values compare favourably with those presented earlier with respect to the Standard DTW approach.

TABLE 3.5: Recorded Run time, z value and calculated run time using S-C Band DTW.

ID.	**Data set**	**Recorded Run time (secs.)**	Constant z **(secs.)**	**Theoretical Run time (secs.)**
1.	SmoothSubspace	4.32	0.0000023700	3.50
2.	ItalyPowerDemand	30.83	0.0000004950	16.69
3.	Libras	4.46	0.0000001890	6.33
4.	SyntheticControl	14.03	0.0000001200	31.26
5.	GunPoint	3.89	0.0000000480	2.40
6.	OliveOil	3.39	0.0000000322	3.12
7.	Trace	6.65	0.0000000244	8.07
8.	ToeSegmentation2	8.70	0.0000000298	8.64
9.	Car	11.38	0.0000000264	12.79
10.	Lightning2	13.52	0.0000000253	15.85
11.	ShapeletSim	23.34	0.0000000259	26.68
12.	DiatomSizeReduction	27.61	0.0000000249	32.92
13.	Adiac	58.89	0.0000000346	50.41
14.	HouseTwenty	226.56	0.0000000249	269.80
15.	PenDigits	1023.42	0.0000014700	1102.30

Figure 3.9 plots the theoretical run times derived using Equation 3.18 with the recorded run time from Table 3.4 using the S-C Band DTW approach. From the Figure, it can be seen that the calculated run time and the recorded run time align well.

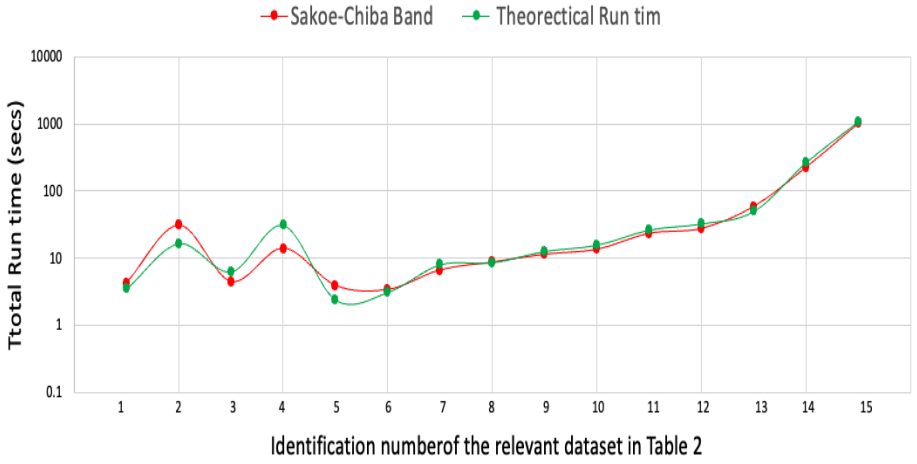

FIGURE 3.9: Recorded and calculated run times (seconds) for S-C Band DTW, using $\bar{z}$ calculated according to time series length, with respect to the fifteen evaluation data sets.

3.5 Standard DTW versus Sakoe-Chiba Band DTW

A comparison of the performance of the Standard DTW approach and the S-C Band DTW approach is presented in this section. The collated recorded run times, and accuracy and F1-scores, with respect to each approach, and each data set are given in Tables 3.6 and 3.7 respectively.

Table 3.6 gives the run time results. Figure 3.10 presents the recorded run times in graph form. In the figure, the x-axis records the identification number of the relevant data set (see Table 3.1) and the y-axis the run time in seconds. From both, the table and the figure, it can be seen that S-C Band DTW approach outperformed the Standard DTW approach. The run-time for the S-C Band DTW approach was reduced to almost one third of the total run time required for Standard DTW. This was particularly noticeable with respect to some of the data sets considered, such as the *Trace*, *ToeSegmentation2* and *Car* data sets.

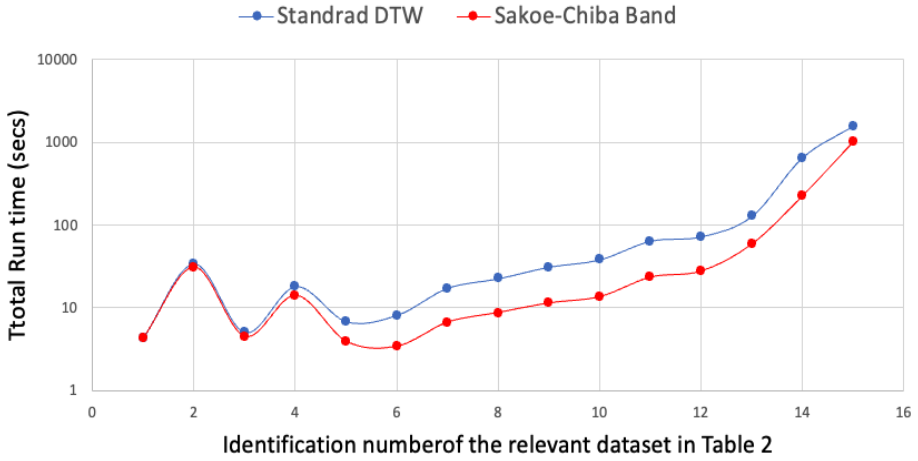

FIGURE 3.10: Comparison of the Standard DTW (blue) and S-C Band DTW (red) recorded run time results for the fifteen evaluation data sets.

TABLE 3.6: Recorded Run times for the Standard and S-C Band DTW approaches.

ID.	Dataset	Standard DTW Run time (secs.)	S-C Band Run time (secs.)
1.	SmoothSubspace	4.36	4.32
2.	ItalyPowerDemand	33.97	30.83
3.	Libras	5.04	4.46
4.	SyntheticControl	18.10	14.03
5.	GunPoint	6.84	3.89
6.	OliveOil	8.15	3.39
7.	Trace	17.30	6.65
8.	ToeSegmentation2	22.52	8.70
9.	Car	31.12	11.38
10.	Lightning2	38.15	13.52
11.	ShapeletSim	64.03	23.34
12.	DiatomSizeReduction	72.29	27.61
13.	Adiac	127.74	58.89
14.	HouseTwenty	653.64	226.56
15.	PenDigits	1569.87	1023.42

Table 3.7 gives the accuracy values and the F1 scores obtained using the Standard DTW approach and the S-C Band approach (values and scores obtained by averaging over the ten folds of the TCV). The figures in parenthesis are the standard deviation values obtained. From the table, it can be seen that the performance using the S-C Band approach is not adversely affected. In some cases, the performance improves as in the case of the *SmoothSubspac*, *GunPoint*, and *ToeSegmentation2* data sets. The reason

for this, it is suggested here, is that the S-C band approach serves to avoid pathological alignments and consequent miss-classifications.

Table 3.7: Recorded Accuracies and F1-Scores for the Standard and S-C Band DTW approaches.

ID	Dataset	Standard DTW		S-C Band	
#	Name	Accuracy (SD)	F1-Score (SD)	Accuracy (SD)	F1-Score (SD)
1	**SmoothSubspace**	91.00 (0.04)	0.91 (0.04)	95.00 (0.02)	0.95 (0.02)
2	**ItalyPowerDemand**	95.61 (0.02)	0.95 (0.02)	95.70 (0.02)	0.95 (0.02)
3	**Libras**	58.89 (0.10)	0.60 (0.11)	63.06 (0.10)	0.60 (0.11)
4	**SyntheticControl**	98.50 (0.01)	0.98 (0.01)	98.50 (0.01)	0.98 (0.01)
5	**GunPoint**	94.50 (0.05)	0.94 (0.05)	97.50 (0.03)	0.97 (0.03)
6	**OliveOil**	86.67 (0.15)	0.86 (0.16)	86.67 (0.15)	0.86 (0.16)
7	**Trace**	99.00 (0.03)	0.99 (0.03)	99.00 (0.03)	0.99 (0.03)
8	**ToeSegmentation2**	88.46 (0.09)	0.88 (0.10)	92.68 (0.07)	0.92 (0.7)
9	**Car**	80.83 (0.07)	0.80 (0.09)	81.67 (0.07)	0.81 (0.08)
10	**Lightning2**	87.76 (0.09)	0.87 (0.08)	87.76 (0.09)	0.87 (0.08)
11	**ShapeletSim**	82.00 (0.10)	0.81 (0.11)	82.00 (0.10)	0.81 (0.11)
12	**DiatomSizeReduction**	99.38 (0.01)	0.99 (0.01)	99.69 (0.01)	0.99 (0.01)
13	**Adiac**	65.30 (0.04)	0.62 (0.04)	65.30 (0.04)	0.62 (0.04)
14	**HouseTwenty**	95.00 (0.05)	0.95 (0.05)	93.00 (0.08)	0.93 (0.08)
15	**PenDigits**	85.47 (0.01)	0.85 (0.01)	87.62 (0.01)	0.87 (0.01)

3.6 Conclusion

In this chapter, an overview of the data sets used throughout the thesis has first been presented. These were taken from The UEA and UCR (the University of East Anglia and the University of California Riverside) Time Series Classification Repository. Then, two benchmarks DTW approaches were presented: (i) the Standard DTW approach and (ii) the Sakoe-Chiba (S-C) Band DTW approach. In both cases, the process, pseudo code, theoretical complexity and actual complexity were discussed. Both cases were also illustrated with a worked example and included an evaluation of the approach. The S-C Band approach featured the idea of a warping window so as to realise efficiency gains. The significance of both approaches was that they are used for comparison purposes as described later in this thesis. The chapter was concluded with a comparison between the operation of the Standard DTW approach and the S-C Band DTW approach in

terms of run time, accuracy and F1-score. As was anticipated, the S-C Band approach outperformed the Standard DTW approach in terms of run time, and achieved an equal, and in some cases better, accuracy and F1 score. Given that the evaluation results obtained were obtained using 15 evaluation data sets, it can be assumed that the results obtained were reliable. The following chapter introduces the first two of the six alternative approaches proposed in this thesis, directed at reducing the complexity of DTW, by splitting time series into sub-sequences: (i) the Sub-Sequence-Based DTW (SSBDTW) approach and (ii) the Fuzzy Sub-Sequence-Based DTW (FSSBDTW) approach.

Chapter 4

Sub-Sequence-Based Dynamic Time Warping Approaches

4.1 Introduction

This chapter presents two segmentation-based approaches to addressing the DTW overhead, the first of a number of alternative approaches considered in this thesis, the Sub-Sequence-Based approach (SSBDTW) and the Fuzzy Sub-Sequence-Based (FSSBDTW) approach [10, 11]. The second is a refinement of the first. The central idea presented in this chapter is, given two time series $S_1 = [p_1, p_2, \ldots, p_x]$ and $S_2 = [q_1, q_2, \ldots, q_x]$, where x is the length of S_1 and S_2 (the assumption is that both time series are of the same length), to segment S_1 and S_2 into two sets of corresponding sub-sequences each of length len, $S_1 = [U_{1_1}, U_{1_2}, \ldots]$ and $S_2 = [U_{2_1}, U_{2_2}, \ldots]$, and then to apply DTW to each sub-sequence pairing U_{1_i}, U_{1_j} where $i = j$. To differentiate between a time series expressed in terms of a number of points and a time series expressed in terms of a number of sequences, the notation L_1 and L_2 will be used for the latter. Thus $L_1 = [U_{1_1}, U_{1_2}, \ldots]$ where $U_{1_1}, U_{1_2} \in S_1$, and $L_2 = [U_{2_1}, U_{2_2}, \ldots]$ where $U_{2_1}, U_{2_2} \in S_2$.

The sub-sequence idea is illustrated in Figure 4.1. From the figure the theoretical complexity of the comparison will then be:

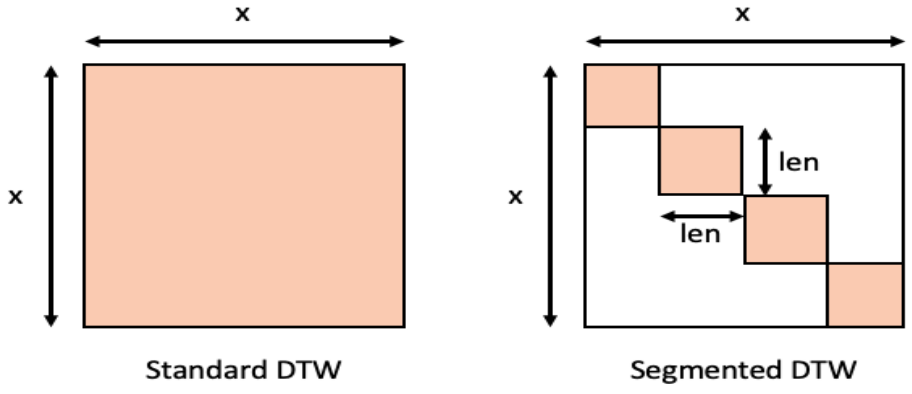

FIGURE 4.1: Comparison between Standard DTW and Segmented DTW.

$$DTW_{complexitySub-Sequence} = O\left(x \times len\right) \tag{4.1}$$

Where len is the length of the sub-sequences. Given that len will be significantly less than x this compares very favourably with the theoretical complexity of Standard DTW given in Equation 3.5 in the previous chapter and presented again below:

$$DTW_{complexityStand} = O\left(x^2\right) \tag{4.2}$$

The challenge is then how best to decide what the most appropriate values of *len* should be.

The work presented in the chapter was directed at providing answers to the first two subsidiary research questions given in Chapter 1:

1. Given that DTW operates using a distance matrix, what are the most appropriate mechanisms, in addition to those described in the literature, for limiting the size of this matrix?
2. Is there any advantage to be gained, in terms of efficiency, from segmenting the distance matrix into a set of sub-matrices?

The rest of this chapter follows a similar format to that presented in Chapter 3. Section 4.2 presents a description of the SSBDTW approach and Section 4.3 the FSSBDTW approach. Both sections comprise six sub-sections: (i) the operation of the proposed approach, (ii) a worked example, (iii) the algorithm expressed in terms of pseudo code, (iv) theoretical complexity calculation, (v) evaluation and (vi) actual complexity calculation. A comparison of the performance of the SSBDTW and FSSBDTW is then presented in Section 4.4. The chapter is concluded in Section 4.5, which provides a summary of the content and the main findings.

4.2 Sub-Sequence-Based Dynamic Time Warping

This section presents the first of the two sub-sequence-based approaches considered in this chapter, the Sub-Sequence-Based DTW (SSBDTW) approach. As noted earlier, the section is organised as follows. Sub-section 4.2.1 describes the operation of SSBDTW approach. For further clarification Sub-section 4.2.2 presents a worked example. The pseudo code for the SSBDTW approach is then presented in Sub-section 4.2.3. Sub-section 4.2.4 discusses the theoretical computational complexity of the SSBDTW approach using "Big O" notation. This is followed by Sub-section 4.2.5 which presents a practical evaluation of the proposed SSBDTW approach. The sub-section is concluded, in Sub-section 4.2.6, with the derivation of a complexity equation based on the work presented in the preceding two sub-sections.

4.2.1 Operation of Sub-Sequence-Based DTW

In this section, the operation of the proposed SSBDTW approach is presented. At a high level, the process is similar to the Standard DTW process described in the preceding chapter; the distinction is the idea of segmenting the time series. Thus, given two time series S_1 and S_2 these are divided into s sub-sequences so that we have $L_1 = [U_{1_1}, U_{1_2}, \ldots U_{1_s}]$ and $L_2 = [U_{2_1}, U_{2_2}, \ldots U_{2_s}]$. DTW is then applied to each sub-sequence pairing $\langle U_{1_i}, U_{1_j} \rangle$ where $i = j$. The final minimum warping distance arrived at will be the accumulated warping distance for all sub-sequences after s applications of DTW. There are two mechanisms whereby s can be defined:

1. **Fixed number**: Directly by specifying a value for s, a number of sub-sequences.
2. **Fixed length**: In terms of a predefined sub-sequence length *len*, such that $s = \frac{x}{len}$, where x is the overall length of the two time series to be considered (assuming they are of equal length).

Both are considered here.

4.2.2 Sub-Sequence-Based DTW Worked Example

Considering the two time series used for the worked example given in sub-sections 3.3.2 and 3.4.2, $S_1 = [1, 2, 2, 3, 2, 1, 1, 0, 1, 0, 3, 2, 4, 2, 0]$ and $S_2 = [1, 2, 4, 3, 3, 0, 3, 3, 1, 2, 1, 1, 3, 4, 2]$, using standard DTW the matrix M will measure 15×15 (the lengths of the two time series); The resulting distance matrix M was given in Figure 2.5 and again in Figure 3.1, and is given again here in Figure 4.2. However; in the case of the sub-sequence-based method the first step is to define the number of splits s. If, for the purposes of this example, it is assumed that $s = 3$, there will be three sub-sequences in each time series, each of length $len = 5$, thus $L_1 = [U_{1_1}, U_{1_2}, U_{1_3}] = [[1, 2, 2, 3, 2], [1, 1, 0, 1, 0], [3, 2, 4, 2, 0]]$ and $L_2 = [U_{2_1}, U_{2_2}, U_{2_3}] = [[1, 2, 4, 3, 3], [0, 3, 3, 1, 2], [1, 1, 3, 4, 2]]$. Figure 4.3 shows the three resulting distance matrices. The warping distance in this case is $2 + 8 + 6 = 16$.

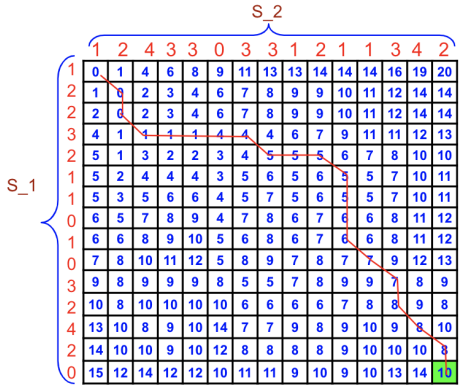

FIGURE 4.2: Distance Matrix and Warping Path (red line) for the example time series S_1 and S_2 generated using Standard DTW (repeat of Figures 3.1 and 4.2).

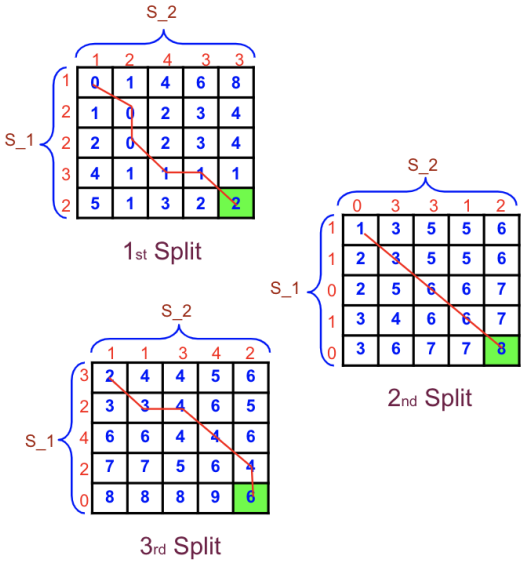

FIGURE 4.3: Distance Matrices and Warping Paths (red lines) for the example time series S_1 and S_2 generated using SSBDTW.

4.2.3 Sub-Sequence-Based DTW Algorithm

This pseudo code for the SSBDTW algorithm, using the Fixed Number approach to specifying s, is given in Algorithm 3. The input, line 1, is two time series $S_1 = [p_1, p_2, \ldots, p_x]$ and $S_2 = [q_1, q_2, \ldots, q_y]$, where x and y are the lengths of S_1 and S_2 respectively, and s is the number of sub-sequences. Note that it is assumed that $x = y$. The first step, line 2, is to define the length len of the sub-sequences; the notation $|S|$ is used to indicate the length (number of points) of a time series S. Next, lines 3, two variables are defined to hold the start and end points of a sub-sequence. The lists L_1 and L_2 are then declared and initialised with empty lists (line 4). Next, lines 5 to 10, L_1 and L_2 are populated with s sub-sequences of length len. Note that the notation $[a : b]$ indicates a sub-sequence starting at index a and ending at index b. The function *append(L,U)* appends segment U to a list L. Next, line 11, a DTW function is called. The DTW function is given on lines 13 and 18. The Standard DTW process (presented previously in Chapter 3) is applied to each sub-sequence pairing. The minimum warping distance will then be the accumulated warping distance for all sub-sequence pairings, held in the variable wd returned at the end of the function and then returned at the end of the process.

Algorithm 3 Sub-Sequence-Based DTW (SSBDTW)

```
1:  input S1,S2,s.
2:  len = int(|S1|/s)            ▷ It is assumed that S1 and S2 are of the same length
3:  start = 0, end = size
4:  L1 = [], L2 = []
5:  for i = 1 to s do
6:      U1_i = [s1_start : s1_end], U2_i = [s2_start : s2_end]
7:      L1 = append(L1,U1_i), L2 = append(L2,U2_i)
8:      start = end
9:      end = end + len
10: end for
11: wd =DTW(L1, L2)
12: return wd

13: function DTW(L1, L2)
14:     for i = 1 to s do
15:         wd = wd+ dtw(U1_i,U2_i)          ▷ The standard DTW Algorithm
16:     end for
17:     return wd
18: end function
```

When using the Fixed Length approach, the desired sub-sequence length len will form part of the input instead of s, and line 2 will be replaced with $s = \text{int}(|S_1|/len)$.

4.2.4 Theoretical Time Complexity of Sub-Sequence-Based DTW

Previously the time complexity for comparing two time series using standard DTW was given by $O(x \times y)$, which equates to $O(x^2)$ given the assumption that all our time series are of equal length. The time complexity when using the proposed splitting approach ($DTW_{complexitySSBDTW}$) can be expressed as follows:

$$DTW_{complexitySSBDTW} = O\left(len^2 \times s\right) \tag{4.3}$$

Where len is the pre-specified sequence length and s is the number of sequences (segments). However, given that $len \times s = x$, this can be rewritten as:

$$DTW_{complexitySSBDTW} = O\left(x \times len\right) \tag{4.4}$$

Substituting this into Equation 3.8 given in Chapter 3 for the complexity when using 1NN classification, and assuming cross validation, we get:

$$O\left(r \times DTW_{complexitySSBDTW} \times t \times numFolds\right) \tag{4.5}$$

Where r is the number of examples in the "data bank" D, t is the number of previously unseen examples and $numFolds$ is the number of folds used in the cross validation (it is assumed that each fold will be of equal size). When using ten cross validation, where consequently, D is split into tenths and thus $r = \frac{9 \times |D|}{10}$, $t = \frac{|D|}{10}$ and the number of fold will equal 10, this will translate to:

$$O\left(\frac{9 \times |D|}{10} \times DTW_{complexitySSBDTW} \times \frac{|D|}{10} \times 10\right) \tag{4.6}$$

which can be simplified to:

$$O\left(\frac{9 \times |D|^2}{10} \times DTW_{complexitySSBDTW}\right) \tag{4.7}$$

4.2.5 Evaluation of Sub-Sequence-Based DTW

The evaluation of the proposed SSBDTW approach is presented in this section. The evaluation was conducted using 1NN classification and fifteen selected data sets from the UEA and UCR Time Series Classification repository [14] in the same manner as reported on in Chapter 3. Experiments were conducted comparing the operation of SSBDTW with: (i) the Standard DTW, the benchmark approach (Standard DTW) and (ii) S-C Band DTW benchmark approach. To define the Sakoe-Chiba band warping window, $\ell = 10\%$ was used as proposed in [81, 92]. The objectives of the evaluation were:

1. To determine the most suitable mechanism for selecting a value for s (fixed number or fixed length); and what the most appropriate value for s should be.
2. To evaluate the run-time advantages gained using the SSBDTW approach compared to the Standard DTW and the S-C Band DTW approaches from the previous chapter.
3. To determine whether the classification accuracy of the proposed SSBDTW approach was commensurate with that obtained using standard DTW and S-C Band DTW.

The results are considered in further detail below. For the evaluation, a desktop computer was used with: Apple M1 chip with 8 core CPU, 8 core GPU, 16 core Neural Engine, 16GB unified memory, and 512GB SSD storage. The same set up as used with respect to the evaluations reported on in the previous Chapter. The evaluation metrics collected were again total run time (seconds), accuracy and F1-score; the later derived from a confusion matrix [33, 38]. The accuracy and F1-score values presented later in this section are all average values, collected using TCV [62].

Selection of The s Parameter

For the experiments conducted to determine the most appropriate value for s, the first of the above evaluation objectives, both effectiveness in terms of accuracy and F1 score, and efficiency in terms of run time, were considered. Recall that we have two mechanisms for defining s, the number of sub-sequences into which a time series is to be segmented: (i) fixed number where the value for s is pre-sepcified directly, and (ii) fixed length where the desired length of the sub-sequence is pre-specified from which a value for s can be derived. For the first, a range of values for s was experimented with, from 1 to 10, increasing in steps of 1. Note that $s = 1$ is equivalent to not segmenting at all (in other words Standard DTW). For the second, a range of values for *len* was used from 10 to 50 points, increasing in steps of 10 points. The anticipation was that as s increased the run time would decrease in a corresponding manner, whilst accuracy would remain the same or better for the higher values of s.

The run time results are presented in Tables 4.1 and 4.2. Tables 4.1 gives the results using the fixed number mechanism to define s, and Table 4.2 the results using the fixed length mechanism for defining s. In the tables, a "-" character indicates that no result was obtained because the time series was too short for the given value of s or *len*. From the tables, it can be seen that, as expected, as s increased the recorded run time correspondingly decreased. However, where $x \leq 60$ points (such as in the case of *PenDigits, SmoothSubspace, ItlayPowerDemand, Libras and SyntheticControl*), the selected value of s or *len* was found to have less of an impact than when $x > 60$ points. Therefore, in the case of *PenDigits*, further experiments using $len = 4$ were conducted.

TABLE 4.1: Recorded run time (Secs) using fixed number time series sub-sequences with a range of values for s; the "-" character indicates that the time series for a given data set is too short with respect to the suggested value for s.

	s									
Data set	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
SmoothSubspace	4.36	4.38	4.51	-	-	-	-	-	-	-
ItlayPowerDemand	33.97	33.25	32.31	34.33	35.41	35.71	-	-	-	-
Libras	5.04	4.89	4.87	5.11	5.25	5.49	5.64	5.79	5.95	6.25
SyntheticControl	18.10	15.16	14.88	14.93	15.27	15.62	16.62	17.07	17.54	17.91
GunPoint	6.84	5.61	5.03	4.68	4.54	4.47	4.38	4.30	4.26	4.20
OliveOil	8.15	4.58	3.28	3.09	2.44	2.11	1.98	1.84	1.70	1.64
Trace	17.30	10.22	8.73	6.23	5.57	5.02	4.79	4.64	4.91	4.52
ToeSegmentation2	22.52	13.09	9.06	7.09	6.82	6.22	5.20	5.02	4.93	4.79
Car	31.12	17.97	11.37	9.75	6.76	6.54	6.19	5.44	5.12	4.81
Lightning2	38.15	19.77	13.88	11.31	9.18	7.78	7.35	6.36	5.75	5.62
ShapeletSim	64.03	34.33	24.13	18.90	16.00	13.97	12.36	11.81	9.77	9.23
DiatomSizeRed	72.29	46.51	30.83	27.05	22.10	21.01	18.38	17.32	16.87	16.40
Adiac	127.74	93.53	74.83	70.03	63.94	51.99	41.94.87	36.09	31.30	27.14
HouseTwenty	653.64	334.36	224.80	169.77	133.00	108.00	93.00	82.71	74.35	67.45
PenDigits	1569.87	1036.28	-	-	-	-	-	-	-	-

The s and *len* values that produced the best accuracy and F1 score results are given in Tables 4.3. From the table, it can be seen that the fixed number mechanism for specifying s tended to produce better results. However, from the table it can also be seen that there was no single best value for s or *len* that could be identified. For future work (see Chapter 7) it is suggested that it might be possible to learn a best value of s using a training data set, as in the case of the work on learning warping windows presented in [81].

TABLE 4.2: Recorded run time (Secs) using fixed length time series sub-sequences with a range of values for *len*; the "-" character indicates that the time series for a given data set is too short with respect to the suggested value for *len*.

	len				
Data set	**10**	**20**	**30**	**40**	**50**
SmoothSubspace	1.68	-	-	-	-
ItalyPowerDemand	21.17	-	-	-	-
Libras	4.77	7.27	6.58	-	-
SyntheticControl	7.58	7.90	7.03	-	-
GunPoint	7.26	6.74	6.94	6.56	6.92
OliveOil	1.81	1.77	1.81	1.90	1.96
Trace	7.12	7.27	6.58	6.76	7.19
ToeSegment2	7.58	7.90	7.03	7.52	7.63
Car	4.72	4.25	4.41	4.72	5.04
Lightning2	5.49	4.92	5.26	5.48	5.83
ShapeletSim	12.25	11.96	11.71	12.31	13.41
DiatomSizeRed	21.29	19.72	21.04	22.33	24.25
Adiac	96.55	91.99	101.00	98.32	98.71
HouseTwenty	19.47	16.57	17.03	20.43	22.65
PenDigits	-	-	-	-	-

Performance Comparison

Figure 4.4 and Table 4.4 present a comparison between SSBDTW, Standard DTW and S-C Band DTW in terms of accuracy, F1 score and run time (objectives two and three for the evaluation). Figure 4.4 gives the average run time performance of the three approaches. From the figure, it can be seen that for the first three data sets, the smallest data sets, there was little difference between the recorded run times, this was to be expected. For the remaining twelve data sets, in seven cases the proposed SSBDTW approach produced the best run time result, in two cases, the S-C Band DTW approach produced the best run time result and in the remaining three cases there was no difference between SSBDTW and S-C Band DTW. The values for s and *len* used were those reported in Table 4.3. For the S-C Band DTW approach $\ell = 10\%$ was used as proposed in [81, 92].

Table 4.4 present the comparison between SSBDTW, Standard DTW and S-C Band DTW in terms of accuracy and F1 score. The values in parentheses are the standard deviations recorded after averaging over the ten folds of the TCV. Best results are highlighted in bold font. From the table, it can be seen that the proposed SSBDTW approach produced the best results with respect to thirteen of the fifteen data sets considered; for the remaining two cases, an almost identical performance was recorded. Where the performance was improved, it was conjectured that this was because the effect of noise was reduced when using the segmentation.

The results highlighted the issue of selecting the best value for s. As noted earlier, there is no single best value for s. In some cases, there are a range of values for s that give the same accuracy and F1 score (for example the *GunPoint* and *OliveOil* data sets). From the results obtained it was conjectured that the sub-sequence-based approach might not work well for short or medium time series (less than or equal to 60 points). However, from Table 4.4, it can be seen that, even when considering short time series, accuracy was maintained if not improved.

Table 4.3: Overall best accuracy and F1 results using SSBFTW (Fixed Number and Fixed Length), best values highlighted in bold font.

ID #	Data set	Fixed Number			Fixed Length		
		s	**ACC**	**F1**	*len*	**Acc**	**F1**
1	**SmoothSubspace**	3	**98.67 (0.03)**	**0.98 (0.03)**	5	**98.67 (0.03)**	**0.98 (0.03)**
2	**ItalyPowerDemand**	6	**96.34 (0.02)**	**0.96 (0.02)**	4	**96.34 (0.02)**	**0.96 (0.02)**
3	**Libras**	6	**64.44 (0.11)**	**0.64 (0.11)**	10	62.78 (0.11)	0.62 (0.11)
4	**SyntheticControl**	3	**98.50 (0.01)**	**0.98 (0.01)**	20	98.33 (0.01)	0.98 (0.01)
1	**GunPoint**	4, 7, 8, 10	98.50 (0.02)	0.98 (0.02)	40	**99.47 (0.02)**	**0.99 (0.02)**
2	**OliveOil**	4, 8, 10	88.33 (0.13)	0.88 (0.13)	10 - 50	**90.95 (0.16)**	**0.91 (0.16)**
3	**Trace**	2	**99.00 (0.03)**	**0.99 (0.03)**	40, 50	97.50 (0.03)	0.98 (0.04)
4	**ToeSegmentation2**	7	**93.38 (0.04)**	**0.93 (0.04)**	50	92.75 (0.06)	0.92 (0.06)
5	**Car**	5	82.50 (0.07)	0.82 (0.07)	40	**83.33 (0.10)**	**0.82 (0.11)**
6	**Lighting2**	3	**91.79 (0.06)**	**0.90 (0.06)**	10	83.30 (0.06)	0.83 (0.06)
7	**DiatomSizeReduction**	3, 6, 10	**100.00 (0.00)**	**1.00 (0.00)**	10 - 50	**100 (0.00)**	**1.00 (0.00)**
8	**ShapeletSim**	9	**90.00 (0.06)**	**0.90 (0.06)**	40	89.97 (0.06)	0.89 (0.06)
9	**Adiac**	7	**65.81 (0.03)**	**0.64 (0.03)**	10	65.51 (0.04)	0.63 (0.04)
10	**HouseTwenty**	2	**95.00 (0.05)**	**0.95 (0.05)**	10	93.71 (0.06)	0.94 (0.06)
1	**PenDigits**	2	**88.46 (0.01)**	**0.88 (0.01)**	4	**88.46 (0.01)**	**0.88 (0.01)**

4.2.6 Sub-Sequence-Based DTW Run time Equation

From the run times presented in Tables 4.1 and 4.2 in the forgoing sub-section, an actual complexity (run time) equation for SSBDTW can be derived in a similar manner as described with respect to Standard DTW and S-C Band DTW in Chapter 3. Given Equations 4.4 and 4.7 the actual complexity can be defined as follows:

$$runtime = \frac{9 \times |D|^2}{100} \times (x \times len) \times z \tag{4.8}$$

$$z = \frac{runtime \times 100}{9 \times |D|^2 \times s \times len^2} \tag{4.9}$$

Where, from Chapter 3, z is a constant that represents the time to process a single cell in the matrix M. In Chapter 3 a set of rules were derived to determine the z value for short, medium and long time series for both Standard DTW and S-C Band DTW. Using the z values for Standard DTW, and incorporating these into Equation 4.9 we can calculate the run time using SSBDTW given a specific data set. Figure 4.5 shows a

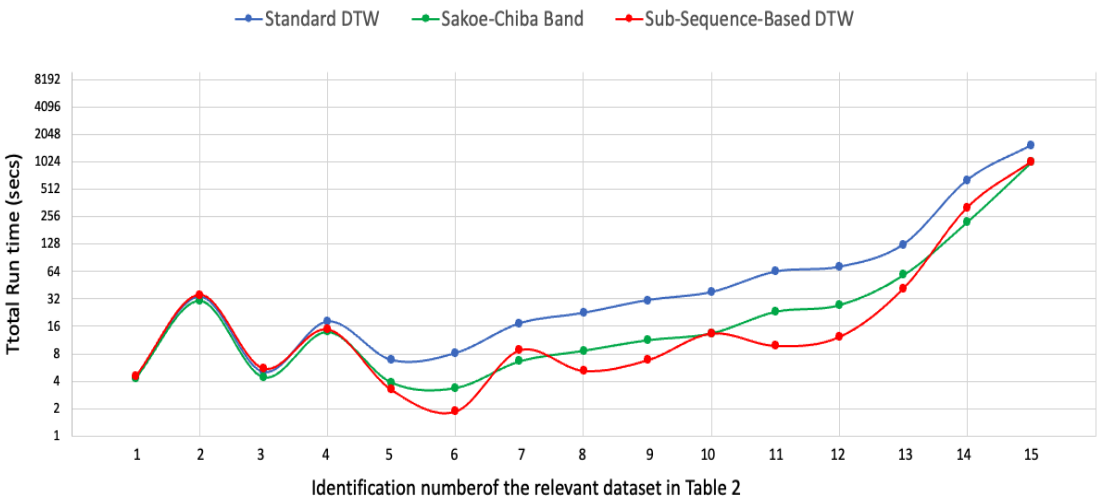

FIGURE 4.4: Run time results for Standrad DTW, DTW coupled with Sakoe-Chiba Band and Sub-Sequence-Based DTW.

plot of the calculated run times derived using Equation 4.8, and the z values obtained earlier, against the recorded run times from Table 4.1 and Table 4.2. From the figure, it can be seen that the calculated run times and the recorded run times are almost aligned, thus confirming the validity of Equation 4.9.

