الجمهورية الجزائرية الديمقراطية الشعبية
Peoples Democratic Republic of Algeria
وزارة التعليم العالي و البحث العلمي
Ministry of Higher Education and Scientific Research

جامعة 20 أوت 1955-سكيكدة كلية العلوم
University of 20 Août 1955-SKIKDA Faculty of Science

قسم الإعلام الآلي
Department of Computer Science

A thesis submitted in partial fulfillment

of the requirements for the degree of Master's (LMD) in Computer Science

Minor : Advanced software engineering and application

Time Series Classification : Comparative Study of Some Matching Methods

Presented by :
KHELIL Hind

Supervised by :
Dr. BOUGUEROUA Salah

Academic year : 2024/2025

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Abstract

Time series data, characterized by sequential measurements over time, are fundamental in various domains such as finance, healthcare, and environmental monitoring. The increasing volume of time series data, driven by IoT and sensor technologies, necessitates efficient and accurate analysis methods. However, challenges such as high dimensionality, noise, temporal misalignments, and varying lengths complicate classification tasks.

This thesis investigates prominent time series classification techniques, evaluating methods including Euclidean distance, Dynamic Time Warping (DTW), Longest Common Subsequence (LCSS), and the Shape Exchange Algorithm (SEA) using the 1-Nearest Neighbor (1NN) classifier on 85 datasets from the UCR Archive.

We propose two novel variants, PAA-SEA and RDP-SEA, though experiments confirm the superiority of the original SEA. Additionally, we explore a hybrid DTW-LCSS approach, which outperforms standalone methods.

We also assess the previously untested MSR (Mean Squared Residue) method in this field, demonstrating competitive accuracy.

Keywords: Time series, Time series classification,Time series mining, dimensionality reduction, distance/similarity measure.

ملخص

تمثّل بيانات السلاسل الزمنية، التي تتكون من قياسات متتابعة عبر الزمن، عنصراً أساسياً في مجالات متعددة مثل: التمويل، والرعاية الصحية، والمراقبة البيئية. ومع تزايد حجم هذه البيانات، مدفوعاً بتقنيات إنترنت الأشياء (IoT) وأجهزة الاستشعار، أصبحت الحاجة إلى أساليب تحليل دقيقة وفعالة أكثر إلحاحاً. ومع ذلك، فإن التحديات - مثل الأبعاد العالية، والضوضاء، واختلاف المحاذاة الزمنية، وتفاوت الأطوال - تُعقّد مهام التصنيف.

تهدف هذه الأطروحة إلى دراسة تقنيات تصنيف السلاسل الزمنية، مع تقييم طرق مثل: المسافة الإقليدية، والالتواء الزمني الديناميكي ، (DTW) وأطول سلسلة مشتركة ، (LCSS) وخوارزمية تبادل الشكل (SEA)، باستخدام مصنّف الجار الأقرب (1-NN) على 85 مجموعة بيانات من أرشيف UCR. نُقدّم نموذجين جديدين (PAA-SEA) و (RDP-SEA) ، على الرغم من أن النتائج تُظهر تفوّق الأسلوب الأصلي (SEA). كما نستكشف نهجاً هجيناً يجمع بين (DTW) و (LCSS)، والذي يتفوّق على الطرق المنفردة.

نقوم أيضاً بتقييم أسلوب (MSR) الذي لم يُختبر سابقاً في هذا المجال، حيث يظهر دقّة تنافسية.

كلمات مفتاحية: السلاسل الزمنية، تصنيف السلاسل الزمنية، تنقيب السلاسل الزمنية، اختزال الأبعاد، مقاييس المسافة/التشابه.

CHAPTER 1

GENERAL INTRODUCTION

1.1 Context and problematic

Time series data, consisting of sequential measurements collected over time [19], is ubiquitous in fields such as finance (e.g., currency trading), medicine (electrocardiograms (ECG), electroencephalograms (EEG), astronomy, and weather forecasting (e.g. daily recording of temperature and humidity) [21]. In fact, time-series data permeate almost all domains of scientific inquiry and industrial practice [19].

At the same time, time series datasets are growing much larger. With the rise of IoT devices and sensors, enormous amounts of this data are generating every day [59].

The ability to analyze and classify these data accurately is crucial for extracting meaningful patterns [19], predicting future trends [21], and detecting anomalies [11]. These tasks play a vital role in decision-making processes in industries [35]. However, time series data presents unique challenges, including high dimensionality [51], noise [17], temporal misalignments [63], and varying lengths [48], which complicate traditional analysis methods.

To address these challenges, researchers have developed various distance measures and dimensionality reduction techniques for time series classification.

Methods such as dynamic time warping (DTW), longest common sequence (LCSS) [62], and the Shape Exchange Algorithm (SEA) [5] offer different trade-offs between accuracy, computational efficiency, and robustness to distortions. Despite their widespread use, comparative studies have shown that there is no universally superior method [17], and the choice of technique often depends on the characteristics of the specific data set, including length, noise level, and temporal variability.

1.2 Thesis Objectives

This thesis aims to study different methods for classifying time series data. We compare traditional and advanced techniques, propose new improvements, and test new approaches to enhance classification accuracy. The main goals are listed below:

1. Investigate time series analysis, time series mining tasks and explore common methods for classification.
2. Compare well-known techniques like Euclidean distance, Canberra distance, Manhattan distance, and Earth Mover's Distance (EMD) using the 1-Nearest Neighbor (1NN) classifier on 85 UCR Archive datasets.
3. Examine advanced methods, including Dynamic Time Warping (DTW), Local Extrema DTW (LE-DTW), Shape Exchange Algorithm (SEA), and Longest Common Subsequence (LCSS).
4. Propose two new approaches, PAA-SEA and RDP-SEA, as potential improvements to the SEA method.
5. Explore combining DTW with LCSS to enhance classification performance.
6. Test MSR (Mean Squared Residue), a method not previously used in time series classification, to evaluate its effectiveness.

1.3 Thesis Outline

This thesis is structured as follows:

1. **Time Series Background** (chapter 2)
    - Introduction to fundamental concepts, properties, and challenges of time series data.
    - Discussion of key tasks in time series mining (clustering, classification, forecasting, anomaly detection, and motif discovery).
    - Review of preprocessing and representation techniques for handling high dimensionality and noise.
2. **Time Series Classification** (chapter 3)
    - Exploration of dimensionality reduction methods (PAA, LE, RDP, and SAX) and their impact on classification.
    - Examination of distance/similarity measures (Euclidean, Manhattan, Canberra, DTW, LCSS, SEA) with their respective strengths and limitations.
3. **Results and Discussion** (chapter 4)
    - Time series classification results from UCR Archive benchmarking
    - Comparative analysis of classification error rates across different methods and variants.
    - Presentation of a hybrid approach, combining Dynamic Time Warping (DTW) and Longest Common Subsequence (LCSS), balanced via parameter $\alpha \in [0, 1]$.
4. **General Conclusion and Perspectives**
    - Summary of key findings.
    - Future directions and perspectives.

CHAPTER 2

TIME SERIES BACKGROUND

2.1 Introduction

Time series data is a collection of measurements taken over time, used in many fields like finance, medicine, weather forecasting, and more. Analyzing this data helps us understand trends, make predictions, and detect unusual patterns. However, working with large time series datasets brings challenges like high dimensionality, noise, and storage limitations.

This chapter explains the basics of time series, its properties, applications, and common analysis techniques. We also explore key tasks in time series mining, such as clustering, classification, forecasting, and anomaly detection, along with methods to represent and preprocess the data efficiently.

2.2 Definition

A time series represents a collection of values obtained from sequential measurements over time [19].

$$T = \{(t_1, v_1), \ldots, (t_n, v_n)\}, \quad v_i \in \mathbb{R},$$
$$\text{where } t_1 < t_2 < \cdots < t_n. \text{ [65]}$$

Time series are used in statistics, signal processing, pattern recognition, econometrics, mathematical finance, weather forecasting, earthquake prediction, electroencephalography (EEG), control engineering, astronomy, communications engineering, and largely in any domain of applied science and engineering which involves temporal measurements [64].

Two main goals of managing time series data are the **effectiveness** and the **efficiency**, and the two key aspects towards achieving them are: **representation methods**, and **similarity measures** [63].

There are two types of time series data: A **univariate time series** is defined as an ordered sequence $X = \{x_1, x_2, \ldots, x_n\}$ where $x_i \in \mathbb{R}$ and $n = |X|$ denotes its length. A **multivariate time series** (or c-dimensional) is represented as $X = [X^{(1)}, X^{(2)}, \ldots, X^{(c)}]$, where each $X^{(j)}$ is a univariate series of length n [21][48].An illustration of these types is shown in (Figure 2.1).

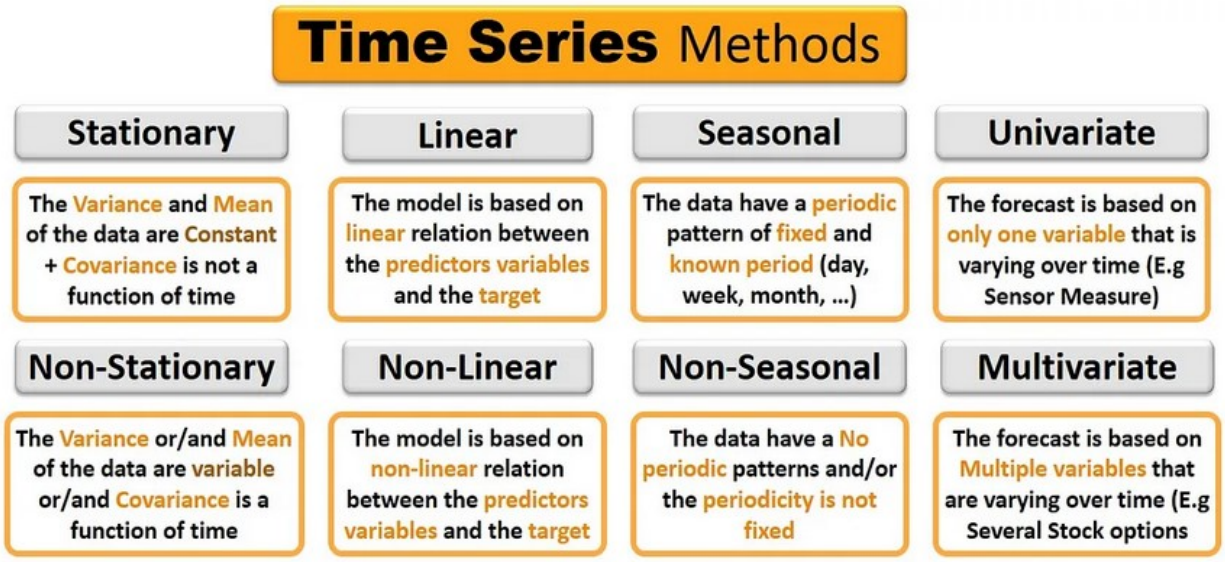

Figure 2.1: Times series types [22].

2.3 Properties and Challenges of Large Time Series

Before developing methods for time series data mining, it is essential to identify the key challenges that need to be addressed. Large time series datasets typically exhibit several characteristics, including extremely high dimensionality, noise, outliers, and dynamic behavior. One of the most

significant challenges in time series data mining is comparing two or more time series that may be shifted or scaled in time or amplitude. These challenges arise from the inherent properties of large time series, which we outline below [38].

Properties and Challenges

1. **High Dimensionality:** In time series data, each observation can be considered a dimension, leading to extremely high dimensionality, especially in large datasets [38][52]. Visualizing time series with tens of thousands of observations can already be difficult [44][38].and processing and storing such high-dimensional raw data can be computationally expensive. As a result, there is a need for efficient data representations or abstractions to reduce complexity. Additionally, the core philosophy of data mining suggests that analyzing raw data directly is often impractical due to potential information loss and slow processing speeds [38].

2. **Noise and Outliers:** Time series data often contains additive white noise alongside characteristic patterns[19]. For effective data mining, techniques must be robust enough to handle this noise, especially when focusing on global patterns. Furthermore, as datasets grow larger, sensitivity to measurement errors and outliers increases. However, longer time series also provide an advantage: they allow for better differentiation between outliers and rare but meaningful events. These rare outcomes, which might be misclassified as outliers in smaller datasets, can offer valuable insights into heterogeneity [38].

3. **Scale of Large Time Series:** The term "large" in the context of time series can refer to datasets with up to 1 trillion (1,000,000,000,000) time series objects. Storing 1 trillion time series objects requires approximately 7.2 terabytes of storage space [38][52].

4. **Data Streams and Storage Limitations:** The rapid growth of time series data often outpaces our ability to process and store it efficiently. In many cases, the data must be reduced or compressed immediately to manage storage requirements. A prime example of such data is streaming data, which is generated continuously and at highly variable rates. Examples include computer network traffic data or sensor data from physical networks, which produce non-stop

streams of information [24].

2.4 Applications of Time Series

Time series analysis is used in many fields [61], Figure 2.2 highlights key domains of application:

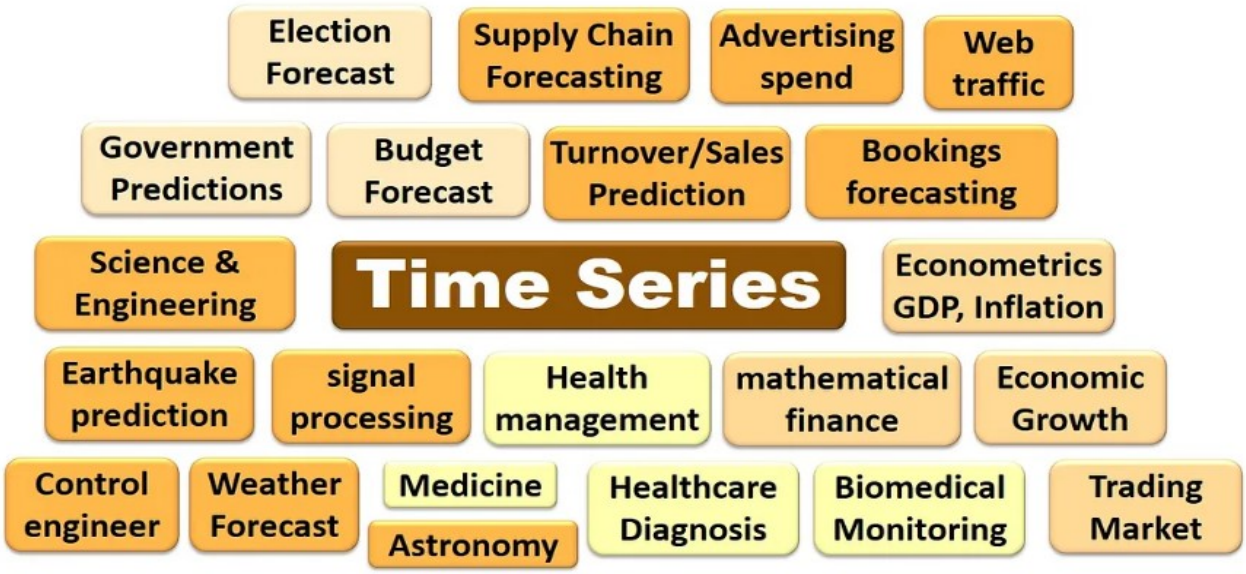

Figure 2.2: Time Series use cases and domains of applications [22].

2.4.1 Time series in Financial and Business Domain

The majority of business, investment, and financial choices are based on forecasts of future needs and developments in the financial sector. Forecasting and time series analysis are essential procedures for understanding the dynamic and significant behavior of financial markets. An expert can make the necessary predictions for significant financial applications in a variety of domains, including risk evolution, option pricing and trading, portfolio design, etc., by analyzing financial data.

2.4.2 Time Series in Medicine

Medicine uses time series to track patient health over time. For example, doctors study heart rate changes with breathing patterns to detect heart problems. In epidemiology, time series helps

monitor diseases by keeping regular records.

In medicine, diagnostic tests such as ECG (electrocardiogram, for heart activity) and EEG (electroencephalogram, for brain activity) rely on time series analysis. These signals are recorded using specialized medical devices (electrocardiographs for ECG and electroencephalographs for EEG). With wearable health devices, people can now track their health continuously, improving early diagnosis.

2.4.3 Time Series in Astronomy

Astronomers use time series to study space. Sunspot records date back to 800 BC in China! Scientists track stars, supernovae, and cosmic events by analyzing patterns in light waves. Today, machine learning helps classify space objects using time series data.

2.4.4 Time Series in Weather Forecasting

Weather forecasting improved with data collection. Ancient scientists studied weather, but modern weather stations now record real-time data worldwide. This helps predict daily weather and climate changes.

2.4.5 Time series in Business Development

Businesses use time series to predict future trends. It helps track growth, detect sales trends, and identify seasonal patterns. Companies rely on these forecasts to plan production and marketing effectively.