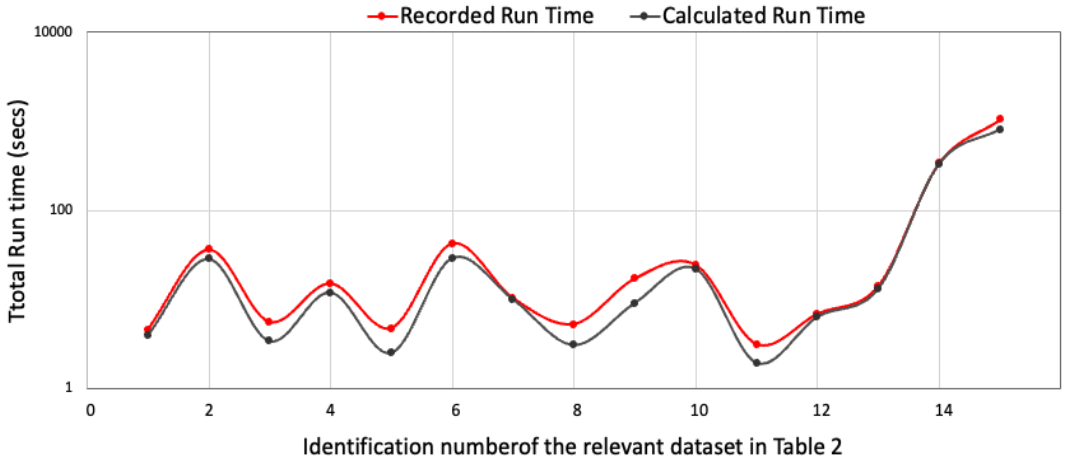

FIGURE 4.5: Recorded Run time results (seconds) and Calculated Run time results using SSBDTW with respect to the fifteen evaluation data sets.

4.3 Fuzzy Sub-Sequence-Based Dynamic Time Warping

A possible criticism of the SSBDTW approach described in the foregoing is that the segment boundaries are "crisp" boundaries. This may not be the most appropriate form of segmentation. It was, therefore, suggested that a more appropriate approach might be one that featured fuzzy boundaries. To this end, the Fuzzy Sub-Sequence-Based DTW approach (FSSBDTW) was investigated and developed. Detail concerning this approach is presented in this section. The section is structured in a similar manner to the previous section. The section commences, Sub-section 4.3.1 with an overview of the FSSBDTW approach. A worked example is given in Sub-section 4.3.2 and the associated pseudo code in Sub-section 4.3.3. The theoretical time complexity for the

TABLE 4.4: Accuracy and F1-Score with standard deviation for fifteen time series data set using Standard DTW, Sakoe-Chiba Band and Sub-Sequence-Based DTW.

ID	**Dataset**	**Standard DTW**		**S-C Band $\ell = 10$**		**SSBDTW**	
#	**Name**	**Acc** (SD)	**F1** (SD)	**Acc** (SD)	**F1** (SD)	**Acc** (SD)	**F1** (SD)
1	**SmoothSubspace**	91.00 (0.04)	0.91 (0.04)	95.00 (0.02)	0.95 (0.02)	**98.67 (0.03)**	**0.98 (0.03)**
2	**ItalyPowerDemand**	95.61 (0.02)	0.95 (0.02)	95.70 (0.02)	0.95 (0.02)	**96.34 (0.02)**	**0.96 (0.02)**
3	**Libras**	58.89 (0.10)	0.60 (0.11)	63.06 (0.11)	0.60 (0.11)	**64.44 (0.11)**	**0.64 (0.11)**
4	**SyntheticControl**	**98.50 (0.01)**	**0.98 (0.01)**	**98.50 (0.01)**	**0.98 (0.01)**	**98.50 (0.01)**	**0.98 (0.01)**
5	**GunPoint**	94.50 (0.05)	0.94 (0.05)	97.50 (0.03)	0.97 (0.03)	**98.50 (0.02)**	**0.98 (0.02)**
6	**OliveOil**	86.67 (0.15)	0.86 (0.16)	86.67 (0.15)	0.86 (0.16)	**88.33 (0.13)**	**0.88 (0.13)**
7	**Trace**	**99.00 (0.03)**	**0.99 (0.03)**	**99.00 (0.03)**	**0.99 (0.03)**	**99.00 (0.03)**	**0.99 (0.03)**
8	**ToeSegmentation2**	88.46 (0.09)	0.88 (0.10)	92.68 (0.07)	0.92 (0.7)	**93.38 (0.04)**	**0.93 (0.04)**
9	**Car**	80.83 (0.07)	0.80 (0.09)	81.67 (0.07)	0.81 (0.08)	**82.50 (0.07)**	**0.82 (0.07)**
10	**Lightning2**	87.76 (0.09)	0.87 (0.08)	87.76 (0.09)	0.87 (0.08)	**91.79 (0.06)**	**0.90 (0.06)**
11	**ShapeletSim**	82.00 (0.10)	0.81 (0.11)	82.00 (0.10)	0.81 (0.11)	**90.00 (0.06)**	**0.89 (0.08)**
12	**DiatomSizeReduction**	99.38 (0.01)	0.99 (0.01)	99.69 (0.01)	0.99 (0.01)	**100.00 (0.00)**	**1.00 (0.00)**
13	**Adiac**	65.30 (0.04)	0.62 (0.04)	65.30 (0.04)	0.62 (0.04)	**65.81 (0.03)**	**0.64 (0.03)**
14	**HouseTwenty**	95.00 (0.05)	0.95 (0.05)	93.00 (0.08)	0.93 (0.08)	**95.00 (0.05)**	**0.95 (0.05)**
15	**PenDigits**	85.47 (0.01)	0.85 (0.01)	87.62 (0.01)	0.87 (0.01)	**88.46 (0.01)**	**0.88 (0.01)**

approach is then discussed in Sub-section 4.3.4, whilst the evaluation of the proposed approach is reported on in Section 4.3.5. The section is concluded with a discussion of the derivation of an actual complexity equation (as opposed to one using Big O notation) in Sub-section 4.3.6.

4.3.1 Operation of Fuzzy Sub-Sequence-Based DTW

The fundamental idea underpinning the proposed FSSBDTW approach was the same as that for the SSBDTW approach described above; to segment (divide) the input time series into s sub-sequences and then apply DTW to correlated sub-sequence pairs. The final minimum warping distance arrived at will then be the accumulated warping distance for each sub-sequence pairing of s applications of DTW.

However, the distinction between the proposed FSSBDTW approach and the previous SSBDTW approach, is that the proposed approach features fuzzy boundaries as opposed to crisp boundaries. Intuitively, good points at which to cut the time series is where they meet, or at least at points where the distance between corresponding pairs

of points is at a minimum. By incorporating a degree of *fuzziness* as to where the segment boundaries are located it should be possible to derive a "best" segmentation. This is defined by specifying a *tail*, of length t points, measured backwards from a variable $maxLen$ defining the maximum permitted length of a segment, within which the cut should be applied. Thus, the cut should fall between $maxLen - t$ and $maxLen$, the latter measured from the start of the time series on the first iteration and the end of the previous segment on further iterations. The split point will be selected according to the minimum distance associated with the points between $maxLen - t$ and $maxLen$. There are then three options as to how the split point is allocated to a segment or segments:

Option A add to the preceding segment only.

Option B add to the succeeding segment only.

Option C add to both the preceding and the succeeding segments.

A worked example to clarify the process is given in the following sub-section, Sub-Section 4.3.2.

4.3.2 Fuzzy Sub-Sequence-Based DTW Example

If we assume two times $S_1 = [1, 2, 2, 3, 2, 1, 1, 0, 1, 0, 3, 2, 4, 2, 0]$ and $S_2 = [1, 2, 4, 3, 3, 0, 3, 3, 1, 2, 1, 1, 3, 4, 2]$, the same two time series as used in the worked example presented in Sub-section 4.2.2 above, and $maxLen = 10$ and $t = 3$; we derive a "distance list" D that hold the absolute difference as shown in Figure 4.6. If we examine the values in the tail the minimum value of 0 is at index 9, this is then the split point. We then have three options as to how to allocate the split point to a segment or segments; as itemised above and as also shown in Figure 4.6.

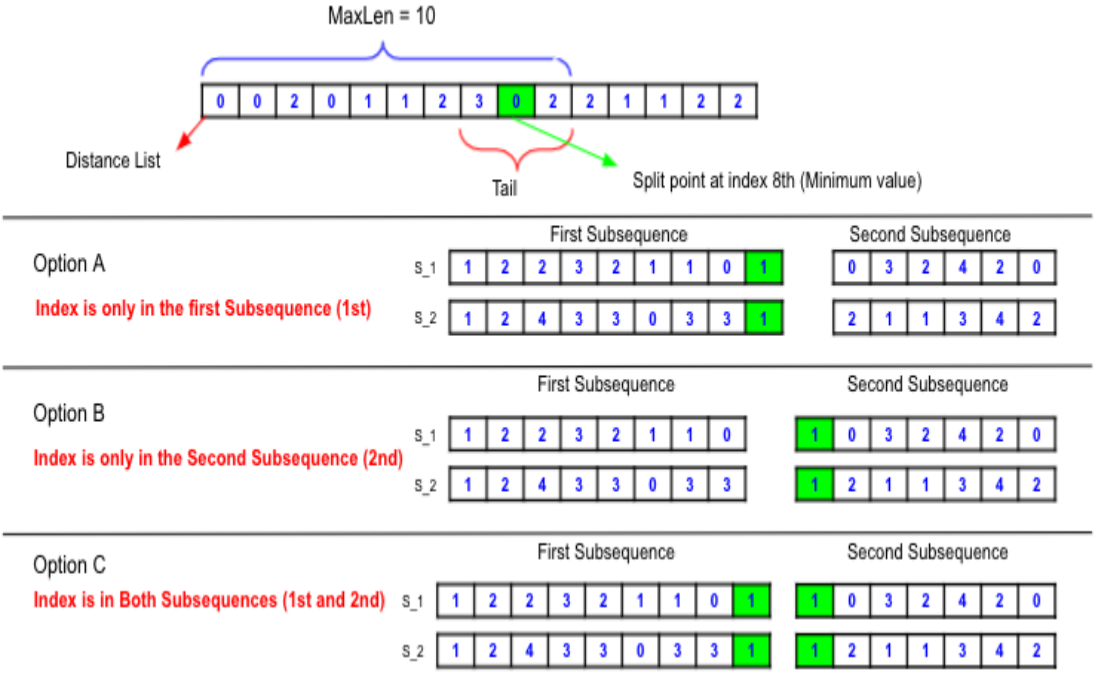

FIGURE 4.6: Segmentation example given two time series S_1 and S_2, and Options A, B or C.

4.3.3 Fuzzy Sub-Sequence-Based DTW Algorithm

This pseudo code for the FSSBDTW algorithm, using Option A, is given in Algorithm 4. The input, line 1, is: (i) two time series $S_1 = [p_1, p_2, \ldots, p_x]$ and $S_2 = [q_1, q_2, \ldots, q_y]$, (ii) the desired maximum length $maxLen$ of a segment and (iii) the length of the tail t. Note that, as in previous algorithms presented in this thesis, it is assumed that S_1 and S_2 are of the same length. Next, as in the case of the SSBDTW algorithm presented in Algorithm 3, two variables are defined to hold the start and end points of a sub-sequence (line 2), and the lists L_1 and L_2 are declared and initialised with empty lists (line 3). The difference list D is then populated using the genDiffList function. This is given in lines 18 to 24. Next, lines 6 to 16, L_1 and L_2 are populated in an iterative manner with a series of sub-sequences. On each iteration, a list $Tail$ is produced (line 7) holding the difference values from D in the tail of the current segment. The index associated with the minimum value in the tail is then selected (line 8) and used to define an end index, *end*, for the current segment. If this index is greater than the end of the input time series the final index is used. Thus, where an exact segmentation of a time series can not be achieved, a "short" segment will be included at the end of the segment collection. The start and end index values are used to generate two sub-sequences, U_1 and U_2, which are appended to L_1 and L_2. Note that $s_{1_{start}}, s_{1_{end}} \in S_1$, and $s_{2_{start}}, s_{2_{end}} \in S_2$. On line 14 the start index value is updated ready for the next iteration. Next, line 16, a DTW function is called, the same as that given in Algorithm 3,. The minimum warping distance will then be the accumulated warping distance for all sub-sequence pairings, held in the variable *wd* returned at the end of the function, and then returned at the end of the process.

For Option B, add the split point to the proceeding segment only, line 12 would need to be replaced with $U_1 = [s_{1_{start}} : s_{1_{end-1}}]$, $U_2 = [s_{2_{start}} : s_{2_{end-1}}]$, and line 14 with $start = end$. For Option C, add the split point to both the preceding and the proceeding segments, line 14 would need to be replaced with $start = end$.

4.3.4 Time Complexity of Fuzzy Sub-Sequence-Based DTW

The theoretical time complexity will be similar to that derived for the SSBDTW approach and is given in Equation 4.10:

$$DTW_{complexitySSBDTW} = O\left(x \times len\right) \tag{4.10}$$

except the length variable *len* will be replaced with an average length:

$$DTW_{complexityFSSBDTW} = O\left(x \times \overline{len}\right) \tag{4.11}$$

The TCV complexity when using 1NN is then given by:

$$O\left(\frac{9 \times |D|^2}{10} \times DTW_{complexityFSSBDTW}\right) \tag{4.12}$$

4.3.5 Evaluation of Fuzzy Sub-Sequence-Based DTW

In this section, the evaluation of the proposed FSSBDTW approach is presented. The same fifteen evaluation data sets selected from the UEA-UCR Time Series Classification repository were used as those used for the evaluation of the SSBDTW algorithm reported on earlier in this chapter, and for the evaluation of Standard DTW and S-C Band DTW reported on in the previous chapter. As before 1NN classification and TCV was adopted. The objectives of the evaluation were:

Algorithm 4 Fuzzy Sub-Sequence-Based DTW (FSSBDTW)

input $S_1, S_2, maxLen, t$
$start = 0$, $end = 0$
$L_1 = []$, $L_2 = []$
$D = \text{genDiffList}(L_1, L_2)$
$i = 0$
while $Start < |S_1|$ **do**
    $Tail = [d_{start+maxLen-t} : d_{start+maxLen}]$
    $end =$ Index associated with minimum value in $Tail$
    if $end > |S_1|$ **then**
        $end = |S_1|$
    end if
    $U_1 = [s_{1_{start}} : s_{1_{end}}]$, $U_2 = [s_{2_{start}} : s_{2_{end}}]$
    $L_1 =$ append(L_1,U_1), $L_2 =$ append(L_2,U_2)
    $start = end + 1$
end while
$wd =\text{dtw}(L_1, L_2)$
return wd

function genDiffList(L_1, L_2)
    $D = \{d_1, d_2, \ldots, d_{|S_1|}\}$
    for $i = 0$ to $|S_1|$ **do**
        $d_i = \text{abs}(s_{1_i} - s_{2_i})$ ($s_{1_i} \in S_1$, $s_{2_i} \in S_2$)
    end for
    return D
end function

1. To determine the best values for the *maxLen* parameter, the maximum sub-sequence length, and the t parameter, the sub-sequence tail.

2. To determine the best location for the split point to be included, Options A, B or C.

3. To compare the operation of the proposed FSSBDTW mechanism in terms of efficiency and effectiveness of the proposed approach with the operation of Standard DTW and S-C Band DTW.

Selection of The *maxLen* and t Parameters, and Options A, B or C

With respect to the most appropriate values for *maxLen* and t, and whether to use Options A, B or C, the first two of the above evaluation objectives, a range of values for *maxLen* were considered from 10 to 50, increasing in steps of 10, $maxLen = \{10, 20, 30, 40, 50\}$; and a range of values for t from 1 to 10, incrementing in steps of 1, $t = \{1, 2, 3, ..., 10\}$. A summary of the best results obtained, including the best parameters settings, run time, accuracy and F1-score are presented in Table 4.5. From the table, it can be seen that no "best" parameters could be identified; the best classification results were obtained using different settings. However, inspection of the table would suggest $maxLen = 40$, $t = 2$ and Option A.

TABLE 4.5: Best values for parameters *maxLen* and t, and best option for the point allocation strategy in terms of run time, accuracy and F1 results for each data set.

ID #	**Data set**	**Length** *maxLen*	**Tail** t	**Option** *A,B or C*	**Run time** (Secs)	**Acc** (SD)	**F1** (SD)
1	**SmoothSubspace**	6	2	A	3.03	98.67 (0.03)	0.98 (0.03)
2	**ItalyPowerDemand**	5	2	C	39.66	97.17 (0.01)	0.97 (0.01)
3	**Libras**	9	2	A	5.71	65.83 (0.11)	0.64 (0.11)
4	**SyntheticControl**	20	2	A	16.07	98.50 (0.01)	0.98 (0.01)
1	**GunPoint**	40	2	A	3.81	99.50 (0.01)	0.99 (0.01)
2	**OliveOil**	30	2	A	1.88	90.00 (0.11)	0.90 (0.11)
3	**Trace**	80	2	A	7.19	99.00 (0.03)	0.99 (0.03)
4	**ToeSegmentation2**	50	2	B	8.19	93.42 (0.04)	0.93 (0.04)
5	**Car**	60	5	A	6.88	86.67 (0.06)	0.86 (0.06)
6	**Lighting2**	60	2	A	8.11	89.23 (0.08)	0.89 (0.08)
7	**DiatomSizeReduction**	40	2	A	24.18	100.00 (0.00)	1.00 (0.00)
8	**ShapeletSim**	40	2	B	15.79	93.00 (0.04)	0.93 (0.04)
9	**Adiac**	40	8	C	88.20	68.12 (0.02)	0.66 (0.03)
10	**HouseTwenty**	40	5	A	50.64	95.00 (0.05)	0.95 (0.05)
1	**PenDigits**	8	1	A	480.17	88.46 (0.01)	0.88 (0.01)

Performance Comparison

For the comparison of the performance of the proposed FSSBDTW approach compared to Standard DTW and S-C Band DTW the best run time, accuracy measure and F1 score for each of the approaches were considered. The comparison of the best run-time results is presented in Figure 4.7. In Figure 4.7 total run time is given on the y-axis and the data set identifiers on x-axis. The identifiers are ordered according to their perceived complexity as described in Section 3.2 and illustrated in Table 3.1. From the figure, it can firstly be seen that for short time series (data sets 1, 2, 3 and 4) there was little to choose between the approaches in terms of run time. For longer time series (data sets 5 to 15), the performance of FSSBDTW outperformed that of Standard DTW in all cases. With respect to S-C Band DTW, the FSSBDTW approach was more efficient in six of the eleven cases, and less efficient in only one case (for the remaining four cases the efficiency was about the same).

Table 4.6 gives the accuracy and F1 results for the FSSBDTW, Standard DTW and S-C Band DTW approaches. In the table, the figures in parenthesis give the standard deviations recorded after averaging over the ten folds of the TCV. Best results are highlighted in bold font. From the table, it can be seen that the proposed FSSBDTW

approach, in thirteen of the fifteen cases, produced a better performance than that produced using either Standard DTW or S-C Band DTW; in the remaining two cases, an identical performance was recorded. Where the performance improved over Standard DTW and S-C Band DTW it was conjectured that this was because the effect of noise was reduced when using splitting.

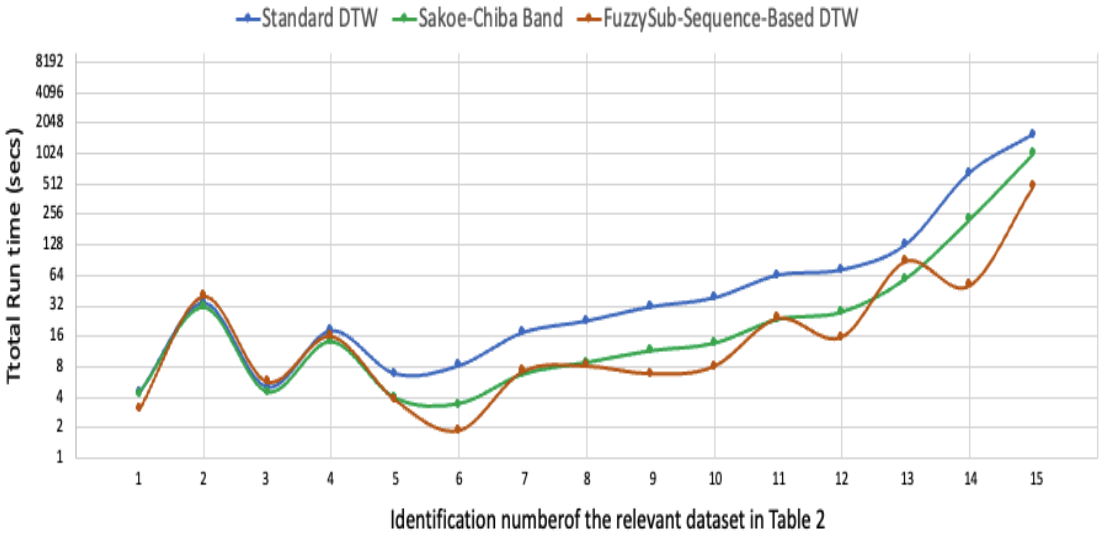

FIGURE 4.7: Run time results for Standard DTW, S-C Band DTW and FSSBDTW.

4.3.6 Fuzzy Sub-Sequence-Band Run time Equation

As in the case of the work reported with respect to the Standard DTW, SC Band DTW and SSBDTW approaches, using the run times presented in Table 4.5 an actual complexity (run time) equation for the FSSBDTW approach can be derived. The actual complexity can be defined using Equations 4.11 as follows:

$$runtime = \frac{9 \times |D|^2}{100} \times \left(x \times \overline{len}\right) \times z \tag{4.13}$$

Where, as before, z is a constant that represents the time to process a single cell in the matrix M. As we do not know what $\overline{len}$ will be in advance we can use $maxLen - t/2$ as an approximation. Thus we get:

$$runtime = \frac{9 \times |D|^2}{100} \times \left(x \times \left(maxLan - \frac{t}{2}\right)\right) \times z \tag{4.14}$$

Using the z values for standard DTW from Chapter 3 a set of run times can be calculated. Figure 4.9 shows a plot of the calculated run times and the recorded run times from Table 4.5. From the Figure, it can be seen that the calculated run times and the recorded run times are similar, therefore, indicating the veracity of the above equation.

4.4 Sub-Sequence-Based DTW Versus Fuzzy Sub-Sequence-Based DTW

A comparison of the performance of the proposed approaches, SSBDTW and FSSB-DTW, is presented in this section. The collated recorded run times, and accuracy and

TABLE 4.6: Accuracy and F1-Score using Standard DTW, S-C Band DTW and FSSBDTW.

ID #	Data set Name	Standard DTW		S-C Band $\ell = 10$		FSSBDTW	
		Acc (SD)	**F1** (SD)	**Acc** (SD)	**F1** (SD)	**Acc** (SD)	**F1** (SD)
1	**SmoothSubspace**	91.00 (0.04)	0.91 (0.04)	95.00 (0.02)	0.95 (0.02)	**98.67 (0.03)**	**0.98 (0.03)**
2	**ItalyPowerDemand**	95.61 (0.02)	0.95 (0.02)	95.70 (0.02)	0.95 (0.02)	**97.17 (0.01)**	**0.97 (0.01)**
3	**Libras**	58.89 (0.10)	0.60 (0.11)	63.06 (0.11)	0.60 (0.11)	**65.83 (0.11)**	**0.64 (0.11)**
4	**SyntheticControl**	**98.50 (0.01)**	**0.98 (0.01)**	**98.50 (0.01)**	**0.98 (0.01)**	**98.50 (0.01)**	**0.98 (0.01)**
5	**GunPoint**	94.50 (0.05)	0.94 (0.05)	97.50 (0.03)	0.97 (0.03)	**99.50 (0.01)**	**0.99 (0.01)**
6	**OliveOil**	86.67 (0.15)	0.86 (0.16)	86.67 (0.15)	0.86 (0.16)	**90.00 (0.11)**	**0.90 (0.11)**
7	**Trace**	**99.00 (0.03)**	**0.99 (0.03)**	**99.00 (0.03)**	**0.99 (0.03)**	**99.00 (0.03)**	**0.99 (0.03)**
8	**ToeSegmentation2**	88.46 (0.09)	0.88 (0.10)	92.68 (0.07)	0.92 (0.7)	**93.42 (0.04)**	**0.93 (0.04)**
9	**Car**	80.83 (0.07)	0.80 (0.09)	81.67 (0.07)	0.81 (0.08)	**86.67 (0.06)**	**0.86 (0.06)**
10	**Lightning2**	87.76 (0.09)	0.87 (0.08)	87.76 (0.09)	0.87 (0.08)	**89.23 (0.08)**	**0.89 (0.08)**
11	**ShapeletSim**	82.00 (0.10)	0.81 (0.11)	82.00 (0.10)	0.81 (0.11)	**90.00 (0.06)**	**0.89 (0.08)**
12	**DiatomSizeReduction**	99.38 (0.01)	0.99 (0.01)	99.69 (0.01)	0.99 (0.01)	**100.00 (0.00)**	**1.00 (0.00)**
13	**Adiac**	65.30 (0.04)	0.62 (0.04)	65.30 (0.04)	0.62 (0.04)	**65.81 (0.03)**	**0.64 (0.03)**
14	**HouseTwenty**	95.00 (0.05)	0.95 (0.05)	93.00 (0.08)	0.93 (0.08)	**95.00 (0.05)**	**0.95 (0.05)**
15	**PenDigits**	85.47 (0.01)	0.85 (0.01)	87.62 (0.01)	0.87 (0.01)	**89.64 (0.01)**	**0.89 (0.01)**

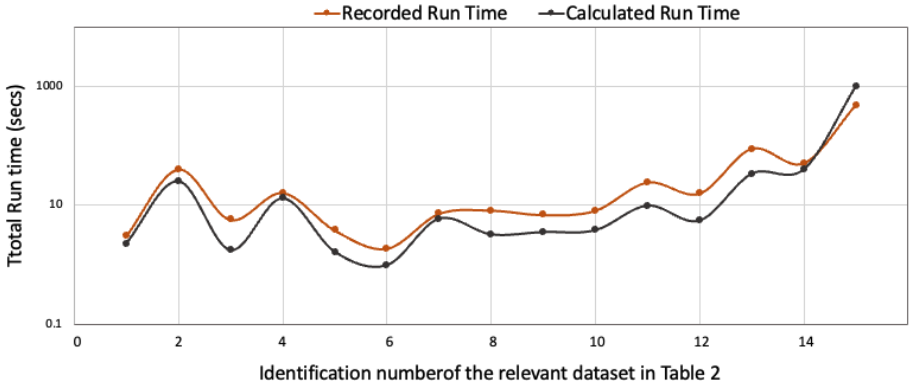

FIGURE 4.8: Recorded Run time results (seconds) and Calculated Run times (seconds) using FSSBDTW with the fifteen evaluation data sets.

F1-scores, with respect to each approach, and each data set are given in Tables 4.7 and 4.8 respectively.

Table 4.7 gives the run time results. Figure 4.9 presents the same recorded run time results in graph form. In the figure, the x-axis records the identification number of the relevant data set (see Table 3.1) and the y-axis the run time in seconds. From both the table and figure, it can be seen that FSSBDTW approach and SSBDTW approach are very similar; however, FSSBDTW outperforms the SSBDTW in the case of the *HouseTwenty* and *PenDigits* data sets.

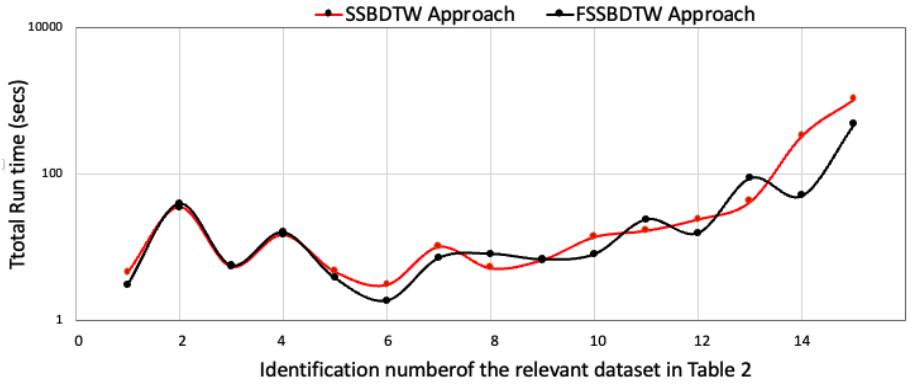

FIGURE 4.9: Comparison of the SSBDTW approach (red) and FSSBDTW approach (black) recorded run time results for fifteen evaluation data sets.

TABLE 4.7: Recorded Run times for the SSBDTW and FSSBDTW approaches.

ID.	**Data set**	**SSBDTW Run time (secs.)**	**FSSBDTW Run time (secs.)**
1.	SmoothSubspace	4.51	3.03
2.	ItalyPowerDemand	35.97	39.66
3.	Libras	5.49	5.71
4.	SyntheticControl	14.88	16.07
5.	GunPoint	4.68	3.81
6.	OliveOil	3.09	1.88
7.	Trace	10.22	7.19
8.	ToeSegmentation2	5.20	8.19
9.	Car	6.76	6.88
10.	Lightning2	13.88	8.11
11.	ShapeletSim	24.13	24.18
12.	DiatomSizeReduction	16.87	15.79
13.	Adiac	41.94	88.20
14.	HouseTwenty	334.36	50.64
15.	PenDigits	1036.28	480.17

Table 4.8 gives the accuracy values and the F1 scores obtained using the SSBDTW and FSSBDTW approaches (values and scores obtained by averaging over the ten folds of the TCV). In the table, the figures in parentheses are the standard deviation values obtained. From the table, it can be seen that the FSSBDTW approach outperforms the SSBDTW approach in most cases. In some cases, the performance is the same as in the case of the *SmoothSubspac* and *DiatomSizeReduction* data sets. The reason for this, is due to the flexibility of the FSSBDTW process where the alignment can not be adversely affected by the splitting process.

TABLE 4.8: Recorded Accuracies and F1-Scores for the SSBDTW and FSSBDTW approaches.

ID	Data set	SSBDTW		FSSBDTW	
#	Name	**Accuracy** (SD)	**F1-Score** (SD)	**Accuracy** (SD)	**F1-Score** (SD)
1	**SmoothSubspace**	**98.67** (**0.03**)	**0.98** (**0.03**)	**98.67** (**0.03**)	**0.98** (**0.03**)
2	**ItalyPowerDemand**	96.34 (0.02)	0.96 (0.02)	**97.17** (**0.01**)	**0.97** (**0.01**)
3	**Libras**	62.78 (0.11)	0.62 (0.11)	**65.83** (**0.11**)	**0.64** (**0.11**)
4	**SyntheticControl**	98.33 (0.01)	0.98 (0.01)	**98.50** (**0.01**)	**0.98** **(0.01)**
5	**GunPoint**	99.47 (0.02)	0.99 (0.02)	**99.50** (**0.01**)	**0.99** (**0.01**)
6	**OliveOil**	88.33 (0.13)	0.88 (0.13)	**90.00** (**0.11**)	**0.90** (**0.11**)
7	**Trace**	97.50 (0.03)	0.97 (0.03)	**99.00** (**0.03**)	**0.99** (**0.03**)
8	**ToeSegmentation2**	92.75 (0.06)	0.92 (0.06)	**93.42** (**0.04**)	**0.93** (**0.4**)
9	**Car**	83.33 (0.10)	0.82 (0.11)	**86.67** (**0.06**)	**0.86** (**0.06**)
10	**Lightning2**	83.30 (0.06)	0.83 (0.06)	**89.23** (**0.06**)	**0.89** (**0.08**)
11	**ShapeletSim**	89.97 (0.06)	0.89 (0.06)	**93.00** (**0.04**)	**0.93** (**0.04**)
12	**DiatomSizeReduction**	**100.00** (**0.00**)	**1.00** (**0.00**)	**100.00** (**0.00**)	**1.00** (**0.00**)
13	**Adiac**	65.51 (0.04)	0.62 (0.04)	**68.12** (**0.04**)	**0.66** (**0.04**)
14	**HouseTwenty**	93.71 (0.06)	0.93 (0.06)	**95.00** (**0.05**)	**0.95** (**0.05**)
15	**PenDigits**	88.46 (0.01)	0.88 (0.01)	**89.64** (**0.01**)	**0.89** (**0.01**)

4.5 Conclusion

The chapter has presented two proposed segmentation-based approaches directed at speeding up DTW in the context of 1NN classification: (i) the Sub-Sequence-Based DTW (SSBDTW) approach and (ii) the Fuzzy Sub-Sequence-Based DTW (FSSBDTW) approach. The fundamental idea was that by applying DTW to pairs of corresponding sub-sequences significant efficiency gains could be made. The reported evaluation demonstrated that this was indeed the case. In addition, it was found that accuracy gains were also made in comparison with Standard DTW and S-C Band DTW. It was conjectured that this was because the comparison of short time series sub-sequences was more resilient to noise. The distinction between SSBDTW and FSSBDTW was that the first included crisp boundaries while the second included fuzzy boundaries. Comparison of SSBDTW and FSSBDTW indicated that FSSBDTW was more effective (more accurate) since the splitting process was more flexible. In addition, although the reported evaluation indicated that both tended to have similar run times, it could be argued that FSSBDTW was more efficient with respect to long time series. The following Chapter

presents two further proposed approaches directed at reducing the number of DTW process required in the context of 1NN classification by limiting the number of candidates to be considered for comparison: (i) Candidate Reduction Based on Euclidean Distance (CRBED) and (ii) Candidate Reduction Based on Lower Bounding (CRBLB).

Chapter 5

Candidate Reduction Based on Euclidean Distance and Lower Bounding

5.1 Introduction

Chapter 3 presented an analysis of the Standard DTW and S-C Band DTW approaches. These were essentially two benchmark approaches, any approach proposed in later chapters had to improve on these approaches to be worthwhile. The analysis demonstrated the considerable computational overhead that DTW entailed. Chapter 4 presented two segmentation-based DTW approaches designed to address the DTW overhead, the Sub-Sequence-Based DTW (SSBDTW) approach and the Fuzzy Sub-Sequence-Based DTW (FSSBDTW) approach. The distinction was in how the sub-sequences were defined. All four approaches were analysed in the context of kNN classification using $k = 1$. FSSBDTW was found to be the most effective approach.

This chapter presents two approaches to further reduce the DTW complexity assuming that we wish to apply DTW in the context of kNN classification (where $k = 1$) and that the DTW approach to be used is FSSBDTW (because it was found to be the best performing approach as reported in Chapter 4). The fundamental idea is to reduce the number of applications of DTW by reducing the number of comparisons that need to be considered. In other words, by introducing a pre-processing step for each previously unseen record to be labelled (each *query* time series) whereby the time series in the kNN bank are prioritised and only the r most significant time series are considered. The question is then how best to reduce the number of time series to be considered without actually invoking DTW. Two approaches are presented: (i) Candidate Reduction Based on Euclidean Distance (CRBED) and (ii) Candidate Reduction Based on Lower Bounding (CRBLB). Note that the CRBED approach as presented here was first published in [12]. The two approaches are fully described by coupling them with FSSBDTW. Thus when we refer to the "CRBED approach", this is Candidate Reduction Based on Euclidean Distance coupled with FSSBDTW. Similarly when we refer to the "CRBLB approach", this is Candidate Reduction Based on Lower Bounding coupled with FSSBDTW.

The work presented in this chapter, therefore, provides an answer to the third subsidiary question that this thesis seeks to address:

Is it possible to utilise knowledge of a given application domain to limit the DTW processing required? For example, in the case of kNN classification, using early

abandonment with respect to time series that are clearly not going to be very similar to the query time series.

The rest of this chapter follows a similar format to that presented in Chapters 3 and 4. Section 5.2 presents the proposed CRBED approach and Section 5.3 the proposed CRBLB approach. Each section follows a similar structure: (i) the operation of the proposed approach, (ii) pseudo code, (iii) theoretical complexity calculation and (iv) evaluation. Section 5.4 provides a comparison of the two proposed approaches. The chapter is concluded in Section 5.5 where a brief summary is provided. Note that comparison with the approaches presented in Chapter 4 is left till Chapter 7.

5.2 Candidate Reduction Based on Euclidean Distance Approach

This section presents the first of the two candidate reduction approaches considered in this chapter, the Candidate Reduction Based on Euclidean Distance (CRBED) approach. The fundamental idea is to use a cheaper mechanism for comparing time series than DTW, namely Euclidean distance measurement. Recall from earlier work presented in this thesis that the complexity of DTW is given by:

$$DTW_{complexityStand} = O(x^2) \tag{5.1}$$

where x is the length of the time series to be compared (assuming both time series are of the same length). The fundamental Euclidean Distance time series comparison mechanism is given in Algorithm 5. The input is two time series S_1 and S_2 of equal length. The accumulated difference in distance, between each point $p_i \in S_1$ with each corresponding point $p_j \in S_2$, is then determined. On completion, the accumulated distance is a measure of the similarity between the two time series. If the difference is 0 the two time series are identical. The complexity of Euclidean distance time series comparison is given by:

$$EuclideanComparison_{complexity} = O(x) \tag{5.2}$$

A significant difference when compared with the complexity of DTW, especially when considering long time series.

Algorithm 5 Eulidean Distance Time Series Comparison

```
1: input S1, S2
2: dist = 0
3: for ∀p_i ∈ S1 and ∀p_j ∈ S2 do
4:     dist = dist + abs(p_i − p_j)
5: end for
6: return dist
```

The remainder of this section is organised as follows. Sub-section 5.2.1 describes the operation of the CRBED approach. The pseudo code for the approach is then presented in Sub-section 5.2.2. Sub-section 5.2.3 discusses the theoretical computational complexity of the approach using "Big O" notation. This is followed by Sub-section 5.2.4 where a practical evaluation of CRBED is presented.

5.2.1 Operation of the Candidate Reduction Based on Euclidean Distance Approach

In this sub-section, the operation of the proposed CRBED approach is presented when coupled with FSSBDTW and used for the purpose of 1NN classification. A block-diagram outlining the overall process is presented in Figure 5.1. The input is a set D of r time series $D = \{S_1, S_2, \ldots S_r\}$. The process then comprises two stages.

Stage 1 Parameter learning to find the best parameters.

Stage 2 Classification of a given query time series $\mathfrak{t}$.

It should be noted that Stage 1 is only undertaken once, whilst Stage 2 may be applied as many times as required. Unless, of course, more pre-labelled data becomes available when it may be desirable to repeat Stage 1.

Recall, from the work presented in Chapter 4 that the FSSBDTW approach utilised two parameters:

1. *maxLength*: The maximum permitted length of a segment.
2. t: The length of the tail within which the "cut" should be applied.

Previously the values for these parameters were pre-specified as advocated in [10] and [11]. In other words, they were assigned values in an intuitive manner. Finding the best parameter was not an option given the computational complexity of DTW. However, by adopting a cheaper (if less accurate) comparison mechanism, such as Euclidean distance comparison, parameter tuning becomes a realistic option. Recall also that we have the parameter $\mathfrak{r}$, the number of time series to be retained. Thus there are three parameters to be considered. We thus have a three-dimensional parameter space $|I| \times |T| \times |R|$ where:

I is the set of values to be considered for parameter *maxLength*, $I = \{maxLen_1, \ldots, maxLen_{|I|}\}$.

T is the set of values to be considered for parameter t, $T = \{t_1, \ldots, t_{|T|}\}$.

R is the set of values to be considered for parameter $\mathfrak{r}$, $R = \{\mathfrak{r}_1, \ldots, \mathfrak{r}_{|R|}\}$.

Preliminary experiments, using classification accuracy and F-1 score, indicated that there was no global "peak" in this parameter space, but instead many local maxima with, typically, one that was better than the rest. This precluded any form of "hill-climbing" strategy. An exhaustive search strategy was therefore adopted for Stage 1. What the preliminary experiments did indicate was that the sets I, T and R, the parameter space, should be defined as follows: $I = \{10, 20, 30, 40, 50\}$ ($|I| = 5$), $T = \{1, 2, \ldots, 9, 10\}$ ($|T| = 10$) and $R = \{1, 2, \ldots, 49, 50\}$ ($|R| = 50$).

Once the most appropriate parameters settings had been identified the classification of previously unseen (query) time series could commence, Stage 2. Given a query time series $\mathfrak{t}$ the first step (Stage 2.1) is to prune those time series in D that are unlikely to provide a good match using Euclidean Distance time series comparison. As noted above, Euclidean Distance measurement is significantly more efficient than DTW in terms of run-time (although less accurate). This then provided a reduced data set $D' = \{S_1, S_2, \ldots, S_{\mathfrak{r}}\}$. To obtain the final classification, Stage 2.2, kNN was applied in the same manner as before using FSSBDTW as described in Chapter 4 (Algorithm 4).

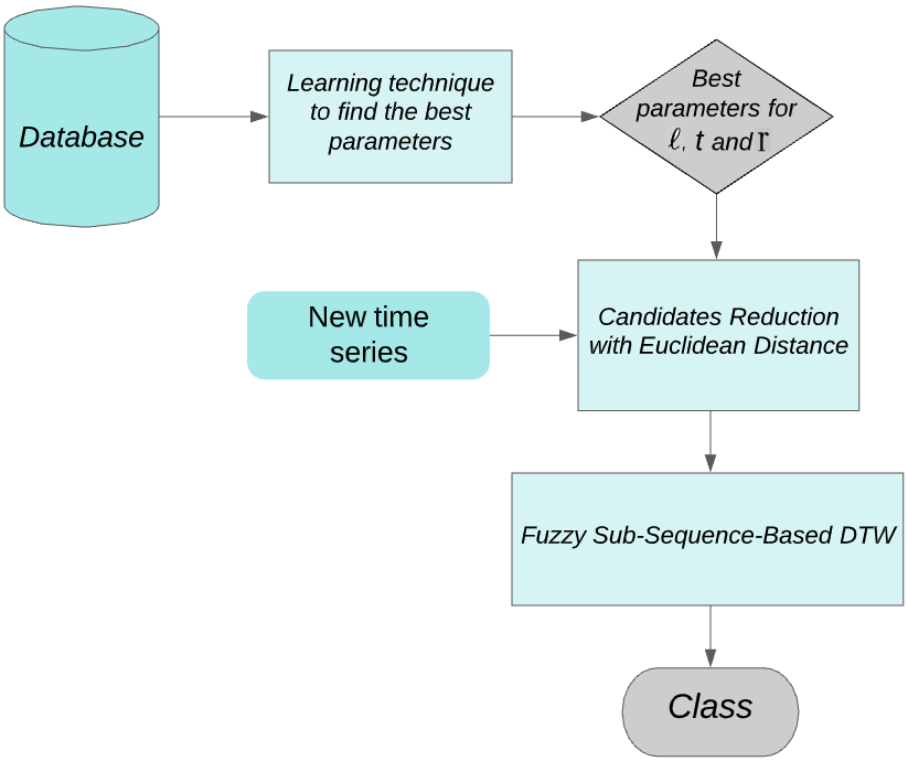

FIGURE 5.1: Schematic illustrating operation of Candidate Reduction Based on Euclidean Distance approach

5.2.2 Candidate Reduction Based on Euclidean Distance Approach Algorithm

This section presents the pseudo code for Stage 1 and Stage 2 of the above process. The pseudo code for Stage 1, parameter learning, is given in Algorithm 6. The input, line 1, is a collection of pre-labelled time series D_{train} to form the initial kNN bank, a second collection of pre-labelled time series D_{test} to form a test set, and the sets I, T and R. The two sets D_{train} and D_{test} are derived from D such that $D_{train} \cap D_{test} = \emptyset$ and $D_{train} \cup D_{test} = D$. There are a variety of ways in which D can be divided, but with respect to the evaluation presented later in this chapter a 9:1 training/test split was used. Next, line 2, a list in which to hold the best parameters is defined (initialised with three zeros), followed (line 3) by the definition of a variable *bestAccuracy* to hold the best accuracy obtained so far. This is followed (lines 4 to 13) by a set of three nested loops whereby the entire search space is processed. On each iteration, the function $kNN\_Euclidean$ is called with D_{train}, D_{test} and the selected parameters ($maxLen_i$, t_i and ɾ_i). The function returns the classification accuracy, the number of true positives and true negatives. This is then used to update the bestAccuracy variable and the best parameter list. Once the entire search space has been processed the best parameters will have been identified. Stage 1 is then complete and the identified parameters are returned (line 14). A similar algorithm is used with respect to the CRBLB approach described in the following section.

Stage 2 is concerned with the classification of an individual query time series ʈ. The pseudo code for the algorithm is presented in Algorithm 7. The input (line 1) is the labelled data set D, the query time series ʈ and the parameters identified in Stage 1, $maxLen$, t and ɾ. We start, lines 2 to 7, by applying the candidate reduction and generating $D' = \{S_1, S_2, \ldots, S\}$. Then, lines 8, the function $classification$ is called

Algorithm 6 Parameter Learning, Stage 1

1: **input** $D_{train}, D_{test}, I, T, R$.
2: $bestParameters = [0, 0, 0]$
3: $bestAccuarcy = 0$
4: **for** $\forall maxLen_i \in I$ **do**
5: **for** $\forall t_i \in T$ **do**
6: **for** $\forall$ $\Gamma_i \in R$ **do**
7: $newAccuracy =$ kNN_Euclidean$(D_{train}, D_{test}, maxLen_i, t_i, \Gamma_i)$
8: **if** $newAccuracy > bestAccuracy$ **then**
9: $bestAcuuracy = newAccuracy$
10: $bestParameters = [maxLen_i, t_i, _i]$
11: **end if**
12: **end for**
13: **end for**
14: **end for**
15: **return** $bestParameters$

which returns the final class. The pseudo code for the $classification$ function is given in Algorithm 8; this function is also used with respect to the CRBLB approach described later in this chapter. The function determines the best match in D' with t and retains the associated class c for this best match, which is returned in line 11. Note that the matching is done according to the best obtained warping distance ($bestWD$) generated using the FSSBDTW algorithm as presented in the previous Chapter.

Algorithm 7 Candidate Reduction Based on Euclidean Distance (CRBED), Stage 2.1

1: **input** $D, \text{t}, maxLen, t$,
2: $R =$ empty list to hold references to time series in D
3: **for** $\forall S_i \in D$ **do**
4: $dist = euclideanDistance(S_i, \text{t})$
5: $R = R$ with S_i inserted according to $dist$
6: **end for**
7: $D' =$ top Γ from R
8: **return** $classification(D', \text{t}, maxLen, t)$ (Algorithm 8)

Algorithm 8 Candidate Reduction Based on Euclidean Distance (CRBED), Stage 2.2

1: **input** $D', \text{t}, maxLen, t$
2: $bestWD = \infty$
3: $c =$ default class
4: **for** $\forall S_i \in D'$ **do**
5: $wd = FSSDDTW(S_i, \text{t}, maxLen, t)$
6: **if** $wd < bestWD$ **then**
7: bestWD = wd
8: $c =$ class associated with S_i
9: **end if**
10: **end for**
11: **return** c

5.2.3 Theoretical Time Complexity of Candidate Reduction Based on Euclidean Distance Approach

In Chapters 3 and 4 the practice was to present the time complexity of a particular approach assuming TCV and 1NN classification. We will do the same here. We will first consider the two stages independently and then provide an overall theoretical complexity.

Commencing with Stage 1, parameter tuning, from the foregoing, the time complexity for comparing two time series using Euclidean distance was:

$$EuclideanComparison_{complexity} = O\left(x\right) \tag{5.3}$$

Assuming a 9:1 training test set split the number of comparisons will be:

$$numberOfComparisons = \frac{|D|}{10} \times \frac{9 \times |D|}{10} = \frac{9 \times |D|^2}{100} \tag{5.4}$$

The number of parameter combinations is given by:

$$numberParameterCombinations = |I| \times |T| \times |R| \tag{5.5}$$

The complexity of Stage 1 is thus given by:

$$\begin{aligned} stage1_{complexity} = \ & numberParameterCombinations \times numberOfComparisons \\ & \times EuclideanComparison_{complexity} \end{aligned} \tag{5.6}$$

Thus:

$$stage1_{complexity} = O\left(|I| \times |T| \times |R| \times \frac{9 \times |D|^2}{100} \times x\right) \tag{5.7}$$

From the foregoing, the sets I, T and R were designed to be pre-specified and hard coded into the Stage 1 process. Suggested values for the sets I, T and R were presented above. Using these values, $|I| = 5$, $|T| = 10$ and $|R| = 50$. The above thus becomes:

$$stage1_{complexity} = O\left(2500 \times \frac{9 \times |D|^2}{100} \times x\right) \tag{5.8}$$

which simplifies to:

$$stage1_{complexity} = O\left(225 \times |D|^2 \times x\right) \tag{5.9}$$

When using TCV, the parameter tuning will be conducted 10 times. Hence the above will become:

$$stage1TCV_{complexity} = O\left(10 \times 225 \times |D|^2 \times x\right) = O\left(2250 \times |D|^2 \times x\right) \tag{5.10}$$

Considering Stage 2, the number of Euclidean distance comparisons to decide (generate) D' will equate to $|D|$. The complexity to conduct the candidate reduction will thus be:

$$\begin{aligned} candidateReduction_{complexity} &= O\left(|D| \times EuclideanComparison_{complexity}\right) \\ &= O\left(|D| \times x\right) \end{aligned} \tag{5.11}$$

The determination of the class of a query time series will then require $|D'|$ comparisons, or more simply $\mathfrak{r}$ comparisons. The complexity for class determination, assuming the FSSBDTW approach, will then be:

$$classDetermination_{complexity} = O\left(\mathfrak{r} \times DTW_{complexityFSSBDTW}\right) \tag{5.12}$$

which, from Equation 4.11, equates to:

$$classDetermination_{complexity} = O\left(\mathfrak{r} \times x \times \overline{len}\right) \tag{5.13}$$

where x is the length of a time series and $\overline{len}$ is the average length of a segment. The overall complexity of Stage 2 will then equate to:

$$stage2_{complexity} = candidateReduction_{complexity} + classDetermination_{complexity} \tag{5.14}$$

Thus:

$$\begin{aligned} stage2_{complexity} &= O\left(\left(|D| \times x\right) + \left(\mathfrak{r} \times x \times \overline{len}\right)\right) \\ &= O\left(x\left(|D| + \left(\mathfrak{r} \times \overline{len}\right)\right)\right) \end{aligned} \tag{5.15}$$

If we are undertaking TCV the number of classifications will equate to $|D|$. Thus the above becomes:

$$stage2TCV_{complexity} = O\left(\left(x \times |D|\right)\left(|D| + \left(\mathfrak{r} \times \overline{len}\right)\right)\right) \tag{5.16}$$

The overall complexity of the CRBED approach, assuming TCV, will then be:

$$CRBED\_TCV_{complexity} = stage1TCV_{complexity} + stage2TCV_{complexity} \tag{5.17}$$

In other words:

$$\begin{aligned} CRBED\_TCV_{complexity} &= O\left(\left(2250 \times |D|^2 \times x\right) + \left(x \times |D|\right)\left(|D| + \left(\mathfrak{r} \times \overline{len}\right)\right)\right) \\ &= O\left(\left(x \times |D|\right)\left(\left(2250 \times |D|\right) + |D| + \left(\mathfrak{r} \times \overline{len}\right)\right)\right) \\ &= O\left(\left(x \times |D|\right)\left(\left(2251 \times |D|\right) + \left(\mathfrak{r} \times \overline{len}\right)\right)\right) \end{aligned} \tag{5.18}$$

5.2.4 Evaluation of Candidate Reduction Based on Euclidean Distance Approach

The evaluation of the proposed CRBED approach is presented in this sub-section. The evaluation was conducted using 1NN classification and the fifteen selected data sets from the UEA and UCR Time Series Classification repository [14] in the same manner as reported on in Chapters 3 and 4. Experiments were conducted comparing the operation of the proposed CRBED approach with the two benchmark approaches from Chapter 3: (i) Standard DTW and (ii) S-C Band DTW. With respect to the later, the Sakoe-Chiba Band warping window size, $\ell = 10\%$, was defined in the same way as before using the suggested mechanism proposed in [81, 92]. The objectives of the evaluation were:

1. **Operational Analysis**: To analyse the operation of the proposed CRBED approach.

2. **Efficiency Comparison**: To evaluate the run-time advantages gained using the CRBED approach compared to the Standard DTW and S-C Band DTW benchmark approaches.

3. **Effectiveness Comparison**: To determine whether the classification accuracy of the proposed CRBED approach was commensurate with that obtained using Standard DTW and S-C Band DTW approaches.

For the evaluation, a desktop computer was used with an Apple M1 processor with 8 core CPU, 8 core GPU, 16 core Neural Engine, 16GB unified memory, and 512GB SSD storage. The same set up as used with respect to the evaluations reported on in previous Chapters. The evaluation metrics collected were again total run time (seconds), accuracy and F1-score; the later derived from a confusion matrix [33, 38]. The accuracy and F1-score values presented later in this section are all average values, collected using TCV [62, 89].

TABLE 5.1: Best values for parameters $maxLen$, t and $\mathfrak{r}$ in terms of run time, accuracy and F1 results using 15 evaluation data sets and the CRBED approach.

ID #	**Data set**	**Length** $maxLen$	**Tail** t	**Retained Candidates** $\mathfrak{r}$	**Run time** (Secs)	**Acc** (SD)	**F1** (SD)
1	**SmoothSubspace**	5	1	3	0.92	98.33 (0.03)	0.98 (0.03)
2	**ItalyPowerDemand**	6	3	3	13.77	97.17 (0.01)	0.97 (0.01)
3	**Libras**	10	3	4	2.64	65.83 (0.11)	0.65 (0.11)
4	**SyntheticControl**	30	2	22	12.10	98.50 (0.01)	0.98 (0.01)
5	**GunPoint**	40	2	8	2.66	99.50 (0.01)	0.99 (0.01)
6	**OliveOil**	40	9	4	1.88	90.00 (0.11)	0.90 (0.11)
7	**Trace**	70	2	10	4.72	99.00 (0.03)	0.99 (0.03)
8	**ToeSegmentation2**	30	3	3	2.32	92.72 (0.04)	0.92 (0.04)
9	**Car**	60	6	13	5.15	88.33 (0.06)	0.86 (0.06)
10	**Lighting2**	80	2	4	3.00	87.76 (0.07)	0.87 (0.07)
11	**ShapeletSim**	40	1	26	14.94	89.50 (0.06)	0.89 (0.06)
12	**DiatomSizeReduction**	40	2	1	2.98	100.00 (0.00)	1.00 (0.00)
13	**Adiac**	50	1	1	8.40	65.83 (0.03)	0.64 (0.03)
14	**HouseTwenty**	40	1	9	19.04	93.71 (0.04)	0.93 (0.04)
15	**PenDigits**	5	2	1	120.37	89.64 (0.01)	0.89 (0.01)

Operational Analysis

The performance of the proposed CRBED approach, in terms of run time, accuracy and the F1 measure, is considered here together with the nature of the derived values for $maxLen$, t and $\mathfrak{r}$. Table 5.1 gives the results obtained. Columns 3, 4 and 5 give the

maxLen, t and γ values in each case. Column 6 gives the run time in seconds. Columns 7 and 8 give the accuracy values and F1 scores obtained; the numbers in parentheses give the associated standard deviation. As before, there is no definitive value for *maxLen* or t. It is also interesting to note the range of values γ. This all supports the intuition of including a parameter tuning stage in the CRBED approach.

Efficiency Comparison

Figure 5.2 gives a comparison of the run time performance of the three approaches considered here. The x-axis represents the data set identification number given in Table 5.1, and the y-axis represents the run time in seconds. The values for *maxLen*, t and γ that were used to obtain the run times were those also reported in Table 5.1. As noted above, for the S-C Band DTW approach $\ell = 10\%$ was used as suggested in [81, 92]. From the figure, it can be seen that for all data sets the proposed CRBED approach produced the best run times despite, the effort directed at parameter tuning and candidate reduction. This is particularly evident with respect to the larger data sets, *Adiac*, *HouseTwenty* and *PenDigits*. In Chapters 3 and 4, the recorded run times were used to determine and test an actual run time equation from the theoretical complexity equation. This was done by determining the value of a constant z, the approximate run time for a single application of DTW. However, in the case of the CRBED approach we have both Euclidean distance and DTW applications, thus two constants z_1 and z_2. Determining the proportion of the run time to be allocated to the calculation of z_1, and the proportion to be allocated to the calculation of z_2, is not straight forward. Hence, unlike in Chapters 3 and 4, an actual run time equation is not presented here (this is left as a challenge for future work).

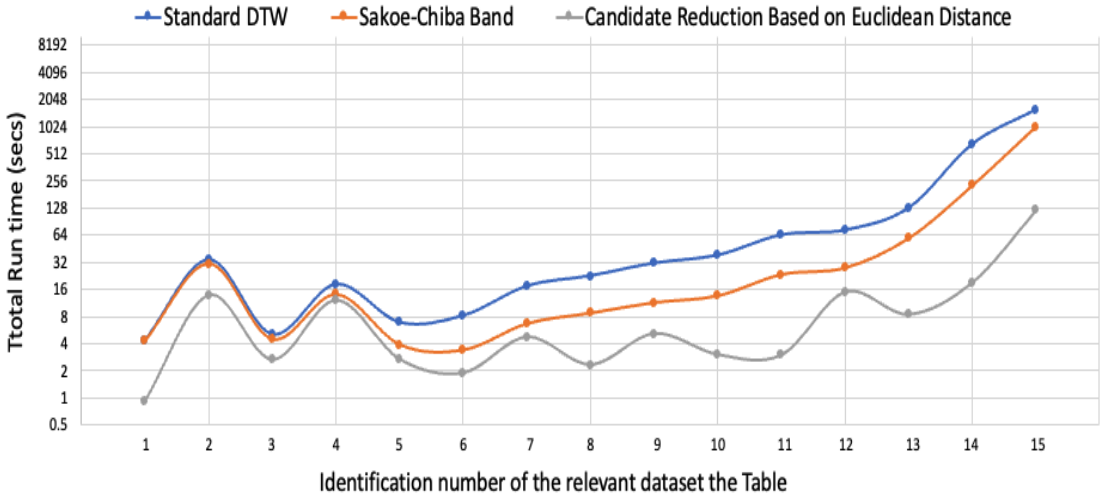

FIGURE 5.2: Comparison of Run time results using Standard DTW, S-C Band DTW and the CRBED approach.

Effectiveness Comparison

Table 5.2 presents a comparison between CRBED, Standard DTW and S-C Band DTW in terms of accuracy and F1 score. The values in parenthesis are the standard deviations recorded after averaging over the ten folds of the TCV. Best results are highlighted in bold font. From the table, it can be seen that the proposed CRBED approach produced the best results with respect to fourteen of the fifteen data sets considered; for the

TABLE 5.2: Accuracy and F1-Score with standard deviation for the fifteen evaluation data set using Standard DTW, S-C Band DTW and CRBED (best results in bold font).

ID	**Data set**	**Standard DTW**		**S-C Band $\ell = 10$**		**CRBED**	
#	**Name**	**Acc** (SD)	**F1** (SD)	**Acc** (SD)	**F1** (SD)	**Acc** (SD)	**F1** (SD)
1	**SmoothSubspace**	91.00 (0.04)	0.91 (0.04)	95.00 (0.02)	0.95 (0.02)	**98.33 (0.03)**	**0.98 (0.03)**
2	**ItalyPowerDemand**	95.61 (0.02)	0.95 (0.02)	95.70 (0.02)	0.95 (0.02)	**97.17 (0.01)**	**0.97 (0.01)**
3	**Libras**	58.89 (0.10)	0.60 (0.11)	63.06 (0.11)	0.60 (0.11)	**65.83 (0.11)**	**0.65 (0.11)**
4	**SyntheticControl**	**98.50 (0.01)**	**0.98 (0.01)**	**98.50 (0.01)**	**0.98 (0.01)**	**98.50 (0.01)**	**0.98 (0.01)**
5	**GunPoint**	94.50 (0.05)	0.94 (0.05)	97.50 (0.03)	0.97 (0.03)	**99.50 (0.01)**	**0.99 (0.01)**
6	**OliveOil**	86.67 (0.15)	0.86 (0.16)	86.67 (0.15)	0.86 (0.16)	**90.11 (0.13)**	**0.90 (0.11)**
7	**Trace**	**99.00 (0.03)**	**0.99 (0.03)**	**99.00 (0.03)**	**0.99 (0.03)**	**99.00 (0.03)**	**0.99 (0.03)**
8	**ToeSegmentation2**	88.46 (0.09)	0.88 (0.10)	92.68 (0.07)	0.92 (0.7)	**92.72 (0.04)**	**0.92 (0.04)**
9	**Car**	80.83 (0.07)	0.80 (0.09)	81.67 (0.07)	0.81 (0.08)	**88.33 (0.06)**	**0.86 (0.06)**
10	**Lightning2**	87.76 (0.09)	0.87 (0.08)	87.76 (0.09)	0.87 (0.08)	**87.76 (0.07)**	**0.87 (0.07)**
11	**ShapeletSim**	82.00 (0.10)	0.81 (0.11)	82.00 (0.10)	0.81 (0.11)	**89.50 (0.06)**	**0.89 (0.08)**
12	**DiatomSizeReduction**	99.38 (0.01)	0.99 (0.01)	99.69 (0.01)	0.99 (0.01)	**100.00 (0.00)**	**1.00 (0.00)**
13	**Adiac**	65.30 (0.04)	0.62 (0.04)	65.30 (0.04)	0.62 (0.04)	**65.83 (0.03)**	**0.64 (0.03)**
14	**HouseTwenty**	**95.00 (0.05)**	**0.95 (0.05)**	93.00 (0.08)	0.93 (0.08)	93.71 (0.04)	0.93 (0.04)
15	**PenDigits**	85.47 (0.01)	0.85 (0.01)	87.62 (0.01)	0.87 (0.01)	**89.64 (0.01)**	**0.89 (0.01)**

remaining one case, *HouseTwenty*, a second best performance was recorded. Where the performance was improved, it was postulated that reducing the number of records considered in the FSSBDTW process prevented anomalous matches.

5.3 Candidate Reduction Based on Lower Bounding Approach

In this section, the proposed Candidate Reduction Based on Lower Bounding (CRBLB) approach is presented. A criticism that may be directed at the CRBED approach presented in the previous section was that the γ parameter was tuned using a hard-coded parameter space measuring $\{1, 2, \ldots, 49, 50\}$. However, it might be possible that the best value for γ lies outside of this space. The proposed CRBLB approach addresses this deficiency by considering an alternative candidate reduction mechanism.

The fundamental idea of the proposed CRBLB approach is the same as that underpinning the CRBED approach described above; to reduce (filter) the number of candidates in the input time series data set D to produce a reduced set of time series D'

to which FSSBDTW can then be applied ($D' \subset D$). However, instead of doing this by prioritising the time series in the kNN bank according to similarity with the query time series t, measured using Euclidean distance, the idea is to use a bounding mechanism. Given a set of integers a lower bound is any integer that is less than equal to all the integers in the set. An upper bound is then any integer that is greater than or equal to every integer in the set. We say that the set is "bounded" by the two values. The same can be applied to time series. The relevance is that we can think of the lower and upper bounds as defining an envelope surrounding t. The lower and upper bounds can be thought of as time series in their own right. Then, for a time series from the kNN bank to be a potential match for a query time series, it must fall entirely within this band. Thus the banding principle can be used as a candidate reduction mechanism. There are many ways whereby such banding can be defined and implemented. Examples can be found in [61, 68, 80, 115]. With respect to the work presented in this chapter the LB_Keogh method [59] was adopted because, in many respects, it is the simplest to implement. Recall that LB_Keogh was described in sub-section 2.3.4 Using LB_Keogh the band is defined by a parameter w, which is essentially an offset, applied along the time dimension to each point in t to form the upper and lower bound "tram lines".

The reminder of this section is structured in a similar manner to the previous section. The section commences, Sub-section 5.3.1 with an overview of the CRBLB approach. The pseudo code for the proposed CRBLB approach is then presented in Sub-section 5.3.2. This is followed by a discussion of theoretical time complexity for the proposed approach in Sub-section 5.3.3. An evaluation of the CRBLB approach is then reported on in Section 5.3.4.

5.3.1 Operation of Candidate Reduction Based on Lower Bounding Approach

In this sub-section, the operation of the proposed CRBLB approach is presented when coupled with FSSBDTW and used for the purpose of 1NN classification. A block-diagram outlining the overall process is presented in Figure 5.3. The input, as in the case of the CRBED approach, is a set D of r time series $D = \{S_1, S_2, \ldots S_r\}$. The process then comprises two stages.

Stage 1 Parameter learning to find the best parameters.

Stage 2 Classification of a given query t time series using CRBLB and the application of FSSBDTW.

Stage 1 is similar to that described for CRBED except that the parameter γ (the number of "best matched" time series to be selected) was not considered, instead the parameter w, the bounding offset parameter, was considered. A parameter space measuring $|I| \times |T| \times |W|$ was thus used where:

I is the set of values to be considered for parameter $maxLength$, $I = \{maxLen_1, \ldots, maxLen_{|I|}\}$.

T is the set of values to be considered for parameter t, $T = \{t_1, \ldots, t_{|T|}\}$.

W is the set of values to be considered for parameter w, $W = \{w_1, \ldots, w_{|W|}\}$.