2.5 Tasks in Time Series Mining

Time series mining is a broad field that focuses on extracting meaningful patterns, knowledge, and insights from sequential data. The primary tasks in time series mining include indexing (Query by Content), classification, clustering, anomaly detection, forecasting, and segmentation [19][45][38].

2.5.1 Indexing (Query by Content)

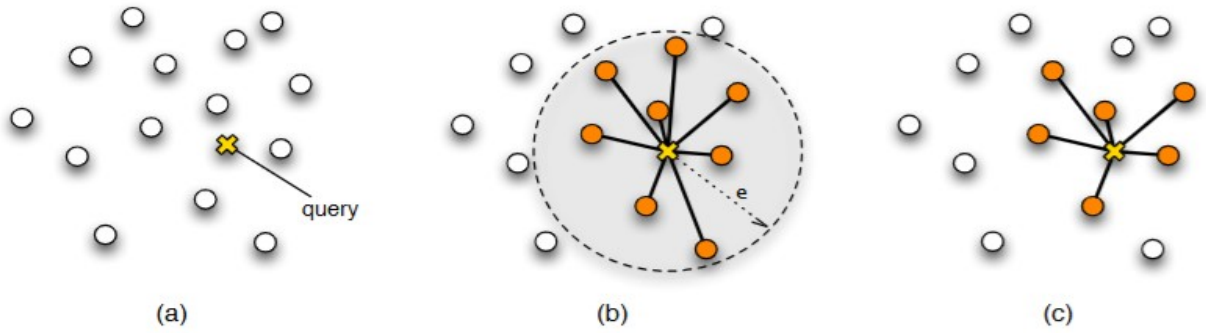

Figure 2.3: Query-by-content in 2D feature space: (a) Query transformation, (b) ε-range search returns series within radius ε, and (c) K-NN search finds K closest neighbors. Each point represents a time series with extracted features [19].

Query by content in time series refers to the process of searching for similar or relevant time series data based on their content or characteristics. Instead of relying on metadata or textual descriptions, query by content involves comparing the actual data values of time series to find matches or patterns, as illustrated in (Figure 2.3).

The process typically involves defining a query time series, which serves as a template or reference for the search. Algorithms then analyze the query time series and compare it to a database or collection of other time series data to identify similar patterns or sequences. Similarity measures such as Euclidean distance, dynamic time warping (DTW), or other distance metrics are often used to quantify the similarity between time series.

Query by content in time series has applications in various domains, including signal processing, pattern recognition, anomaly detection, and data mining. It allows researchers, analysts, and practitioners to efficiently search and retrieve relevant time series data based on their content, enabling tasks such as pattern matching, trend analysis, and predictive modeling [54][19].

2.5.2 Clustering

Clustering is the process of finding natural groups, called clusters, in a dataset [19]. As shown in Figure 2.4, Clustering is similar to classification that categorizes data into groups; however, these groups are not predefined, but rather defined by the data itself, based on the similarity between time series. There are two main ways to do clustering: Partitional Clustering and Hierarchical Clustering [54].

Hierarchical Clustering starts by figuring out how similar each data point is to every other point. Then, it merges similar points together until it forms groups. This doesn't need us to say how many groups there will be beforehand.

Partitional Clustering, often using the K-means algorithm, tries to find groups by minimizing how spread out the data points are within each group. But it needs us to say how many groups we want before it starts.

Clustering is used in many fields like biology, medicine, anthropology, marketing, and economics [54].

The time series clustering task can be divided into two sub-tasks: **Whole series clustering** , **Subsequence clustering** [19].

Whole clustering is when we group similar time series together based on their overall similarity. Subsequence clustering, on the other hand, looks at smaller chunks of time series data within a larger set.

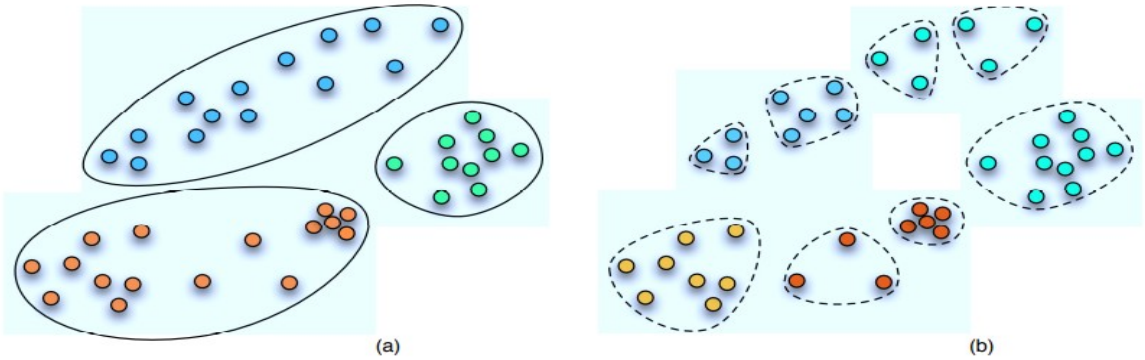

Figure 2.4: Clustering results with different parameters: (a) $N = 3$ clusters and (b) $N = 8$ clusters, demonstrating how parameter selection affects output quality.[19].

2.5.3 Classification

The classification task is about giving labels to different sets of data. Unlike clustering, where the groups aren't known beforehand, in classification, we already know the groups [19]. As illustrated in Figure 2.5, this process typically involves three main steps: training on labeled data, predicting labels for new data, and optionally refining class boundaries.

The aim is to train a model to recognize patterns or features in the time series data that distinguish one class from another. Once the model is trained, it can predict the class of new, unlabeled time series data based on these learned patterns. Time series classification has applications in various fields, including finance, healthcare, manufacturing, and environmental monitoring.

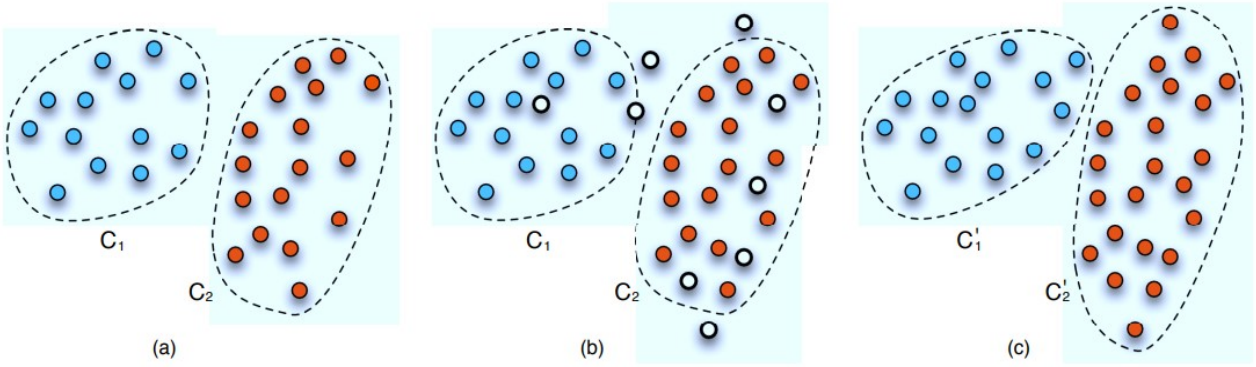

Figure 2.5: The three main steps of a classification task. (a) A training set consisting of two pre-labeled classes C1 and C2 is entered into the system. The algorithm will first try to learn what the characteristic features distinguishing one class from another are; they are represented here by the class boundaries. (b) An unlabeled dataset is entered into the system that will then try to automatically deduce which class each datapoint belongs to. (c) Each point in the set entered has been assigned to a class. The system can then optionally adapt the classes boundaries.[19]

2.5.4 Prediction (Forecasting)

Prediction can be viewed as a type of clustering or classification. But the difference is, when we predict, we're talking about what will happen later, not what's happening right now[54]. As shown in Figure 2.6, this involves analyzing input patterns to forecast future values. The task of predic-

tion is aimed at explicitly modeling such variable dependencies to forecast the next few values of a series [19]. We use prediction for lots of things, Its applications include obtaining forewarning of natural disasters (flooding, hurricane, snowstorm, etc), epidemics, stock crashes, etc. Many time series prediction applications can be seen in economic domains. prediction helps us plan for things by looking at how things have happened before and figuring out what might happen next. For example, in places where people buy and sell electricity, like in competitive energy markets, predicting how much electricity will be needed is really important. We look at things like how much electricity was used in the past, what the weather was like, and how much it cost, to guess how much electricity we'll need in the future. Another example is predicting how many accessories for cell phones will be sold based on how many cell phones were sold before [54].

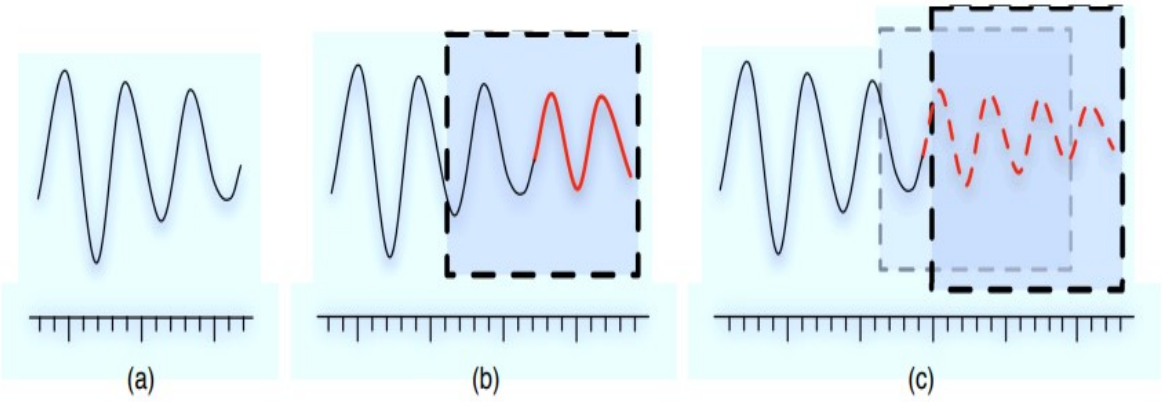

Figure 2.6: A typical example of the time series prediction task. (a) The input time series may exhibit a periodical and thus predictable structure. (b) The goal is to forecast a maximum number of upcoming datapoints within a prediction window. (c) The task becomes really hard when it comes to having recursive prediction, i.e. the long term prediction of a time series implies reusing the earlier forecast values as inputs in order to go on predicting [19].

2.5.5 Summarization (segmentation)

Summarization or (segmentation) , Summarization in time series data mining involves condensing large volumes of time-stamped data into more manageable and interpretable forms, such as rep-

resentative patterns, key features, or summary statistics. As demonstrated in Figure 2.7, this process transforms noisy, high-dimensional time series (a) into simplified representations (b) while preserving essential features. These summaries provide insights into the underlying patterns, trends, and characteristics of the time series data, facilitating analysis, visualization, and decision-making.

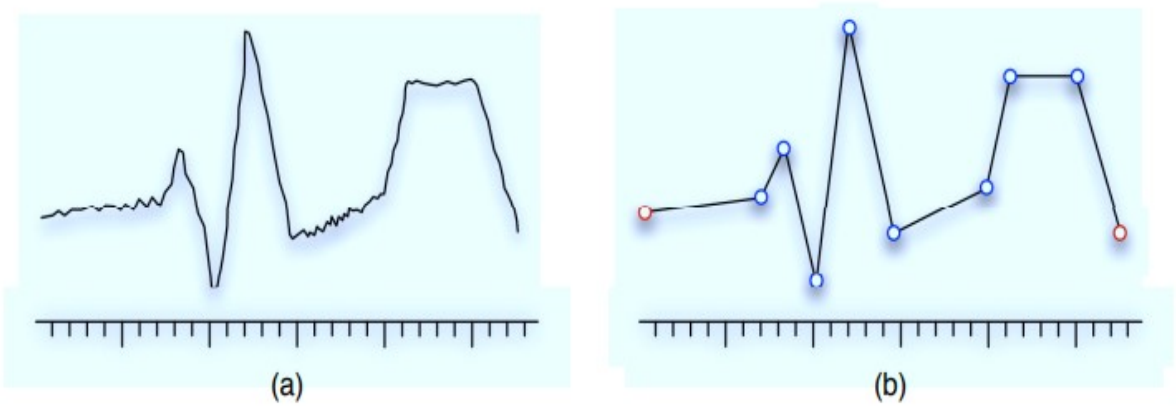

Figure 2.7: Example of application of a segmentation system. From (a) usually noisy time series containing a very large number of datapoints, the goal is to find (b) the closest approximation of the input time series with the maximal dimensionality reduction factor without loosing any of its essential features [19].

Anomaly detection and motif discovery are special cases of summarization where only anomalous more [54]. The task's goal is to reduce the difference between a simplified version and the original time series.The most common method used is Piecewise Linear Approximation (PLA), which involves dividing the series into segments and fitting a polynomial model to each segment [19].

2.5.6 Anomaly Detection (Interestingness Detection)

Anomaly detection, also known as Interestingness Detection, is a technique used to identify patterns or instances that deviate significantly from the norm within a dataset. As illustrated in Figure 2.8, this typically involves modeling standard periodic behavior and detecting subsequences that diverge from this pattern. These anomalies, or outliers, can represent data points that are rare, unexpected, or indicative of potential issues or anomalies in a system.

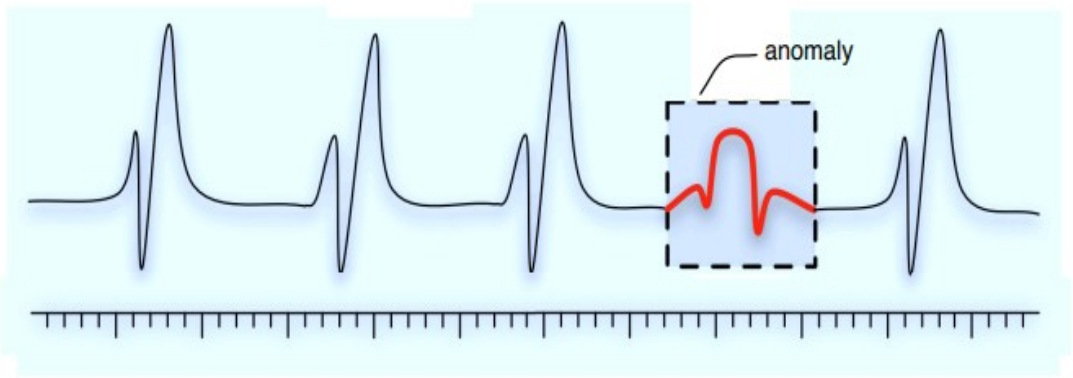

Figure 2.8: An idealized example of the anomaly detection task. A long time series which exhibits some kind of periodical structure can be modeled thanks to a reduced pattern of "standard" behavior. The goal is thus to find subsequences which does not follow the model and may therefore be considered as anomalies. [19]

Anomaly detection has numerous applications across various domains, including cybersecurity, fraud detection, network monitoring, healthcare, and industrial systems. In cybersecurity, anomaly detection helps identify unusual behavior or malicious activities within a network, such as unauthorized access attempts or abnormal data transfer patterns. In fraud detection, it can flag suspicious transactions or activities that deviate from typical behavior, helping financial institutions prevent fraudulent activities [11].

2.5.7 Motif Discovery

Motif discovery is the process of identifying recurring subsequences, known as motifs, within a longer time series. As shown in Figure 2.9, this task involves finding all recurrent subsequences in a time series, which can vary in length and may overlap, presenting significant computational complexity.

This idea originally came from studying DNA sequences, where scientists look for recurring genetic patterns. In time series data, motifs are short sequences that appear multiple times and can help identify important trends.

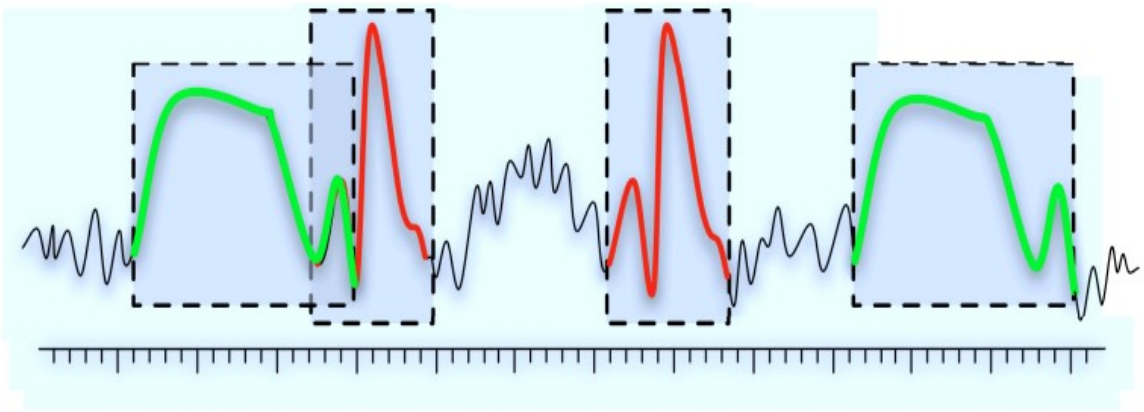

Figure 2.9: The task of motif discovery consists in finding every subsequence that appears recurrently in a longer time series. These subsequences are named motifs. This task exhibits a high combinatorial complexity as several motifs can exist within a single series, motifs can be of various lengths and even overlap. [19]

Researchers realized that traditional methods of grouping time series data often produced meaningless results [35] [19]. To solve this problem, motif discovery was introduced as a better way to find useful patterns. Some early methods used random selection techniques to identify motifs, but these approaches were not always accurate because they relied on probability [19].