Preliminary experiments using selected parameter settings indicated that in this case the sets I, T and W should be defined as follows: $I = \{10, 20, \ldots, 90, 100\}$ ($|I| = 10$)

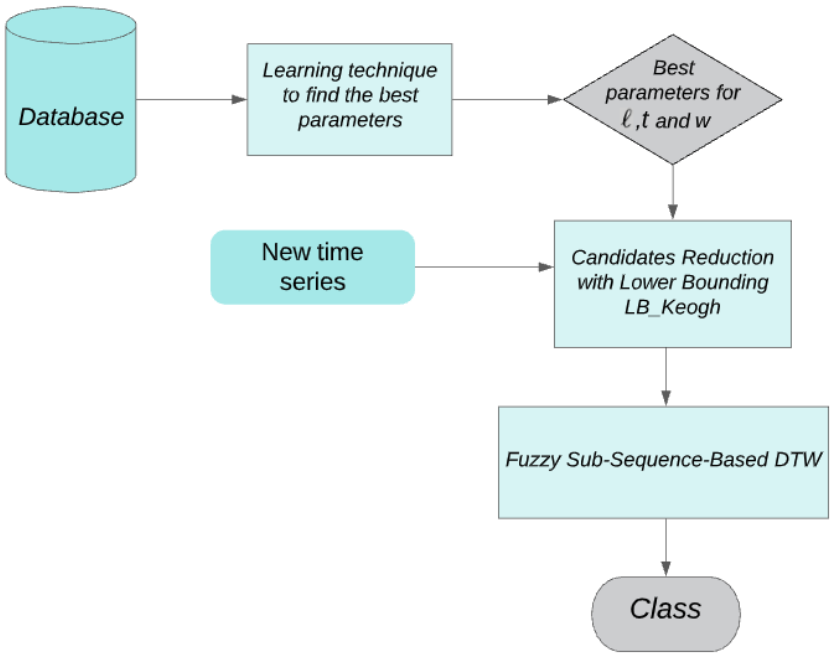

FIGURE 5.3: Schematic illustrating operation of Candidate Reduction Based on Lower Bounding appoach

for medium and long time series, and $I = \{5, 6, 7, 8, 9, 10\}$ ($|I| = 6$) for short time series; $T = \{1, 2, \ldots, 9, 10\}$ ($|T| = 10$); and $W = \{5, 10, \ldots, 45, 50\}$ ($|W| = 10$). Note that a different range of values for I was used with respect to the CRBED approach. The different ranges were used purely as a consequence of the preliminary experiments. Note that a short time series was defined earlier, in Chapter 3, as a time series where $x < 15$ as in the case of the *PenDigits* and *SmoothSubspace* data sets. A medium or long time series is then a time series where $x \geq 15$. Recall also that the parameter generation only needs to be conducted once (unless the set D is added to, or changed in some way).

Stage 2 comprises three steps; (i) lower and upper bound generation (Stage 2.1), (ii) candidate reduction (Stage 2.2) and (ii) classification (Stage 2.3). We commence, Stage 2.1, by generating the upper and lower bounds on either side of $\mathfrak{t}$. The upper and lower bound time series, L and U, comprise a sequence of points such that $L = \{l_1, l_2, \ldots, l_x\}$ and $U = \{u_1, u_2, \ldots, u_x\}$ (where x is the length of $\mathfrak{t}$). As noted above, the values in L and U are determined by applying a parameter w to $\mathfrak{t}$. The parameter w defines a number of units of time, one or more. The value for a point l_i in U was determined as follows:

$$l_i = \begin{cases} \mathfrak{t}_{i-w} & \text{if } (\mathfrak{t}_{i-w} > \mathfrak{t}_i) \\ \mathfrak{t}_i & \text{otherwise} \end{cases} \tag{5.19}$$

The first describes the downward slope situation, the second the default case. With respect to the lower bound a similar argument applies:

$$u_i = \begin{cases} \mathfrak{t}_{i-w} & \text{if } (\mathfrak{t}_{i-w} < \mathfrak{t}_i) \\ \mathfrak{t}_i & \text{otherwise} \end{cases} \tag{5.20}$$

In this case the first describes the upward slope situation.

Next the candidate reduction is conducted (Stage 2.2) where the time series in D that are unlikely to provide a good match are pruned, using the LB_Keogh lower bounding

method, to give $D' = \{S_1, S_2, \dots\}$. Note that in [59] Stages 2.1 and 2.2 are combined and the actual bounds are not specifically calculated. However, the approach presented here avoids some repeat calculation and thus, it is suggested, is more efficient. A time series S_i from D is pruned if it is not entirely contained within the band surrounding t. This is determined using a distance measurement, LB_{sum} calculated as follows (where $u_i \in U$ and $l_i \in L$):

$$LB_{sum} = \sum_{\forall p_i \in S_i} dist(p_i) \tag{5.21}$$

$$dist(p_i) = \begin{cases} p_i - u_i & \text{if } (p_i > u_i) \\ l_i - p_i & \text{if } (p_i < l_i) \\ 0 & \text{otherwise} \end{cases} \tag{5.22}$$

If LB_{sum} is equal to zero, then S_i is within the band and therefore not pruned; otherwise S_i is pruned. The process is illustrated in Figure 5.4 which gives two examples taken from [59]. In the figure, the blue vertical lines indicated where distance measurements are made because S_i (labeled as "time series" in the figure), is outside of the band surrounding t (labeled as "new time series" in the figure).

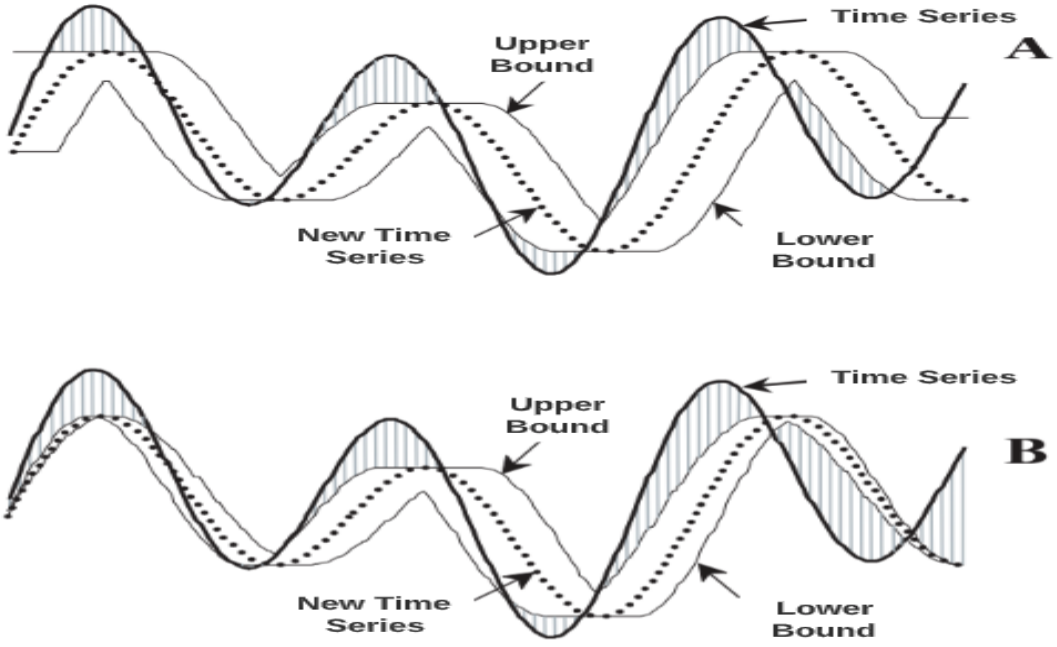

FIGURE 5.4: Two examples of the application of LB_Keogh Lower Bound mechanism [59].

To obtain the final classification, Stage 2.3, kNN is applied in the same way as before using FSSBDTW as described in Chapter 4 (Algorithm 4).

5.3.2 Candidate Reduction Based on Lower Bounding Algorithm

This section presents the pseudo code for Stage 2 of the above process. The algorithm for Stage 1 is essentially the same as Algorithm 6 and is therefore not considered further here. Stage 2 is where a previously unseen query time series t is labeled with a class label c taken from a set of labels C. The pseudo code for the algorithm is presented in Algorithm 9. The input (line 1) is the labelled data set D, the query time series t,

Algorithm 9 Candidate Reduction Based on Lower Bounding Algorithm

```
1: input D, t, maxLen, t, w
2: % Stage 2.1
3: U = generateUpperBound(t, w)
4: L = generateLowerBound(t, w)
5: % Stage 2.2
6: D' = empty set to hold retained time series
7: c = default class
8: bestSum = ∞
9: for ∀S_i ∈ D do
10:     boundSum = boundingSum(S_i, U, L)
11:     if boundSum == 0 then
12:         D' = D' with S_i appended.
13:     else
14:         if boundSum < bestSum then
15:             bestSum = boundSum
16:             c = class associated with S_i
17:         end if
18:     end if
19: end for
20: % Stage 2.3
21: if D' ≠ ∅ then
22:     c = classification(D', t, maxLen, t) (Algorithm 8)
23: end if
24: return c
```

the *maxLen*, t and w parameters identified in Stage 1. The first step (Stage 2.1) is to define and populate the point series U and P, the upper and lower bound sequences (pseudo time series). The sets are defined in lines 3 and 4 through calls to the functions *generateUpperBound* and *generateLowerBound*. Next, the pruned set D' is generated (Stage 2.2) in lines 6 to 19. Note that the code takes into account the potential for the set D' to be empty in which case the class associated with the nearest time series to t is allocated to t. If the set D' is not empty DTW is applied using the classification function presented earlier with respect to the CRBED approach (Algorithm 8).

The pseudo code for the functions *generateUpperBound* and *generateLowerBound* are given in Algorithms 10 and 11. The two algorithms are essentially the same except for line 9, but both have been included here for reasons of completeness.

The pseudo code for the *boundingSum* function is given in Algorithm 12. The input is a time series $S \in D$ and the lower and upper bound sequences U and L. A simple comparison is then conducted between each point p_i in S and each corresponding point in U and L. The process was illustrated in Figure 5.4. It is of course possible to integrate Algorithms 11, 10 and 12 as in the case of [59].

5.3.3 Theoretical Time Complexity of Candidate Reduction Based on Lower Bounding Approach

In this sub-section, the theoretical time complexity of the CRBLB approach is considered. As in the case of the discussion regarding the theoretical complexity of the CRBED approach, the complexity of the CRBLB approach will be presented by considering the two stages independently and then deriving an overall theoretical complexity.

Algorithm 10 Generate Upper Bound

```
1: input D, t, w
2: U = empty "time series" representing the upper bound
3: i = 0
4: while i < w do
5:     u_i = t_i (u_i ∈ U)
6:     i++
7: end while
8: while i ≤ x do
9:     if t_i < t_{i−w} (downward slope) then u_i = t_{i−w}
10:    else
11:        u_i = t_i
12:    end if
13: end while
14: return U
```

Algorithm 11 Generate Lower Bound

```
1: input D, t, w
2: L = empty "time series" representing the lowerr bound
3: i = 0
4: while i < w do
5:     l_i = t_i (l_i ∈ L)
6:     i++
7: end while
8: while i ≤ x do
9:     if t_i > t_{i−w} (upward slope) then
10:        l_i = t_{i−w}
11:    else
12:        l_i = t_i
13:    end if
14: end while
15: return L
```

Starting with Stage 1, the complexity is similar to Stage 1 of the CRBED approach. This was expressed in Equation 5.6 as follows.

$$stage1_{complexity} = numberParameterCombinations \times numberOfComparisons \times EuclideanComparison_{complexity} \tag{5.23}$$

However, in the case of the CRBLB approach, the number of parameter combinations is equivalent to:

$$numberParameterCombinations = |I| \times |T| \times |W| \tag{5.24}$$

where I is the set of values to be considered for parameter $maxLength$, $I = \{maxLen_1, \ldots, maxLen_{|I|}\}$; T is the set of values to be considered for parameter t, $T = \{t_1, \ldots, t_{|T|}\}$ and W is the set of values to be considered for parameter w, $W = \{w_1, \ldots, w_{|W|}\}$. Thus with reference to Equation 5.7 the complexity of Stage 1 is given by:

Algorithm 12 Bounding Sum

```
1: input S, U, L
2: boundSum = 0
3: for ∀p_i ∈ S do
4:     if p_i > u_i (u_i ∈ U) then
5:         dist = p_i − u_i
6:     else if p_i < l_i (l_i ∈ L) then
7:         dist = l_i − p_i
8:     else
9:         dist = 0
10:    end if
11:    boundSum = boundSum + dist
12: end for
13: return boundSum
```

$$stage1_{complexity} = O\left(|I| \times |T| \times |W| \times \frac{9 \times |D|^2}{100} \times x\right) \tag{5.25}$$

As noted earlier, the sets I, T and W have been hard coded into the algorithm, hence $|I| = 10$ (6 for short time series), $|T| = 10$ and $|W| = 10$. The above, assuming long time series, thus becomes:

$$stage1_{complexity} = O\left(90 \times |D|^2 \times x\right) \tag{5.26}$$

When using TCV, the parameter tuning will be conducted 10 times. Hence the above will become:

$$stage1TCV_{complexity} = O\left(10 \times 90 \times |D|^2 \times x\right) = O\left(900 \times |D|^2 \times x\right) \tag{5.27}$$

For short time series ($x < 15$) the above will be:

$$stage1TCV_{complexity} = O\left(540 \times |D|^2 \times x\right) \tag{5.28}$$

Considering Stage 2 we have three components, lower and upper bound generation (Stage 2.1), candidate reduction (Stage 2.2) and classification (Stage 2.3). Considering each in turn, generating the upper and lower bounds will entail a complexity of:

$$boundGeneration_{complexity} = O\left(2x \times |D|\right) \tag{5.29}$$

where x is the length of a time series and $|D|$ is the number of time series in D. The complexity of the candidate reduction will be the same as for the CRBED approach:

$$candidateReduction_{complexity} = O\left(x \times |D|\right) \tag{5.30}$$

The class determination complexity, using FSSBDTW, is also the same as in the case of the CRBED approach:

$$classDetermination_{complexity} = O\left(\Gamma \times x \times \overline{len}\right) \tag{5.31}$$

The overall complexity of Stage 2 will then equate to:

$$stage2_{complexity} = \begin{array}{l} boundGeneration_{complexity} + candidateReduction_{complexity} + \\ classDetermination_{complexity} \end{array} \quad (5.32)$$

Thus:

$$\begin{aligned} stage2_{complexity} &= O\left((2x \times |D|) + (x \times |D|) + \left(\Gamma \times x \times \overline{len}\right)\right) \\ &= O\left(x\left((x \times |D|) + |D| + \left(\Gamma \times \overline{len}\right)\right)\right) \end{aligned} \quad (5.33)$$

If we are undertaking TCV, the number of classifications will equate to $|D|$. Thus the above becomes:

$$stage2TCV_{complexity} = O\left((x \times |D|)\left((x \times |D|) + |D| + \left(\Gamma \times \overline{len}\right)\right)\right) \quad (5.34)$$

The overall complexity of the CRBLB approach, assuming TCV, will then be:

$$CRBLB\_TCV_{complexity} = stage1TCV_{complexity} + stage2TCV_{complexity} \quad (5.35)$$

In other words, assuming long time series:

$$\begin{array}{l} CRBLB\_TCV_{complexity} = \\ O\left(\left(900 \times |D|^2 \times x\right) + \left((x \times |D|)\left((x \times |D|) + |D| + \left(\Gamma \times \overline{len}\right)\right)\right)\right) = \\ O\left(\left(900 \times |D|^2 \times x\right) + \left(x^2 \times |D|^2\right) + \left(x \times |D|^2\right) + \left(x \times |D| \times \Gamma \times \overline{len}\right)\right) = \\ O\left((x \times |D|)\left((900 \times |D|) + (x \times |D|) + |D| + \left(\Gamma \times \overline{len}\right)\right)\right) \end{array} \quad (5.36)$$

For short time series this will equate to:

$$CRBLB\_TCV_{complexity} = O\left((x \times |D|)\left((540 \times |D|) + (x \times |D|) + |D| + \left(\Gamma \times \overline{len}\right)\right)\right) \quad (5.37)$$

Comparing the overall theoretical complexity for CRBLB with that calculated for CRBED given in Equation 5.18 and given again below, it can be seen that the theoretical complexity of CRBLB is greater than that for CRBED, although not significantly so. Of course, the complexity in both cases will be affected by the nature of the parameter space, but the proposed parameter space settings do seem to produce good results as demonstrated by the evaluations reported on in this chapter. In practice, TCV would not be used, thus serving to reduce the above two complexities, but the relative difference between the two will remain the same.

$$CRBED\_TCV_{complexity} = O\left((x \times |D|)\left((2251 \times |D|) + \left(\Gamma \times \overline{len}\right)\right)\right) \quad (5.38)$$

5.3.4 Evaluation of the Candidate Reduction Based on Lower Bounding Approach

In this section, the evaluation of the proposed CRBLB approach is presented. The same fifteen selected data sets from the UEA-UCR Time Series Classification repository as

used with respect to earlier experiments reported in this thesis were used. As before, 1NN classification and TCV was adopted. The evaluation objectives, as in the case of the CRBED evaluation reported on in the previous section, were:

1. **Operational Analysis**: To analyse the operation of the proposed CRBLB approach.
2. **Efficiency Comparison**: To evaluate the run-time advantages gained using the CRBLB approach compared to the Standard DTW and S-C Band DTW benchmark approaches.
3. **Effectiveness Comparison**: To determine whether the classification accuracy of the proposed CRBLB approach was commensurate with that obtained using standard DTW and S-C Band DTW approache.

The results obtained are discussed in the remainder of this sub-section.

Operational Analysis

The performance of the proposed CRBLB approach, in terms of run time, accuracy and the F1 measure, is considered in this sub-section together with the nature of the derived values for $maxLen$, t and w. Table 5.3 gives the results obtained. The layout of the table is the same as Table 5.1 used with respect to the evaluation of the CRBED approach. Columns 3, 4 and 5 give the $maxLen$, t and w values. Column 6 of the table gives the run time in seconds. Columns 7 and 8 give the accuracy values and F1 scores obtained; the numbers in parentheses give the associated standard deviation. As before, there is no definitive value for $maxLen$ or t. It is also interesting to note the range of values for w. Best classification results were obtained using different settings.

Efficiency Comparison

Figure 5.2 gives a comparison of the run time performance of the proposed CRBLB approach compared with the Standard DTW and S-C Band DTW benchmark approaches; the last with $\ell = 10\%$ as suggested in [81, 92]. The same value for ℓ as used with respect to the evaluation of the CRBED approach discussed in the previous section. As before, the x-axis represents the data set identification number as itemised previously in Tables 5.1 and 5.3, and the y-axis represents the run time in seconds. The values for $maxLen$, t and w that were used in each case were those reported in Table 5.3. From the figure, it can be seen that for all data sets, the proposed CRBLB approach was the most efficient in all cases; although not particularly evident with respect to the largest data set, *PenDigits*. For similar reasons as in the case of the CRBED approach, the run time values obtained were not used to calculate an actual run time equation because of the challenge of deriving such an equation which was considered outside of the scope of this thesis, and thus left as a topic for future work (see Chapter 7.

Effectiveness Comparison

Table 5.2 present a comparison between CRBLB, Standard DTW and S-C Band DTW in terms of accuracy and F1 score. As before, the values in parenthesis are the standard deviations recorded after averaging over the TCV folds. Best results are highlighted in bold font. From the table, it can be seen that the proposed CRBLB approach, in seven of the fifteen cases, produced a better performance than that produced using either

TABLE 5.3: Best values for parameters $maxLen$, t and w in terms of run time, accuracy and F1 results using 15 evaluation data sets and the CRBLD approach.

ID #	**Data set**	**Length** $maxLen$	**Tail** t	**Offset** w	**Run time** (Secs)	**Acc** (SD)	**F1** (SD)
1	**SmoothSubspace**	6	2	15	0.92	96.97 (0.03)	0.96 (0.03)
2	**ItalyPowerDemand**	6	3	20	13.20	95.98 (0.02)	0.95 (0.02)
3	**Libras**	9	1	50	2.14	59.44 (0.09)	0.59 (0.09)
4	**SyntheticControl**	30	1	15	6.44	98.17 (0.02)	0.98 (0.02)
5	**GunPoint**	40	2	50	1.26	95.50 (0.04)	0.95 (0.04)
6	**OliveOil**	40	2	5	1.32	88.33 (0.11)	0.88 (0.11)
7	**Trace**	70	2	30	1.97	99.00 (0.03)	0.99 (0.03)
8	**ToeSegmentation2**	30	3	20	1.61	88.49 (0.06)	0.88 (0.06)
9	**Car**	60	5	20	1.76	82.50 (0.09)	0.82 (0.09)
10	**Lighting2**	80	2	40	2.14	87.63 (0.07)	0.87 (0.07)
12	**ShapeletSim**	40	2	5	13.12	89.50 (0.07)	0.89 (0.07)
11	**DiatomSizeReduction**	40	2	15	6.64	100.00 (0.00)	1.00 (0.00)
13	**Adiac**	50	2	15	19.51	66.58 (0.03)	0.64 (0.03)
14	**HouseTwenty**	40	1	20	18.24	93.04 (0.05)	0.93 (0.05)
15	**PenDigits**	5	2	20	869.43	85.00 (0.01)	0.85 (0.01)

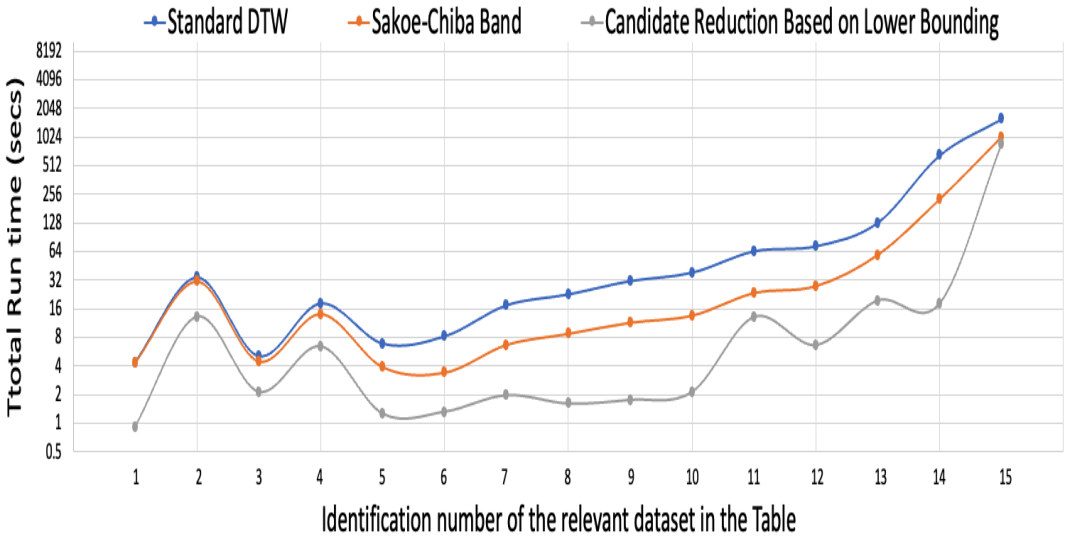

FIGURE 5.5: Comparison of Run time results using Standard DTW, S-C Band DTW and the CRBLB approach.

Standard DTW or S-C Band DTW; in the other eight cases, an almost identical performance was recorded. Where the performance improved over Standard DTW and/or

TABLE 5.4: Accuracy and F1-Score with standard deviation for the fifteen evaluation data set using Standard DTW, S-C Band DTW and CRBLB (best results in bold font).

ID	**Data set**	**Standard DTW**		**S-C Band $\ell = 10$**		**CRBLB**	
#	**Name**	**Acc** (SD)	**F1** (SD)	**Acc** (SD)	**F1** (SD)	**Acc** (SD)	**F1** (SD)
1	**SmoothSubspace**	91.00 (0.04)	0.91 (0.04)	95.00 (0.02)	0.95 (0.02)	**96.97 (0.03)**	**0.96 (0.03)**
2	**ItalyPowerDemand**	95.61 (0.02)	0.95 (0.02)	95.70 (0.02)	0.95 (0.02)	**95.98 (0.02)**	**0.95 (0.02)**
3	**Libras**	58.89 (0.10)	0.60 (0.11)	**63.06 (0.11)**	**0.60 (0.11)**	59.44 (0.09)	0.59 (0.09)
4	**SyntheticControl**	**98.50 (0.01)**	**0.98 (0.01)**	**98.50 (0.01)**	**0.98 (0.01)**	**98.17 (0.02)**	**0.98 (0.02)**
5	**GunPoint**	94.50 (0.05)	0.94 (0.05)	**97.50 (0.03)**	**0.97 (0.03)**	95.50 (0.04)	0.95 (0.04)
6	**OliveOil**	86.67 (0.15)	0.86 (0.16)	86.67 (0.15)	0.86 (0.16)	**88.33 (0.11)**	**0.88 (0.11)**
7	**Trace**	**99.00 (0.03)**	**0.99 (0.03)**	**99.00 (0.03)**	**0.99 (0.03)**	**99.00 (0.03)**	**0.99 (0.03)**
8	**ToeSegmentation2**	88.46 (0.09)	0.88 (0.10)	**92.68 (0.07)**	**0.92 (0.7)**	88.49 (0.06)	0.88 (0.06)
9	**Car**	80.83 (0.07)	0.80 (0.09)	81.67 (0.07)	0.81 (0.08)	**82.50 (0.09)**	**0.82 (0.09)**
10	**Lightning2**	**87.76 (0.09)**	**0.87 (0.08)**	**87.76 (0.09)**	**0.87 (0.08)**	**87.63 (0.07)**	**0.87 (0.07)**
11	**ShapeletSim**	82.00 (0.10)	0.81 (0.11)	82.00 (0.10)	0.81 (0.11)	**89.50 (0.07)**	**0.89 (0.07)**
12	**DiatomSizeReduction**	99.38 (0.01)	0.99 (0.01)	99.69 (0.01)	0.99 (0.01)	**100.00 (0.00)**	**1.00 (0.00)**
13	**Adiac**	65.30 (0.04)	0.62 (0.04)	65.30 (0.04)	0.62 (0.04)	**66.58 (0.03)**	**0.64 (0.03)**
14	**HouseTwenty**	**95.00 (0.05)**	**0.95 (0.05)**	93.00 (0.08)	0.93 (0.08)	93.04 (0.05)	0.93 (0.05)
15	**PenDigits**	85.47 (0.01)	0.85 (0.01)	**87.62 (0.01)**	**0.87 (0.01)**	85.00 (0.01)	0.85 (0.01)

S-C BAND DTW it was conjectured that this was because the FSSBDTW approach reduced the effect of noise (as noted in Chapter 4).

5.4 Candidate Reduction Based on Euclidean Distance vs Candidate Reduction Based on Lower Bounding

A comparison of the performance of the CRBED approach and the CRBLB approach is presented in this section. The collated recorded run times, and accuracy and F1-scores, with respect to each approach, and each data set are given in Tables 5.5 and 5.6 respectively.

Table 5.5 gives the run time results. Figure 5.6 presents the recorded run times in graph form. In the figure, the x-axis records the identification number of the relevant data set, and the y-axis the run time in seconds. From both the table and the figure, it can be seen that the performance of both CRBED and CRBLB, in terms of run time, are very similar; however, it can be argued (from the recorded results) that the CRBED approach is more efficient than the CRBLB approach. This is because of the extra work

undertaken by the CRBLB approach to apply bounding, which is more expensive than a simple Euclidean distance comparison (see the theoretical complexity comparison given at the end of Sub-section 5.3.3). This is especially evident with respect to the *Adiac* and *PenDigits* data sets (numbers 13 and 15), where the number of records is large.

TABLE 5.5: Recorded Run times for the CRBED and CRBLB approaches.

ID.	Data set	CRBED Run time (secs.)	CRBLB Run time (secs.)
1.	SmoothSubspace	**0.92**	**0.92**
2.	ItalyPowerDemand	13.77	**13.20**
3.	Libras	2.64	**2.14**
4.	SyntheticControl	12.10	**6.44**
5.	GunPoint	2.66	**1.26**
6.	OliveOil	1.88	**1.32**
7.	Trace	4.72	**1.97**
8.	ToeSegmentation2	2.32	**1.61**
9.	Car	5.15	**1.76**
10.	Lightning2	3.00	**2.14**
11.	ShapeletSim	14.94	**13.12**
12.	DiatomSizeReduction	**2.98**	6.64
13.	Adiac	**8.40**	19.51
14.	HouseTwenty	19.04	**18.24**
15.	PenDigits	**120.37**	869.43

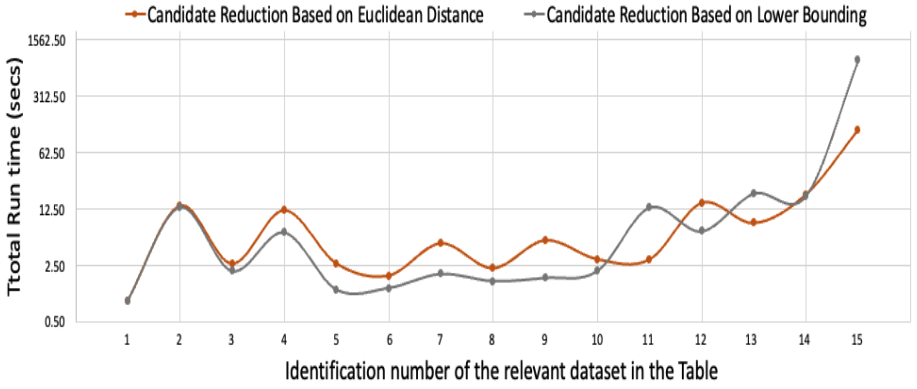

FIGURE 5.6: Comparison of the CRBED approach (brown) and CRBLB approach (grey) recorded run time results for fifteen evaluation data sets.

Table 5.6 gives a comparison of the accuracy values and the F1 scores obtained using the CRBED and the CRBLB approaches (recall that the values and scores were obtained by averaging over the ten folds of the TCV). As before, the figures in parentheses are the standard deviation values obtained. Best results are given in bold font. From the table, it can be seen that the performance using the CRBED approach was better than in the case of the CRBLB approach in most cases. In three cases, the performance was the same. In one case, the *Adiac* data set, the CRBLB approach produced a better accuracy (but not F1 score).

5.5 Conclusion

The chapter has presented two proposed candidate reduction approaches directed at speeding up DTW in the context of 1NN classification: (i) the Candidate Reduction

Table 5.6: Recorded Accuracy and F1-Scores for the CRBED and CRBLB approaches (best results in bold font).

ID #	Dataset Name	CRBED		CRBLB	
		Accuracy (SD)	**F1-Score** (SD)	**Accuracy** (SD)	**F1-Score** (SD)
1	**SmoothSubspace**	**98.33 (0.03)**	**0.98 (0.03)**	96.97 (0.03)	0.96 (0.03)
2	**ItalyPowerDemand**	**97.17 (0.01)**	**0.97 (0.01)**	95.98 (0.02)	0.95 (0.02)
3	**Libras**	**65.83 (0.11)**	**0.65 (0.11)**	59.44 (0.09)	0.59 (0.09)
4	**SyntheticControl**	**98.50 (0.01)**	**0.98 (0.01)**	98.17 (0.02)	0.98 (0.02)
5	**GunPoint**	**99.50 (0.01)**	**0.99 (0.01)**	95.50 (0.04)	0.95 (0.04)
6	**OliveOil**	**90.11 (0.13)**	**0.90 (0.11)**	88.33 (0.11)	0.88 (0.11)
7	**Trace**	**99.00 (0.03)**	**0.99 (0.03)**	**99.00 (0.03)**	**0.99 (0.03)**
8	**ToeSegmentation2**	**92.72 (0.04)**	**0.92 (0.04)**	88.49 (0.06)	0.88 (0.06)
9	**Car**	**88.33 (0.06)**	**0.86 (0.06)**	82.50 (0.09)	0.82 (0.09)
10	**Lightning2**	**87.76 (0.07)**	**0.87 (0.07)**	87.63 (0.07)	0.87 (0.07)
11	**ShapeletSim**	**89.50 (0.06)**	**0.89 (0.08)**	**89.50 (0.07)**	**0.89 (0.07)**
12	**DiatomSizeReduction**	**100.00 (0.00)**	**1.00 (0.00)**	**100.00 (0.00)**	**1.00 (0.00)**
13	**Adiac**	65.81 (0.03)	**0.64 (0.03)**	**66.58 (0.03)**	**0.64 (0.03)**
14	**HouseTwenty**	**93.71 (0.04)**	**0.93 (0.04)**	93.04 (0.05)	0.93 (0.05)
15	**PenDigits**	**89.64 (0.01)**	**0.89 (0.01)**	85.00 (0.01)	0.85 (0.01)

Based on Euclidean Distance (CRBED) approach and (ii) the Candidate Reduction Based on Lower Bounding (CRBLB) approach. The fundamental idea was that these approaches work as filtering techniques so that the number of DTW processes that needed to be applied would be reduced. DTW needs only be applied to the retained candidates. The work described assumed that any required DTW would be conducted using the Fuzzy Sub-Sequence-Based DTW (FSSBDTW) approach from the previous chapter, although the proposed approaches could easily be adapted for use with other DTW variations. The reported evaluation demonstrated that by applying candidate reduction significant efficiency gains could be made. In addition, it was found that accuracy gains were also made over Standard DTW and S-C Band DTW. The distinction between CRBED and CRBLB was that the first adopted a Euclidean distance based approach to filter the candidates; while CRBLB used a LB_Keogh lower bounding method. Comparison of CRBED and CRBLB indicated that CRBED was more accurate. In addition, although the recorded runtimes were very similar in most cases, it could also be argued that CRBED showed a better run time performance when it comes to data

sets that featured large numbers of records. The following Chapter introduces two further approaches directed at reshaping time series: (i) Exact Discriminator-Based DTW approach and (ii) Distance Profile-Based DTW approach.

Chapter 6

Pruning Approaches

6.1 Introduction

In the previous chapter, two mechanisms for reducing the computational complexity of DTW, assuming a kNN classification scenario, were considered using candidate reduction: (i) Candidate Reduction Based on Euclidean Distance (CRBED) and (ii) Candidate Reduction Based on Lower Bounding (CRBLB). The fundamental idea, given a kNN bank of labelled time series D with r records, and a previously unseen time series t, was to reduce the number of records in D to r in such a way that only the most likely matches for t were retained. This chapter explores an alternative whereby the DTW computational complexity can be reduced, given a kNN classification scenario, by reducing (pruning) the number of points in the time series to be considered from x to x', as opposed to reducing the number of time series in D. By reducing the length of the time series from x to x' the size of the DTW distance matrix is reduced as shown in Figure 6.1.

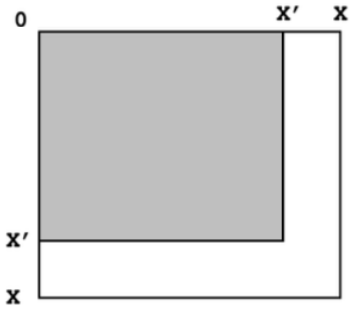

FIGURE 6.1: Reduction in size of DTW distance matrix by reducing the time series length from x to x'

The first idea considered was to identify specific sub-sequences of indexes (locations) within the time series within D that would best serve to discriminate between classes, one sub-sequence (location) per class. Once the sequences have been identified, D can be "reformatted" to form D' comprised of only the identified sequences. We refer to such sub-sequences as *discriminators*. The second idea considered is to translate D and t into a "distance profile" format; what a distance profile is will become clear later in this chapter. Both are used to reformat D as a consequence of which each time series in D is significantly shortened, and therefore requires less processing. Note that given a new time series t this also needs to be reformatted to give t', DTW is then applied to the reduced data set D to obtain a label for t' (t).

The idea of discriminators has similarities with the concept of motifs [5, 56, 102, 116]. A motif is a time series sub-sequence that occurs frequently in a data set D and, because it occurs frequently, is deemed to be a good indicator of class [124]. The opposite is a discord [20, 101]. A discord is a time series sub-sequence that occurs infrequently in a data set D and, because it occurs infrequently, is deemed to be a good indicator of class [101]. Note the two schools of thought; a motif is the opposite of a discord. The distinction between a discriminator as envisioned in this chapter, and motifs and discords, is that the location of the latter can be anywhere within a collection of time series D, while discriminators always occur at the same locations within D. To put it another way, a discriminator is a time series sub-sequence that is a good indicator of class that occurs at a constant location within a set of time series. The challenge is how these discriminators (locations) should be identified. We can conceive of two categories of approach, (i) exact approaches and (ii) approximate approaches. The distinction between the two is that the first entails an exhaustive search. Only the first is considered here, the Exact Discriminator-Based DTW (EDBDTW) approach. Approximate approaches are left as a consideration for further work (see Chapter 7). The distance profile idea is based on work using matrix profiles to identify motifs and discords [4, 127, 128]. The proposed distance profile approach is referred to as the Distance Profile-Based DTW (DPBDTW) approach [9].

Earlier in this thesis, Chapters 3 and 4, four DTW approaches were considered. Two benchmark approaches, Standard DTW and S-C Band DTW, and two approaches based on segmenting time series, the Sub-Sequence-Based DTW (SSBDTW) approach and the Fuzzy Sub-Sequence-Based DTW (FSSBDTW). Of these, FSSBDTW was found to be both the most efficient and the most effective. Hence, with respect to the work presented in this chapter FSSBDTW was used to obtain classifications, although any other time series comparison method could be applied. Thus in the remainder of this chapter reference to either the EDBDTW or DPBDTW approach should be interpreted as entailing FSSBDTW in the context of kNN classification where $k = 1$.

Given the foregoing, the work presented in this chapter was directed at providing answers to the subsidiary research questions four and five given in Chapter 1:

Is it possible to only consider sub-sequences that are repeated in instances (records) of the same class, so that DTW can be applied only to such sub-sequences instead of the entire time series?

and

Is it possible to identify mechanisms for transforming time series from their original form and apply DTW to this new form?

The rest of this chapter is organised as follows. Sections 6.2 and 6.3 present the proposed EDBDTW and DPBDTW approaches respectively. Both sections are structured in a similar manner. Sections 6.4 then provides a comparison of the operation of the proposed EDBDTW and DPBDTW approaches. Note that comparison with the approaches presented in Chapters 4 and 5 is left till Chapter 7. The chapter is concluded in Section 6.5, with a summary of the main findings.

6.2 Exact Discriminator-Based DTW approach

This section presents the first of the two time series pruning mechanisms considered in this chapter, exact discriminator identification. As noted above the exact mechanism

is an exhaustive mechanism that assumes a fixed discriminator length δ, and typically one discriminator per time series. Note that $\delta < x$, where x is the length of a time series in the database D (as before, given a particular application, it is assumed that all time series are of the same length). The fundamental idea is that a discriminator will be defined by the location, within a set of time series associated with a class, where the time series are most similar. This location is then where DTW can best be applied to obtain a good classification. The exact discriminator identification mechanism was built into the Exact Discriminator-Based DTW (EDBDTW) approach. The approach is fully described and evaluated in this section.

The section is organised in a similar manner to the previous sections describing approaches to enhancing the efficiency of DTW presented earlier in this thesis. Sub-section 6.2.1 describes the operation of the EDBDTW approach. For further clarification, Sub-section 6.2.2 presents a worked example. The pseudo code for the EDBDTW approach is presented in Sub-section 6.2.3. Sub-section 6.2.4 then discusses the theoretical computational complexity of the proposed approach using "Big O" notation. This is then followed by Sub-section 6.2.5 which presents a practical evaluation of the approach, which concludes the section.

6.2.1 Operation of Exact Discriminator-Based DTW Approach

In this section, the operation of the proposed Exact Discriminator-Based DTW (EDBDTW) approach is presented. A block diagram describing the approach is given in Figure 6.2. The process comprises four stages.

Stage 1 Parameter learning to find the best parameters (the loop at the top of Figure 6.2).

Stage 2 Discriminator location using identified est parameters.

Stage 3 Data set D transformation to give D'.

Stage 4 Classification of a given query t time series with respect to D' through the application of FSSBDTW.

Note that stages 1, 2 and 3 are only undertaken once, unless more training data becomes available. Stage 4 can then be applied as required.

During Stage 1, the values for the following parameters are learnt: (i) δ, the length of a discriminator expressed as a percentage of x, the length of a time series in D (as before, it is assumed that all time series in D are of the same length), (ii) $maxLength$, the maximum length of a FSSBDTW segment and (iii) t, the length of the tail used with respect to FSSBDTW. As before, during Stage 1, a training and test set are used with Euclidean distance time series comparison (because this is cheaper than DTW). A $9:1$ training and test set split was used with respect to the evaluation presented later in this chapter. In Stage 2, the learnt parameter δ is used to identify the discriminator locations, which are then used to reformat D into D' in Stage 3. Only one discriminator is typically learnt for each class. Thus if we have three classes we anticipate that we will have three discriminators as illustrated in Figure 6.3. In the figure, the identified discriminators are surrounded by bounds (stadia). The discriminators, for each class, are identified by comparing all time series of length δ at each location (index) in D to every other time series of length δ at each other location, and choosing the location where the time series are most similar. Stage 4 is where the desired classification takes place.

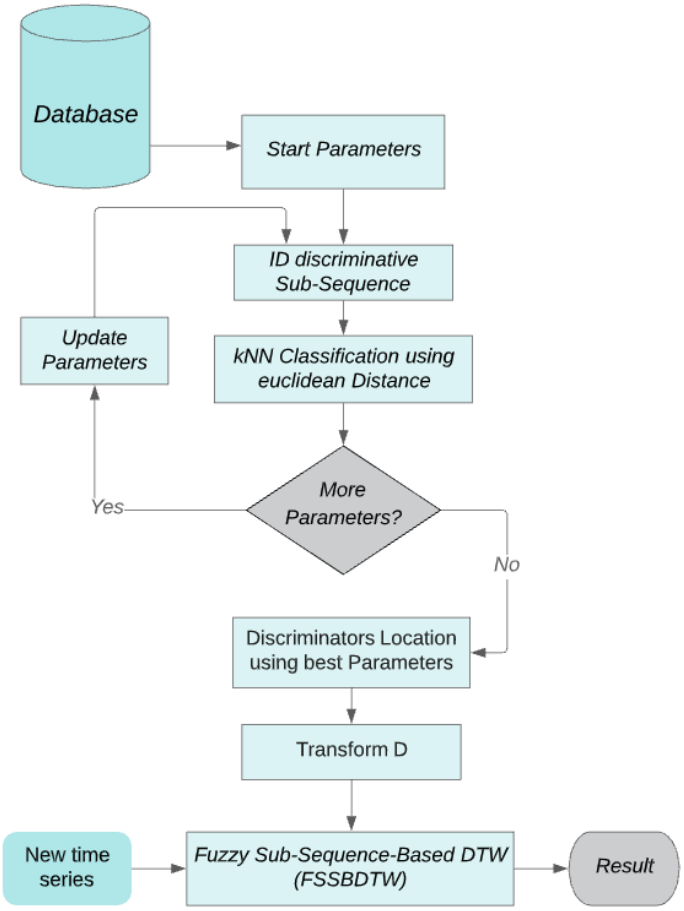

FIGURE 6.2: Schematic outlining the operation of the EDBDTW approach

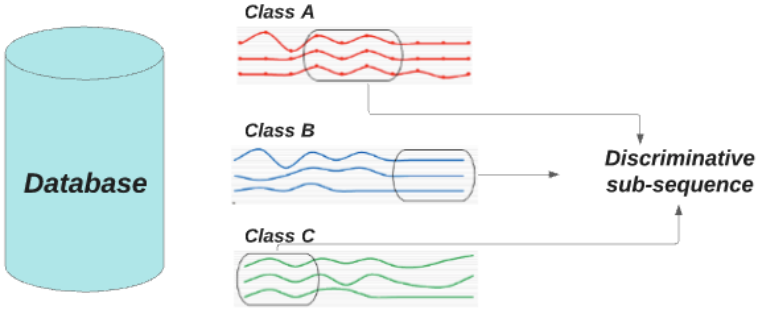

FIGURE 6.3: Example discriminators

6.2.2 Exact Discriminator-Based DTW Approach Worked Example

In this section, a worked example of the process of exact discriminator extraction is presented. Assuming the set of classes $C = \{blue, red\}$, a discriminator length $\delta = 4$, and the set $D = \{\langle S_1, c_1\rangle, \ldots, \langle S_6, c_6\rangle\}$ as follows:

i	S_i	c_i
1	$[1,3,2,1,3,2,1,3,1,2]$	red
2	$[2,1,1,1,2,2,2,1,3,1]$	red
3	$[4,5,5,6,5,3,4,5,5,6]$	blue
4	$[3,3,4,5,5,4,4,5,4,5]$	blue
5	$[3,3,1,2,2,3,2,3,2,3]$	red
6	$[5,5,3,3,5,4,3,5,3,3]$	blue

We commence by dividing the input set D into a set of sets $\mathbf{A}$ so that the time series associated with each class are grouped. In this example, we will get $\mathbf{A} = \{A_{blue}, A_{red}\}$ such that A_{blue} will be defined as follows:

i	S_i	c_i
1	$[4,5,5,6,5,3,4,5,5,6]$	blue
2	$[3,3,4,5,5,4,4,5,4,5]$	blue
3	$[5,5,3,3,5,4,3,5,3,3]$	blue

and A_{red} as follows:

i	S_i	c_i
1	$[1,3,2,1,3,2,1,3,1,2]$	red
2	$[2,1,1,1,2,2,2,1,3,1]$	red
5	$[3,3,1,2,2,3,2,3,2,3]$	red

We then process $\mathbf{A}$ starting with A_{blue}. For all time series in A_{blue} up to, but not including the last time series, we accumulate the absolute distances between points and store them in a set B. Thus for the first two time series in A_{blue} the distances will be.

i	$abs(S_1 - S_2);\ S_1, S_2 \in A_{blue}$
1	$[4,5,5,6,5,3,4,5,5,6]$
2	$[3,3,4,5,5,4,4,5,4,5]$
distance 1-2	$[1,2,1,1,0,1,0,0,1,1]$

and for the next two:

i	$abs(S_2 - S_3);\ S_2, S_3 \in A_{blue}$
2	$[3,3,4,5,5,4,4,5,4,5]$
3	$[5,5,3,3,5,4,3,5,3,3]$
distance 2-3	$[2,2,1,2,0,0,1,0,1,2]$

Adding these two together we get $B = \{3,4,2,3,0,1,1,0,2,3\}$. Note that there is a one-to-one correspondence between the index locations in B and the time series in A_{blue}. Given $\delta = 4$ we can identify seven sub-sequences in B:

i	w_i
1	$3,4,2,3$
2	$4,2,3,0$
3	$2,3,0,1$
4	$3,0,1,1$
5	$0,1,1,0$
6	$1,1,0,2$
7	$1,0,2,3$

Thus $W = \{w_1, w_2, w_3, w_4, w_5, w_6, w_7\}$. We then define and populate a set $E = \{dist_1, dist_2, dist_3, dist_4, dist_5, dist_6, dist_7\}$ with a one-to-one correspondence with the content of W. Each $dist_i$ in E holds the sum of the values in w_i. Thus $E = \{12, 9, 6, 5, 2, 4, 6\}$. The lowest distance in E is 2, which corresponds to w_5 which in turn corresponds to indexes $[5, 6, 7, 8]$ (assuming we start counting from 1). Thus for the class A_{blue} the differentiator sub-sequence is at indexes $[5, \ldots, 8]$.

Considering the class A_{red}, $B = \{2, 4, 1, 1, 1, 1, 1, 4, 3, 3\}$ and thus we have the sub-sequences:

i	w_i
1	2, 4, 1, 1
2	4, 1, 1, 1
3	1, 1, 1, 1
4	1, 1, 1, 1
5	1, 1, 1, 4
6	1, 1, 4, 3
7	1, 4, 3, 3

We therefore have $E = \{8, 7, 4, 4, 7, 9, 11\}$. In this case, the lowest value is 4, which equates to w_3 and w_4, thus we have two discriminators at indexes $[3, \ldots, 6]$ and $[4, \ldots, 7]$. Note that in practice it is unusual to have two or more discriminators for a class.

When we have a new time series $\mathfrak{t}$, we find the similarity by only comparing the differentiators in $\mathfrak{t}$ and D. So, no need to compare the whole of $\mathfrak{t}$ with the whole of the time series in D.

6.2.3 Exact Discriminator-Based DTW Approach Algorithm

This section presents the pseudo code for Stages 1, 2, 3 and 4 of the proposed EDBDTW approach starting with Stage 1, parameter learning. The pseudo code for Stage 1 is given in Algorithm 13. The inputs, line 1, are: (i) a collection of pre-labelled time series, D_{train}, which will form the initial kNN bank, (ii) a second collection of pre-labelled time series, D_{test}, to form a test set, (iii) the size, $|C|$, of the set of classes C, (iv) a set I of possible values for the FSSBDTW *maxlength* parameter, (v) a set T of possible values for the FSSBDTW t tail parameter and (vi) a set Δ of the possible values for the discriminator length parameter δ. The algorithm loops through the different parameter combinations. On each iteration, a set DIS describing the identified discriminator locations is generated (line 7). Each discriminator is defined by a start and an end index, thus $DIS = \{\langle start_1, end_1\rangle, \langle start_2, end_2\rangle, \ldots\}$ where $\langle start_1, end_1\rangle$ is the discriminator associated with class $c_1 \in C$,and $\langle start_2, end_2\rangle$ is the discriminator associated with $c_2 \in C$, and so on. The process for doing this will be presented later in this section (Algorithm 14). The set DIS is then used to translate the input sets D_{train} and D_{test} into the sets D'_{train} and D'_{test} (lines 8 and 9). On line 10, the function $kNN\_Euclidean$ is called with D'_{train}, D'_{test} and the selected values for the parameters for $maxLen$ and t. The function returns the classification accuracy, the number of true positives and true negatives. This is then used to update the *bestAccuracy* variable and the best parameter list. Once the entire search space has been processed the best parameters will have been identified. Stage 1 is then complete and the identified parameters are returned (line 18). Initial experiments indicated that the following definitions for I, T and Δ were the most appropriate: $I = \{10, 20, \ldots, 90, 100\}$, $T = \{1, 2, \ldots, 9, 10\}$ and $\Delta = \{5\%, 10\%, \ldots, 90\%, 95\%\}$.

Algorithm 13 EDBDTW Parameter Learning, Stage 1

1: **input** $D_{train}, D_{test}, |C|, I, T, \Delta$.
2: $bestParameters = [0, 0, 0]$
3: $bestAccuarcy = 0$
4: **for** $\forall maxLen_i \in I$ **do**
5: **for** $\forall t_i \in T$ **do**
6: **for** $\forall \delta_i \in \Delta$ **do**
7: $DIS = exactDiscriminatorrID(D_{train}, |C|, \delta_i)$ (Algorithm 14)
8: $D'_{train} = D_{train}$ transformed according to DIS
9: $D'_{test} = D_{test}$ transformed according to DIS
10: $newAccuracy =$ kNN_Euclidean(D'_{train}, D'_{test}, $maxLen_i$, t_i)
11: **if** $newAccuracy > bestAccuracy$ **then**
12: $bestAcuuracy = newAccuracy$
13: $bestParameters = [maxLen_i, t_i, \delta_i]$
14: **end if**
15: **end for**
16: **end for**
17: **end for**
18: **return** $bestParameters$

Stage 2, discriminator identification, is where the discriminators are identified. Note that discriminator locations were identified during the Stage 1 parameter tuning, but not using the entire data set, hence the process is repeated in Stage 2. Recall also that the discriminators are defined by a tuple of the form $\langle start_i, end_i \rangle$, where $start_1$ and end_i are the start and end indexes for the discriminator. The pseudo code for the discriminator identification is given in Algorithm 14. The inputs (line 1) are: (i) the data set $D = \{\langle S_1, c_i \rangle, \langle S_2, c_2 \rangle, \ldots \langle S_r, c_r \rangle\}$, (ii) the number of classes in C and (iii) the desired discriminator length δ as discovered during Stage 1. The first step (line 2) is to declare an empty set DIS in which to hold the identified discriminator locations, which is to be populated as the process progresses. Next (lines 3 to 6), the time series in D are grouped according to their associated class and placed in a set of sets $\mathbf{A} = \{A_1, A_2, \ldots A_{|C|}\}$, where set A_i holds the collection of time series associated with class c_i. This set of sets is then processed, lines 7 to 22, so as to generate DIS. For each set of time series $A_i \in \mathbf{A}$, associated with a particular class c_i, a temporary array B of length r is generated (lines 8 to 13), which holds the accumulated distances for each index in the time series in A_i. Thus, the accumulated distances between the time series for index 1 (time point 1), index 2, and so on, up to index x (the assumption is that the input time series are all of the same length). The array B is then, line 14, divided into a set of overlapping sub-sequences, $W = \{w_1, w_2, \ldots, w_s\}$, each of length δ. A temporary array E is then created, lines 15 to 18, to hold accumulated distances (sums of distances) held in each sub-sequence $w_j \in W$. Note that there is a one-to-one correspondence between W and E. The start index associated with the distance sequence $w_j \in W$ that has the lowest accumulated distance $dist_j \in E$ is then selected as the start index of the location for the discriminator for the time series in A_i (line 19). It is then a simple step to find the end index by adding δ to $start$ (line 20). At the end of the process, the set DIS will be fully populated with a set of discriminators. To obtain a more comprehensive understanding of the operation of Stage 2 the reader might find it useful to return to the worked example given in the precious sub-section.

Stage three involves transforming the set D into a set of discriminators D' according

Algorithm 14 Exact Discriminator Identification (Stage 2)

1: **input** $D, |C|, \delta$
2: $DIS = \emptyset$
3: $\mathbf{A}$ = Temporary set of sets length $|C|$ to hold sets of time series
4: **for** $\forall \langle S_j, c_j \rangle \in D$ **do** ▷ Populate A
5: $A_i = A_i \cup \langle S_j, c_j \rangle$ $(i = j)$
6: **end for**
7: **for** $\forall A_i \in \mathbf{A}$ **do** ▷ Populate B
8: B = Temporary array $[dist_1, dist_2, \ldots, dist_r]$, $dist_i = 0$
9: **for** $S_j \in A_i$, $j = 0$ to $j = r - 1$ **do**
10: **for** $\forall p_x \in S_j$ and $\forall q_x \in S_{j+1}$ **do**
11: $dist_x \in B = dist_x + abs(p_x - q_x)$
12: **end for**
13: **end for**
14: $W = [w_1, w_2, \ldots, w_s]$, array of sub-sequences in B, each of length $\bar{l}$
15: E = Temporary array $[dist_1, dist_2, \ldots, dist_s]$
16: **for** $\forall w_j \in W$ **do** ▷ Populate E
17: $dist_j \in E = \sum_{i=0}^{i=\bar{l}} p_i \in w_j$
18: **end for**
19: $start$ = start index of $w_j \in W$ with lowest distance value $dist_j \in E$.
20: $end = start + \delta$
21: $DIS = DIS \cup \langle start, end \rangle$
22: **end for**
23: **return** DIS

Algorithm 15 EDBDTW Parent Code (Stages 1, 2 and 3)

1: **input** $D_{train}, D_{test}, |C|, I, T, \Delta$.
2: $[maxLength, t, \delta] = paramLearn(D_{train}, D_{test}, |C|, I, T, \Delta)$ ▷ Stage 1 Algorithm 13
3: $DIS = exactDiscriminatorID(D_{train}, |C|, \delta)$ ▷ Stage 2 Algorithm 14
4: $D' = D_{train} \cup D_{test}$ transformed using DIS ▷ Stage 3
5: **return** $D', DIS, maxLength, t$

to the content of DIS. Note that $D = D_{train} \cup D_{test}$. This is a fairly straightforward process. Once Stage 3 is complete, classification of unseen time series can commence (Stage 4). The pseudo code given in Algorithm 15 outlines the "parent" code for stages 1, 2 and 3. The input is the same as for Algorithm 13. The output is D'. The parent algorithm also returns: (i) the set DIS because this will be required to transform any previously unseen time series to be labeled, and (ii) the parameters $maxLength$ and t required by FSSBDTW.

The final stage of the proposed EDBDTW approach, Stage 4, is the classification of previously unseen time series. This final stage is also relatively straight forward when compared to Stages 1 and 2. The pseudo code is given in Algorithm 16. The input (line 1) is the kNN bank D', the set of discriminator locations DIS, the values for the parameters $maxLength$ and t used by FSSBDTW, and the time series to be labelled $\mathfrak{t}$. The first step, line 2, is to transform $\mathfrak{t}$ into $\mathfrak{t}'$ according to the content of DIS. Then, lines 4 to 11 we determine the best match in D' with $\mathfrak{t}'$ and retain the associated class c, which is returned in line 12. Note that the matching is done using the best warping distance ($bestWD$) obtained using the FSSBDTW algorithm presented previously in Chapter 4.

Algorithm 16 EDBDTW Classification (Stage 4)

```
1: input D', DIS, maxLength, t, ţ.
2: ţ' = ţ transformed using DIS
3: bestWD = ∞
4: c = default class
5: for ∀S_i ∈ D' do
6:     wd = FSSDDTW(S_i, ţ', maxLen, t)
7:     if wd < bestWD then
8:         bestWD = wd
9:         c = class associated with S_i ∈ D'
10:    end if
11: end for
12: return c
```

6.2.4 Theoretical Time Complexity of Exact Discriminator-Based DTW Approach

In this section, the theoretical time complexity of the EDBDTW approach is presented, assuming TCV and 1NN classification, and Big O ("in the Order of") notation. Each of the four stages will first be considered independently and then an overall complexity equation derived. Commencing with Stage 1, parameter tuning, and assuming a 9:1 training-test set split, the complexity is similar to Stage 1 of the CRBED approach described in Chapter 5 (Equation 5.6):

$$\begin{aligned} stage1_{complexity} = \; & numberParameterCombinations \times numberOfComparisons \\ & \times EuclideanComparison_{complexity} \end{aligned} \tag{6.1}$$

where:

$$numberParameterCombinations = |I| \times |T| \times |\Delta| \tag{6.2}$$

$$numberOfComparisons = \frac{|D|}{10} \times \frac{9 \times |D|}{10} = \frac{9 \times |D|^2}{100} \tag{6.3}$$

$$EuclideanComparison_{complexity} = O\left(\bar{\delta}\right) \tag{6.4}$$

Note that for determining the Euclidean distance comparison complexity the average value of δ is used ($\bar{\delta}$). The Stage 1 complexity equation thus becomes:

$$stage1_{complexity} = O\left(|I| \times |T| \times |\Delta| \times \frac{9 \times |D|^2}{100} \times \bar{\delta}\right) \tag{6.5}$$

Recall that the sets I, T and Δ were pre-specified, and that $|I| = 10$, $|T| = 10$ and $|\Delta| = 17$. The above thus becomes:

$$stage1_{complexity} = O\left(10 \times 10 \times 17 \times \frac{9 \times |D|^2}{100} \times \bar{\delta}\right) \tag{6.6}$$

which simplifies to:

$$stage1_{complexity} = O\left(153 \times |D|^2 \times \bar{\delta}\right) \tag{6.7}$$

When using TCV, the parameter tuning will be conducted 10 times. Hence the above will become:

$$\begin{aligned} stage1TCV_{complexity} &= O\left(10 \times 153 \times |D|^2 \times \bar{\delta}\right) \\ &= O\left(1530 \times |D|^2 \times \bar{\delta}\right) \end{aligned} \tag{6.8}$$

The complexity of Stage 2, identification of discriminator locations (each defined by a start and end index), depends on the number of candidates to be considered and the number of records in D. The number of candidates, given a single time series $S_i \in D$, will be equivalent to $x - \delta + 1$. However, the value of δ will be learnt from a set of values Δ, thus using a mean value of δ will again be more appropriate. The complexity of Stage 3 will thus be given by:

$$\begin{aligned} stage2_{complexity} &= O\left(|D| \times \left(x - \bar{\delta} + 1\right)\right) \\ &= O\left(|D| \times \left(x - \bar{\delta} + 1\right)\right) \end{aligned} \tag{6.9}$$

When using TCV this will be conducted 10 times for both D_{train} and D_{test} ($D = D_{train} \cup D_{test}$):

$$stage2TCV_{complexity} = O\left(10 \times |D| \times \left(x - \bar{\delta} + 1\right)\right) \tag{6.10}$$

The complexity of Stage 3, where D is transformed into D' is simply given by:

$$stage3_{complexity} = O\left(|D|\right) \tag{6.11}$$

When using TCV this will be conducted 10 times:

$$stage3TCV_{complexity} = O\left(10 \times |D|\right) \tag{6.12}$$

The determination of the class of a query time series $\mathfrak{t}$ will then require $|D'|$ comparisons. The complexity for class determination, assuming the FSSBDTW approach, will therefore be:

$$stage4_{complexity} = O\left(|D'| \times DTW_{complexityFSSBDTW}\right) \tag{6.13}$$

The complexity of FSSBDTW was previously given in Equation 4.11 in Chapter 4 as:

$$DTW_{complexityFSSBDTW} = O\left(x \times \overline{len}\right) \tag{6.14}$$

where x was the length of the time series to be labelled and $\overline{len}$ was the average length of a segment. However, in the case of the EDBDTW approach, the length of the time series has been reduced. If we assume that the identified discriminators are non-overlapping and that we have one discriminator per class, not necessarily the case as illustrated in the worked example, then the number of points in each time series in D' (the length of the time series) will be $|C| \times \bar{\delta}$. The classification complexity thus becomes:

$$stage4_{complexity} = O\left(|D'| \times |C| \times \bar{\delta} \times \overline{len}\right) \tag{6.15}$$

If we are undertaking TCV, the number of classifications will equate to $|D|$. Thus:

$$stage4TCV_{complexity} = O\left(|D| \times |D'| \times |C| \times \bar{\delta} \times \overline{len}\right) \tag{6.16}$$

Note that in this case $|D'|$ is equivalent to $|D|$. Thus:

$$stage4TCV_{complexity} = O\left(|D|^2 \times |C| \times \bar{\delta} \times \overline{len}\right) \tag{6.17}$$

The overall complexity of the EDBDTW approach, assuming TCV, will then be:

$$\begin{aligned} EDBDTW\_TCV_{complexity} = stage1TCV_{complexity} + stage2TCV_{complexity} + \\ stage3TCV_{complexity} + stage4TCV_{complexity} \end{aligned} \tag{6.18}$$

In other words:

$$\begin{aligned} EDBDTW\_TCV_{complexity} = \; & O(\left(1530 \times |D|^2 \times \bar{\delta}\right) + \left(10 \times |D| \times \left(x - \bar{\delta} + 1\right)\right) + \\ & (10 \times |D|) + \left(|D|^2 \times |C| \times \bar{\delta} \times \overline{len}\right)) \\ = \; & O(|D| \times \left(1530 \times |D| \times \bar{\delta}\right) + \left(10 \times \left(x - \bar{\delta} + 1\right)\right) + \\ & 10 + \left(|D| \times |C| \times \bar{\delta} \times \overline{len}\right)) \end{aligned} \tag{6.19}$$

6.2.5 Evaluation of Exact Discriminator-Based DTW Approach

The evaluation of the proposed EDBDTW approach is presented in this sub-section. The evaluation was conducted using 1NN classification and the fifteen selected data sets from the UEA and UCR Time Series Classification repository [14] in the same manner as reported on in Chapters 3, 4 and 5. The objectives of the evaluation were:

1. **Operational Analysis**: To analyse the operation of the proposed EDBDTW approach.
2. **Efficiency Comparison**: To evaluate the run-time advantages gained using the EDBDTW approach compared to the Standard DTW and the S-C Band DTW benchmark approaches.
3. **Effectiveness Comparison**: To determine whether the classification accuracy of the proposed EDBDTW approach was commensurate with that obtained using standard DTW and S-C Band DTW.

For the S-C Band approach a warping window of $\ell = 10\%$ was again used as proposed in [81, 92]. The experimental set up was the same as that used with respect to the evaluations reported on in earlier chapters. A desktop computer was used with an Apple M1 8 core CPU processor, an 8 core GPU, a 16 core Neural Engine, 16GB unified memory, and 512GB Solid State Drive (SSD).

Operational Analysis

The performance, in terms of run time, accuracy and the F1 measure, for the proposed EDBDTW approach, is considered here together with the nature of the derived values for $maxLen$, t and δ. Table 6.1 gives the results obtained. Columns 3, 4 and 5 give the $maxLen$, t and δ values in each case. Column 6 gives the run time in seconds. Columns 7 and 8 give the accuracy values and F1 scores obtained. As in the case of the previous

TABLE 6.1: Best values for parameters *maxLen*, t and δ in terms of run time, accuracy and F1 score for each data set.

ID #	**Data set**	*maxLen*	**Tail** t	δ	**Run time** (Secs)	**Acc** (SD)	**F1** (SD)
1	**SmoothSubspace**	5	2	95%	3.04	95.00 (0.02)	0.95 (0.04)
2	**ItalyPowerDemand**	6	3	55%	15.87	96.35 (0.02)	0.96 (0.02)
3	**Libras**	10	3	60%	3.47	64.17 (0.12)	0.61 (0.12)
4	**SyntheticControl**	30	2	90%	10.68	95.17 (0.03)	0.95 (0.03)
5	**GunPoint**	40	2	45%	1.63	98.00 (0.02)	0.98 (0.02)
6	**OliveOil**	40	9	95%	2.18	90.00 (0.11)	0.90 (0.11)
7	**Trace**	70	2	80%	5.88	99.00 (0.02)	0.99 (0.02)
8	**ToeSegmentation2**	30	3	75%	6.05	92.21 (0.04)	0.92 (0.04)
9	**Car**	60	6	85%	6.19	83.33 (0.06)	0.83 (0.06)
10	**Lighting2**	80	2	90%	8.25	89.17 (0.08)	0.89 (0.08)
11	**ShapeletSim**	40	1	85%	14.64	87.00 (0.07)	0.87 (0.07)
12	**DiatomSizeReduction**	20	2	20%	4.81	100.00 (0.00)	1.00 (0.00)
13	**Adiac**	10	2	100%	88.20	65.81 (0.03)	0.62 (0.04)
14	**HouseTwenty**	300	5	45%	53.32	96.25 (0.05)	0.96 (0.05)
15	**PenDigits**	5	2	100%	480.17	89.19 (0.01)	0.89 (0.01)

evaluation results presented in this thesis, the numbers in parentheses give standard deviations. From the table, and as experienced with respect to the parameter settings associated with the earlier approaches presented in this thesis, there is no definitive value for *maxLen*, t or δ. This supports the intuition of including a parameter tuning stage in the EDBDTW approach. Although not shown in the table, as was anticipated, the results obtained indicated that run time decreased as δ decreased, whilst accuracy was only affected in a marginal manner.

From Table 6.1 it is also interesting to note that in many cases the best value for δ is high. In 10 of the 15 cases, the best identified value for δ was greater than 70%. In other words, no significant reduction of the time series length was being made. Given several classes, it might be the case that the entire data set is still being examined! In two cases, *Adiac* and *PenDSigits* the best value for δ was found to be 100%; in other words, no saving was being made. In only one case, *DiatomSizeReduction*, where the best value for δ was found to be 20%, was any significant saving being made in terms of time series length. It is also worth noting, by comparing back to Table 4.5 in Chapter 4, that different best values for *maxLength* and t were obtained using the EDBDTW approach compared to using FSSBDTW in isolation. This, it was conjectured, was because if the interplay between the parameters *maxLength* and t, and δ, in the case of the EDBDTW approach. It also might be conjectured that there was a certain degree of "volatility" with respect to the parameters *maxLength*, t and δ.

Efficiency Comparison

Figure 6.4 gives a comparison of the run time performance of the Standard DTW, S-C Band DTW and EDBDTW Approaches. The x-axis represents the data set identification numbers as listed in Table 6.1, and the y-axis represents the run time in seconds. The values for $maxLen$, t and δ that were used were the best performing values as reported in Table 6.1. From the figure, it can be seen that for all the data sets considered, except the *Adiac* and *PenDigits* data sets, the proposed EDBDTW approach produced the best run times despite the effort directed at parameter tuning, discriminator location identification and data set transformation. In the case of the *PenDigits* data set, a very similar run time was recorded. Referring back to Table 6.1 the reason for the poor result with respect to the *Adiac* and *PenDigits* data sets was that the best value for δ of 100% was used. Hence, effort was directed at finding best parameters, identifying the location of the discriminator and transforming D into D', with no advantage gained because the entire time series were still being processed. In addition, referring back to the discussion presented in the previous sub-section, in many cases, the value for δ was high and consequently, the length of the time series were not significantly reduced. Hence an argument can be made that a significant proportion of the efficiency gains obtained using the EDBDTW approach, compared to Standard DTW and S-C Band DTW, were as a result of using FSSBDTW rather than the discriminator concept. However, this argument should be tempered with the observation that some of the data sets considered featured time series that were relatively short (see Table 3.1 presented in Chapter 3). For example, the time series in the *PenDigits* data set only featured 8 points, hence it should come as no surprise that the best value for δ was found to be 100%.