Over time, new techniques were developed to improve motif discovery. Some focused on finding approximate motifs, which are useful when working with complex data, such as protein folding studies. Others treated motif discovery as a problem of organizing data in a multi-dimensional space, but these methods required careful selection of pattern lengths to work well.

Further improvements introduced ways to summarize motifs, making it easier to understand large datasets. However, one challenge in motif discovery is that small changes in pattern length can significantly affect results. To address this, researchers developed methods that can recognize motifs even if they stretch or shrink slightly.

More recent advancements have focused on improving speed and efficiency, allowing motif discovery to work with very large datasets. Some methods speed up the process by stopping early when the necessary patterns are found. Others allow users to provide additional information to guide the

search for motifs, making the results more relevant.

Motif discovery is useful in many areas. In anomaly detection, it helps define normal patterns so that unusual behaviors can be spotted. In time series classification, it speeds up the process by identifying key patterns that represent different categories, making it easier to organize and analyze large amounts of data [19] [50].

2.6 Preprocessing

In real-world applications, time series data often comes from sensors or live measurements and typically contains noise and outliers. To address these issues, preprocessing is applied. Noise can be reduced using signal processing techniques, such as digital filters or wavelet methods, while approaches like Independent Component Analysis (ICA) help extract the main patterns from the data. Many time series representations inherently reduce noise during transformation.

Another challenge is scaling differences between time series, where values may range differently (e.g., 0–100 vs. 0–1). Normalization methods, such as scaling to a fixed range (e.g. 0 to 1) or adjusting to zero mean and unit variance, are commonly used. However, normalization must be applied carefully, for example, normalizing a nearly flat but noisy series to unit variance can distort its shape, and normalizing very short segments may make distinct series appear identical.

Finally, time series often vary in length, requiring resampling (time adjustment) to standardize their sizes for comparison. A straightforward and effective approach is down-sampling, which shortens longer series while maintaining robustness [19].

2.7 Representation

Time series data has many dimensions because each point in the series is treated as a separate dimension. To make working with time series data more efficient and manageable, we need methods that reduce the number of dimensions. However, this remains a challenge in time series research. By reducing a time series with n data points to k (where k is much smaller than n), we can make computations much faster, lowering the complexity from O(n) to O(k). At the same time, we must

ensure that we do not lose too much important information and that the key patterns in the data are preserved. Another important aspect is that the reduced data should still be easy to interpret and visualize.

A good representation of time series data should not only reduce dimensions but also be quick to compute, maintain both small and large patterns in the data, and be resistant to noise, which refers to small, random changes that do not reflect meaningful trends. Over time, researchers have developed various techniques to represent and organize time series data. Some methods transform the original data into a different form to reduce its size before applying an indexing method, while others focus on improving the efficiency of searching through the data.

One practical way to categorize these techniques was suggested by [17]. Their approach helps researchers choose the right method by classifying them into three main types (Figure 2.10). Data-adaptive techniques adjust based on the specific data being analyzed, non-data-adaptive techniques follow fixed rules regardless of the data, and model-based techniques use mathematical models to represent time series. This classification helps in selecting the best method depending on the purpose and requirements of the analysis [38].

2.7.1 Non Data Adaptive Representation Techniques

Non-data adaptive representation techniques use fixed settings that don't change based on the data. They always work the same way, no matter what the data looks like. Some of these methods work by analyzing the data's frequency patterns. The idea is that any time series can be broken down into a mix of sine and cosine waves [38]. An example of this is the Discrete Fourier Transform (DFT), introduced by [1], which keeps only a few important Fourier coefficients. These coefficients are complex numbers that represent the most significant sine and cosine waves in the data. This helps reduce the size of the time series while keeping its main characteristics.

Another method, Discrete Wavelet Transform (DWT), was suggested as an alternative to DFT by researchers [27] and [9]. Unlike DFT, which focuses on global patterns, DWT captures both overall shapes and smaller details within the time series. DWT uses special mathematical functions called "wavelets" to break down the data. However, one drawback of this method is that it requires the

data length to be a power of two. A well-known type of wavelet is the "Haar" wavelet, introduced by [58] for time series analysis.

Several non-data-adaptive methods have been designed specifically for time series mining. One of the most popular is Piecewise Aggregate Approximation (PAA), introduced by Keogh and Pazzani [36]. PAA works by dividing a time series into equal-sized segments and computing the average value of each segment. These average values, known as PAA coefficients, form a simplified version of the original time series. Despite its simplicity, PAA performs well compared to more complex techniques. A major advantage of PAA is that all segments have the same length, making indexing easier.

An improved version of PAA, called Adaptive Piecewise Constant Approximation (APCA), was later introduced by Keogh et al [34]. Unlike PAA, APCA allows segments of different lengths, adjusting them to reduce reconstruction errors. In addition to storing the mean value of each segment, APCA also keeps track of its length.

Another method, Principal Component Analysis (PCA), is a technique borrowed from general data analysis. PCA reduces the number of dimensions by keeping only the most important components of the data while ignoring less significant ones. This method relies on mathematical transformations to find new, uncorrelated variables called "principal components". However, PCA does not consider the time relationships between data points, which can lead to incorrect results if two time series are similar but shifted in time. To fix this,[42] introduced a modified version of PCA that includes Dynamic Time Warping (DTW) before applying PCA, allowing the method to handle time shifts more effectively.

Unlike all these previous methods, some researchers, such as [53] argue that there is no need to reduce the number of dimensions. Instead, they suggest using a "clipped representation," where time series data is stored in a more compact format using bits instead of real numbers. This reduces storage space while keeping the data length unchanged [38].

2.7.2 Data Adaptive Representation Techniques

Data adaptive representation techniques are (more) sensitive to the nature of the data.They do not use fixed parameters but adjust to fit the data. However, most non-adaptive methods can be made adaptive by adding steps that respond to the data. A useful method called Singular Value Decomposition (SVD) also known as Latent Semantic Indexing, helps represent data efficiently with minimal error. It combines patterns in the data to keep the most important information. Researchers have improved SVD further by adding adjustments (called SVDD) to reduce errors even more.

Another popular method, Symbolic Aggregate Approximation (SAX), simplifies data by turning numbers into symbols (like letters). First, the data is split into equal sections, and averages are calculated (PAA). These averages are then converted into symbols (such as "a," "b," or "c") using predefined cutoff points. Storing symbols instead of raw data saves space. A faster version, iSAX, helps manage huge datasets. The number of symbols (α) can be chosen automatically using Minimum Description Length (MDL), which balances accuracy and simplicity [38].

2.7.3 Model Based Representation Techniques

The model-based approach assumes that a time series is created by an underlying model. The goal is to find the right model and its parameters to represent the data. Two time series are considered similar if they come from the same model with the same parameters.

There are different types of models used for time series, such as statistical models that extract features,ARMA models, Markov Chains (MCs), and Hidden Markov Models (HMMs). Markov Chains are simpler than HMMs, so they work better for short time series, but they cannot capture complex patterns as well.

Another example of a model-based method is Time Series Bitmaps, which helps visualize time series data [38][19].

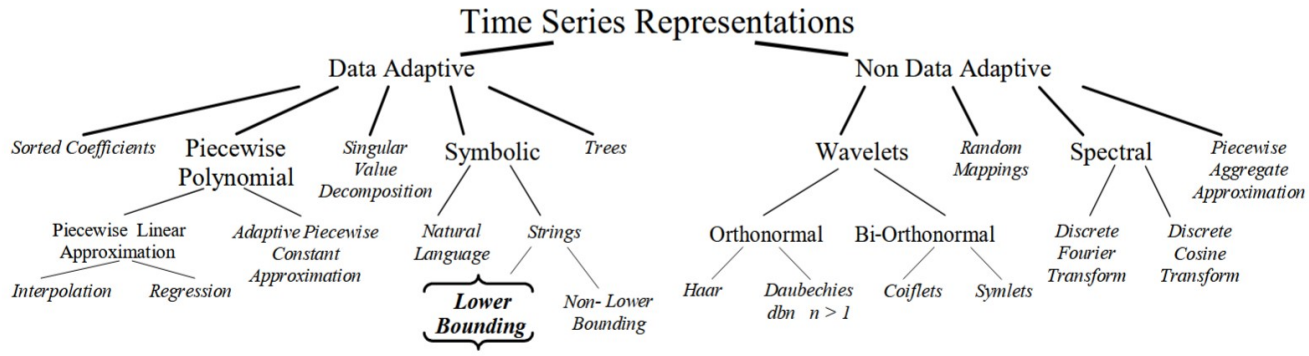

Figure 2.10: A hierarchy of time series representation. The leaf nodes refer to the actual representation, and the internal nodes refer to the classification of the approach [46].

2.8 Conclusion

Time series data is essential for analyzing trends and making predictions across various domains. This chapter introduced the fundamentals of time series, including its definition, challenges (such as high dimensionality and noise), and real-world applications. We also discussed different analysis techniques and key tasks like clustering, classification, and forecasting. Additionally, we covered preprocessing and representation methods to handle large datasets effectively. Understanding these concepts is crucial for working with time series data and extracting meaningful insights. In the next chapter, we will explore time series classification techniques and their practical implementations.

CHAPTER 3

TIME SERIES CLASSIFICATION

3.1 Introduction

This chapter explains time series classification, a machine learning method for analyzing data over time, with important uses like improving energy efficiency in smart homes [20]. It covers techniques to simplify time series data, including PAA, LE (Local Extrema), and RDP (Ramer-Douglas-Peucker), SAX (Symbolic Aggregate Approximation). The chapter also explores ways to measure similarity between time series, from simple methods like Euclidean distance to more flexible approaches like DTW and LCSS that can handle time shifts. SEA (Shape Exchange Algorithm), is also explained. Finally, the chapter introduces the UCR Time Series Archive, a collection of standard datasets used for testing classification methods.

3.2 Definition

Classification is the process of assigning a label to an unlabeled observation by utilizing patterns from the training data [20]. The classification of time series data has significant real-world applications, including household device identification for energy efficiency and carbon footprint reduction [20]. By analyzing power consumption patterns, machine learning models can classify

electrical appliances, enabling optimized energy usage and sustainability initiatives [4]. Such approaches contribute to smart home automation and environmental conservation [32]. Time series classification thus serves as a key tool in addressing modern energy challenges through data-driven insights.

Two critical components underpin this process: **dimensionality reduction** (simplifying data while preserving essential features) and **distance/similarity measures** (quantifying the resemblance between time series). These components will be explored in detail in the subsequent sections.

3.3 Dimensionality Reduction Techniques

Modern data analysis revolves around massive datasets characterized not only by growing sample sizes but also by an increasing number of features [2]. While high-dimensional data (common in time series and large-scale datasets) theoretically contain more information, they also introduce noise, redundancy, and computational challenges. The sheer volume and complexity of such data complicate analysis, visualization, and storage, often leading to the "curse of dimensionality" [2][43].

To lessen these difficulties, **dimensionality reduction** techniques aim to simplify data by retaining only the most informative features. By eliminating noise and redundancy, these methods enhance the efficiency and accuracy of machine learning algorithms while reducing computational overhead. However, the key challenge lies in preserving the essential structure and meaning of the original data.

Briefly, the main goal of dimensionality reduction in time series is to reduce the number of variables (features) while preserving essential information.

3.3.1 Piecewise Aggregate Approximation (PAA)

Piecewise Aggregate Approximation (PAA) is a dimensionality reduction technique for time-series data that allows efficient similarity search and pattern recognition in large databases [33]. The method simplifies time series while preserving essential shape characteristics, as demonstrated in Figure 3.1.

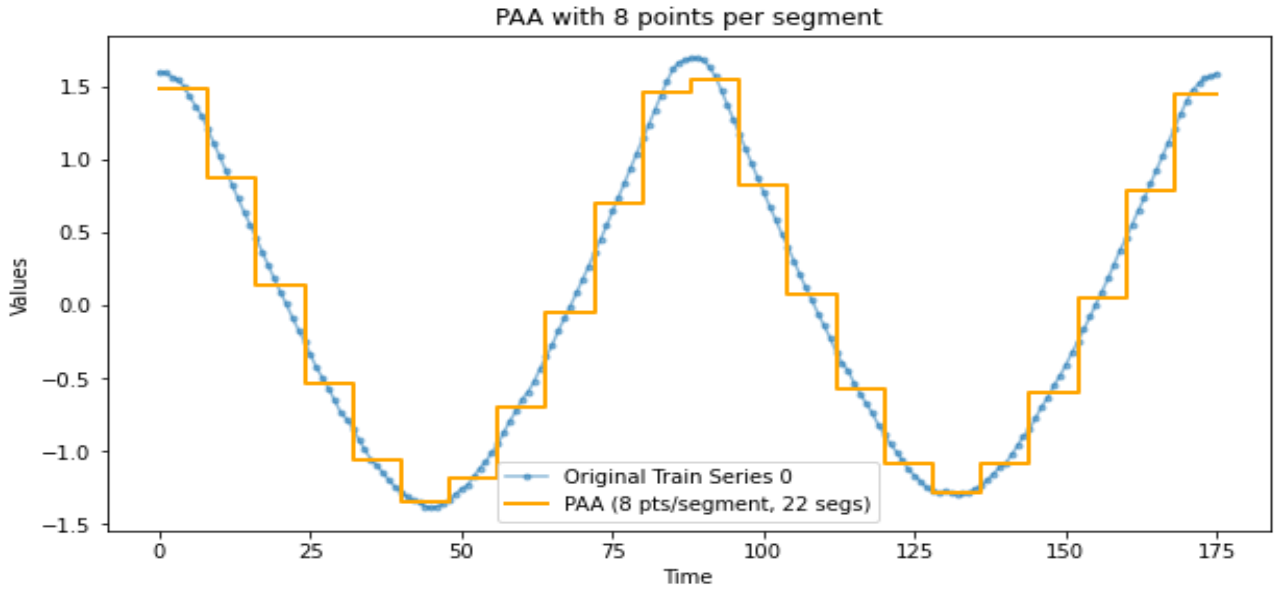

Figure 3.1: Piecewise Aggregate Approximation (PAA) transformation of the *Adiac* dataset (UCR Archive), showing the original series and its 22-segment approximation.

Formal Definition

PAA reduces a time series $T = \{t_1, t_2, \ldots, t_n\}$ of length n to a compressed representation $\bar{T} = \{\bar{t}_1, \bar{t}_2, \ldots, \bar{t}_M\}$ of length M (where $M \ll n$) by dividing T into M equal-sized segments and computing the mean of each segment:

$$\bar{t}_i = \frac{M}{n} \sum_{j=\left\lfloor \frac{n}{M}(i-1) \right\rfloor + 1}^{\left\lfloor \frac{n}{M} i \right\rfloor} t_j \tag{3.1}$$

where:

- n: Original time series length
- M: Reduced dimensionality
- Segment size: $w = \frac{n}{M}$ (if n is divisible by M)

Properties and variants

The Piecewise Aggregate Approximation technique exhibits several important properties that make it valuable for time series analysis. First, it provides effective **dimensionality reduction**, compressing an original time series of length n to a reduced representation of M points where $M \ll n$. This compression maintains the essential shape characteristics while significantly decreasing storage requirements.

Several enhanced versions of PAA have been proposed like: Linear Statistical Feature based Piecewise Aggregate Approximation (LSF_PAA) [28], Square root Statistical Feature based Piecewise Aggregate Approximation (SSF_PAA) [28], Adaptive piecewise aggregate approximation (APAA) [23].

A crucial theoretical property is its **distance preservation** capability, where the Euclidean distance between two PAA-reduced series forms a lower bound of the true Euclidean distance between the original time series [33]. This enables exact similarity search in the reduced dimensional space.

The algorithm demonstrates excellent computational characteristics with **linear complexity** of $O(n)$ in both time and space. This efficiency makes PAA particularly suitable for large-scale time series databases where computational resources are constrained.

3.3.2 Local Extrema (LE)

Local extrema is a dimensionality reduction technique used in time series analysis to simplify data while retaining key structural patterns. As shown in Figure 3.2, this method identifies critical points peaks (maxima) and valleys (minima) that define the essential shape and trends of the time series. By focusing on these points, the technique reduces computational complexity without losing significant information, making it particularly useful for long or high-dimensional datasets [41].