In Chapters 3 and 4, the recorded run times were used to determine and test an actual runtime equation founded on the theoretical complexity equation. This was done by determining the value of a constant z, the approximate run time for a single application of DTW. However, in the case of the EDBDTW approach we have both Euclidean distance and DTW comparisons, thus two constants z_1 and z_2. Furthermore, we have the four stages of the proposed EDBDTW approach to consider. Thus, determining the proportion of the recorded run times to be allocated to the calculation of z_1 and z_2 is not straight forward. Hence, unlike in Chapters 3 and 4, an actual run time equation is not presented here.

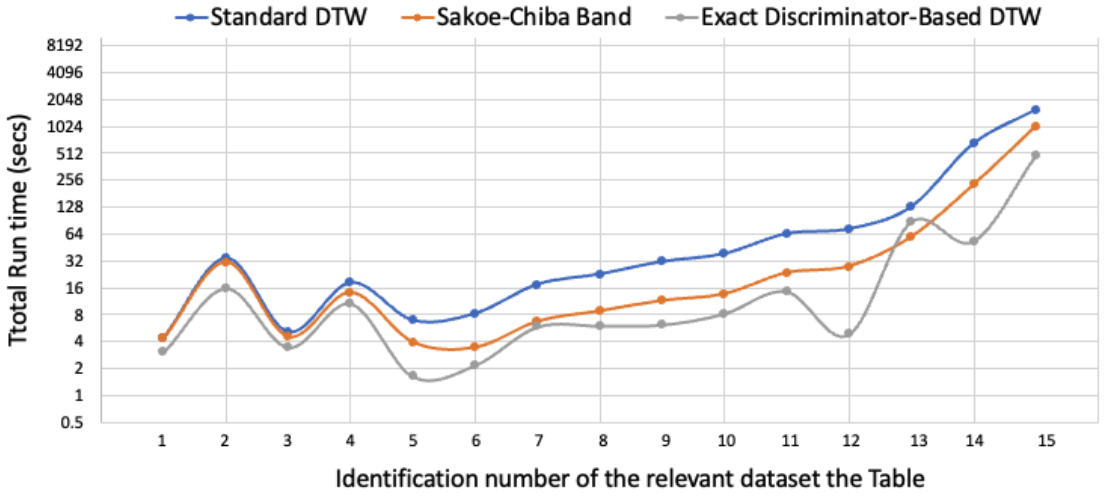

FIGURE 6.4: Run time results using the Standard DTW and S-C Band DTW benchmark approaches and the proposed EDBDTW Approach.

TABLE 6.2: Accuracy and F1-Score, with standard deviation, for fifteen time series data sets using Standard DTW, S-C Band DTW and EDBDTW.

ID	**Data set**	**Standard DTW**		**S-C Band $\ell = 10$**		**EDBDTW**	
#	**Name**	**Acc** (SD)	**F1** (SD)	**Acc** (SD)	**F1** (SD)	**Acc** (SD)	**F1** (SD)
1	**SmoothSubspace**	91.00 (0.04)	0.91 (0.04)	**95.00** (**0.02**)	**0.95** (**0.02**)	**95.00** **(0.02)**	**0.95** **(0.02)**
2	**ItalyPowerDemand**	95.61 (0.02)	0.95 (0.02)	95.70 (0.02)	0.95 (0.02)	**96.35** **(0.02)**	**0.96** **(0.02)**
3	**Libras**	58.89 (0.10)	0.60 (0.11)	63.06 (0.11)	0.60 (0.11)	**64.17** **(0.12)**	**0.61** **(0.12)**
4	**SyntheticControl**	**98.50** **(0.01)**	**0.98** **(0.01)**	**98.50** **(0.01)**	**0.98** **(0.01)**	95.17 (0.03)	0.95 (0.03)
5	**GunPoint**	94.50 (0.05)	0.94 (0.05)	97.50 (0.03)	0.97 (0.03)	**98.00** **(0.02)**	**0.98** **(0.02)**
6	**OliveOil**	86.67 (0.15)	0.86 (0.16)	86.67 (0.15)	0.86 (0.16)	**90.11** **(0.11)**	**0.90** **(0.11)**
7	**Trace**	**99.00** **(0.03)**	**0.99** **(0.03)**	**99.00** **(0.03)**	**0.99** **(0.03)**	**99.00** **(0.02)**	**0.99** **(0.02)**
8	**ToeSegmentation2**	88.46 (0.09)	0.88 (0.10)	**92.68** (**0.07**)	**0.92** (**0.7**)	**92.21** **(0.04)**	**0.92** **(0.04)**
9	**Car**	80.83 (0.07)	0.80 (0.09)	81.67 (0.07)	0.81 (0.08)	**83.33** **(0.06)**	**0.83** **(0.06)**
10	**Lightning2**	87.76 (0.09)	0.87 (0.08)	87.76 (0.09)	0.87 (0.08)	**89.17** **(0.08)**	**0.89** **(0.08)**
11	**ShapeletSim**	82.00 (0.10)	0.81 (0.11)	82.00 (0.10)	0.81 (0.11)	**87.00** **(0.07)**	**0.87** **(0.07)**
12	**DiatomSizeReduction**	99.38 (0.01)	0.99 (0.01)	99.69 (0.01)	0.99 (0.01)	**100.00** **(0.00)**	**1.00** **(0.00)**
13	**Adiac**	65.30 (0.04)	0.62 (0.04)	65.30 (0.04)	0.62 (0.04)	**65.81** **(0.03)**	**0.64** **(0.03)**
14	**HouseTwenty**	95.00 (0.05)	0.95 (0.05)	93.00 (0.08)	0.93 (0.08)	**96.25** (**0.05**)	**0.96** (**0.05**)
15	**PenDigits**	85.47 (0.01)	0.85 (0.01)	87.62 (0.01)	0.87 (0.01)	**89.64** **(0.01)**	**0.89** **(0.01)**

Effectiveness Comparison

Table 6.2 presents the comparison between EDBDTW, Standard DTW and S-C Band DTW in terms of accuracy and F1 score. The values in parentheses are the standard deviations recorded after averaging over the ten folds of the TCV. Best results are highlighted in bold font. From the table, it can be seen that the proposed EDBDTW approach produced the best results with respect to fourteen of the fifteen data sets considered; for the remaining one case (the *SyntheticControl* data set), a similar performance was recorded. Where the performance was improved, and not with standing that in some case $\delta = 100\%$, it can be argued that in some cases the reshaped time series according to the identified discriminators served to produce a more representative data bank for use with respect to kNN classification, hence a better classification was produced given previously unseen time series.

6.3 Distance Profile-Based DTW

A criticism of the EDBDTW approach described in the previous section is that in many cases the value of δ is such that there is little (in some cases no) reduction in the overall length x of the time series in D. Hence, in many cases there is little advantage to be gained. This section presents an alternative approach, Distance Profile-Based DTW (DPBDTW), to reduce x by recasting the data into a *distance profile* format. An idea influenced by the work on matrix profiles for motif (and discord) discovery [4, 127, 128]. A distance profile, in the context of the work presented here, is a data structure that serves to reduce the lengths of the time series to be considered by $n-1$, where n is a pre-defined time series sub-sequence length [129]. The reduction in size is thus dependent on the selected value for n. Given a kNN classification scenario both the data bank D, and the previously unseen time series $\mathfrak{t}$ to be classified, have to be converted into distance profiles, Π and $\mathfrak{t}'$ which can then be compared to retrieve the class label associated with the most similar row in Π. The idea is illustrated in Figure 6.5. In the figure, the x-axis represents the time series indexes and the y-axis represents the values. At the top of the figure, we have an input time series S_i, at the bottom, we have the same time series transformed into a distance profile P_i (note the change in shape and the reduced length of $n-1$).

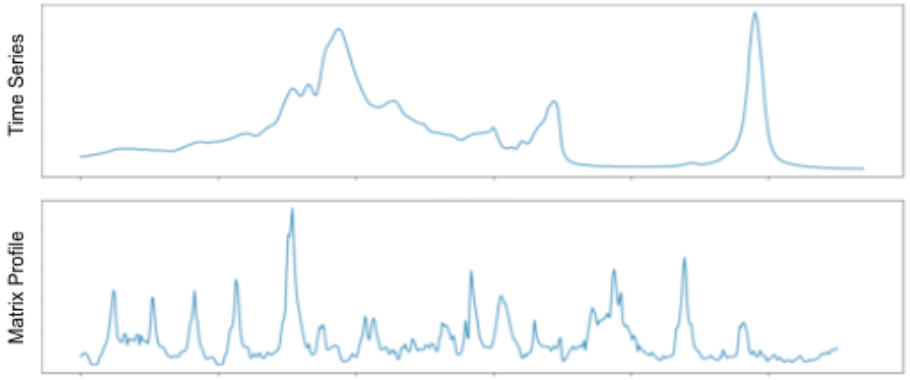

FIGURE 6.5: Distance profile generation. Top: the original time series. Bottom: the resulting distance profile.

The remainder of this section is organised in a similar manner to the previous section. The section commences, Sub-section 6.3.1 with an overview of the operation of the proposed DPBDTW mechanism. For further clarification, a worked example is given in Sub-section 6.3.2 and the associated pseudo code in Sub-section 6.3.3. The theoretical computational complexity for the proposed DPBDTW approach is then discussed in Sub-section 6.3.4. A practical evaluation of the proposed approach is reported on in Section 6.3.5.

6.3.1 Operation of Distance Profile-Based DTW

In this section, the operation of the proposed DPBDTW approach is presented. As noted in the introduction to this section, the fundamental idea is to recast the input data into a distance profile. A distance profile is a $r \times (x - n + 1)$ matrix where r, the number of rows in the profile, is the number of time series (records) to be considered, x is the length of the time series and n is a pre-defined time series sub-sequence length. Given a data set $D = \{S_1, S_2, \ldots, S_r\}$ each time series S_i is represented by a row in the associated distance profile. Each column in the distance profile represents a sub-sequence in a time

series S_i. To generate a row in a distance profile, S_i is first segmented (divided) into $x - n + 1$ overlapping sub-sequences. The elements in a row in the distance profile are then the Euclidean distances between the first sub-sequence and all the sub-sequences in S_i (thus the first element, where the first sub-sequence is compared to itself, will always have the value zero).

A schematic of the overall DPBDTW process is given in Figure 6.6. From the figure, it can be seen that the DPBDTW approach comprises three stages:

Stage 1 Parameter learning to find best parameters (the loop at the top of Figure 6.6).

Stage 2 Data set D transformation to give Π.

Stage 3 Classification of a given query $\mathfrak{t}$ time series through the application of FSSBDTW.

Note that, as in the earlier proposed approaches that featured parameter learning and data transformation, Stages 1 and 2 are only undertaken once (unless of course more training data becomes available). Stage 3 can then be applied as and when required.

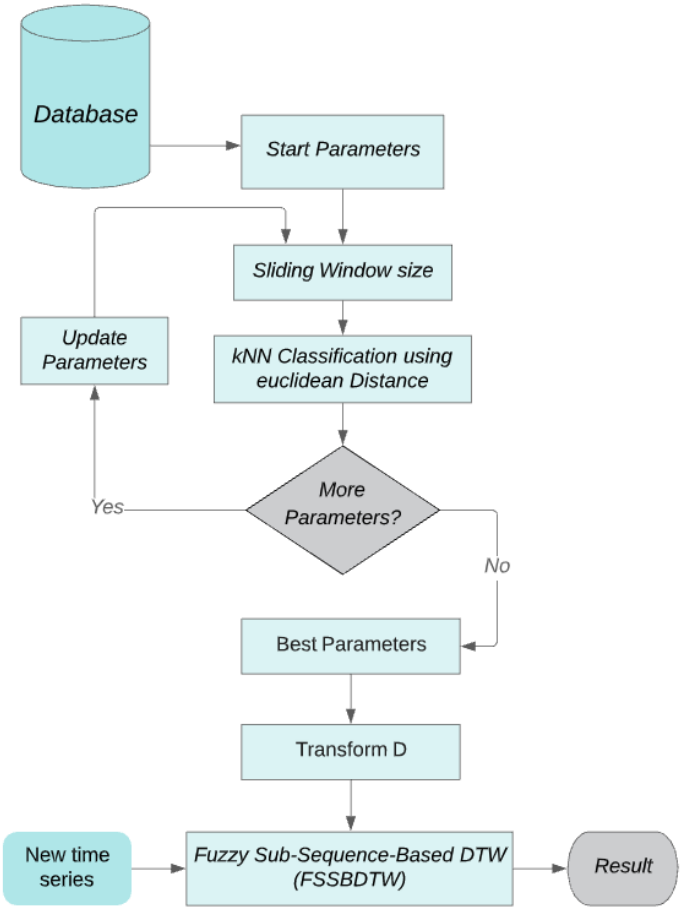

FIGURE 6.6: Schematic outlining the operation of the DPBDTW approach

During Stage 1, the values for the following parameters are learnt: (i) n, the time series sub-sequence length, (ii) $maxLength$, the maximum length of the FSSBDTW segment and (iii) t the length of the tail used with respect to the FSSBDTW approach. As before, during Stage 1, a training and test set are used with Euclidean distance time series comparison. A $9:1$ training and test set split was used with respect to the

evaluation presented later in this section. In Stage 2, the learnt parameter n is used to generate a distance profile Π. Stage 3 is where the desired classification takes place. A worked example to clarify the process is given in the following sub-section, Sub-Section 6.3.2.

6.3.2 Distance Profile-Based DTW Worked Example

As in the case of the EDBDTW approach presented in the previous section, this sub-section presents a worked example of the operation of the DPBDTW approach. For the worked example the same input data set, $D = \{S_1, S_2, \dots\}$, as used previously is considered here:

i	S_i	c_i
1	$[1, 3, 2, 1, 3, 2, 1, 3, 1, 2]$	red
2	$[2, 1, 1, 1, 2, 2, 2, 1, 3, 1]$	red
3	$[4, 5, 5, 6, 5, 3, 4, 5, 5, 6]$	blue
4	$[3, 3, 4, 5, 5, 4, 4, 5, 4, 5]$	blue
5	$[3, 3, 1, 2, 2, 3, 2, 3, 2, 3]$	red
6	$[5, 5, 3, 3, 5, 4, 3, 5, 3, 3]$	blue

The first step is to segment the records in $D = \{S_1, S_2, \dots\}$. Each time series $S_i \in D$ is segmented into $x - n + 1$ sub-sequences each of length n, and the result stored in a set $\Omega = \{W_1, W_2, \dots, W_{x-n+1}\}$. Thus, in the case of $S_1 = [1, 3, 2, 1, 3, 2, 1, 3, 1, 2]$, and assuming $n = 4$, the following will be obtained:

i	W_i
1	$[1, 3, 2, 1]$
2	$[3, 2, 1, 3]$
3	$[2, 1, 3, 2]$
4	$[1, 3, 2, 1]$
5	$[3, 2, 1, 3]$
6	$[2, 1, 3, 1]$
7	$[1, 3, 1, 2]$

We then process W and determine the Euclidean distances between w_1 and all of the sub-sequences in W. Thus the Euclidean distance between $w1$ and w_2 is $abs(1-3) + abs(3-2) + abs(2-1) + abs(1-3) = 2+1+1+2 = 6$. This is repeated for all $w_i \in W$. On completion, the distances will be $[0, 6, 5, 0, 6, 3, 2]$. The process is repeated for all $S_i \in D$. The results are then stored in a *Distance Profile* $\Pi = \{P_1, P_2, \dots\}$ (note that there is a direct correspondence between each P_i and S_i). In the case of the worked example, Π will be of the form:

i	S'_i	c_i
1	$[0, 6, 5, 0, 6, 3, 2]$	red
2	$[0, 2, 3, 4, 2, 3, 2]$	red
3	$[0, 3, 5, 6, 5, 3, 0]$	blue
4	$[0, 2, 5, 5, 3, 4, 3]$	blue
5	$[0, 3, 4, 4, 3, 3, 3]$	red
6	$[0, 4, 7, 3, 3, 5, 2]$	blue

Note that in the above distance profile the first column comprises all zero values. This is because the first column gives the distance between each segment W_1 and itself, hence we would expect this to be zero.

Given a previously unseen time series, $\mathfrak{t}$, this will be compared with the contents of Π using DTW. In the case of the evaluation presented later in this section FSSBDTW was used because of its good performance as established in Chapter 4. To do this $\mathfrak{t}$ has to also first be transformed into a distance profile (of one row).

6.3.3 Distance Profile-Based DTW Algorithm

This pseudo code for the DPBDTW approach is given in this sub-section. The algorithm for Stage 1 is similar to that given in Algorithm 13 for the EDBDTW approach except that: (i) $|C|$ is not required as an input; (ii) Δ is replaced with N, the set of possible values for n; (iii) line 7 references the function $distanceProfileGen(D, n)$ (algorithm 17 presented below). Through a process of experimentation it was found that $N = \{5, 10, \ldots, 95, 100\}$ was an appropriate setting for N. The same settings for I and T as used previously were adopted; $I = \{10, 20, \ldots, 90, 100\}$ and $T = \{1, 2, \ldots, 9, 10\}$.

The pseudo code for Stage 2, distance matrix generation, is given in Algorithm 17. The input is a data set $D = \{S_1, S_2, \ldots, S_r\}$ and the n parameter value identified in Stage 1. Note, D could comprise only one record ($r = 1$) when considering a previously unseen record to be labelled. The output is a distance profile $\Pi = \{P_1, P_2, \ldots, P_r\}$. The distance profile is defined on line 2. Next D is processed record by record (lines 3 to 9). On each iteration, the current time series $S_i \in D$ is segmented into $x - n + 1$ time series to give $\Omega = \{W_1, W_2, \ldots, W_{x-n+1}\}$. The Euclidean distance between $W_1 \in \Omega$ and all $W_j \in \Omega$ is then calculated (including the distance between W_1 and itself, which will, obviously, be zero). To determine the Euclidean distance Algorithm 5 from Chapter 5 was used. The calculated distances are then stored at index i, j in Π. On completion, the populated distance profile is returned (line 10). Although not shown in Algorithm 17., we also retain the class label associated with each row in the distance profile.

Algorithm 17 Distance Profile Generation (Stage 2)

```
1:  input D, n
2:  Π = r × x − n + 1                                   ▷ Empty distance profile
3:  for ∀S_i ∈ D do
4:      Ω = {W_1, W_2, ..., W_{x−n+1}} = S_i segmented into x − n + 1 sub-sequences
5:      for ∀W_j ∈ W do
6:          d = euclideanDistance(W_1, W_j)                          ▷ Algorithm 5
7:          Π_{i,j} = d
8:      end for
9:  end for
10: return Π
```

Algorithm 18 DPBDTW Parent Code (Stages 1 and 2)

```
1: input D_train, D_test, |C|, I, T, N.
2: [maxLength, t, n] = paramLearn(D_train, D_test, I, T, N)          ▷ Stage 1
3: Π = distanceProfileGen(D_train ∪ D_test, n)          ▷ Stage 2 Algorithm 17
4: return Π, maxLength, t, n
```

Algorithm 18 gives the parent code for Stages 1 and 2. The input (line 1) is: (i) a collection of pre-labelled time series, D_{train}, to form the initial kNN bank, (ii) a second collection of pre-labelled time series, D_{test}, to form a test set, (iii) a set I of possible values for the FSSBDTW *maxlength* parameter, (iv) a set T of possible values for the

FSSBDTW t tail parameter and (v) a set N of possible values for the segment length parameter n. The parameter learning is first conducted (line 2) followed by distance profile generation (line 3). The distance profile Π is then returned on line 4, together with the parameters $maxLength$, t and n. The distance profile Π then becomes the kNN bank for classifying previously unseen records. To conduct the desired classification, each previously unseen record has to first be transformed into the distance profile format using the parameter n. For the evaluation presented later in this section, FSSDDTW was used for the actual classification, which requires the parameters $maxLength$ and t. Hence the best values for $maxlength$ and t are also returned by Algorithm 18.

Algorithm 19 DPBDTW Classification (Stage 3)

```
1: input Π, maxLength, t, n, ţ.
2: ţ' = distanceProfileGen(ţ, n)                    ▷ Algorithm 17
3: bestWD = ∞
4: c = default class
5: for ∀P_i ∈ Π do
6:     wd = FSSDDTW(P_i, ţ', maxLen, t)
7:     if wd < bestWD then
8:         bestWD = wd
9:         c = class associated with P_i ∈ D'
10:    end if
11: end for
12: return c
```

The final stage of the proposed DPBDTW approach, Stage 3, is the classification of previously unseen time series. The pseudo code is given in Algorithm 19. The input (line 1) is: (i) the kNN bank Π; (ii) the value for the parameter $maxLength$; used by FSSBDTW; (iii) the value for the parameter t, also used by FSSBDTW; (iv) the value for parameter n used to generate a distance profile for the previously unseen time series, and (v) the time series to be labelled ţ. The first step, line 2, is to transform ţ into a single row distance profile. Then, lines 5 to 11, we determine the best match in Π with ţ$'$ and retain the associated class c, which is returned in line 12.

6.3.4 Time Complexity of Distance Profile Based DTW

In this section, the theoretical time complexity of the DPBDTW approach is presented, again assuming TCV and 1NN classification. The organisation of this section is similar to that used to derive the complexity for the EDBDTW approach given in Sub-section 6.2.4 above. Starting with Stage 1 the complexity will be:

$$\begin{aligned} stage1_{complexity} = \; & numberParameterCombinations \times numberOfComparisons \\ & \times EuclideanComparison_{complexity} \end{aligned} \tag{6.20}$$

where:

$$numberParameterCombinations = |I| \times |T| \times |N| \tag{6.21}$$

$$numberOfComparisons = \frac{|D|}{10} \times \frac{9 \times |D|}{10} = \frac{9 \times |D|^2}{100} \tag{6.22}$$

$$EuclideanComparison_{complexity} = O\left(\bar{n}\right) \tag{6.23}$$

Note that for determining the Euclidean distance comparison complexity the average value of n is used ($\bar{n}$). The Stage 1 complexity equation thus becomes:

$$stage1_{complexity} = O\left(|I| \times |T| \times |N| \times \frac{9 \times |D|^2}{100} \times \bar{n}\right) \tag{6.24}$$

Recall that the sets I, T and N were pre-specified, and that $|I| = 10$, $|T| = 10$ and $|N| = 20$. The above thus becomes:

$$stage1_{complexity} = O\left(10 \times 10 \times 20 \times \frac{9 \times |D|^2}{100} \times \bar{n}\right) \tag{6.25}$$

which simplifies to:

$$stage1_{complexity} = O\left(180 \times |D|^2 \times \bar{n}\right) \tag{6.26}$$

When using TCV, the parameter tuning will be conducted 10 times. Hence the above will become:

$$\begin{aligned} stage1TCV_{complexity} &= O\left(10 \times 180 \times |D|^2 \times \bar{n}\right) \\ &= O\left(1800 \times |D|^2 \times \bar{n}\right) \end{aligned} \tag{6.27}$$

The complexity of Stage 2, distance profile generation, depends on the size of the distance profile Π, which will be dictated by the number of records in D and the number of segments into which each time series is divided times. Thus $|D| \times (x - n + 1)$[1]. Therefore:

$$stage2_{complexity} = O\left(|D| \times (x - \bar{n} + 1)\right) \tag{6.28}$$

Note that the average value $\bar{n}$ for the segment length n is again used here. When using TCV, distance profile generation will be conducted 10 times for both D_{train} and D_{test} ($D = D_{train} \cup D_{test}$). Therefore:

$$stage2TCV_{complexity} = O\left(10 \times |D| \times (x - \bar{n} + 1)\right) \tag{6.29}$$

The determination of the class of a query time series $\mathfrak{t}$ will then require $|\Pi|$ comparisons. In other words, $|D|$. The complexity for class determination, assuming the FSSBDTW approach, will then be:

$$classDetermination_{complexity} = O\left(|D| \times DTW_{complexityFSSBDTW}\right) \tag{6.30}$$

The complexity of FSSBDTW was previously given in Equation 4.11 in Chapter 4 as:

$$DTW_{complexityFSSBDTW} = O\left(x \times \overline{len}\right) \tag{6.31}$$

where x was the length of the time series to be labelled and $\overline{len}$ was the average length of a segment. However, in the case of the DPBDTW approach, the length of the time series will have been reduced to $x - n + 1$. The classification complexity thus becomes:

[1]The parameter r could be used here, but $|D|$ has been used here to maintain consistency with earlier complexity calculations

$$stage3_{complexity} = O\left(|D| \times x - \bar{n} + 1 \times \overline{len}\right) \tag{6.32}$$

Note that the average value for the parameters n is again used here. If we are undertaking TCV, the number of classifications will equate to $|D|$. Thus:

$$\begin{aligned} stage3TCV_{complexity} &= O\left(|D| \times |D| \times x - \bar{n} + 1 \times \overline{len}\right) \\ &= O\left(|D|^2 \times x - \bar{n} + 1 \times \overline{len}\right) \end{aligned} \tag{6.33}$$

The overall complexity of the DPBDTW approach, assuming TCV, will then be:

$$\begin{aligned} DPBDTW\_TCV_{complexity} = \; & stage1TCV_{complexity} + stage2TCV_{complexity} + \\ & stage3TCV_{complexity} \end{aligned} \tag{6.34}$$

In other words:

$$\begin{aligned} DPBDTW\_TCV_{complexity} = \; & O(\left(1800 \times |D|^2 \times \bar{n}\right) + (10 \times |D| \times (x - \bar{n} + 1)) + \\ & \left(|D|^2 \times x - \bar{n} + 1 \times \overline{len}\right)) \\ = \; & O(|D| \times ((1800 \times |D| \times \bar{n}) + (10 \times (x - \bar{n} + 1)) + \\ & \left(|D| \times x - \bar{n} + 1 \times \overline{len}\right))) \end{aligned} \tag{6.35}$$

6.3.5 Evaluation of Distance Profile-Based DTW

In this section, the evaluation of the proposed DPBDTW mechanism is presented. The evaluation was conducted using 1NN classification, and the same fifteen selected data sets from the UEA-UCR Time Series Classification repository as used earlier. In line with the earlier evaluations reported on in this thesis, the objectives of the evaluation were:

1. **Operational Analysis**: To analyse the operation of the proposed DPBDTW approach.
2. **Efficiency Comparison**: To evaluate the run-time advantages gained using the proposed DPBDTW approach in comparison with Standard DTW and S-C Band DTW benchmark approaches presented in Chapter 3.
3. **Effectiveness Comparison**: To determine whether the classification accuracy of the proposed DPBDTW approach was commensurate with that obtained using standard DTW and S-C Band DTW.

For the S-C Band approach, as on previous occasions, a warping window of $\ell = 10\%$ was used as recommended in [81, 92]. The experimental set up was the same as that used with respect to the previous evaluations reported on in this thesis. A desktop computer was used with an Apple M1 8 core CPU processor, an 8 core GPU, a 16 core Neural Engine, 16GB unified memory, and 512GB Solid State Drive (SSD).

Operational Analysis

This sub-section reports on the results obtained to determine the most appropriate value for the parameters n, $maxLen$ and t. A summary of the best results, including the best parameters settings, run time, accuracy and f1-score are presented in Table 6.3. From the table, it can be seen that, in line with earlier evaluation outcomes, no "best" parameters settings could be identified; best classification results were obtained using different settings for each data set. From Table 6.3 it is interesting to note that relatively small best values for n were obtained, meaning that in many cases the length of the distance profile time series was not significantly less than the original time series length. Table 6.4 presents a summary of the effect of distance profiling in terms of the reduction in overall time series length using the best value for n as listed in Table 6.3. The lowest recorded best value for n was 4 with respect to the *PenDigits* data set. For this data set $x = 8$, hence a reduction to $8 - 4 + 1 = 5$, a reduction of 37.5%. The highest recorded best value for n was 70 with respect to the *Adiac* data set. For this data set $x = 176$, hence a reduction to $176 - 70 + 1 = 107$, a reduction of 29.2%.

TABLE 6.3: Best values for parameters $maxLen$, t and n in terms of run time, accuracy and F1 results for each data set.

ID #	**Data set**	**Length** $maxLen$	**Tail** t	Seg. len. n	**Run time** (Secs)	**Acc** (SD)	**F1** (SD)
1	**SmoothSubspace**	10	2	7	1.63	43.00 (0.07)	0.43 (0.07)
2	**ItalyPowerDemand**	4	2	11	21.94	95.16 (0.01)	0.95 (0.01)
3	**Libras**	9	2	15	5.80	65.83 (0.11)	0.65 (0.11)
4	**SyntheticControl**	30	1	10	9.44	89.17 (0.03)	0.89 (0.03)
5	**GunPoint**	40	2	30	4.20	98.00 (0.02)	0.98 (0.02)
6	**OliveOil**	80	2	20	2.04	90.00 (0.08)	0.90 (0.08)
7	**Trace**	50	2	40	5.49	100.00 (0.00)	1.00 (0.00)
8	**ToeSegmentation2**	50	2	15	7.54	88.53 (0.06)	0.88 (0.06)
9	**Car**	50	6	35	7.39	83.33 (0.09)	0.83 (0.09)
10	**Lighting2**	80	2	45	8.22	75.26 (0.13)	0.75 (0.13)
12	**ShapeletSim**	40	1	5	15.21	99.50 (0.01)	0.99 (0.01)
11	**DiatomSizeReduction**	20	2	40	21.21	100.00 (0.00)	1.00 (0.00)
13	**Adiac**	10	2	70	64.28	53.53 (0.06)	0.53 (0.06)
14	**HouseTwenty**	40	2	20	41.91	94.29 (0.06)	0.94 (0.06)
15	**PenDigits**	4	1	4	490.66	88.46 (0.01)	0.88 (0.01)

Comparing the best values for $maxLength$ and t given in Table 6.3 with those given previously in Table 4.5 in Chapter 4, it can be observed that different best values for $maxLength$ and t were obtained using the proposed DPBDTW approach compared to using the FSSBDTW approach in isolation. As in the case of the EDBDTW approach

Table 6.4: Reduction to time series length as a consequence of distance profile generation.

ID #	Data set	Length x	Seg. len n	Reduced length	% reduction
1	**SmoothSubspace**	15	7	9	40.00
2	**ItalyPowerDemand**	24	11	14	41.67
3	**Libras**	45	15	31	31.11
4	**SyntheticControl**	60	10	51	15.00
5	**GunPoint**	150	30	121	19.33
6	**OliveOil**	570	20	551	3.33
7	**Trace**	275	40	236	14.18
8	**ToeSegmentation2**	343	15	329	4.08
9	**Car**	577	35	543	5.89
10	**Lighting2**	637	45	593	6.91
12	**ShapeletSim**	500	5	496	0.80
11	**DiatomSizeReduction**	345	40	306	11.30
13	**Adiac**	176	70	107	39.20
14	**HouseTwenty**	2000	20	1981	0.95
15	**PenDigits**	8	4	5	37.50

considered in the foregoing section, it was conjectured that this was either because of the interplay between the parameters $maxLength$ and t, and n; or because of a certain degree of "volatility" with respect to the parameters $maxLength$, t and n; or a mixture of the two.

Efficiency Comparison

Figure 6.7 gives a comparison of the run time performance of the Standard DTW and S-C Band DTW benchmark approaches and the DPBDTW approach. As in the case of similar graphs presented previously, the x-axis represents the data set identification numbers (see Table 6.3), and the y-axis represents the run time in seconds. The values for $maxLen$, t and n that were used were the best performing values as reported on in Table 6.3. From the figure, it can be seen that for 13 of the 15 data sets considered, the proposed DPBDTW approach offered efficiency gains over the benchmark approaches. The exceptions were the *Libras* data set and the *Adiac* data set. The *Libras* data set was a relatively small data set, 360 records comprised of 45 points each (see Table 3.1 given previously in Chapter 3), hence the advantage offered by the DPBDTW approach my not always materialise in such a small data set. The *Adiac* data set featured an advantageous value for n, it is thus unclear why, in this case, the S-C Band DTW approach produced a slightly better performance.

Effectiveness Comparison

Table 6.5 gives the accuracy and F1 results for the DPBDTW approach in comparison with the benchmark Standard DTW and S-C BAND DTW approaches. The figures in parentheses give the standard deviations recorded after averaging over the ten folds of the TCV. Best results are highlighted in bold font. From the table, it can be seen that the proposed DPBDTW mechanism, in nine of the fifteen cases, produced a better performance than that produced using either Standard DTW or S-C Band DTW (100% accuracy in two cases). In one case, the *ItalyPowerDemand* data set, an identical performance was recorded. In the five remaining cases, DPBDTW did not produce the best results. In one case, *SmoothSubspace*, DPBDTW produced a particularly poor performance. It was conjectured that this was because the *SmoothSubspace* data set featured

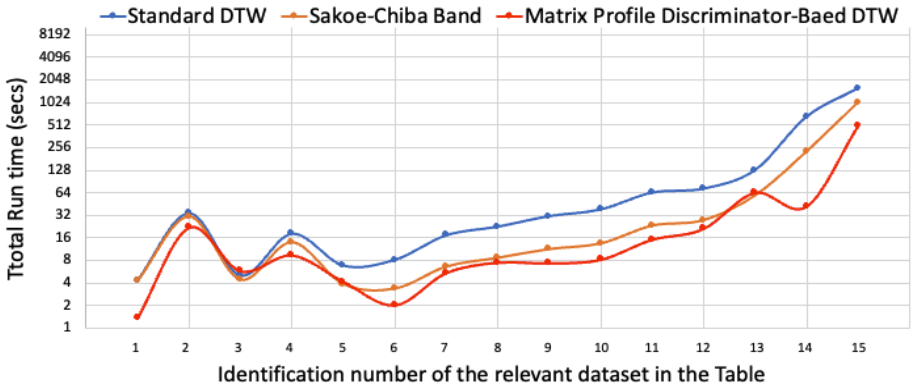

FIGURE 6.7: Run time results using the Standard DTWn and S-C Band DTW benchmark approaches and the proposed DPBDTW approach.

time series of only 15 points, hence reducing the length had an adverse effect. For the remaining five cases, the loss in accuracy when using DPBDTW was less pronounced. There did not seem to be any correlation between x and accuracy that might have indicated that DPBDTW was more suited to time series that featured larger values of x.

6.4 Exact Discriminator-Based DTW versus Distance Profile-Based DTW

A comparison of the performance of the EDBDTW approach and the DPBDTW approach is presented in this section. The collated recorded run times, and accuracy and F1-scores, with respect to each approach, and each data set, are given in Tables 6.6 and 6.7 respectively. The results with respect to the best performing parameters were used in each case.

Table 6.6 gives the run time results. The same results are presented in graphical form in Figure 6.8. In the figure, the x-axis records the identification number of the relevant data set, and the y-axis the run time in seconds. From both the table and the figure, it can be seen that the EDBDTW and DPBDTW approaches both have very similar run times. There is little to choose between them.

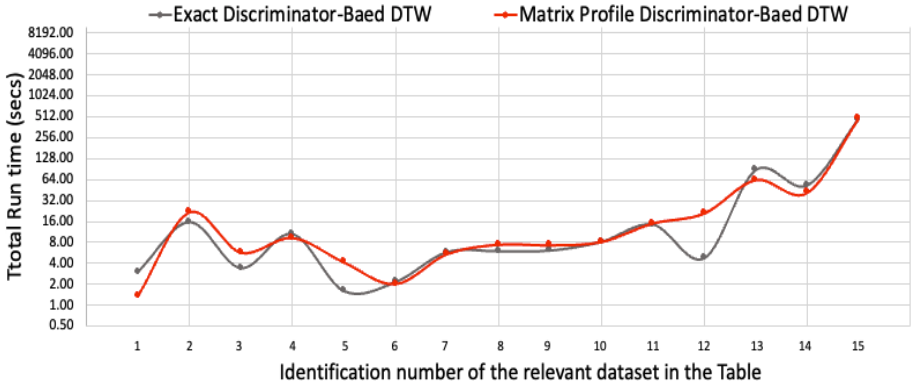

FIGURE 6.8: Comparison of the recorded run time results using the EDBDTW (gray) and DPBDTW (red) approaches with respect to the fifteen evaluation data sets.

Table 6.7 gives the accuracy values and the F1 scores obtained using the EDBDTW and DPBDTW approaches (values and scores obtained by averaging over the ten folds of

Table 6.5: Accuracy and F1-Score using Standard DTW, S-C Band and DPBDTW approaches.

ID #	Data set Name	Standard DTW		S-C Band $\ell = 10$		DPBDTW	
		Acc (SD)	**F1** (SD)	**Acc** (SD)	**F1** (SD)	**Acc** (SD)	**F1** (SD)
1	**SmoothSubspace**	91.00 (0.04)	0.91 (0.04)	**95.00** (**0.02**)	**0.95** (**0.02**)	43.00 (0.07)	0.43 (0.07)
2	**ItalyPowerDemand**	**95.61** (**0.02**)	**0.95** (**0.02**)	**95.70** (**0.02**)	**0.95** (**0.02**)	**95.16** (**0.01**)	**0.95** (**0.01**)
3	**Libras**	58.89 (0.10)	0.60 (0.11)	63.06 (0.11)	0.60 (0.11)	**65.83** (**0.11**)	**0.65** (**0.11**)
4	**SyntheticControl**	**98.50** **(0.01)**	**0.98** **(0.01)**	**98.50** **(0.01)**	**0.98** **(0.01)**	89.17 (0.03)	0.89 (0.03)
5	**GunPoint**	94.50 (0.05)	0.94 (0.05)	97.50 (0.03)	0.97 (0.03)	**98.00** (**0.02**)	**0.98** (**0.02**)
6	**OliveOil**	86.67 (0.15)	0.86 (0.16)	86.67 (0.15)	0.86 (0.16)	**90.00** (**0.0**8)	**0.90** (**0.08**)
7	**Trace**	99.00 (0.03)	0.99 (0.03)	99.00 (0.03)	0.99 (0.03)	**100.00** (**0.00**)	**1.00** (**0.00**)
8	**ToeSegmentation2**	88.46 (0.09)	0.88 (0.10)	**92.68** (**0.07**)	**0.92** (**0.7**)	88.53 (0.06)	0.88 (0.06)
9	**Car**	80.83 (0.07)	0.80 (0.09)	81.67 (0.07)	0.81 (0.08)	**83.33** (**0.09**)	**0.83** (**0.09**)
10	**Lightning2**	**87.76** (**0.09**)	**0.87** (**0.08**)	**87.76** (**0.09**)	**0.87** (**0.08**)	75.26 (0.13)	0.75 (0.13)
11	**ShapeletSim**	82.00 (0.10)	0.81 (0.11)	82.00 (0.10)	0.81 (0.11)	**99.50** (**0.01**)	**0.99** (**0.01**)
12	**DiatomSizeReduction**	99.38 (0.01)	0.99 (0.01)	99.69 (0.01)	0.99 (0.01)	**100.00** (**0.00**)	**1.00** (**0.00**)
13	**Adiac**	**65.30** (**0.04**)	**0.62** (**0.04**)	**65.30** (**0.04**)	**0.62** (**0.04**)	53.53 (0.06)	0.53 (0.06)
14	**HouseTwenty**	**95.00** (**0.05**)	**0.95** (**0.05**)	93.00 (0.08)	0.93 (0.08)	94.29 (0.06)	0.94 (0.06)
15	**PenDigits**	85.47 (0.01)	0.85 (0.01)	87.62 (0.01)	0.87 (0.01)	**88.46** (**0.01**)	**0.88** (**0.01**)

the TCV). The figures in parentheses are the standard deviation values obtained. From the table, it can be seen that the performance using the EDBDTW was better than in the case of the DPBDTW approach in six of the fifteen cases. In another six cases, the performance was the same. In three cases, the *Libras*, *Trace* and *ShapeletSim* data sets, the DPBDTW approach produced a better performance. There was no obvious reason why this should be the case other than as a result of the vagaries of the different data sets involved.

6.5 Conclusion

The chapter has suggested, in the context of kNN classification, two novel mechanisms whereby the length x of the time series to which DTW is to be applied can be reduced. The intuition was that if the time series length was reduced, the DTW distance matrix would be reduced, and hence the DTW process would become more efficient. Of course, any reduction in x should not cause a commensurate loss in classification accuracy.

Table 6.6: Recorded Run times for the EDBDTW and DPBDTW approaches.

ID.	Data set	EDBDTW Run time (secs.)	DPBDTW Run time (secs.)
1.	SmoothSubspace	03.04	1.63
2.	ItalyPowerDemand	15.87	21.94
3.	Libras	3.47	5.80
4.	SyntheticControl	10.68	9.44
5.	GunPoint	1.63	4.20
6.	OliveOil	2.18	2.04
7.	Trace	5.88	5.49
8.	ToeSegmentation2	6.05	7.54
9.	Car	6.19	7.39
10.	Lightning2	8.25	8.22
11.	ShapeletSim	14.64	15.21
12.	DiatomSizeReduction	4.81	21.21
13.	Adiac	88.20	64.28
14.	HouseTwenty	53.32	41.91
15.	PenDigits	480.17	490.66

The first idea promoted in this chapter was that of *discriminators*, a time series subsequence with fixed start and end locations (indexes) within a time series collection (assuming that all time series are of a constant length) to which DTW should be applied, therefore, reducing the overall length of the time series to be considered. The criterion for selecting these indexes was the similarity between time series segments across the input time series with respect to individual classes. The idea was incorporated into the Exact Discriminator-Based DTW (EDBDTW) approach. The second idea promoted in this chapter was to translate the input data into a distance profile format, after first segmenting the data, a consequence of which was reduced time series length. The idea was incorporated into the Distance Profile-Based DTW (DPBDTW) approach. Both approaches featured a parameter learning stage; the discriminator length δ in the case of the EDBDTW approach, and the segment length n in the case of the DPBDTW approach. Both approaches were fully described. Evaluation of the EDBDTW approach indicated that the value for δ could be very large, equivalent to x in some cases, hence the advantages gained were limited; although, overall, a better performance was recorded compared to Standard and S-C Band DTW. Evaluation of the DPBDTW approach was not conclusive; although, as in the case of the EDBDTW approach, a better performance than Standard and S-C Band DTW was observed generally. Comparing EDBDTW and DPBDTW indicated that there was little to differentiate the two approaches (in terms of accuracy, F1 score and run time), although both provided interesting insights into addressing the computational overhead associated with DTW. The following chapter concludes the thesis with a summary and an overview of the main findings in terms of the research question and related subsidiary questions from Chapter 1. The chapter also presents a comparison of the eight different approaches directed at reducing the complexity of DTW that have been proposed and discussed in this thesis, and some possible directions for future work whereby the work presented in this thesis can be extended.

TABLE 6.7: Recorded Accuracy and F1-Scores for the EDBDTW and DPBDTW approaches.

ID	Data set	EDBDTW		DPBDTW	
#	Name	**Accuracy** (SD)	**F1-Score** (SD)	**Accuracy** (SD)	**F1-Score** (SD)
1	**SmoothSubspace**	**95.00 (0.02)**	**0.95 (0.02)**	43.00 (0.07)	0.43 (0.07)
2	**ItalyPowerDemand**	**96.35 (0.02)**	**0.96 (0.02)**	**95.16 (0.01)**	**0.95 (0.01)**
3	**Libras**	64.17 (0.12)	0.61 (0.12)	**65.83 (0.11)**	**0.65 (0.11)**
4	**SyntheticControl**	**95.17 (0.03)**	**0.95 (0.03)**	89.17 (0.03)	0.89 (0.03)
5	**GunPoint**	**98.00 (0.02)**	**0.98 (0.02)**	**98.00 (0.02)**	**0.98 (0.02)**
6	**OliveOil**	**90.11 (0.11)**	**0.90 (0.11)**	**90.00 (0.08)**	**0.90 (0.08)**
7	**Trace**	99.00 (0.02)	0.99 (0.02)	**100.00 (0.00)**	**1.00 (0.00)**
8	**ToeSegmentation2**	**92.21 (0.04)**	**0.92 (0.04)**	88.53 (0.06)	0.88 (0.06)
9	**Car**	**83.33 (0.06)**	**0.83 (0.06)**	**83.33 (0.09)**	**0.83 (0.09)**
10	**Lightning2**	**89.17 (0.08)**	**0.89 (0.08)**	75.26 (0.13)	0.75 (0.13)
11	**ShapeletSim**	87.00 (0.07)	0.87 (0.07)	**99.50 (0.01)**	**0.99 (0.01)**
12	**DiatomSizeReduction**	**100.00 (0.00)**	**1.00 (0.00)**	**100.00 (0.00)**	**1.00 (0.00)**
13	**Adiac**	**65.81 (0.03)**	**0.64 (0.03)**	53.53 (0.06)	0.53 (0.06)
14	**HouseTwenty**	**96.25 (0.05)**	**0.96 (0.05)**	94.29 (0.06)	0.94 (0.06)
15	**PenDigits**	**89.64 (0.01)**	**0.89 (0.01)**	**88.46 (0.01)**	**0.88 (0.01)**

Chapter 7

Conclusion and Future Work

7.1 Introduction

In this Chapter, the work presented in this thesis is concluded. The chapter commences with Section 7.1.1 which provides a summary of the thesis including a comparative review of the eight approaches considered. The main findings and contributions of the thesis will then be presented in Section 7.1.2 with respect to the research question and subsidiary research questions. The chapter is then concluded with potential areas and directions for future research that build upon the work presented in the thesis.

7.1.1 Summary of Thesis

The work presented in this thesis was directed at reducing the computational overhead associated with the application of Dynamic Time Warping (DTW) to compare time series; particularly in the context of 1NN classification which entails significant amounts of time series comparison. The thesis commenced (Chapter 1) with an overview of time series and similarity measurements including DTW and Euclidean Distance (ED). The observation was made that DTW tended to be more accurate although more expensive than ED comparison. Then motivation, research question, subsidiary questions and research methodology were also presented in this first chapter. The fundamental idea was to make the use of DTW more effective and efficient, especially with a view to processing large time series data sets. DTW has quadratic complexity; therefore, when using DTW with larger time series, the complexity increases in a quadratic manner.

Chapter 2 then presented a review of related work. The chapter included an overview of time series classification including consideration of a number of classification approaches: (i) Decision Trees, (ii) Support Vector Machines and (iii) k-Nearest Neighbour. This was followed by a discussion of time series similarity measurement approaches, namely: (i) lock-step measures and (ii) elastic measures. A detailed description of the Dynamic Time Warping (DTW) process was included. The computational complexity of DTW was again high-lighted. The chapter also discussed some previous work directed at addressing DTW complexity starting with the idea of warping windows: (i) predefined warping window, (ii) learnt warping window and (iii) localised windows. It was also noted that an alternative to addressing the computational complexity of DTW was to adopt a lower bounding mechanism, three were considered: (i) LB Yi Lower Bound, (ii) LB Kim Lower Bound and (iii) LB Keogh Lower Bound. The Chapter was concluded with a review of a range of motif discovery mechanisms including: (i) the MK Algorithm and (ii) the Matrix Profile technique. The significance was that the motif idea was a

previously proposed mechanism for reducing DTW complexity that influenced two of the approaches considered later in the thesis in Chapter 6.

An overview of the data sets used throughout the thesis was presented in Chapter 3. In total, fifteen time series data sets were used for evaluation purposes throughout the thesis. These data sets were all taken from The UEA and UCR (the University of East Anglia and the University of California Riverside) Time Series Classification Repository [14]. Data sets were selected so that a range of different sizes in terms of the number of columns and the number of rows were considered. Chapter 3 also presented an analysis of two benchmark DTW approaches:

1. Standard DTW.
2. Sakoe-Chiba Band DTW (S-C Band DTW).

The S-C Band DTW approach featured the idea of a warping window so as to realise efficiency gains. The significance of both approaches was that they could be used for comparison purposes with respect to alternative approaches presented later in the thesis. Any proposed alternative, to be of value, had to outperform the two benchmark approaches. The chapter was concluded with a comparison between the operation of the Standard DTW and S-C Band DTW approaches in terms of run time, accuracy and F1-score. As was anticipated, the S-C Band approach outperformed the Standard DTW approach in terms of run time, and achieved an equivalent, and in some cases better, accuracy and F1 score.

The following three chapters, Chapters 4, 5 and 6, featured six alternative approaches to addressing the computational overhead associated with the application of DTW:

1. Sub-Sequence-Based DTW (SSBDTW).
2. Fuzzy Sub-Sequence-Based DTW (FSSBDTW).
3. Candidate Reduction Based on Euclidean Distance (CRBED).
4. Candidate Reduction Based on Lower Bounding (CRBLB).
5. Exact Discriminator-Based DTW (EDBDTW).
6. Distance Profile-Based DTW (DPBDTW)

The idea underpinning the first two approaches, SSBDTW and FSSBDTW, was that the overall size of the DTW distance matrix could be reduced if a given pair of time series were segmented and the individual segments compared (using DTW). The first adopted a "crisp" boundary between segments, while the second used a "fuzzy boundary". The reported evaluation demonstrated that efficiency gains could indeed be made in comparison with Standard DTW and S-C Band DTW, although not always the case for all data sets. In addition, it was found that accuracy gains were also made (again in comparison with Standard DTW and S-C Band DTW). It was conjectured that this was because the comparison of short time series sub-sequences was more resilient to noise. Comparison of SSBDTW and FSSBDTW indicated that FSSBDTW was more effective (more accurate) since the splitting process was more flexible. In addition, although the reported evaluation indicated that both tended have similar run times, it could be argued that FSSBDTW was more efficient with respect to longer time series.

The next two approaches, CRBED and CRBLB, investigated mechanisms whereby the number of DTW comparison instances could be reduced. The fundamental idea,

given a kNN bank D of time series and a new time series t to be labelled, was to "throw away" the time series in D that were unlikely to be good matches for t leaving a reduced data set D'. Using the CRBED mechanisms the time series in D were compared with t using inexpensive Euclidean distance comparison. Using CRBLB a lower bounding technique was used. In both cases, a reduced data set D' was produced which could then be compared with t using the FSSBDTW approach from the previous chapter. From the reported evaluation it was found that significant efficiency gains could be made. In addition, it was found that accuracy gains were also made with respect to Standard DTW and SC Band DTW. It was conjectured that this was because the comparison was done only on similar candidates, not on all the time series in the data set.

The idea underpinning the last two approaches was, instead of reducing the number of DTW comparisons, to reduce the over number of points in the time series to be considered thereby reducing the size of the DTW distance matrix. Two mechanisms were considered whereby this could be achieved. The first involved finding a location (a sub-sequence) in a set of time series in D associated with a particular class where DTW could best be applied; a location (sub-sequence) which was most representative of the class. The second involved transforming D into a distance profile representation (an idea influenced by existing work on motifs). The two mechanisms were incorporated into two DTW approaches, EDBDTW and DPBDTW. From the reported evaluation, it was found that the accuracy gains were made with respect to Standard DTW and SC Band DTW. It was conjectured, in this case, that this was because the comparison tended to be more resistant to noise. Comparison of EDBDTW and DPBDTW indicated that EDBDTW tended to be more effective, although they both featured similar run times.

Table 7.1: Recorded Run times (Secs) for the fifteen time series data sets using the eight approaches presented in this thesis, best results are highlighted in bold.

ID.	Dataset	DTW	SC-DTW	SSBDTW	FSSBDTW	CRBED	CRBLB	EDBDTW	DPBDTW
1.	Smooth	4.36	4.32	4.51	3.03	**0.92**	**0.92**	3.04	1.63
2.	ItalyPower	33.97	30.83	35.97	39.66	**13.77**	13.20	15.87	21.94
3.	Libras	5.04	4.46	5.49	5.71	**2.64**	2.14	3.47	5.80
4.	Synthetic	18.10	14.03	14.88	16.07	**12.10**	6.44	10.68	9.44
5.	GunPoint	6.84	3.89	4.68	3.81	**2.66**	1.26	1.63	4.20
6.	OliveOil	8.15	3.39	3.09	1.88	**1.88**	1.32	2.18	2.04
7.	Trace	17.30	6.65	10.22	7.19	4.72	1.97	5.88	**5.49**
8.	ToeSegment	22.52	8.70	5.20	**8.19**	2.32	1.61	6.05	7.54
9.	Car	31.12	11.38	6.76	6.88	**5.15**	1.76	6.19	7.39
10.	Lighting2	38.15	13.52	**13.88**	8.11	3.00	2.14	8.25	8.22
11.	Shapelet	64.03	23.34	24.13	24.18	14.94	13.12	14.64	**15.21**
12.	DiatomSize	72.29	27.61	16.87	15.79	**2.98**	6.64	4.81	21.21
13.	Adiac	127.74	58.89	41.94	88.20	8.40	19.51	88.20	64.28
14.	HouseTwenty	653.64	226.56	334.36	50.64	19.04	18.24	**53.32**	41.91
15.	PenDigits	1569.87	1023.42	1036.28	480.17	**120.37**	869.43	480.17	490.66
	Ave. Runtime	178.21	97.42	103.88	50.63	**14.33**	63.98	46.95	47.13

A comparison of the eight approaches considered in this thesis in terms of recorded run times, accuracy and F1 score is given in Tables 7.1, 7.2 and 7.3. In each case, the results were obtained using TCV and the best parameter settings. In each table, best results are highlighted in bold font (by considering best runtime with respect to best accuracy obtained). Note that some approaches feature a run time, however their accuracy was adversely affected; therefore, the performance was balanced to find the best approach. With reference to Table 7.1, it can be seen that CRBED coupled with FSSBDTW was the most efficient (best recorded run time in 9 of the 15 cases). Considering the total run time, last row in Table 7.1, it can be confirmed that the total run time for CRBED coupled with FSSBDTW was the best among the other approaches. Figure 7.1 illustrates the run time performance for all the proposed approaches. From the figure, it can be seen that for the largest data sets (*HouseTwenty* and *PenDigits*) the proposed approach performed much better than for the smaller data sets. Note that CRBED approach (red line) has the best performance among the other approaches.

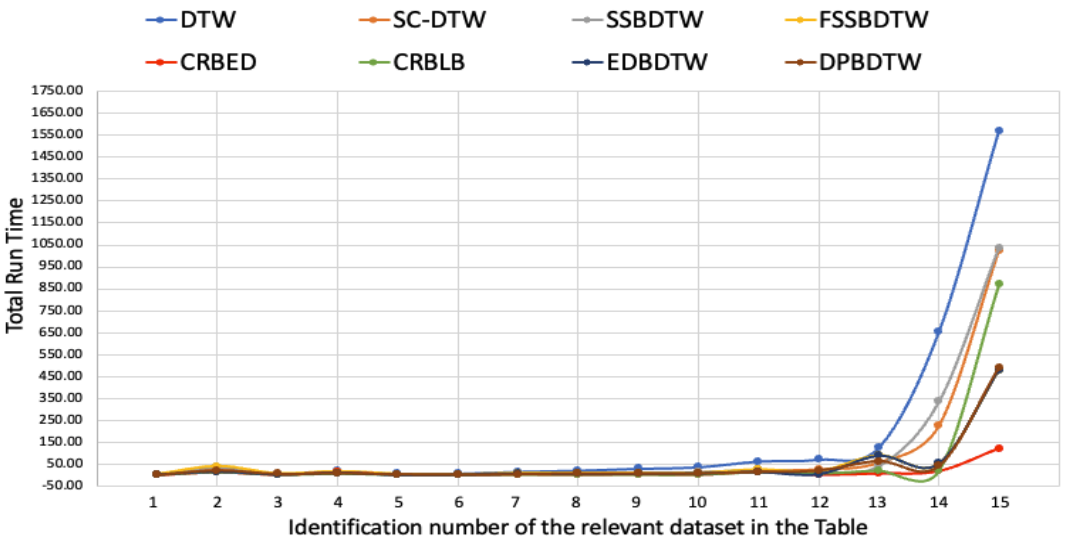

FIGURE 7.1: Comparison of the proposed approaches in term of run time for the fifteen data sets considered.

TABLE 7.2: Accuracy results for the fifteen time series data sets using the eight approaches presented in this thesis, best results are highlighted in bold.

ID.	Dataset	DTW	SC-DTW	SSBDTW	FSSBDTW	CRBED	CRBLB	EDBDTW	DPBDTW
1.	Smooth	91.00 (0.04)	95.00 (0.02)	**98.67** (**0.03**)	**98.67** (**0.03**)	**98.67** (**0.03**)	96.97 (0.03)	95.00 (0.02)	43.00 (0.07)
2.	ItalyPower	95.61 (0.02)	95.70 (0.02)	96.34 (0.02)	**97.17** (**0.01**)	**97.17** (**0.01**)	95.98 (0.02)	96.35 (0.02)	95.16 (0.01)
3.	Libras	58.89 (0.10)	63.06 (0.10)	62.78 (0.11)	**65.83** (**0.11**)	**65.83** (**0.11**)	59.44 (0.09)	64.17 (0.12)	65.83 (0.11)
4.	Synthetic	**98.50** (**0.01**)	**98.50** (**0.01**)	98.33 (0.01)	**98.50** (**0.01**)	**98.50** (**0.01**)	**98.17** (**0.02**)	95.17 (0.03)	**89.17** (**0.03**)
5.	GunPoint	94.50 (0.05)	97.50 (0.03)	**99.47** (**0.02**)	**99.50** (**0.01**)	**99.50** (**0.01**)	95.50 (0.04)	98.00 (0.02)	98.00 (0.02)
6.	OliveOil	86.67 (0.15)	86.67 (0.15)	88.33 (0.13)	90.00 (0.11)	**90.11** (**0.13**)	88.33 (0.011)	90.11 (0.11)	90.00 (0.08)
7.	Trace	99.00 (0.03)	99.00 (0.03)	97.50 (0.03)	99.00 (0.03)	99.00 (0.03)	99.00 (0.03)	99.00 (0.02)	**100.00** (**0.00**)
8.	ToeSegment	88.46 (0.09)	92.68 (0.07)	92.75 (0.06)	**93.42** (**0.04**)	92.72 (0.04)	88.49 (0.06)	92.21 (0.04)	88.53 (0.06)
9.	Car	80.83 (0.07)	81.67 (0.07)	83.33 (0.10)	86.67 (0.06)	**88.33** (**0.06**)	82.50 (0.09)	83.33 (0.06)	83.33 (0.09)
10.	Lighting2	87.76 (0.09)	87.76 (0.09)	83.30 (0.06)	**89.23** (**0.06**)	87.76 (0.07)	87.63 (0.07)	89.17 (0.08)	75.26 (0.13)
11.	Shapelet	82.00 (0.10)	82.00 (0.10)	89.97 (0.06)	93.00 (0.04)	89.50 (0.06)	89.50 (0.07)	87.00 (0.07)	**99.50** (**0.01**)
12.	DiatomSize	99.38 (0.01)	99.69 (0.01)	**100.00** (**0.00**)	**100.00** (**0.00**)	**100.00** (**0.00**)	**100.00** (**0.00**)	**100.00** (**0.00**)	**100.00** (**0.00**)
13.	Adiac	65.30 (0.04)	65.30 (0.04)	65.51 (0.04)	**68.12** (**0.04**)	65.81 (0.03)	66.58 (0.03)	65.81 (0.03)	53.53 (0.06)
14.	HouseTwenty	95.00 (0.05)	93.00 (0.08)	93.71 (0.06)	95.00 (0.05)	93.71 (0.04)	93.04 (0.05)	**96.25** (**0.05**)	94.29 (0.06)
15.	PenDigits	85.47 (0.01)	87.62 (0.01)	88.46 (0.01)	**89.64** (**0.01**)	**89.64** (**0.01**)	85.00 (0.01)	**89.64** (**0.01**)	88.46 (0.01)
	Ave. Acc.	87.89 (0.06)	88.24 (0.06)	89.39 (0.05)	90.64 (0.04)	**91.39** (**0.04**)	89.06 (0.05)	90.09 (0.09)	88.53 (0.05)

Considering Table 7.2, CRBED coupled with FSSBDTW produced the best result in eight of the fifteen cases. This is echoed in Table 7.3 where CRBED coupled with FSSBDTW produced the best F1 score in eight of the fifteen cases. Looking at the average accuracy and F1 score in each case, the last row in Tables 7.2 and 7.3, it can be confirmed that CRBED coupled with FSSBDTW obtained the best results. Thus, overall it can be concluded, from the comparison, that the CRBED approach is both the most effective and efficient approach of the six approaches proposed in this thesis and the two benchmark approaches.

TABLE 7.3: F1-Score results for the fifteen time series data sets using the eight approaches presented in this thesis, best results are highlighted in bold.

ID.	Dataset	DTW	SC-DTW	SSBDTW	FSSBDTW	CRBED	CRBLB	EDBDTW	DPBDTW
1.	Smooth	0.91 (0.04)	0.95 (0.02)	**0.98** (**0.03**)	**0.98** (**0.03**)	**0.98** (**0.03**)	0.96 (0.03)	0.95 (0.02)	0.43 (0.07)
2.	ItalyPower	0.95 (0.02)	0.95 (0.02)	0.96 (0.02)	**0.97** (**0.01**)	**0.97** (**0.01**)	0.95 (0.02)	0.96 (0.02)	0.95 (0.01)
3.	Libras	0.60 (0.011)	0.60 (0.11)	0.62 (0.011)	0.64 (0.11)	**0.65** (**0.11**)	0.59 (0.09)	0.64 (0.12)	**0.65** (**0.11**)
4.	Synthetic	**0.98** (**0.01**)	**0.98** (**0.01**)	**0.98** (**0.01**)	**0.98** (**0.01**)	**0.98** (**0.01**)	**0.98** (**0.02**)	0.95 (0.03)	0.89 (0.03)
5.	GunPoint	0.94 (0.05)	0.97 (0.03)	**0.99** (**0.02**)	**0.99** (**0.01**)	**0.99** (**0.01**)	0.95 (0.04)	0.98 (0.02)	0.98 (0.02)
6.	OliveOil	0.86 (0.16)	0.86 (0.16)	0.88 (0.13)	0.90 (0.11)	**0.90** (**0.11**)	0.88 (0.11)	**0.90** (**0.11**)	**0.90** (**0.08**)
7.	Trace	0.99 (0.03)	0.99 (0.03)	0.97 (0.03)	0.99 (0.03)	0.99 (0.03)	0.99 (0.03)	0.99 (0.02)	**1.00** (**0.00**)
8.	ToeSegment	0.88 (0.10)	0.92 (0.07)	0.92 (0.06)	**0.93** (**0.04**)	0.92 (0.04)	0.88 (0.06)	0.92 (0.04)	0.88 (0.06)
9.	Car	0.80 (0.09)	0.81 (0.08)	0.82 (0.11)	**0.86** (**0.06**)	**0.86** (**0.06**)	0.82 (0.09)	0.83 (0.06)	0.83 (0.09)
10.	Lighting2	0.87 (0.08)	0.87 (0.08)	0.83 (0.06)	**0.89** (**0.06**)	0.87 (0.07)	0.87 (0.07)	**0.89** (**0.08**)	0.75 (0.13)
11.	Shapelet	0.81 (0.011)	0.81 (0.011)	0.89 (0.06)	0.93 (0.04)	0.89 (0.08)	0.89 (0.07)	0.87 (0.07)	**0.99** (**0.01**)
12.	DiatomSize	0.99 (0.01)	0.99 (0.01)	**1.00** (**0.00**)	**1.00** (**0.00**)	**1.00** (**0.00**)	**1.00** (**0.00**)	**1.00** (**0.00**)	**1.00** (**0.00**)
13.	Adiac	0.62 (0.04)	0.62 (0.04)	0.62 (0.04)	**0.66** (**0.04**)	0.64 (0.03)	0.64 (0.03)	0.64 (0.03)	0.53 (0.06)
14.	HouseTwenty	0.95 (0.05)	0.93 (0.08)	0.93 (0.06)	0.95 (0.05)	0.93 (0.04)	0.93 (0.05)	**0.96** (**0.05**)	0.94 (0.06)
15.	PenDigits	0.85 (0.01)	0.87 (0.01)	0.88 (0.01)	**0.89** (**0.01**)	**0.89** (**0.01**)	0.85 (0.01)	**0.89** (**0.01**)	0.88 (0.01)
	Ave. F1.	00.87 (0.06)	00.88 (0.06)	00.89 (0.05)	00.90 (0.04)	**00.91** (**0.04**)	00.89 (0.05)	00.90 (0.09)	00.88 (0.05)

7.1.2 Main finding and Contributions

In this section, the main findings and contributions, arising from the work presented, are considered in terms of the primary and subsidiary research questions. For reference, the primary research question was as follows:

"How can the process of dynamic time warping be applied so that time series can be more effectively and efficiently compared?"

The responses to the subsidiary research questions will be considered first and then the primary research question will be returned to. Thus:

1. *What are the most appropriate mechanisms, in addition to those described in the literature, for limiting the size of the matrix data that needs to be stored?*

   This thesis has suggested several mechanisms whereby the size of the distance matrix, on which DTW is found, can be reduced. As noted in Chapter 1, the length x of any two time series to be compared dictates the size of the distance matrix M ($|M| = x \times x = x^2$). To reduce the size of M the idea of segmenting the input time series and applying DTW to corresponding segments, the SSBDTW and FSSBDTW approaches from Chapter 4, was first proposed. Secondly, the idea of reducing the size of x using discriminators or distance profiles was proposed in Chapter 6. Further advantages could be gained by combing the later with SSBDTW or FSSBDTW. The advantages are illustrated in Figure 7.2. The top left sub-figure shows the search space for standard DTW, the top right sub-figure the search space when segmentation is applied, the bottom left the effect of using discriminators or distance profiles, and the bottom right the effect of combing the two. From the figure, it is clear that all three mechanisms served to reduce the size of the DTW distance matrix. By referring back to the average run times recorded in Table 7.1 it can be seen that the three mechanisms seemed to speed up the

FIGURE 7.2: Comparison of the proposed approaches to reduce the overall size of the DTW distance matrix

DTW process by a factor of three to four times compared to Standard DTW and twice when comparing with S-C Band DTW.

2. *Is there any advantage to be gained from splitting the distance matrix in term of efficiency?*

   The work presented in Chapter 4 clearly demonstrated that segmenting (splitting) the time series offered clear advantages with respect to run time, accuracy and F-score (see Tables 7.1, 7.2 and 7.3). However, in Chapter 4 it was also demonstrated that splitting the time series according to some fixed, pre-specified, time series subsequence length (the SSBDTW approach) did not work as well as when a variable segment length was used (the FSSBDTW approach).

3. *Is it possible to utilise knowledge of a given application domain to limit the DTW processing required. For example, in the case of kNN, using early abandonment with respect to time series that are clearly not going to be very similar to the query time series?*

   The fundamental idea investigated, using kNN classification, was whether it was possible to "throw away" some of the time series to be considered given a new, previously unseen, time series. In other words, whether *candidate reduction* could be successfully applied. Two mechanisms were investigated whereby the kNN bank D could be preprocessed and a subset, D', retained. The first used Euclidean distance comparison (the CRBED mechanisms) and the second lower bounding (the CRBLB mechanism). Both were combined with FSSBDTW. The results, summarized in Table 7.1, clearly demonstrated that efficiency gains could be made. Further, it was demonstrated that efficiency gains could be made without adversely affecting accuracy or F1 score (see Tables 7.2 and 7.3).

4. *Is it possible to only consider sub-sequences that are repeated in instances (records) of the same class, so that DTW can be applied only to such sub-sequences instead of the entire time series?*

   To answer the above subsidiary question, a new approach, the EDBDTW approach was proposed and investigated. The idea was to transform the time series in the input data set D to a new form $D\prime$ which featured shorter time series, compared to the original time series in D, by finding the most similar sub-sequences (discriminators). In other words, the discriminator will be defined by the location within the set of time series associated with a class where the time series are most similar; this is then the location where DTW can best be applied to obtain a good classification. The results obtained indicated that this was indeed the case although an open question remained as to whether the advantages gained were more to do with the use of FSSBDTW than the use of discriminators (this is discussed further in the following section).