Definition and Extrema Types

Given a univariate time series $T = \{q_1, q_2, \ldots, q_n\}$, a point q_i is classified as a **local extremum** if it satisfies one of the following conditions:

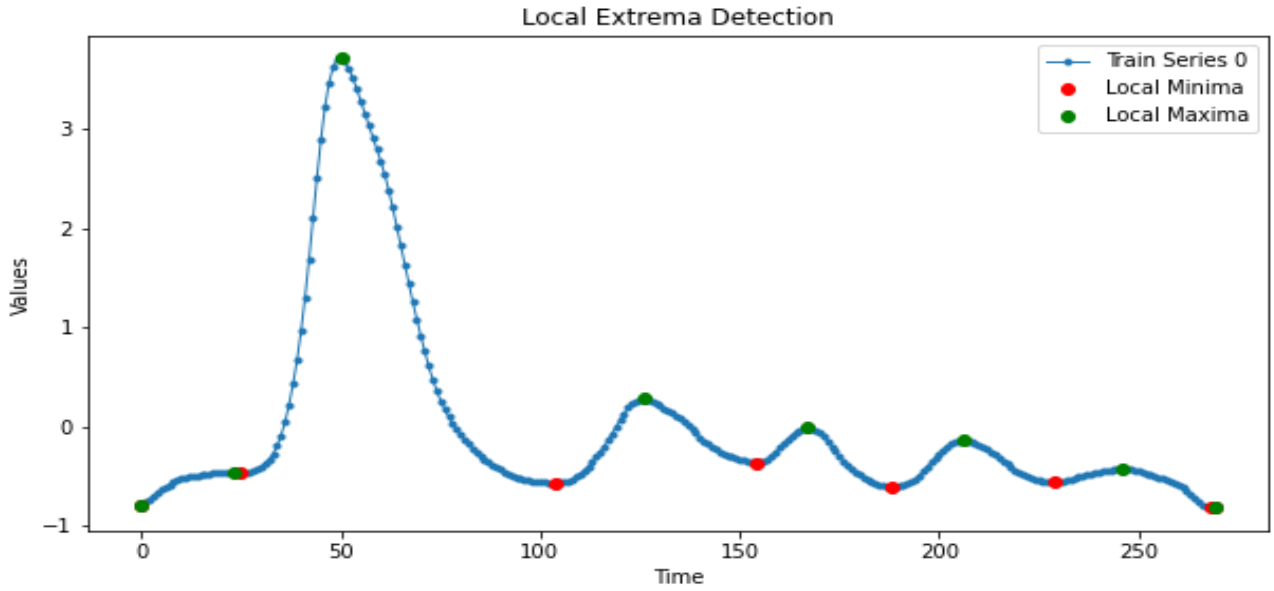

Figure 3.2: Time series from the *FiftyWords* dataset (UCR Archive) with extracted extrema points: local minima (red points), and local maxima (green points).

First/Last Point

- q_1 (first point) or q_n (last point).

Local Minimum (q_{min})

- q_i is smaller than its immediate neighbors:

$$(q_i < q_{i-1} \text{ and } q_i \leq q_{i+1}) \quad \text{OR} \quad (q_i \leq q_{i-1} \text{ and } q_i < q_{i+1}).$$

Local Maximum (q_{max})

- q_i is larger than its immediate neighbors:

$$(q_i > q_{i-1} \text{ and } q_i \geq q_{i+1}) \quad \text{OR} \quad (q_i \geq q_{i-1} \text{ and } q_i > q_{i+1}).$$

Extrema Extraction Process

The algorithm to extract local extrema from a univariate time series $T = \{q_1, q_2, \ldots, q_n\}$ involves the following steps, as illustrated in Figure 3.3:

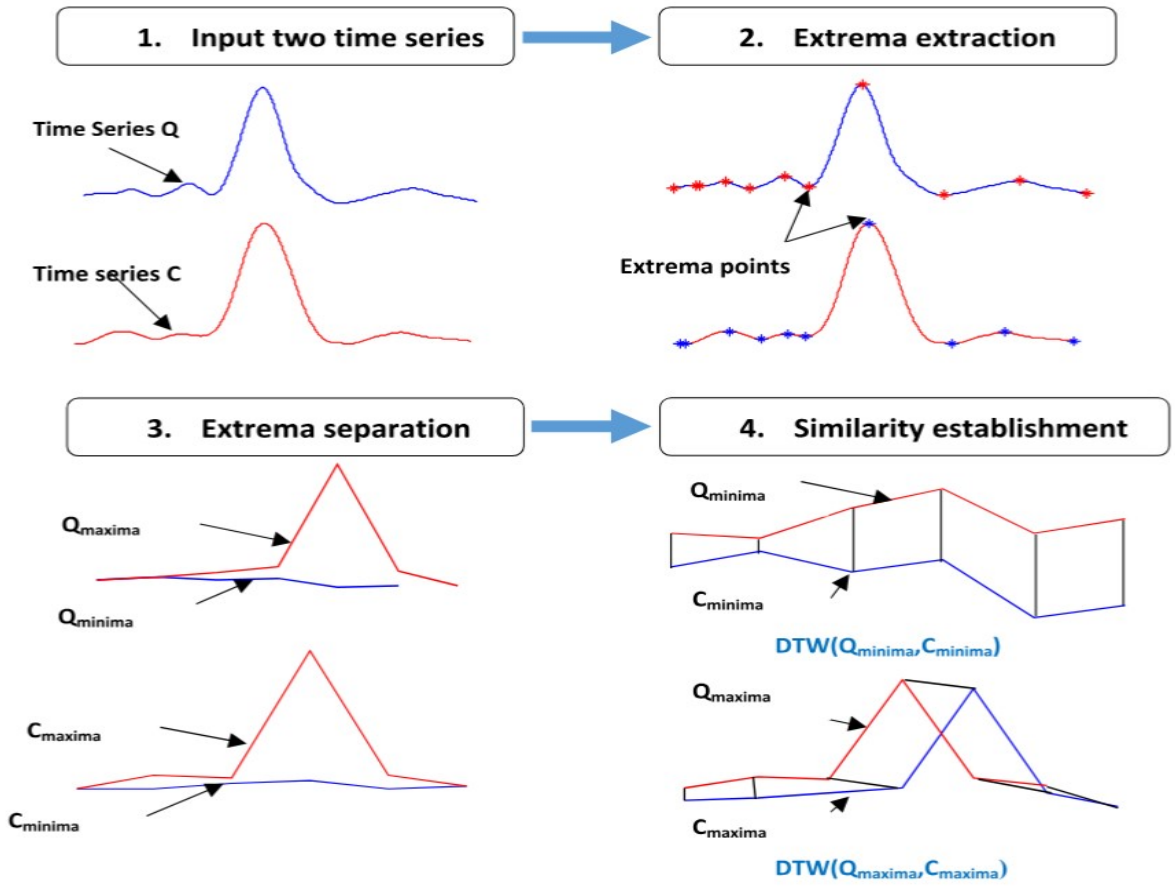

Figure 3.3: Pipeline of LE-DTW. LE-DTW consists of three major steps: (1) Local-extrema extraction. (2) Extrema separation. (3) Similarity establishment [41].

1. **Initialization**:
   - Include the first point q_1
   - Include the last point q_n
2. **Traversal**:
   - Compare q_i with its neighbors q_{i-1} and q_{i+1}
   - Classify q_i as either:
     - A local minimum if:

$$(q_i < q_{i-1} \text{ and } q_i \leq q_{i+1}) \quad \text{OR} \quad (q_i \leq q_{i-1} \text{ and } q_i < q_{i+1})$$

     - A local maximum if:

$$(q_i > q_{i-1} \text{ and } q_i \geq q_{i+1}) \quad \text{OR} \quad (q_i \geq q_{i-1} \text{ and } q_i > q_{i+1})$$

3. **Output**:
   - A reduced representation T' containing only extrema points:

$$T' = \{q_1, q_{\text{min/max}_1}, \ldots, q_{\text{min/max}_k}, q_n\} \tag{3.2}$$

     where k is the number of identified extrema points.

3.3.3 Ramer-Douglas-Peucker (RDP) Algorithm

The Ramer-Douglas-Peucker algorithm simplifies lines by reducing the number of points in a polygon while keeping its shape as close to the original as possible [29][18][13]. Figure 3.4 demonstrates this simplification process applied to a time series from the UCR Archive.

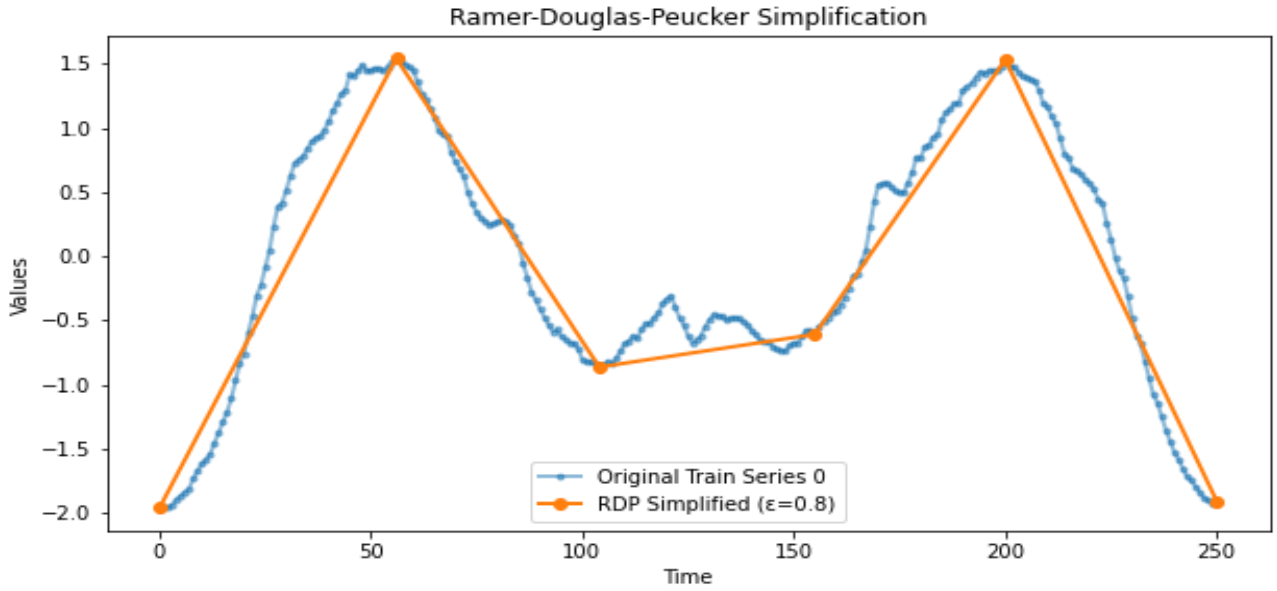

Figure 3.4: Simplification results using the Ramer-Douglas-Peucker algorithm ($\varepsilon = 0.8$) on *Arrow-Head* Training Series 0 (UCR Archive).

How the RDP Algorithm Works

The RDP algorithm simplifies a line by removing unnecessary points while keeping the overall shape. It works step by step as follows:

1. **Start and End Points:** First, it keeps the first and last points of the line.
2. **Find the Farthest Point:** It then looks for the point in the middle that is farthest away from the straight line connecting the start and end points.
3. **Check the Distance (ε):**

- If that farthest point is very close to the straight line (within distance ε), then all other middle points can be safely removed. The simplified line will still look similar to the original.
- If the farthest point is too far (beyond ε), the algorithm keeps that point and repeats the same process on:
  - The section between the start point and this new point.
  - The section between this new point and the end point.

4. **Stop When Done:** The algorithm stops when no more points are too far from the simplified line. The final result is a smoother version of the original line, using only the most important points.

Choosing ε

The RDP algorithm's ε controls simplification: higher values create simpler lines (fewer points), lower values keep more detail. Some versions of RDP can adjust ε automatically based on the data [29].

Several enhanced versions of RDP have been proposed like: Ramer-Douglas-Peucker Dynamic Time Warping (RDP-DTW) [37], Enhanced RDP (ERDP) [30].

3.3.4 Symbolic Aggregate Approximation (SAX)

SAX (Symbolic Aggregate Approximation) transforms a time series of length n into a symbolic string of length w (where $w < n$, and typically $w \ll n$), using an alphabet size a ($a > 2$). As illustrated in Figure 3.5, This discretization process involves an intermediate step the Piecewise Aggregate Approximation (PAA) which offers two key advantages [45]:

- **Dimensionality Reduction**: By first converting the time series into PAA representation, SAX inherits PAA's well-established dimensionality reduction capabilities, effectively compressing the data while preserving its structure.

- **Lower Bounding**: Proving that a symbolic distance measure lower-bounds the original time series distance is challenging. However, by demonstrating that the symbolic distance lower-bounds the PAA distance and leveraging existing PAA lower-bounding proofs SAX ensures meaningful similarity comparisons.

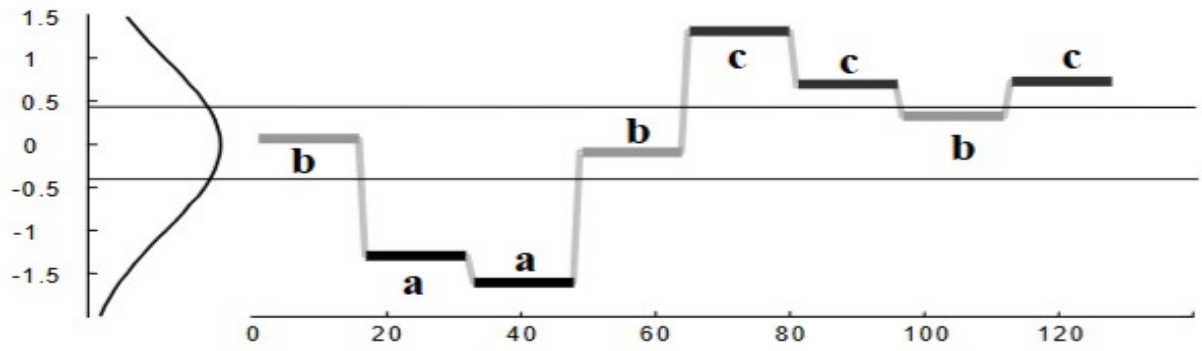

Figure 3.5: SAX discretization process: (1) A time series ($n = 128$) is converted to PAA representation ($w = 8$ segments), then (2) mapped to symbols ($a = 3$) using breakpoints, producing the symbolic word `baabccbc` [45].

Several enhanced versions of SAX have been proposed like: Trend Feature Symbolic Aggregate approximation (TFSAX) [66], Another enhanced version named (SAX_ARM) [49].

3.4 Distance/Similarity Measures

We begin by establishing the formal notation for time series analysis. A distance measure d between two series X and Y is formally defined as a function $d : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$.

Time series distance measures can be taxonomically categorized into: (1) lock-step (Euclidean), (2) sliding (SBD), (3) elastic (DTW), (4) kernel, (5) feature-based, (6) model-based (GMM), and (7) embedding (GRAIL) approaches [48].

In the following subsection, we will briefly outline these taxonomies but focus primarily on *lock-step* and *elastic* measures due to their foundational roles in time-series analysis.

3.4.1 Taxonomy of Time-series Distance Measure

Time-series distance measures can be categorized into seven fundamental approaches [48], as visually summarized in Figure 3.6:

1. **Lock-step measures:** These methods measure the difference between two time series by comparing each point (or step) in one series with the corresponding point in the other. They assume that both series are aligned in time. Common examples of lock-step measures are Euclidean Distance and Manhattan Distance.

2. **Sliding measures:** These methods compare a time series with different shifted versions of another series, then pick the smallest distance found. This makes them strong against time shifts or noise. A popular example is the Shape-Based Distance (SBD), used in the advanced k-Shape [47] clustering algorithm.

3. **Elastic measures:** These methods compare time series by first aligning them in time, allowing the matching of similar patterns even if they are shifted, stretched, or compressed. This flexibility helps overcome timing differences (like delays or speed variations) when measuring similarity.

   Elastic measures can match points in different ways **one-to-one** or **one-to-many** to find the best alignment. For multivariate time series, there are two approaches:

   - **Independent alignment:** Each channel is aligned separately.
   - **Dependent alignment:** All channels are aligned together, preserving their relationships.

4. **Kernel measures:** These methods use a special function to transform time series into a higher-dimensional space, where patterns can be more easily compared. This helps in tasks like clustering, where the data may not be clearly separable in its original form. By applying a non-linear transformation, kernel measures can reveal hidden structures and make clusters more distinct.

5. **Feature-based measures:** These methods work by extracting key characteristics (like average value, trends, or patterns) from time series data. Instead of comparing the raw data directly,

we compare these extracted features using simple distance calculations (like Euclidean distance). This approach has two main benefits:

- **Noise resistance:** Features focus on the important trends and ignore small, random changes.
- **Efficiency:** Working with summarized features speeds up tasks like clustering.

6. **Model-based measures:** These methods try to find the hidden patterns or "rules" that create the time series data. Instead of comparing the raw data directly, they use statistical models like Gaussian Mixture Models (GMM) or Hidden Markov Models (HMM) to represent the data. Then, they measure the difference between these models using simple calculations, such as the Kullback–Leibler (KL) Divergence.

7. **Embedding measures:** These methods convert time series into a simplified, hidden form (called a "latent space"). Instead of comparing the original data, we measure distances in this new space using methods like Euclidean Distance (ED).

   This approach offers three key advantages: noise reduction, better separability, dimensionality reduction

3.4.2 Lock-step Measures

compute distances through direct element-wise comparisons between aligned time points. Characterized by their computational efficiency ($\mathcal{O}(n)$ time complexity, where n represents series length), these measures have become fundamental tools in time-series analysis. The Euclidean Distance (Figure 3.7) stands as the most prominent and widely adopted example of this category, serving as a baseline approach across numerous application domains [48]. These measures fall into several categories [48][10]:

1. **Minkowski Distance Family:** The generalized distance metric defined by the p-norm [10]:

$$D_p(X, Y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p} \tag{3.3}$$

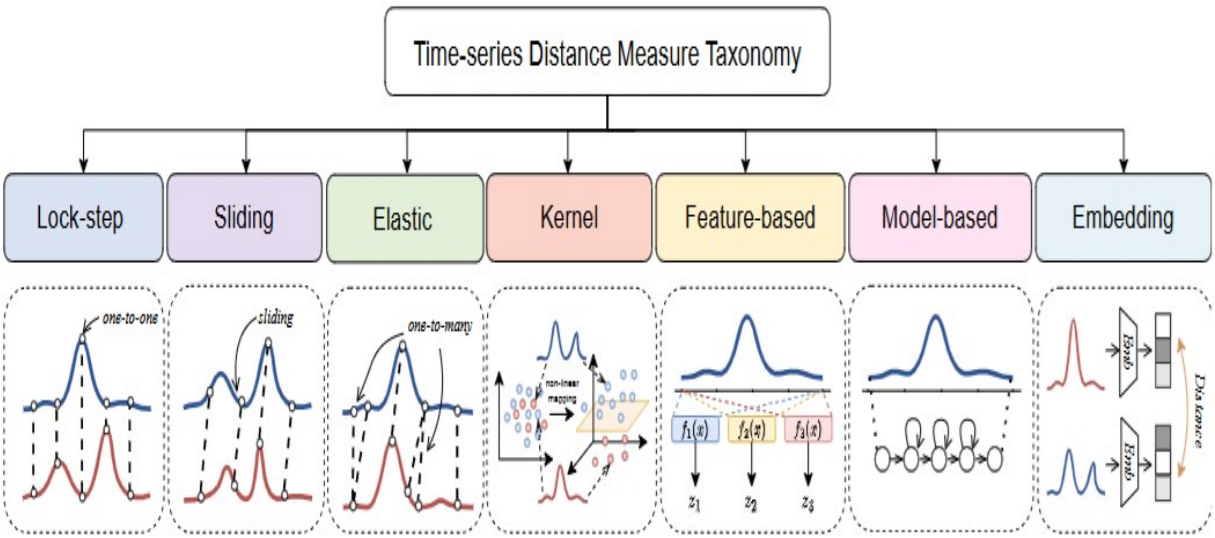

Figure 3.6: Overview of the time-series distance measure taxonomy [48].

Special cases include:

- Euclidean distance ($p = 2$):

$$D_{\text{Euclidean}}(X, Y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{3.4}$$

- Manhattan distance ($p = 1$):

$$D_{\text{Manhattan}}(X, Y) = \sum_{i=1}^{n} |x_i - y_i| \tag{3.5}$$

- Chebyshev distance ($p \to \infty$):

$$D_{\text{Chebyshev}}(X, Y) = \max_i |x_i - y_i| \tag{3.6}$$

2. L_1 **Functions:** The L_1 functions are based on adapted versions of the Manhattan metric. The Sørensen distance is a normalized variant of the L_1 distance, scaling values between 0 and 1, and is commonly used in ecology and environmental sciences. Similarly, the Gower distance normalizes the Manhattan distance by the length of the time series and is effective for datasets

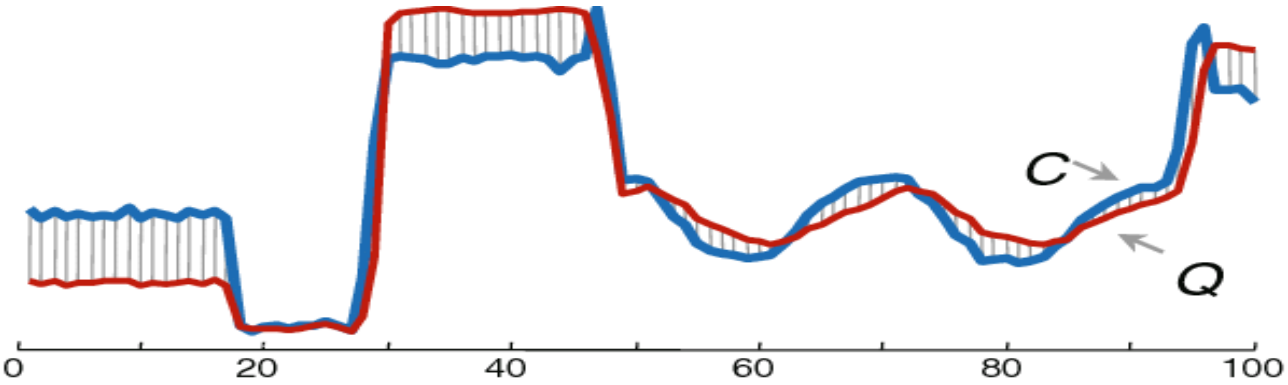

Figure 3.7: The Euclidean distance between two time series [3].

containing both continuous and categorical variables. The Soergel distance normalizes the L_1 distance using the sum of the maximum values of corresponding elements in the two compared time series, while the Kulczynski distance normalizes by the sum of the minimum values. The Canberra distance, defined as:

$$d_{\text{Canberra}}(X, Y) = \sum_{i=1}^{n} \frac{|x_i - y_i|}{|x_i| + |y_i|}, \tag{3.7}$$

is highly sensitive to small changes near zero, making it suitable for data clustered around an origin. The Lorentzian distance applies a logarithmic transformation to the L_1 distance, with an added constant to avoid undefined values, and has demonstrated robustness against noise and outliers. Previous studies have found that the Lorentzian distance, particularly when combined with normalization techniques like z-score, significantly outperforms Euclidean distance in classification tasks [48].

3. **Additional Categories:** While our analysis focuses on the above, other lock-step categories include [10][48]:

   - Intersection Functions
   - Inner Product Functions
   - Squared Chord Functions
   - Squared L_2 Functions

- Shannon's Entropy Functions
- Vicissitude Functions
- Combination Functions

Mathematical formulations for all measures are provided in subsequent tables.

Table 3.1: Lock-steps measures (part 1) including five categories: Minkowski, L_1, Intersection, Inner Product, Square Chord [48]

Method	Formula	Category
Euclidean	$\sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$	Minkowski
Manhattan	$\sum_{i=1}^{n} \lvert x_i - y_i \rvert$	Minkowski
Minkowski	$\left(\sum_{i=1}^{n} \lvert x_i - y_i \rvert^p\right)^{1/p}$	Minkowski
Chebyshev	$\max_i(\lvert x_i - y_i \rvert) = \lim_{p\to\infty} \left(\sum_{i=1}^{n} \lvert x_i - y_i \rvert^p\right)^{1/p}$	Minkowski
Sørensen	$\frac{\sum_{i=1}^{n} \lvert x_i - y_i \rvert}{\sum_{i=1}^{n}(x_i + y_i)}$	L_1
Gower	$\frac{1}{n} \cdot \sum_{i=1}^{n} \lvert x_i - y_i \rvert$	L_1
Soergel	$\frac{\sum_{i=1}^{n} \lvert x_i - y_i \rvert}{\sum_{i=1}^{n} \max(x_i, y_i)}$	L_1
Kulczynski	$\frac{\sum_{i=1}^{n} \lvert x_i - y_i \rvert}{\sum_{i=1}^{n} \min(x_i, y_i)}$	L_1
Canberra	$\sum_{i=1}^{n} \frac{\lvert x_i - y_i \rvert}{x_i + y_i}$	L_1
Lorentzian	$\sum_{i=1}^{n} \ln(1 + \lvert x_i - y_i \rvert)$	L_1
Intersection	$\sum_{i=1}^{n}(\lvert x_i - y_i \rvert)$	Intersection
Wave Hedges	$\sum_{i=1}^{n} \left(1 - \frac{\min(x_i, y_i)}{\max(x_i, y_i)}\right) = \sum_{i=1}^{n} \left(\frac{\lvert x_i - y_i \rvert}{\max(x_i, y_i)}\right)$	Intersection
Czekanowski	$1 - 2\frac{\sum_{i=1}^{n} \min(x_i, y_i)}{\sum_{i=1}^{n}(x_i + y_i)}$	Intersection
Motyka	$1 - \frac{\sum_{i=1}^{n} \min(x_i, y_i)}{\sum_{i=1}^{n}(x_i + y_i)}$	Intersection
Tanimoto	$\frac{\sum_{i=1}^{n}(\max(x_i, y_i) - \min(x_i, y_i))}{\sum_{i=1}^{n} \max(x_i, y_i)}$	Intersection
Inner Product	$\sum_{i=1}^{n} x_i y_i$	Inner Product
Harmonic Mean	$2\sum_{i=1}^{n} \left(\frac{x_i y_i}{x_i + y_i}\right)$	Inner Product
Kumar-Hassebrook	$\frac{\sum_{i=1}^{n}(x_i y_i)}{\sum_{i=1}^{n}(x_i^2 + y_i^2) - \sum_{i=1}^{n}(x_i y_i)}$	Inner Product
Jaccard	$\frac{\sum_{i=1}^{n}(x_i - y_i)^2}{\sum_{i=1}^{n}(x_i^2 + y_i^2) - \sum_{i=1}^{n}(x_i y_i)}$	Inner Product
Cosine	$1 - \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}}$	Inner Product
Dice	$\frac{\sum_{i=1}^{n}(x_i - y_i)^2}{\sum_{i=1}^{n}(x_i^2 + y_i^2)}$	Inner Product
Fidelity	$\sum_{i=1}^{n} \sqrt{x_i y_i}$	Square Chord
Bhattacharyya	$-\ln\left(\sum_{i=1}^{n} \sqrt{x_i y_i}\right)$	Square Chord
Squared-chord	$\sum_{i=1}^{n}(\sqrt{x_i} - \sqrt{y_i})^2$	Square Chord
Hellinger	$\sqrt{2\sum_{i=1}^{n}(\sqrt{x_i} - \sqrt{y_i})^2}$	Square Chord
Matusita	$\sqrt{\sum_{i=1}^{n}(\sqrt{x_i} - \sqrt{y_i})^2}$	Square Chord

Table 3.2: Lock-steps measures (part 2), including three categories: Squared L_2, Shannon's Entropy (Entropy), and Vicissitude [48].

Distance Measure	Formula	Category
Squared Euclidean	$\sum_{i=1}^{n}(x_i - y_i)^2$	Squared L_2
Clark	$\sqrt{\sum_{i=1}^{n}\left(\frac{\lvert x_i-y_i\rvert}{x_i+y_i}\right)^2}$	Squared L_2
Neyman χ^2	$\sum_{i=1}^{n}\frac{(x_i-y_i)^2}{x_i}$	Squared L_2
Pearson χ^2	$\sum_{i=1}^{n}\frac{(x_i-y_i)^2}{y_i}$	Squared L_2
Squared χ^2	$\sum_{i=1}^{n}\frac{(x_i-y_i)^2}{x_i+y_i}$	Squared L_2
Divergence	$2\sum_{i=1}^{n}\frac{(x_i-y_i)^2}{(x_i+y_i)^2}$	Squared L_2
Additive Symmetric χ^2	$\sum_{i=1}^{n}\frac{(x_i-y_i)^2(x_i+y_i)}{x_i y_i}$	Squared L_2
Probabilistic Symmetric χ^2	$2\sum_{i=1}^{n}\frac{(x_i-y_i)^2}{x_i+y_i}$	Squared L_2
Kullback-Leibler	$\sum_{i=1}^{n} x_i \ln\left(\frac{x_i}{y_i}\right)$	Entropy
Jeffreys	$\sum_{i=1}^{n}(x_i - y_i)\ln\left(\frac{x_i}{y_i}\right)$	Entropy
K Divergence	$\sum_{i=1}^{n} x_i \ln\left(\frac{2x_i}{x_i+y_i}\right)$	Entropy
Topsøe	$\sum_{i=1}^{n}\left[x_i \ln\left(\frac{2x_i}{x_i+y_i}\right) + y_i \ln\left(\frac{2y_i}{y_i+x_i}\right)\right]$	Entropy
Jensen Shannon	$\frac{1}{2}\sum_{i=1}^{n}\left[x_i \ln\left(\frac{2x_i}{x_i+y_i}\right) + y_i \ln\left(\frac{2y_i}{y_i+x_i}\right)\right]$	Entropy
Jensen Difference	$\sum_{i=1}^{n}\left[\frac{x_i\ln(x_i)+y_i\ln(y_i)}{2} - \frac{x_i+y_i}{2}\ln\left(\frac{x_i+y_i}{2}\right)\right]$	Entropy
Vicis-Wave Hedges	$\sum_{i=1}^{n}\frac{\lvert x_i-y_i\rvert}{\min(x_i,y_i)}$	Vicissitude
Emanon 2	$2\sum_{i=1}^{n}\frac{(x_i-y_i)^2}{\min(x_i,y_i)^2}$	Vicissitude
Emanon 3	$\sum_{i=1}^{n}\frac{(x_i-y_i)^2}{\min(x_i,y_i)}$	Vicissitude
Emanon 4	$\sum_{i=1}^{n}\frac{(x_i-y_i)^2}{\max(x_i,y_i)}$	Vicissitude
Max-Symmetric χ^2	$\max\left(\sum_{i=1}^{n}\frac{(x_i-y_i)^2}{x_i}, \sum_{i=1}^{n}\frac{(x_i-y_i)^2}{y_i}\right)$	Vicissitude
Min-Symmetric χ^2	$\min\left(\sum_{i=1}^{n}\frac{(x_i-y_i)^2}{x_i}, \sum_{i=1}^{n}\frac{(x_i-y_i)^2}{y_i}\right)$	Vicissitude

3.4.3 Elastic Measures

Lock-step distance measures, such as Euclidean Distance (ED), are often inadequate for comparing time series data with phase misalignments, temporal fluctuations, or varying speeds. As shown in Figure 3.8, these measures fail to account for distortions like stretching or squeezing along the time axis. In contrast, elastic measures, such as Dynamic Time Warping (DTW), employ non-linear alignments to better capture shape similarity between time series.

Elastic measures have proven effective in tasks like classification and clustering, outperforming rigid measures even as data sets size increases. However, a major drawback is their quadratic time complexity ($\mathcal{O}(n^2)$), compared to the linear complexity ($\mathcal{O}(n)$) of ED. This computational overhead, often one to three orders of magnitude higher, limits their use in large-scale applications. To mitigate this, acceleration techniques like lower bounding and early abandoning have been developed, enabling faster elastic distance computations in tasks such as k-nearest neighbor (K-NN) search [48].