5. *Is it possible to identify mechanisms for transforming time series from their original form and apply DTW to this new form?*

   This was investigated by considering the use of distance profiles, transforming the time series in D into a distance profile format and applying DTW to this format (FSSBDTW in the case of the evaluation presented in this thesis). The result was the DPBDTW approach. The reported evaluation indicated that some advantages could be gained; but, as in the case of the EDBDTW approach it was unclear how much of the advantage gained could be attributed to the use of FSSBDTW, and how much to the use of the distance profile idea (this is also discussed further in the following section).

6. *Assuming that the answers to the above entail the use of parameters of various kinds, how can these parameters be optimised?*

   The various approaches proposed and investigated to address the above subsidiary questions did indeed entail the use of a range of parameters and that there was no "one size fits all" values for these parameters. A mechanism was proposed whereby the best parameter settings could be learnt using comparatively inexpensive Euclidean distance time series comparison. This seemed to work well.

Returning to the primary research question. From the foregoing, several approaches were proposed: (i) Sub-Sequence-Based DTW (SSBDTW), (ii) Fuzzy Sub-Sequence-Based DTW approach (FSSBDTW), (iii) Candidate Reduction Based on Euclidean Distance (CRBED), (iv) Candidate Reduction Based on Lower Bounding (CRBLB), (v) Exact Discriminator-Based DTW (EDBDTW) and (vi) Distance Profile-Based DTW (DPBDTW). The experiments and the analyses conducted on those approaches demonstrated that time series can be compared effectively and efficiently using Dynamic time Warping (DTW). Significant performance gains, in terms of run time, accuracy and F1-score, were obtained using these approaches. The best performing approach was the Candidate Reduction Based on Euclidean Distance (CRBED).

The main generic benefits of the work presented in this thesis can be itemized as follows:

1. Reducing the time complexity of the Dynamic Time Warping (DTW) process, so that run time is significantly reduced compared to Standard DTW and S-C Band DTW.

2. Efficiency gains resulting from the use of the proposed mechanisms. Efficiency gains that seemed to be independent of time series length; good results were obtained with respect to both short and long time series. It was also found that the efficiency gains become more pronounced when considering longer time series. From Figure 7.1, it can be seen that for largest data sets (*HouseTwenty* and *PenDigits*) the proposed approaches performed much better than for the smaller data sets.

3. Enhancement in the accuracy and F1-score with respect to both long and short time series data sets was not adversely affected; in some cases, the accuracy and F1 score values obtained exceed those obtained using Standard DTW and S-C Band DTW (an accuracy of 100% was recorded in a few cases).

4. Inexpensive mechanisms for parameter learning using Euclidean distance comparison prior to the application of DTW.

5. Flexibility in the application of the proposed approaches, so that they can be joined with each other as well as with alternative approaches proposed in the literature.

7.1.3 Future Work

The work presented in this thesis has proposed several approaches for speeding up the DTW process. The proposed approaches have demonstrated better performance compared to Standard DTW and SC-Band DTW. The work presented in this thesis has faced some limitations that prevented investigation with respect to very long time series collections, because of the COVID-19 pandemic, which limited access to high performance computing facilities. This investigation has been left as an item for future work. There are also some other related areas that merit further investigation. This concluding sub-section, therefore, itemises a number of potential directions for future work as follows:

1. **Best value for s:** The SSBDTW approach presented in Chapter 4 used a parameter s, the number of segments into which a time series should be divided. It was suggested in Chapter 4 that the best value for s could be learnt in a similar manner to the way that parameters were learnt with respect to the approaches presented in Chapter 5 and 6. It is suggested here that this would be an interesting avenue for future work.

2. **Further investigation using discriminators**. The work presented in Chapter 6 raised the question as to whether the advantages gained from using discriminators might be as a consequence of the use of FSSBDTW rather than the use of the discriminators themselves. This is thus the first suggested avenue for future work.

3. **Further investigation using distance profiles**. The work presented in Chapter 6 also raised the question as to whether the advantages gained from transforming the input data into distance profiles might again be more as a consequence of the use of FSSBDTW than the use of distance profiles. This is thus the third suggested avenue for future work.

4. **Use of Approximate discriminators:** The work in Chapter 6 was directed at finding exact discriminators. Further efficiency gains might be realized if approximate discriminators were found.

5. **Integrating the proposed approaches with each other:** Form the work presented in this thesis, several approaches were introduced. Experiments were conducted that combined CRBED and CRBLB with FSSBDTW, and that combined the use of discriminators and distance profiles with FSSBDTW, but experiments were not conducted combing the mechanisms for reducing the number of time series to be considered (CRBED and CRBLB) with the use of discriminators and distance profiles. It is conjectured that this may provide further efficiency and effectiveness gains and therefore merits further investigation.

6. **Integrating the proposed approaches with alternatives from the literature:** Various alternative mechanisms have been proposed in the literature to reduce the DTW overhead. Of note, is the work on motifs and discords [101, 129]. Other examples are the Itakura parallelogram [55], learning constraints as in [81] or alternative "lower bounding" methods as in [61, 68, 80, 130]. Therefore, a further potential avenue for future work is how these previously proposed mechanisms could be integrated with the approaches presented in this thesis to realise further gains.

7. **Long time series:** We have seen that the complexity of DTW is dictated by the length of the time series to be considered. It was also noted above that the COVID-19 pandemic precluded the investigation of the application of the proposed techniques to long time series. This is thus another potentially fruitful avenue for further work.

8. **Complexity Calculation:** Another possible direction for future work is a more extensive analysis of the time complexity of the proposed approaches. Complexity analysis using "Big O" notation was presented with respect to all the approaches considered. In Chapters 3 and 4 actual complexity equations were derived using a constant z. For the approaches considered in Chapters 5 and 6 an actual complexity equation was not derived because of the challenge of finding values for two constants z_1 and z_2. This is then the final avenue for future work proposed here.

Bibliography

[1] Amaia Abanda, Usue Mori, and Jose A Lozano, *A review on distance based time series classification*, Data Mining and Knowledge Discovery **33** (2019), no. 2, 378–412.

[2] Shaymaa Adnan Abdulrahman, Wael Khalifa, Mohamed Roushdy, and Abdel-Badeeh M Salem, *Comparative study for 8 computational intelligence algorithms for human identification*, Computer Science Review **36** (2020), 100237.

[3] Avinash Achar and Pål Sætrom, *Rna motif discovery: a computational overview*, Biology direct **10** (2015), no. 1, 1–22.

[4] Sara Alaee, Kaveh Kamgar, and Eamonn Keogh, *Matrix profile xxii: Exact discovery of time series motifs under dtw*, 2020 IEEE International Conference on Data Mining (ICDM), IEEE, 2020, pp. 900–905.

[5] Sara Alaee, Ryan Mercer, Kaveh Kamgar, and Eamonn Keogh, *Time series motifs discovery under dtw allows more robust discovery of conserved structure*, Data Mining and Knowledge Discovery **35** (2021), no. 3, 863–910.

[6] Hajar Alhijailan and Frans Coenen, *Effective frequent motif discovery for long time series classification: A study using phonocardiogram*, Proceedings of the 11th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management, SCITEPRESS-Science and Technology Publications, 2019.

[7] Mohammed Ali, Ali Alqahtani, Mark W Jones, and Xianghua Xie, *Clustering and classification for time series data in visual analytics: A survey*, IEEE Access **7** (2019), 181314–181338.

[8] Abdullah Alshehri, *Keyboard usage recognition: A study in pattern mining and prediction in the context of impersonation*, Ph.D. thesis, University of Liverpool, 2018.

[9] Mohammed Alshehri, Frans Coenen, and Keith Dures, *Motif-based classification using enhanced sub-sequence-based dynamic time warping.*

[10] ———, *Effective sub-sequence-based dynamic time warping*, International Conference on Innovative Techniques and Applications of Artificial Intelligence, Springer, 2019, pp. 293–305.

[11] ———, *Sub-sequence-based dynamic time warping.*, KDIR, 2019, pp. 274–281.

[12] ———, *Candidates reduction and enhanced sub-sequence-based dynamic time warping: A hybrid approach*, International Conference on Innovative Techniques and Applications of Artificial Intelligence, Springer, 2020, pp. 273–285.

[13] Anthony Bagnall and Jason Lines, *An experimental evaluation of nearest neighbour time series classification*, arXiv preprint arXiv:1406.4757 (2014).

[14] Anthony Bagnall, Jason Lines, Aaron Bostrom, James Large, and Eamonn Keogh, *The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances*, Data Mining and Knowledge Discovery **31** (2017), no. 3, 606–660.

[15] Anthony J Bagnall and Gareth J Janacek, *Clustering time series from arma models with clipped data*, Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2004, pp. 49–58.

[16] Mridula Batra and Rashmi Agrawal, *Comparative analysis of decision tree algorithms*, Nature inspired computing, Springer, 2018, pp. 31–36.

[17] Mariana Belgiu and Ovidiu Csillik, *Sentinel-2 cropland mapping using pixel-based and object-based time-weighted dynamic time warping analysis*, Remote sensing of environment **204** (2018), 509–523.

[18] Daniel Berrar, *Cross-validation.*, 2019.

[19] Himani Bhavsar and Mahesh H Panchal, *A review on support vector machine for data classification*, International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) **1** (2012), no. 10, 185–189.

[20] Ane Blázquez-García, Angel Conde, Usue Mori, and Jose A Lozano, *A review on outlier/anomaly detection in time series data*, ACM Computing Surveys (CSUR) **54** (2021), no. 3, 1–33.

[21] Andrea Brunello, Enrico Marzano, Angelo Montanari, and Guido Sciavicco, *A novel decision tree approach for the handling of time series*, International Conference on Mining Intelligence and Knowledge Exploration, Springer, 2018, pp. 351–368.

[22] Jimmy Byakatonda, BP Parida, Piet K Kenabatho, and DB Moalafhi, *Analysis of rainfall and temperature time series to detect long-term climatic trends and variability over semi-arid botswana*, Journal of Earth System Science **127** (2018), no. 2, 25.

[23] Eoin Cartwright, Martin Crane, and Heather J Ruskin, *Financial time series: Motif discovery and analysis using valmod*, International Conference on Computational Science, Springer, 2019, pp. 771–778.

[24] Carmelo Cassisi, Marco Aliotta, Andrea Cannata, Placido Montalto, Domenico Patanè, Alfredo Pulvirenti, and Letizia Spampinato, *Motif discovery on seismic amplitude time series: The case study of mt etna 2011 eruptive activity*, Pure and Applied Geophysics **170** (2013), no. 4, 529–545.

[25] Jair Cervantes, Farid Garcia-Lamont, Lisbeth Rodríguez-Mazahua, and Asdrubal Lopez, *A comprehensive survey on support vector machine classification: Applications, challenges and trends*, Neurocomputing **408** (2020), 189–215.

[26] Dar-Jen Chang, Ahmed H Desoky, Ming Ouyang, and Eric C Rouchka, *Compute pairwise manhattan distance and pearson correlation coefficient of data points with*

gpu, 2009 10th ACIS International Conference on Software Engineering, Artificial Intelligences, Networking and Parallel/Distributed Computing, IEEE, 2009, pp. 501–506.

[27] Mu-Yen Chen and Bo-Tsuen Chen, *A hybrid fuzzy time series model based on granular computing for stock price forecasting*, Information Sciences **294** (2015), 227–241.

[28] Padraig Cunningham and Sarah Jane Delany, *k-nearest neighbour classifiers: (with python examples)*, arXiv preprint arXiv:2004.04523 (2020).

[29] Hoang Anh Dau, Anthony Bagnall, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, and Eamonn Keogh, *The ucr time series archive*, IEEE/CAA Journal of Automatica Sinica **6** (2019), no. 6, 1293–1305.

[30] Hoang Anh Dau, Diego Furtado Silva, François Petitjean, Germain Forestier, Anthony Bagnall, and Eamonn Keogh, *Judicious setting of dynamic time warping's window width allows more accurate classification of time series*, 2017 IEEE international conference on big data (big data), IEEE, 2017, pp. 917–922.

[31] Hoang Anh Dau, Diego Furtado Silva, Francois Petitjean, Germain Forestier, Anthony Bagnall, Abdullah Mueen, and Eamonn Keogh, *Optimizing dynamic time warping's window width for time series data mining applications*, Data mining and knowledge discovery **32** (2018), no. 4, 1074–1120.

[32] James J Deng and Clement HC Leung, *Dynamic time warping for music retrieval using time series modeling of musical emotions*, IEEE Transactions on Affective Computing **6** (2015), no. 2, 137–151.

[33] Xinyang Deng, Qi Liu, Yong Deng, and Sankaran Mahadevan, *An improved method to construct basic probability assignment based on the confusion matrix for classification problem*, Information Sciences **340** (2016), 250–261.

[34] Hui Ding, Goce Trajcevski, Peter Scheuermann, Xiaoyue Wang, and Eamonn Keogh, *Querying and mining of time series data: experimental comparison of representations and distance measures*, Proceedings of the VLDB Endowment **1** (2008), no. 2, 1542–1552.

[35] Philippe Esling and Carlos Agon, *Time-series data mining*, ACM Computing Surveys (CSUR) **45** (2012), no. 1, 1–34.

[36] Zakarya Farou and Krisztian Buza, *The warping window size effects the accuracy of person identification based on keystroke dynamics.*, ITAT, 2019, pp. 84–89.

[37] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller, *Deep learning for time series classification: a review*, Data Mining and Knowledge Discovery **33** (2019), no. 4, 917–963.

[38] Tom Fawcett, *An introduction to roc analysis*, Pattern recognition letters **27** (2006), no. 8, 861–874.

[39] Thomas Fischer and Christopher Krauss, *Deep learning with long short-term memory networks for financial market predictions*, European Journal of Operational Research **270** (2018), no. 2, 654–669.

[40] Duarte Folgado, Marília Barandas, Ricardo Matias, Rodrigo Martins, Miguel Carvalho, and Hugo Gamboa, *Time alignment measurement for time series*, Pattern Recognition **81** (2018), 268–279.

[41] Tak-chung Fu, *A review on time series data mining*, Engineering Applications of Artificial Intelligence **24** (2011), no. 1, 164–181.

[42] Zoltan Geler, Vladimir Kurbalija, Mirjana Ivanović, and Miloš Radovanović, *Weighted knn and constrained elastic distances for time-series classification*, Expert Systems with Applications **162** (2020), 113829.

[43] Zoltan Geler, Vladimir Kurbalija, Mirjana Ivanović, Miloš Radovanović, and Weihui Dai, *Dynamic time warping: Itakura vs sakoe-chiba*, 2019 IEEE International Symposium on INnovations in Intelligent SysTems and Applications (INISTA), IEEE, 2019, pp. 1–6.

[44] Tomasz Górecki and Maciej Łuczak, *The influence of the sakoe–chiba band size on time series classification*, Journal of Intelligent & Fuzzy Systems **36** (2019), no. 1, 527–539.

[45] Ashish Gupta, Hari Prabhat Gupta, Bhaskar Biswas, and Tanima Dutta, *Approaches and applications of early classification of time series: A review*, IEEE Transactions on Artificial Intelligence (2020).

[46] Sepand Haghighi, Masoomeh Jasemi, Shaahin Hessabi, and Alireza Zolanvari, *Pycm: Multiclass confusion matrix library in python*, Journal of Open Source Software **3** (2018), no. 25, 729.

[47] Fatma A Hashim, Mai S Mabrouk, and Walid Al-Atabany, *Review of different sequence motif finding algorithms*, Avicenna journal of medical biotechnology **11** (2019), no. 2, 130.

[48] Zhenwen He, Chonglong Wu, Gang Liu, Yiping Tian, Xialin Zhang, and Qiyu Chen, *Review on geoscience time series big data similarity measurement and index method*, **39** (2020), no. 4, 44–50.

[49] K Hebbrecht, M Stuivenga, Tom Birkenhäger, M Morrens, EI Fried, B Sabbe, and EJ Giltay, *Understanding personalized dynamics to inform precision medicine: a dynamic time warp analysis of 255 depressed inpatients*, BMC medicine **18** (2020), no. 1, 1–15.

[50] Zhu Hejun and Zhu Liehuang, *Encrypted network behaviors identification based on dynamic time warping and k-nearest neighbor*, Cluster Computing **22** (2019), no. 2, 2571–2580.

[51] Matthieu Herrmann and Geoffrey I Webb, *Early abandoning and pruning for elastic distances*, arXiv preprint arXiv:2102.05221 (2021).

[52] Yuxiu Hua, Zhifeng Zhao, Rongpeng Li, Xianfu Chen, Zhiming Liu, and Honggang Zhang, *Deep learning with long short-term memory for time series prediction*, IEEE Communications Magazine **57** (2019), no. 6, 114–119.

[53] Shujun Huang, Nianguang Cai, Pedro Penzuti Pacheco, Shavira Narrandes, Yang Wang, and Wayne Xu, *Applications of support vector machine (svm) learning in cancer genomics*, Cancer genomics & proteomics **15** (2018), no. 1, 41–51.

[54] MA Ismail and Samia Gad, *Off-line arabic signature recognition and verification*, Pattern Recognition **33** (2000), no. 10, 1727–1740.

[55] Fumitada Itakura, *Minimum prediction residual principle applied to speech recognition*, IEEE Transactions on Acoustics, Speech, and Signal Processing **23** (1975), no. 1, 67–72.

[56] Ali Jazayeri and Christopher C Yang, *Motif discovery algorithms in static and temporal networks: A survey*, Journal of Complex Networks **8** (2020), no. 4, cnaa031.

[57] Houtan Jebelli, Mohammad Mahdi Khalili, Sungjoo Hwang, and S Lee, *A supervised learning-based construction workers' stress recognition using a wearable electroencephalography (eeg) device*, Construction research congress, vol. 2018, 2018, pp. 43–53.

[58] Eamonn Keogh and Shruti Kasetty, *On the need for time series data mining benchmarks: a survey and empirical demonstration*, Data Mining and knowledge discovery **7** (2003), no. 4, 349–371.

[59] Eamonn Keogh and Chotirat Ann Ratanamahatana, *Exact indexing of dynamic time warping*, Knowledge and information systems **7** (2005), no. 3, 358–386.

[60] Amin Keshavarzi, Abolfazl Toroghi Haghighat, and Mahdi Bohlouli, *Enhanced time-aware qos prediction in multi-cloud: a hybrid k-medoids and lazy learning approach (qopc)*, Computing **102** (2020), no. 4, 923–949.

[61] Sang-Wook Kim, Sanghyun Park, and Wesley W Chu, *An index-based approach for similarity search supporting time warping in large sequence databases*, Data Engineering, 2001. Proceedings. 17th International Conference on, IEEE, 2001, pp. 607–614.

[62] Ron Kohavi et al., *A study of cross-validation and bootstrap for accuracy estimation and model selection*, Ijcai, vol. 14, Montreal, Canada, 1995, pp. 1137–1145.

[63] Georgios N Kouziokas, *Svm kernel based on particle swarm optimized vector and bayesian optimized svm in atmospheric particulate matter forecasting*, Applied Soft Computing **93** (2020), 106410.

[64] Hardy Kremer, Stephan Günnemann, Anca-Maria Ivanescu, Ira Assent, and Thomas Seidl, *Efficient processing of multiple dtw queries in time series databases*, International Conference on Scientific and Statistical Database Management, Springer, 2011, pp. 150–167.

[65] Neha Kulkarni, *Effect of dynamic time warping using different distance measures on time series classification*, International Journal of Computer Applications **975** (2017), 8887.

[66] Roshan Kumari and Saurabh Kr Srivastava, *Machine learning: A review on binary classification*, International Journal of Computer Applications **160** (2017), no. 7.

[67] Brecht Laperre, Jorge Amaya, and Giovanni Lapenta, *Dynamic time warping as a new evaluation for dst forecast with machine learning*, arXiv preprint arXiv:2006.04667 (2020).

[68] Daniel Lemire, *Faster retrieval with a two-pass dynamic-time-warping lower bound*, Pattern recognition **42** (2009), no. 9, 2169–2180.

[69] Hailin Li and Cheng Wang, *Similarity measure based on incremental warping window for time series data mining*, IEEE Access **7** (2018), 3909–3917.

[70] Xiaosheng Li and Jessica Lin, *Linear time motif discovery in time series*, Proceedings of the 2019 SIAM International Conference on Data Mining, SIAM, 2019, pp. 136–144.

[71] Zheng-xin Li, Shi-hui Wu, Yu Zhou, and Chao Li, *A combined filtering search for dtw*, 2017 2nd International Conference on Image, Vision and Computing (ICIVC), IEEE, 2017, pp. 884–888.

[72] Zhengxin Li, Jiansheng Guo, Hailin Li, Tao Wu, Sheng Mao, and Feiping Nie, *Speed up similarity search of time series under dynamic time warping*, IEEE Access **7** (2019), 163644–163653.

[73] Michele Linardi, Yan Zhu, Themis Palpanas, and Eamonn Keogh, *Matrix profile goes mad: variable-length motif and discord discovery in data series*, Data Mining and Knowledge Discovery **34** (2020), no. 4, 1022–1071.

[74] Jason Lines, Luke M Davis, Jon Hills, and Anthony Bagnall, *A shapelet transform for time series classification*, Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining, 2012, pp. 289–297.

[75] Qian Liu, Carson K Leung, and Pingzhao Hu, *A two-dimensional sparse matrix profile densenet for covid-19 diagnosis using chest ct images*, IEEE Access **8** (2020), 213718–213728.

[76] Vinh-Trung Luu, Germain Forestier, Jonathan Weber, Paul Bourgeois, Fahima Djelil, and Pierre-Alain Muller, *A review of alignment based similarity measures for web usage mining*, Artificial Intelligence Review **53** (2020), no. 3, 1529–1551.

[77] Ruizhe Ma, Azim Ahmadzadeh, Soukaina Filali Boubrahimi, and Rafal A Angryk, *Segmentation of time series in improving dynamic time warping*, 2018 IEEE International Conference on Big Data (Big Data), IEEE, 2018, pp. 3756–3761.

[78] Fabian Moerchen, Ingo Mierswa, and Alfred Ultsch, *Understandable models of music collections based on exhaustive feature generation with temporal statistics*, Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, 2006, pp. 882–891.

[79] Meinard Müller, *Dynamic time warping*, Information retrieval for music and motion (2007), 69–84.

[80] Happy Nath and Ujwala Baruah, *Evaluation of lower bounding methods of dynamic time warping (dtw)*, Evaluation **94** (2014), no. 20.

[81] Vit Niennattrakul and Chotirat Ann Ratanamahatana, *Learning dtw global constraint for time series classification*, arXiv preprint arXiv:0903.0041 (2009).

[82] Rubén Nogales and Marco Benalcázar, *Real-time hand gesture recognition using knn-dtw and leap motion controller*, Conference on Information and Communication Technologies of Ecuador, Springer, 2020, pp. 91–103.

[83] Kuldip K Paliwal, Anant Agarwal, and Sarvajit S Sinha, *A modification over sakoe and chiba's dynamic time warping algorithm for isolated word recognition*, Signal Processing **4** (1982), no. 4, 329–333.

[84] Harsh H Patel and Purvi Prajapati, *Study and analysis of decision tree based classification algorithms*, International Journal of Computer Sciences and Engineering **6** (2018), no. 10, 74–78.

[85] Angkoon Phinyomark and Erik Scheme, *An investigation of temporally inspired time domain features for electromyographic pattern recognition*, 2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), IEEE, 2018, pp. 5236–5240.

[86] Lawrence R Rabiner and Biing-Hwang Juang, *Fundamentals of speech recognition*, vol. 14, PTR Prentice Hall Englewood Cliffs, 1993.

[87] Thanawin Rakthanmanon, Bilson Campana, Abdullah Mueen, Gustavo Batista, Brandon Westover, Qiang Zhu, Jesin Zakaria, and Eamonn Keogh, *Searching and mining trillions of time series subsequences under dynamic time warping*, Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2012, pp. 262–270.

[88] KM Rashid and J Louis, *Window-warping: A time series data augmentation of imu data for construction equipment activity identification*, ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction, vol. 36, IAARC Publications, 2019, pp. 651–657.

[89] David R Roberts, Volker Bahn, Simone Ciuti, Mark S Boyce, Jane Elith, Gurutzeta Guillera-Arroita, Severin Hauenstein, José J Lahoz-Monfort, Boris Schröder, Wilfried Thuiller, et al., *Cross-validation strategies for data with temporal, spatial, hierarchical, or phylogenetic structure*, Ecography **40** (2017), no. 8, 913–929.

[90] Salla Ruuska, Wilhelmiina Hämäläinen, Sari Kajava, Mikaela Mughal, Pekka Matilainen, and Jaakko Mononen, *Evaluation of the confusion matrix method in the validation of an automated system for measuring feeding behaviour of cattle*, Behavioural processes **148** (2018), 56–62.

[91] Giovanni Saggio, Pietro Cavallo, Mariachiara Ricci, Vito Errico, Jonathan Zea, and Marco E Benalcázar, *Sign language recognition using wearable electronics: implementing k-nearest neighbors with dynamic time warping and convolutional neural network algorithms*, Sensors **20** (2020), no. 14, 3879.

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

[93] Sonali T Saste and SM Jagdale, *Comparative study of different techniques in speaker recognition*, International Journal of Advanced Engineering, Management and Science **3** (2017), no. 3, 239807.

[94] Pratap Chandra Sen, Mahimarnab Hajra, and Mitadru Ghosh, *Supervised classification algorithms in machine learning: A survey and review*, Emerging technology in modelling and graphics, Springer, 2020, pp. 99–111.

[95] Pavel Senin, *Dynamic time warping algorithm review*, Information and Computer Science Department University of Hawaii at Manoa Honolulu, USA **855** (2008), 1–23.

[96] Ravi Shankar and Archana Venkataraman, *A deep-bayesian framework for adaptive speech duration modification*, arXiv preprint arXiv:2107.04973 (2021).

[97] Alex Sherstinsky, *Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network*, Physica D: Nonlinear Phenomena **404** (2020), 132306.

[98] Ahmed Shifaz, Charlotte Pelletier, Francois Petitjean, and Geoffrey I Webb, *Elastic similarity measures for multivariate time series classification*, arXiv preprint arXiv:2102.10231 (2021).

[99] Diego F Silva and Gustavo EAPA Batista, *Speeding up all-pairwise dynamic time warping matrix calculation*, Proceedings of the 2016 SIAM International Conference on Data Mining, SIAM, 2016, pp. 837–845.

[100] Diego F Silva, Rafael Giusti, Eamonn Keogh, and Gustavo EAPA Batista, *Speeding up similarity search under dynamic time warping by pruning unpromising alignments*, Data Mining and Knowledge Discovery (2018), 1–29.

[101] Diego Furtado Silva and Gustavo EAPA Batista, *Elastic time series motifs and discords*, 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA), IEEE, 2018, pp. 237–242.

[102] Maria Inês Silva and Roberto Henriques, *Exploring time-series motifs through dtwsom*, 2020 International Joint Conference on Neural Networks (IJCNN), IEEE, 2020, pp. 1–8.

[103] Amanpreet Singh, Narina Thakur, and Aakanksha Sharma, *A review of supervised machine learning algorithms*, 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom), IEEE, 2016, pp. 1310–1315.

[104] Yash Singhal, Ayushi Jain, Shrey Batra, Yash Varshney, and Megha Rathi, *Review of bagging and boosting classification performance on unbalanced binary classification*, 2018 IEEE 8th International Advance Computing Conference (IACC), IEEE, 2018, pp. 338–343.

[105] Madan Somvanshi, Pranjali Chavan, Shital Tambade, and SV Shinde, *A review of machine learning techniques using decision tree and support vector machine*, 2016 International Conference on Computing Communication Control and automation (ICCUBEA), IEEE, 2016, pp. 1–7.

[106] Max Spooner and Murat Kulahci, *Monitoring batch processes with dynamic time warping and k-nearest neighbours*, Chemometrics and Intelligent Laboratory Systems **183** (2018), 102–112.

[107] Gian Antonio Susto, Angelo Cenedese, and Matteo Terzi, *Time-series classification methods: Review and applications to power systems data*, Big data application in power systems (2018), 179–220.

[108] Chang Wei Tan, Christoph Bergmeir, Francois Petitjean, and Geoffrey I Webb, *Monash university, uea, ucr time series regression archive*, arXiv e-prints (2020), arXiv–2006.

[109] Chang Wei Tan, Matthieu Herrmann, Germain Forestier, Geoffrey I Webb, and François Petitjean, *Efficient search of the best warping window for dynamic time warping*, Proceedings of the 2018 SIAM International Conference on Data Mining, SIAM, 2018, pp. 225–233.

[110] Chang Wei Tan, Geoffrey I Webb, and François Petitjean, *Indexing and classifying gigabytes of time series under time warping*, Proceedings of the 2017 SIAM international conference on data mining, SIAM, 2017, pp. 282–290.

[111] Jingren Tang, Hong Cheng, Yang Zhao, and Hongliang Guo, *Structured dynamic time warping for continuous hand trajectory gesture recognition*, Pattern Recognition **80** (2018), 21–31.

[112] Ling Tang, Huiling Lv, Fengmei Yang, and Lean Yu, *Complexity testing techniques for time series data: A comprehensive literature review*, Chaos, Solitons & Fractals **81** (2015), 117–135.

[113] Kashvi Taunk, Sanjukta De, Srishti Verma, and Aleena Swetapadma, *A brief review of nearest neighbor algorithm for learning and classification*, 2019 International Conference on Intelligent Computing and Control Systems (ICCS), IEEE, 2019, pp. 1255–1260.

[114] Romain Tavenard and Laurent Amsaleg, *Improving the efficiency of traditional dtw accelerators*, Knowledge and Information Systems **42** (2015), no. 1, 215–243.

[115] Nguyen Cong Thuong and Duong Tuan Anh, *Comparing three lower bounding methods for dtw in time series classification*, Proceedings of the Third Symposium on Information and Communication Technology, 2012, pp. 200–206.

[116] Sahar Torkamani and Volker Lohweg, *Survey on time series motif discovery*, Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery **7** (2017), no. 2, e1199.

[117] Tuan Minh Tran, Xuan-May Thi Le, Hien T Nguyen, and Van-Nam Huynh, *A novel non-parametric method for time series classification based on k-nearest neighbors and dynamic time warping barycenter averaging*, Engineering Applications of Artificial Intelligence **78** (2019), 173–185.

[118] Akshay Utture and V Krishna Nandivada, *Efficient lock-step synchronization in task-parallel languages*, Software: Practice and Experience **49** (2019), no. 9, 1379–1401.

[119] Greg Van Houdt, Carlos Mosquera, and Gonzalo Nápoles, *A review on the long short-term memory model.*, Artif. Intell. Rev. **53** (2020), no. 8, 5929–5955.

[120] Renzhong Wang and Dan Tao, *Dtw-knn implementation for touch-based authentication system*, 2019 5th International Conference on Big Data Computing and Communications (BIGCOM), IEEE, 2019, pp. 318–322.

[121] Xiaoyue Wang, Abdullah Mueen, Hui Ding, Goce Trajcevski, Peter Scheuermann, and Eamonn Keogh, *Experimental comparison of representation methods and distance measures for time series data*, Data Mining and Knowledge Discovery **26** (2013), no. 2, 275–309.

[122] Ian H Witten, Eibe Frank, Mark A Hall, and Christopher J Pal, *Data mining: Practical machine learning tools and techniques*, Morgan Kaufmann, 2016.

[123] Tzu-Tsung Wong and Nai-Yu Yang, *Dependency analysis of accuracy estimates in k-fold cross validation*, IEEE Transactions on Knowledge and Data Engineering **29** (2017), no. 11, 2417–2427.

[124] Wen-Jie Xie, Rui-Qi Han, and Wei-Xing Zhou, *Time series classification based on triadic time series motifs*, International Journal of Modern Physics B **33** (2019), no. 21, 1950237.

[125] Zhengzheng Xing, Jian Pei, and Eamonn Keogh, *A brief survey on sequence classification*, ACM Sigkdd Explorations Newsletter **12** (2010), no. 1, 40–48.

[126] Peng Yang, Lei Xie, Qiao Luan, and Wei Feng, *A tighter lower bound estimate for dynamic time warping*, 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, IEEE, 2013, pp. 8525–8529.

[127] Chin-Chia Michael Yeh, Nickolas Kavantzas, and Eamonn Keogh, *Matrix profile vi: Meaningful multidimensional motif discovery*, 2017 IEEE international conference on data mining (ICDM), IEEE, 2017, pp. 565–574.

[128] Chin-Chia Michael Yeh, Yan Zhu, Liudmila Ulanova, Nurjahan Begum, Yifei Ding, Hoang Anh Dau, Diego Furtado Silva, Abdullah Mueen, and Eamonn Keogh, *Matrix profile i: all pairs similarity joins for time series: a unifying view that includes motifs, discords and shapelets*, 2016 IEEE 16th international conference on data mining (ICDM), Ieee, 2016, pp. 1317–1322.

[129] Chin-Chia Michael Yeh, Yan Zhu, Liudmila Ulanova, Nurjahan Begum, Yifei Ding, Hoang Anh Dau, Zachary Zimmerman, Diego Furtado Silva, Abdullah Mueen, and Eamonn Keogh, *Time series joins, motifs, discords and shapelets: a unifying view that exploits the matrix profile*, Data Mining and Knowledge Discovery **32** (2018), no. 1, 83–123.

[130] Byoung-Kee Yi, HV Jagadish, and Christos Faloutsos, *Efficient retrieval of similar time sequences under time warping*, Data Engineering, 1998. Proceedings., 14th International Conference on, IEEE, 1998, pp. 201–208.

[131] Shuai Zheng, Kosta Ristovski, Ahmed Farahat, and Chetan Gupta, *Long short-term memory network for remaining useful life estimation*, 2017 IEEE international conference on prognostics and health management (ICPHM), IEEE, 2017, pp. 88–95.

[132] Yan Zhu, Chin-Chia Michael Yeh, Zachary Zimmerman, Kaveh Kamgar, and Eamonn Keogh, *Matrix profile xi: Scrimp++: time series motif discovery at interactive speeds*, 2018 IEEE International Conference on Data Mining (ICDM), IEEE, 2018, pp. 837–846.