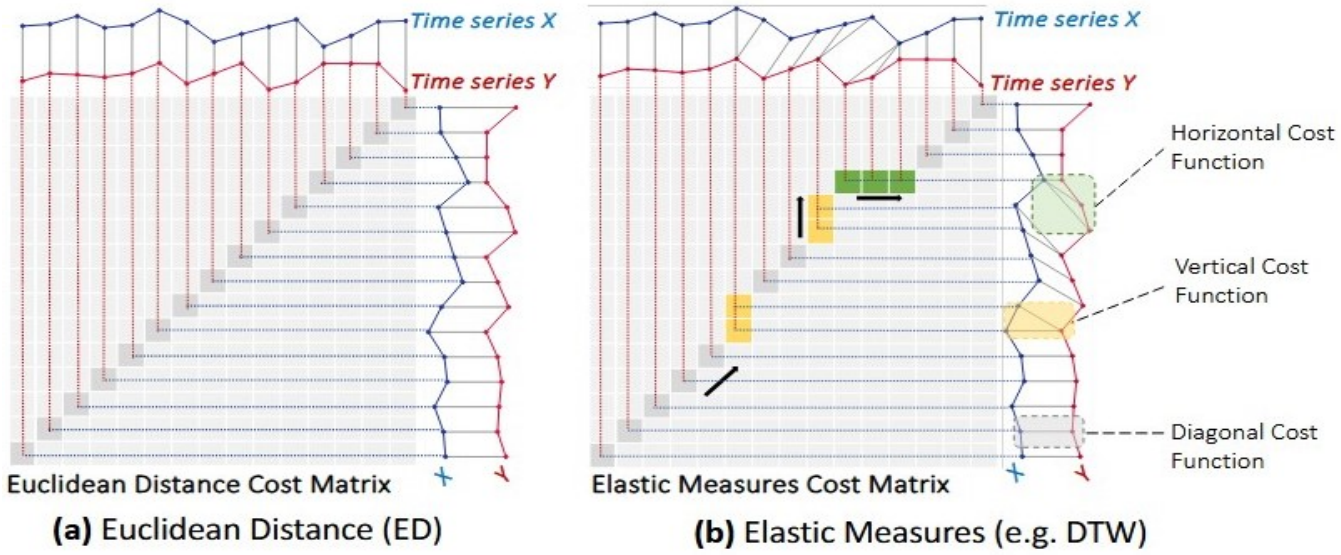

Figure 3.8: Alignments and cost matrices of Euclidean Distance (ED) and Dynamic Time Warping (DTW) [48].

3.4.3.1 Dynamic Time Warping (DTW)

Dynamic Time Warping (DTW) is a method for measuring similarity between two temporal sequences by aligning their data points in a way that minimizes overall differences (Figure 3.10). Unlike simpler approaches such as **Euclidean distance**, which require sequences to have identical lengths and perfect alignment (Figure 3.7), DTW can effectively compare sequences of different lengths and those with timing variations. This flexibility makes DTW particularly valuable in fields such as **speech recognition**, **gesture analysis**, and **financial time series analysis**, where temporal misalignments are common [25].

How Does DTW Work?

The DTW algorithm operates in four key steps [25], as illustrated in Figure 3.9:

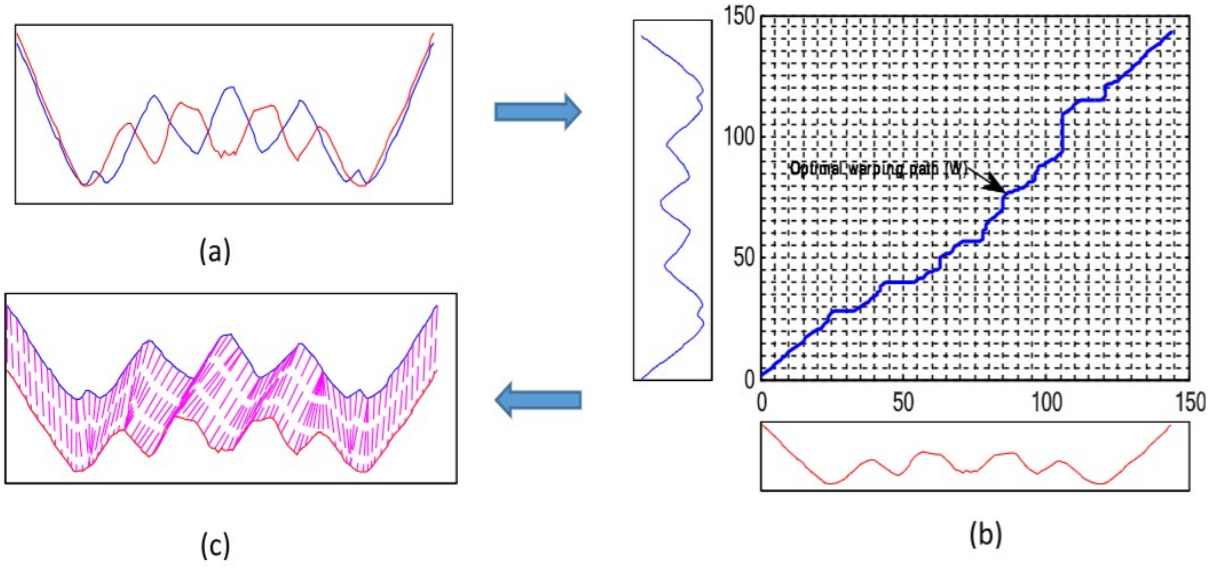

Figure 3.9: A visual example of how the DTW algorithm works. (a) Two given time series, (b) the optimal warping path, and (c) the alignment between these series [41].

1. **Distance Matrix Construction**

   The first step involves constructing a distance matrix between the two sequences $A = \{a_1, a_2, \ldots, a_n\}$ and $B = \{b_1, b_2, \ldots, b_m\}$. Each matrix element $C(i, j)$ represents the

distance (typically Euclidean) between corresponding points a_i and b_j:

$$C(i,j) = \text{distance}(a_i, b_j) = |a_i - b_j|$$

2. **Cost Matrix and Accumulated Cost**

   A cost matrix D is created by accumulating the minimum distances from the start to each point (i,j):

$$D(i,j) = C(i,j) + \min \begin{cases} D(i-1,j) & \text{(insertion)} \\ D(i,j-1) & \text{(deletion)} \\ D(i-1,j-1) & \text{(match)} \end{cases}$$

   Boundary conditions initialize the matrix:

$$\begin{aligned} D(1,1) &= C(1,1) \\ D(i,1) &= D(i-1,1) + C(i,1) \quad \text{for } i = 2,\ldots,n \\ D(1,j) &= D(1,j-1) + C(1,j) \quad \text{for } j = 2,\ldots,m \end{aligned}$$

3. **Optimal Path Finding**

   The optimal alignment path is found by backtracking from $D(n,m)$ to $D(1,1)$, following the direction of minimum cost at each step.

4. **Warping Path**

   The resulting warping path $W = \{(i_1,j_1),(i_2,j_2),\ldots,(i_L,j_L)\}$ shows how to stretch or compress the time axis of one sequence to best match the other.

DTW Variants

Several enhanced versions of DTW have been developed to address specific limitations of the basic algorithm. Constrained DTW (cDTW) improves computational efficiency by restricting warping paths within a Sakoe-Chiba band, while Weighted DTW (WDTW) [31] introduces position-dependent penalties to better preserve temporal relationships. Derivative DTW replaces raw values

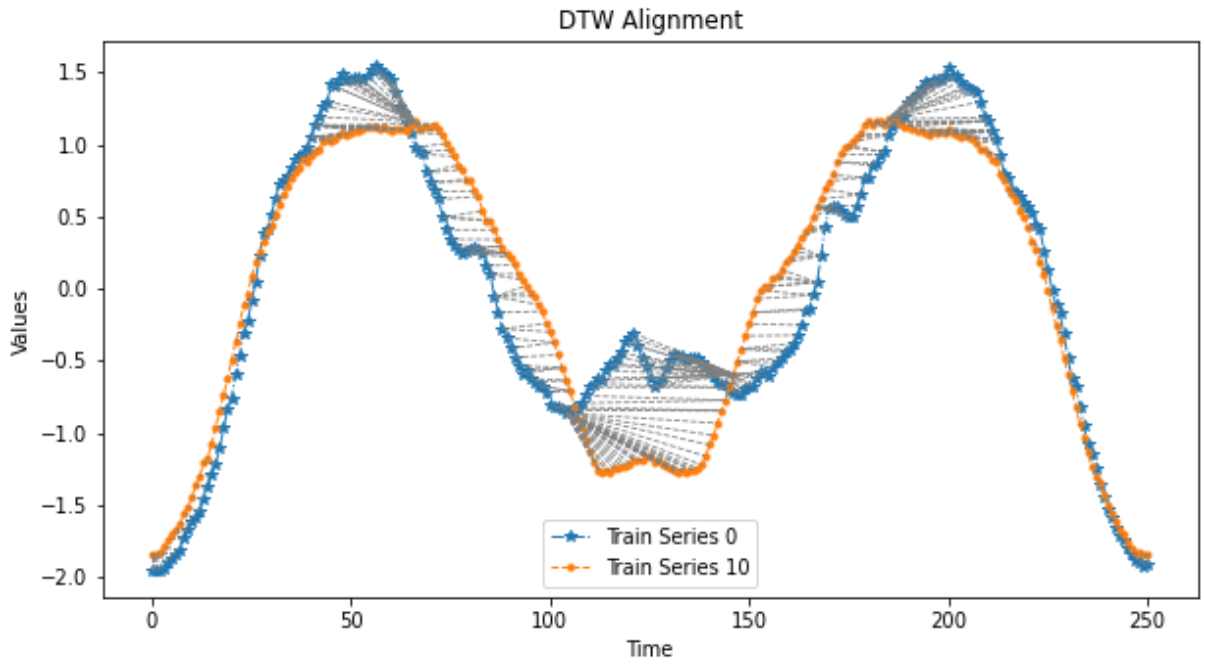

Figure 3.10: Alignment of *ArrowHead* Training Series (0 and 10) Using Dynamic Time Warping (DTW).

with first derivatives for improved shape matching, particularly useful for noisy datasets. These variants maintain DTW's core dynamic programming approach while offering specialized optimizations for different application scenarios [48].

3.4.3.2 Longest Common Subsequence (LCSS)

The Longest Common Subsequence (LCSS) is a classic problem in computer science that employs a point-to-point similarity definition and an edit-based distance measurement approach [56][57]. As illustrated in Figure 3.11 [60], LCSS was originally designed for comparing sequences of characters, identifying the longest subsequence shared by two sequences while accommodating noise and distortions by allowing non-matching elements to be skipped. The similarity in LCSS is defined strictly as the equality of characters (or values, in the case of time series).

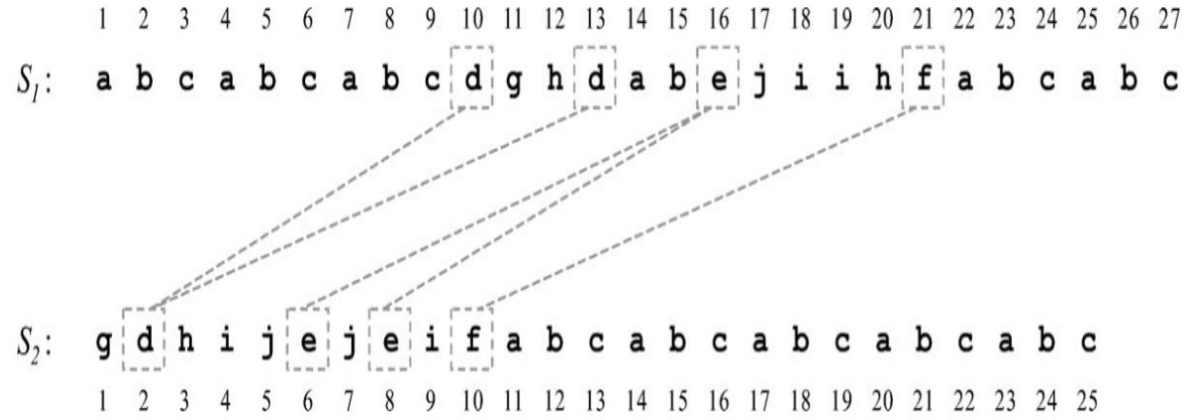

Figure 3.11: Constrained LCS example for strings S_1 and S_2 . Standard LCS yields length 15 (abcabcabcabcabc), while pattern-constrained LCS (requiring subsequence *def*) gives length 12 (dhejifabcabc/gdejifabcabc) [60].

Formulation for Character Sequences

Given two character sequences:

- $S_X = (x_1, x_2, \ldots, x_n)$ of length n,
- $S_Y = (y_1, y_2, \ldots, y_m)$ of length m,

their LCSS, denoted LCSS(S_X, S_Y), is computed as $M(n, m)$, where $0 \leq M(n, m) \leq \min(n, m)$. The recurrence relation for $M(i, j)$ is defined as:

$$M(i,j) = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0, \\ 1 + M(i-1, j-1) & \text{if } x_i = y_j \text{ for } i, j \geq 1, \\ \max\{M(i-1, j), M(i, j-1)\} & \text{if } x_i \neq y_j \text{ for } i, j \geq 1. \end{cases} \tag{3.8}$$

The normalized similarity score $\text{Sim}(S_X, S_Y) = \frac{2 \cdot \text{LCSS}(S_X, S_Y)}{m+n}$ ranges between 0 and 1, with higher values indicating greater similarity.

Adaptation for Time Series

To apply LCSS to time series, the similarity condition is relaxed. Two data points x_i and y_j are deemed similar if their absolute difference is within a threshold ϵ:

$$M(i,j) = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0, \\ 1 + M(i-1, j-1) & \text{if } |x_i - y_j| \leq \epsilon \text{ for } i, j \geq 1, \\ \max\{M(i-1,j), M(i,j-1)\} & \text{if } |x_i - y_j| > \epsilon \text{ for } i, j \geq 1. \end{cases} \tag{3.9}$$

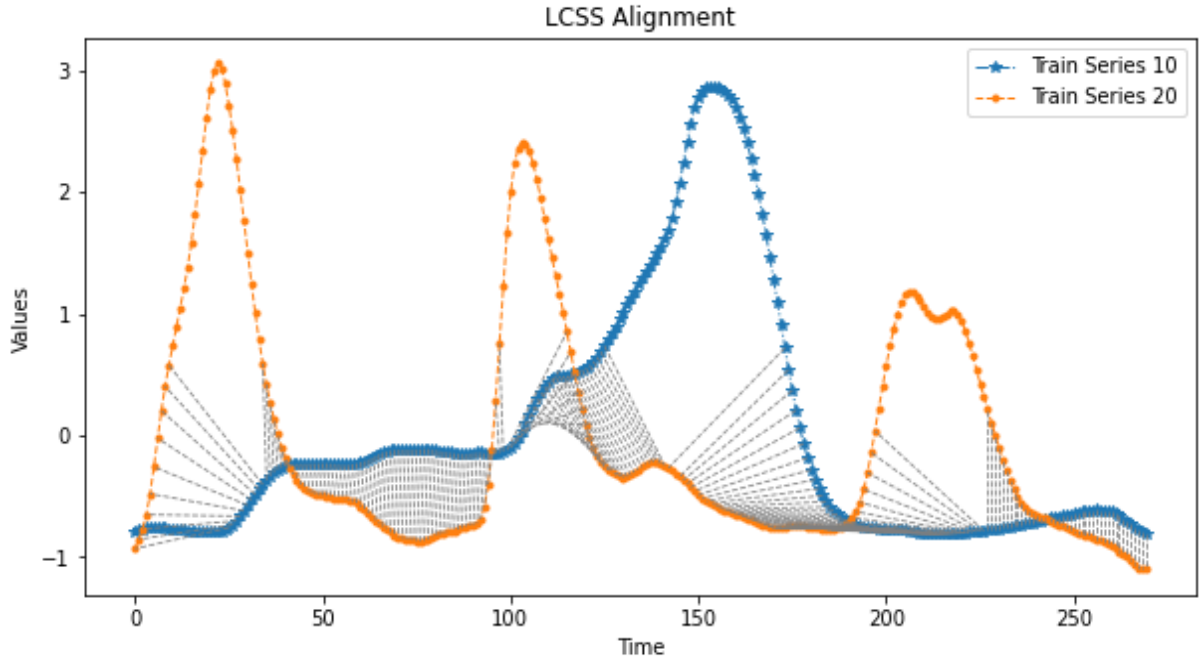

Figure 3.12: Alignment of *FiftyWords* Training Series (10 and 20) Using Longest Common Subsequence (LCSS).

3.4.4 Shape Exchange Algorithm (SEA)

The Shape Exchange Algorithm (SEA), introduced by Boucheham [5], is a novel method designed for matching quasi-periodic time series patterns, particularly effective in domains like electrocardiogram (ECG) analysis. SEA leverages selection sort-based signatures derived from magnitude-sorted time series, enabling robust point-to-point matching of patterns even when the time series differ in length, phase, or number of periods.[5].

Algorithm Steps

SEA operates in four key stages [5]:

1. **Signature Establishment**

   - **Input**: Time series $X = (x_i, t_i)_{i=1}^{N}$ (magnitude x_i at time index t_i).
   - **Process**: Sort X by magnitude to generate a **signature** S_X. This involves reordering the sequence (x_i, t_i) such that $x_1 \leq x_2 \leq \cdots \leq x_N$.
   - **Output**:
     - Sorted magnitudes: $S_X = \{x_{(1)}, x_{(2)}, \ldots, x_{(N)}\}$
     - Corresponding time indices: $T'_X = \{t_{(1)}, t_{(2)}, \ldots, t_{(N)}\}$

   The signature S_X serves as a global, stable descriptor of the time series, invariant to temporal shifts.

2. **Signature Exchange** Given two time series X and Y:

   - Reconstruct X_{rec} using S_Y (magnitudes of Y) and T'_X (time indices of X)
   - Reconstruct Y_{rec} using S_X (magnitudes of X) and T'_Y (time indices of Y)

3. **Reconstruction and Comparison**

   - **Reconstruction**: Sort X_{rec} and Y_{rec} back to the original time order using their respective time indices T_X and T_Y

- **Matching**: Compare the original time series (X, Y) with their reconstructed counterparts (X_{rec}, Y_{rec}) using metrics:
  - **Percent Root Difference (PRD)**:

$$\text{PRD}(X, X_{\text{rec}}) = \sqrt{\frac{\sum_{i=1}^{N} |x_i - x_{\text{rec},i}|^2}{\max\left(\sum_{i=1}^{N} |x_i - \bar{x}|^2, \sum_{i=1}^{N} |x_{\text{rec},i} - \bar{x}_{\text{rec}}|^2\right)}} \times 100\% \quad (3.10)$$

  - **Correlation Coefficient (Corr)**:

$$\text{Corr}(X, X_{\text{rec}}) = \frac{\text{cov}(X, X_{\text{rec}})^2}{\text{var}(X) \cdot \text{var}(X_{\text{rec}})} \quad (3.11)$$

Handling Time Series of Different Lengths

For X (length N) and Y (length M, $N \neq M$):

- **Linear Mapping:** Normalize signatures S_X and S_Y to a common length.
  - Expand the shorter series or compress the longer series using interpolation.
  - Ensures compatibility during signature exchange.

Applications

- **Novelty Detection:**
  - Identifies abnormal patterns (e.g., PVC beats in ECG) by isolating reconstruction errors (Figure 3.13).
- **Person Identification:**
  - Discriminates individuals using ECG signatures (Corr > 0.99 for same person vs. < 0.98 for others).
- **Medical Diagnostics:**
  - Enables detailed comparison of cardiac cycles for arrhythmia detection.

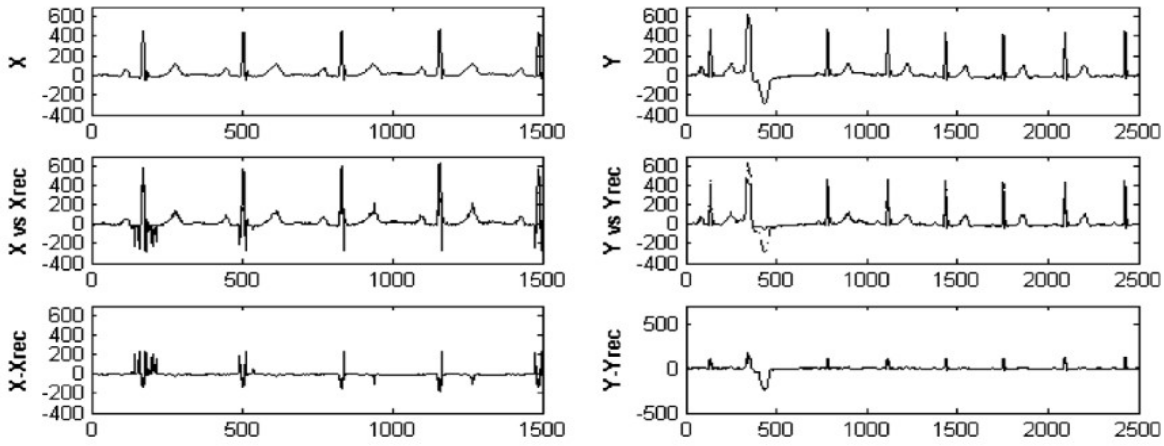

Figure 3.13: Ilustration of novelty detection in electrocardiogram traces [5].

SEA Variants:

The Shape Exchange Algorithm (SEA) has inspired several variants designed to enhance its efficiency, scalability and applicability to different time series matching scenarios. Two notable variants are FastSEA [39] and LMDS-SEA [40], each addressing specific limitations of the original SEA method.

1. **FastSEA:** was introduced by Lahreche [39] to overcome the quadratic complexity ($O(N^2)$) of the original SEA by replacing its sorting mechanism with the **Counting Sort algorithm**, reducing the time complexity to **linear** ($O(N)$). The key improvements include:

    - **Efficient Sorting:** When Lahreche use the **Counting Sort** instead of **Selection Sort**, significantly speeding up the signature establishment phase.
    - **Preserved Accuracy:** Despite the acceleration, FastSEA maintains the same effectiveness as SEA in aligning quasi-periodic time series (e.g. ECG signals).
    - **Scalability:** FastSEA can process very long time series (100,000 samples) in seconds, making it practical for big data applications.

      As noted in [39]: *"For instance, when the length of the time series is 1000 samples, the execution time is 3,060587 and 0,953463 seconds for SEA and FastSEA respectively. However, for 100000 samples the execution time is 14542,21608 seconds (4 hours) for SEA and 44,978761 seconds for*

FastSEA."

2. **LMDS-SEA:** The Shape Exchange Algorithm (SEA) is highly effective for aligning quasi-periodic time series (e.g., ECG signals) but struggles with general time series specially in time series classification task. To address this limitation, Lahreche and Boucheham introduced LMDS-SEA (Local Matching with Distance Selection Shape Exchange Algorithm) [40], which enhances SEA through two ideas:

   - **Local Matching (LM-SEA)**.
   - **Distance Selection**.

3. **FANSEA:** FANSEA introduced by Boucheham [7], is an efficient time series matching technique that combines data reduction (using a variant of the FAN line simplification algorithm to extract key points) with the SEA alignment method. It speeds up comparisons by using only a small fraction of the original data (reducing samples by up to 80%, and time by up to 95 %) while maintaining accuracy, making it ideal for large or complex time-series datasets like ECG signals.

4. **ASEAL (Approximate Shape Exchange Algorithm):** Proposed by Boucheham [6], ASEAL combines the SEA alignment method with data reduction using the DPSimp algorithm (a variant of the Douglas-Peucker line simplification). Key features include:

   - **Data Reduction:** DPSimp retains perceptually significant points, reducing the dataset by up to 90% while preserving shape integrity. As mentioned in [6]: *"For instance, the data reduction percentage reaches 90% for time series of length 50,000 samples"*.
   - **Efficiency:** ASEAL achieves up to 97% faster processing than SEA by operating on reduced data, with minimal loss in alignment accuracy.
   - **Robustness:** Handles complex quasi-periodic time series (e.g., ECG signals) with amplitude/time shifts, phase shifts, and variable lengths, even for arbitrary numbers of periods.

As demonstrated in [6], ASEAL maintains SEA alignment quality while drastically reducing resource usage: *"We stress the point that another very important result this study came up to is the demonstration that ASEAL can perform comparable to those of SEA matchings in terms of the alignment quality and the correlation factors for data memory savings up to 90% and processing time savings reaching 97% that of SEA. Speci cally, the Execution time/Time series lengthratio is below 10% for ASEAL, whereas it reaches 230% for SEA."*

3.4.5 Earth Mover's Distance (EMD)

The Earth Mover's Distance (EMD) [55] is a measure of dissimilarity between two frequency distributions, densities, or measures over a metric space D. EMD measures the **minimum cost** to transform one distribution into another [14].

Distance Formula

For two distributions P and Q with weights $\{w_{p_i}\}$ and $\{w_{q_j}\}$, EMD is computed as:

$$\text{EMD}(P, Q) = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} f_{ij}\, d_{ij}}{\sum_{i=1}^{m} \sum_{j=1}^{n} f_{ij}} \tag{3.12}$$

where:

- f_{ij}: Flow of mass from supply p_i to demand q_j
- d_{ij}: Ground distance between p_i and q_j (e.g., Euclidean distance)

Constraints

The optimization is subject to:

$$f_{ij} \geq 0 \quad \text{(non-negative flow)} \tag{3.13}$$

$$\sum_{j} f_{ij} \leq w_{p_i} \quad \text{(supply limit)} \tag{3.14}$$

$$\sum_{i} f_{ij} \leq w_{q_j} \quad \text{(demand limit)} \tag{3.15}$$

$$\sum_{i,j} f_{ij} = \min\left(\sum_{i} w_{p_i}, \sum_{j} w_{q_j}\right) \tag{3.16}$$

3.5 The UCR Time Series Archive

The UCR Time Series Archive, first introduced in 2002, is one of the most important resources for time series data mining research. Over 1,000 published papers have used at least one dataset from this archive, making it a standard benchmark for testing algorithms [15].

3.5.1 Key Features of the Archive

- **Free and Easy to Use:** The datasets are openly available in a simple file format, making them accessible to everyone.
- **Continuous Growth:**
  - Started with 16 datasets (2002);
  - Expanded to 45 datasets (later years);
  - Grew to 85 datasets (2015);
  - Now includes 128 datasets (latest expansion).

3.5.2 Data Format

All datasets in the UCR Archive follow a standardized structure. As shown in Figure 3.14, each data set consists of two TSV files: a training partition named [DatasetName]_TRAIN.tsv

and a testing partition named [DatasetName]_TEST.tsv. For example, the Fungi dataset includes Fungi_TRAIN.tsv and Fungi_TEST.tsv files [16].

"There is one time series exemplar per row. The first value in the row is the class label (an integer between 1 and the number of classes). The rest of the row are the data sample values" [16].

Table 3.3 summarizes the 128 UCR Archive datasets, detailing each dataset's name, type (e.g., Image, Sensor), training/test set sizes, class count, and time series length.

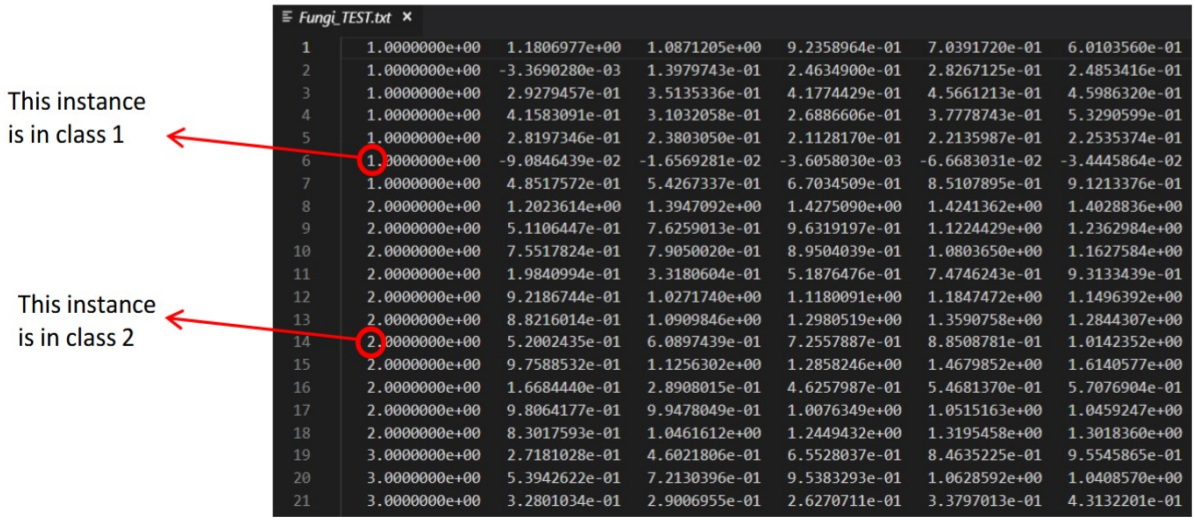

Figure 3.14: Example time series from the Fungi dataset in the UCR Archive, showing class-labeled training/test partitions [16].

Table 3.3: UCR Time Series Archive Datasets [16]

ID	Type	Name	Train	Test	Class	Length
1	Image	Adiac	390	391	37	176
2	Image	ArrowHead	36	175	3	251
3	Spectro	Beef	30	30	5	470
4	Image	BeetleFly	20	20	2	512
5	Image	BirdChicken	20	20	2	512
6	Sensor	Car	60	60	4	577
7	Simulated	CBF	30	900	3	128
8	Sensor	ChlorineConcentration	467	3840	3	166
9	Sensor	CinCECGTorso	40	1380	4	1639
10	Spectro	Coffee	28	28	2	286

Table 3.3 – continued from previous page

ID	Type	Name	Train	Test	Class	Length
11	Device	Computers	250	250	2	720
12	Motion	CricketX	390	390	12	300
13	Motion	CricketY	390	390	12	300
14	Motion	CricketZ	390	390	12	300
15	Image	DiatomSizeReduction	16	306	4	345
16	Image	DistalPhalanxOutlineAgeGroup	400	139	3	80
17	Image	DistalPhalanxOutlineCorrect	600	276	2	80
18	Image	DistalPhalanxTW	400	139	6	80
19	Sensor	Earthquakes	322	139	2	512
20	ECG	ECG200	100	100	2	96
21	ECG	ECG5000	500	4500	5	140
22	ECG	ECGFiveDays	23	861	2	136
23	Device	ElectricDevices	8926	7711	7	96
24	Image	FaceAll	560	1690	14	131
25	Image	FaceFour	24	88	4	350
26	Image	FacesUCR	200	2050	14	131
27	Image	FiftyWords	450	455	50	270
28	Image	Fish	175	175	7	463
29	Sensor	FordA	3601	1320	2	500
30	Sensor	FordB	3636	810	2	500
31	Motion	GunPoint	50	150	2	150
32	Spectro	Ham	109	105	2	431
33	Image	HandOutlines	1000	370	2	2709
34	Motion	Haptics	155	308	5	1092
35	Image	Herring	64	64	2	512
36	Motion	InlineSkate	100	550	7	1882
37	Sensor	InsectWingbeatSound	220	1980	11	256
38	Sensor	ItalyPowerDemand	67	1029	2	24
39	Device	LargeKitchenAppliances	375	375	3	720
40	Sensor	Lightning2	60	61	2	637
41	Sensor	Lightning7	70	73	7	319
42	Simulated	Mallat	55	2345	8	1024
43	Spectro	Meat	60	60	3	448
44	Image	MedicalImages	381	760	10	99
45	Image	MiddlePhalanxOutlineAgeGroup	400	154	3	80
46	Image	MiddlePhalanxOutlineCorrect	600	291	2	80
47	Image	MiddlePhalanxTW	399	154	6	80
48	Sensor	MoteStrain	20	1252	2	84

Table 3.3 – continued from previous page

ID	Type	Name	Train	Test	Class	Length
49	ECG	NonInvasiveFetalECGThorax1	1800	1965	42	750
50	ECG	NonInvasiveFetalECGThorax2	1800	1965	42	750
51	Spectro	OliveOil	30	30	4	570
52	Image	OSULeaf	200	242	6	427
53	Image	PhalangesOutlinesCorrect	1800	858	2	80
54	Sensor	Phoneme	214	1896	39	1024
55	Sensor	Plane	105	105	7	144
56	Image	ProximalPhalanxOutlineAgeGroup	400	205	3	80
57	Image	ProximalPhalanxOutlineCorrect	600	291	2	80
58	Image	ProximalPhalanxTW	400	205	6	80
59	Device	RefrigerationDevices	375	375	3	720
60	Device	ScreenType	375	375	3	720
61	Simulated	ShapeletSim	20	180	2	500
62	Image	ShapesAll	600	600	60	512
63	Device	SmallKitchenAppliances	375	375	3	720
64	Sensor	SonyAIBORobotSurface1	20	601	2	70
65	Sensor	SonyAIBORobotSurface2	27	953	2	65
66	Sensor	StarLightCurves	1000	8236	3	1024
67	Spectro	Strawberry	613	370	2	235
68	Image	SwedishLeaf	500	625	15	128
69	Image	Symbols	25	995	6	398
70	Simulated	SyntheticControl	300	300	6	60
71	Motion	ToeSegmentation1	40	228	2	277
72	Motion	ToeSegmentation2	36	130	2	343
73	Sensor	Trace	100	100	4	275
74	ECG	TwoLeadECG	23	1139	2	82
75	Simulated	TwoPatterns	1000	4000	4	128
76	Motion	UWaveGestureLibraryAll	896	3582	8	945
77	Motion	UWaveGestureLibraryX	896	3582	8	315
78	Motion	UWaveGestureLibraryY	896	3582	8	315
79	Motion	UWaveGestureLibraryZ	896	3582	8	315
80	Sensor	Wafer	1000	6164	2	152
81	Spectro	Wine	57	54	2	234
82	Image	WordSynonyms	267	638	25	270
83	Motion	Worms	181	77	5	900
84	Motion	WormsTwoClass	181	77	2	900
85	Image	Yoga	300	3000	2	426
86	Device	ACSF1	100	100	10	1460

Table 3.3 – continued from previous page

ID	Type	Name	Train	Test	Class	Length
87	Sensor	AllGestureWiimoteX	300	700	10	Vary
88	Sensor	AllGestureWiimoteY	300	700	10	Vary
89	Sensor	AllGestureWiimoteZ	300	700	10	Vary
90	Simulated	BME	30	150	3	128
91	Traffic	Chinatown	20	343	2	24
92	Image	Crop	7200	16800	24	46
93	Sensor	DodgerLoopDay	78	80	7	288
94	Sensor	DodgerLoopGame	20	138	2	288
95	Sensor	DodgerLoopWeekend	20	138	2	288
96	EOG	EOGHorizontalSignal	362	362	12	1250
97	EOG	EOGVerticalSignal	362	362	12	1250
98	Spectro	EthanolLevel	504	500	4	1751
99	Sensor	FreezerRegularTrain	150	2850	2	301
100	Sensor	FreezerSmallTrain	28	2850	2	301
101	HRM	Fungi	18	186	18	201
102	Trajectory	GestureMidAirD1	208	130	26	Vary
103	Trajectory	GestureMidAirD2	208	130	26	Vary
104	Trajectory	GestureMidAirD3	208	130	26	Vary
105	Sensor	GesturePebbleZ1	132	172	6	Vary
106	Sensor	GesturePebbleZ2	146	158	6	Vary
107	Motion	GunPointAgeSpan	135	316	2	150
108	Motion	GunPointMaleVersusFemale	135	316	2	150
109	Motion	GunPointOldVersusYoung	136	315	2	150
110	Device	HouseTwenty	40	119	2	2000
111	EPG	InsectEPGRegularTrain	62	249	3	601
112	EPG	InsectEPGSmallTrain	17	249	3	601
113	Traffic	MelbournePedestrian	1194	2439	10	24
114	Image	MixedShapesRegularTrain	500	2425	5	1024
115	Image	MixedShapesSmallTrain	100	2425	5	1024
116	Sensor	PickupGestureWiimoteZ	50	50	10	Vary
117	Hemodynamics	PigAirwayPressure	104	208	52	2000
118	Hemodynamics	PigArtPressure	104	208	52	2000
119	Hemodynamics	PigCVP	104	208	52	2000
120	Device	PLAID	537	537	11	Vary
121	Power	PowerCons	180	180	2	144
122	Spectrum	Rock	20	50	4	2844
123	Spectrum	SemgHandGenderCh2	300	600	2	1500
124	Spectrum	SemgHandMovementCh2	450	450	6	1500

Table 3.3 – continued from previous page

ID	Type	Name	Train	Test	Class	Length
125	Spectrum	SemgHandSubjectCh2	450	450	5	1500
126	Sensor	ShakeGestureWiimoteZ	50	50	10	Vary
127	Simulated	SmoothSubspace	150	150	3	15
128	Simulated	UMD	36	144	3	150

3.6 Conclusion:

This chapter has explored the fundamental concepts and techniques of time series classification, emphasizing its critical role in extracting meaningful patterns from temporal data. We began by defining time series classification and its practical applications, particularly in energy efficiency and smart home automation, where accurate classification of power consumption patterns can lead to significant sustainability benefits. Key to this process are dimensionality reduction techniques, which address the challenges of high-dimensional, noisy time series data. We examined three prominent methods: Piecewise Aggregate Approximation (PAA), which simplifies time series through segment averaging while preserving essential shape characteristics. Local Extrema (LE), which retains only critical points (peaks and valleys) to capture structural trends. Ramer-Douglas-Peucker (RDP), an iterative line-simplification algorithm that balances fidelity and compression. These techniques enable efficient processing and storage without sacrificing discriminative features, making them indispensable for large-scale time series analysis.

We then discussed distance and similarity measures, categorizing them into lock-step (e.g., Euclidean distance), elastic (e.g., DTW, LCSS), and also the SEA(Shape Exchange Algorithm). While lock-step measures are computationally efficient, elastic measures like DTW accommodate temporal misalignments, proving superior for applications with phase variations. The Shape Exchange Algorithm (SEA) further demonstrated how innovative signature-based matching can handle quasi-periodic patterns, offering robustness in domains like ECG analysis.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Introduction

This chapter presents a comprehensive empirical evaluation of various distance metrics for time series classification using the 1-Nearest Neighbor (1NN) classifier across 85 datasets from the UCR benchmark archive. The analysis systematically compares the performance of fundamental metrics (Euclidean, Manhattan, Canberra, EMD), SEA, MSR, elastic measures (DTW, LCSS), and our proposed hybrid approach (DTW_LCSS). Through extensive experimentation, we examine how different metrics perform across diverse dataset characteristics, including temporal alignment requirements, noise sensitivity, and shape complexity. The results are presented through comparative error rate tables, performance trend visualizations, and detailed analysis of method-specific strengths. Additionally, we introduce the application interface developed to facilitate practical implementation of these classification approaches.

4.2 Comparison of some basic metrics

This section empirically evaluates four fundamental distance metrics (Euclidean, Manhattan, Canberra, and Earth Mover's Distance) for time series classification using 1-Nearest Neighbor (1-

NN) classifiers across 85 UCR benchmark datasets. Results highlight significant performance variations based on dataset characteristics.

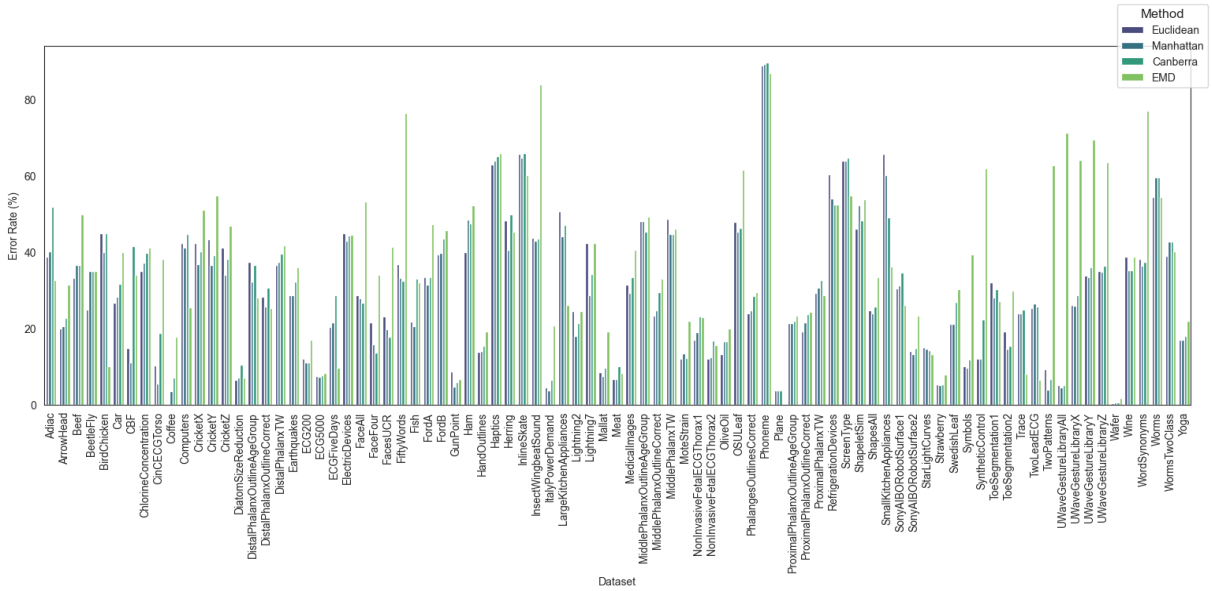

Figure 4.1: Comparative error rates of Euclidean, Manhattan, Canberra, and Earth Mover's Distance across 85 UCR datasets.

Table 4.1: Classification Error Rates of Basic Methods

Base name	Euclidean	Manhattan	Canberra	EMD
Adiac	38.87%	40.15%	51.92%	**32.74%**
ArrowHead	**20.00%**	20.57%	22.86%	31.43%
Beef	**33.33%**	36.67%	36.67%	50.00%
BeetleFly	**25.00%**	35.00%	35.00%	35.00%
BirdChicken	45.00%	40.00%	45.00%	**10.00%**
Car	**26.67%**	28.33%	31.67%	40.00%
CBF	14.78%	**11.11%**	41.56%	34.11%
ChlorineConcentration	**35.00%**	37.27%	39.79%	41.17%
CinCECGTorso	10.29%	**5.58%**	18.84%	38.33%
Coffee	**0.00%**	3.57%	7.14%	17.86%
Computers	42.40%	41.20%	44.80%	**25.60%**
CricketX	42.31%	**36.92%**	40.26%	51.03%
CricketY	43.33%	**36.67%**	39.23%	54.87%

Table 4.1: Classification Error Rates of Basic Methods (continued)

Base name	Euclidean	Manhattan	Canberra	EMD
CricketZ	41.28%	**34.10%**	38.21%	46.92%
DiatomSizeReduction	**6.54%**	7.19%	10.46%	7.19%
DistalPhalanxOutlineAgeGroup	37.41%	32.37%	36.69%	**28.06%**
DistalPhalanxOutlineCorrect	28.26%	25.72%	30.80%	**25.36%**
DistalPhalanxTW	**36.69%**	37.41%	39.57%	41.73%
Earthquakes	**28.78%**	**28.78%**	32.37%	35.97%
ECG200	12.00%	**11.00%**	**11.00%**	17.00%
ECG5000	7.51%	**7.40%**	7.62%	8.27%
ECGFiveDays	20.33%	21.49%	28.80%	**9.76%**
ElectricDevices	44.94%	**42.94%**	44.33%	44.68%
FaceAll	28.64%	27.87%	**26.75%**	53.37%
FaceFour	21.59%	15.91%	**13.64%**	34.09%
FacesUCR	23.07%	19.85%	**17.90%**	41.32%
FiftyWords	36.92%	33.19%	**32.53%**	76.48%
Fish	21.71%	**20.57%**	33.14%	32.00%
FordA	33.48%	**31.59%**	33.41%	47.35%
FordB	**39.38%**	39.88%	43.58%	45.68%
GunPoint	8.67%	**4.67%**	6.00%	6.67%
Ham	**40.00%**	48.57%	47.62%	52.38%
HandOutlines	**13.78%**	14.05%	15.41%	19.19%
Haptics	**62.99%**	63.96%	65.26%	65.91%
Herring	48.44%	**40.62%**	50.00%	45.31%
InlineSkate	65.82%	64.73%	66.00%	**60.18%**
InsectWingbeatSound	43.84%	**43.08%**	43.54%	83.99%
ItalyPowerDemand	4.47%	**3.79%**	6.61%	20.70%
LargeKitchenAppliances	50.67%	44.27%	47.20%	**26.13%**
Lightning2	24.59%	**18.03%**	21.31%	24.59%
Lightning7	42.47%	**28.77%**	34.25%	42.47%
Mallat	8.57%	**7.55%**	9.68%	19.23%
Meat	**6.67%**	**6.67%**	10.00%	8.33%
MedicalImages	31.58%	**29.34%**	33.42%	40.66%
MiddlePhalanxOutlineAgeGroup	48.05%	48.05%	**45.45%**	49.35%
MiddlePhalanxOutlineCorrect	**23.37%**	24.74%	29.55%	32.99%
MiddlePhalanxTW	48.70%	**44.81%**	**44.81%**	46.10%
MoteStrain	**12.14%**	13.42%	12.22%	21.96%

Table 4.1: Classification Error Rates of Basic Methods (continued)

Base name	Euclidean	Manhattan	Canberra	EMD
NonInvasiveFetalECGThorax1	**17.10%**	19.08%	23.21%	22.90%
NonInvasiveFetalECGThorax2	**12.01%**	12.52%	16.79%	15.57%
OliveOil	**13.33%**	16.67%	16.67%	20.00%
OSULeaf	47.93%	**45.45%**	46.28%	61.57%
PhalangesOutlinesCorrect	**23.89%**	24.83%	28.55%	29.60%
Phoneme	89.08%	89.40%	89.72%	**86.97%**
Plane	3.81%	3.81%	3.81%	**0.00%**
ProximalPhalanxOutlineAgeGroup	**21.46%**	**21.46%**	21.95%	23.41%
ProximalPhalanxOutlineCorrect	**19.24%**	21.65%	23.71%	24.40%
ProximalPhalanxTW	29.27%	30.73%	32.68%	**28.78%**
RefrigerationDevices	60.53%	54.13%	**52.53%**	**52.53%**
ScreenType	64.00%	64.00%	64.80%	**54.93%**
ShapeletSim	**46.11%**	52.22%	48.33%	53.89%
ShapesAll	24.83%	**24.00%**	25.83%	33.50%
SmallKitchenAppliances	65.87%	60.27%	49.07%	**36.27%**
SonyAIBORobotSurface1	30.45%	31.28%	34.61%	**26.12%**
SonyAIBORobotSurface2	14.06%	**13.22%**	14.90%	23.29%
StarLightCurves	15.12%	14.72%	14.17%	**13.20%**
Strawberry	5.41%	**5.14%**	5.41%	7.84%
SwedishLeaf	**21.12%**	**21.12%**	26.88%	30.24%
Symbols	10.05%	**9.65%**	11.96%	39.50%
SyntheticControl	**12.00%**	**12.00%**	22.33%	62.00%
ToeSegmentation1	32.02%	28.07%	30.26%	**27.19%**
ToeSegmentation2	19.23%	**14.62%**	15.38%	30.00%
Trace	24.00%	24.00%	25.00%	**8.00%**
TwoLeadECG	25.29%	26.51%	25.81%	**6.58%**
TwoPatterns	9.32%	**3.88%**	6.75%	62.80%
UWaveGestureLibraryAll	5.19%	**4.55%**	5.08%	71.33%
UWaveGestureLibraryX	26.07%	**25.88%**	28.70%	64.29%
UWaveGestureLibraryY	33.84%	**33.39%**	36.01%	69.63%
UWaveGestureLibraryZ	35.04%	**34.95%**	36.40%	63.71%
Wafer	**0.45%**	0.47%	0.60%	1.78%
Wine	38.89%	**35.19%**	**35.19%**	38.89%
WordSynonyms	38.24%	**36.52%**	37.46%	77.12%
Worms	**54.55%**	59.74%	59.74%	**54.55%**

Table 4.1: Classification Error Rates of Basic Methods (continued)

Base name	Euclidean	Manhattan	Canberra	EMD
WormsTwoClass	**38.96%**	42.86%	42.86%	40.26%
Yoga	**16.97%**	17.10%	18.03%	22.00%
Total Wins	**29**	**37**	**9**	**16**

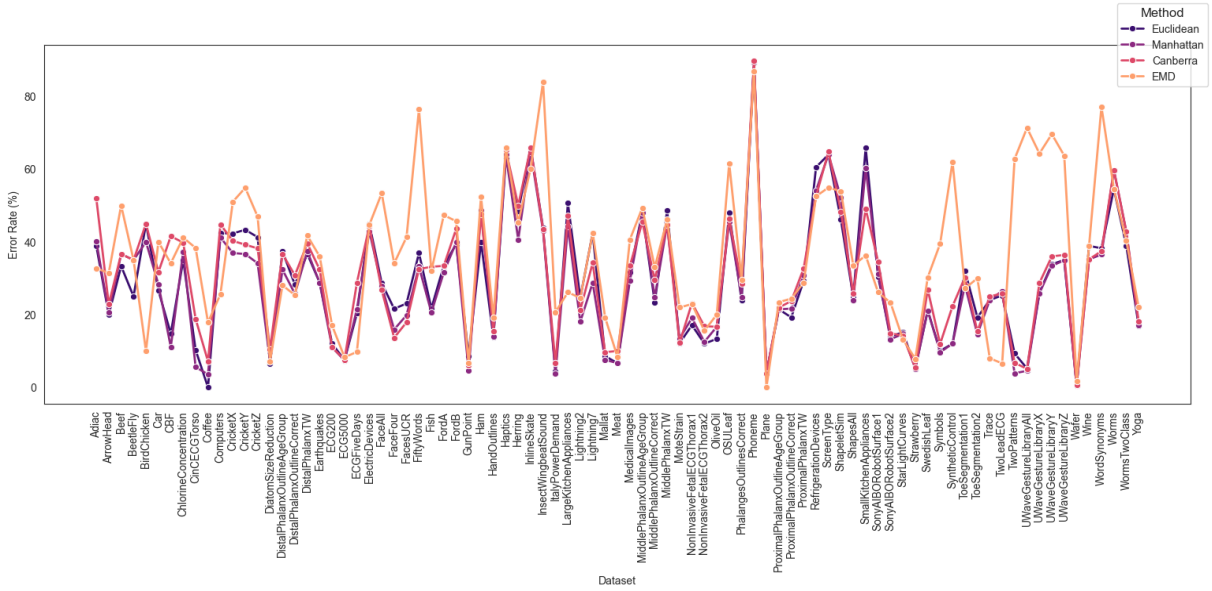

Figure 4.2: Error rate trends for Euclidean, Manhattan, Canberra, and EMD methods.

4.2.1 Discussion

The results shown in Table 4.1 and Figure 4.1 indicate that Manhattan distance achieved the lowest error rates in 37 out of 85 datasets (mean 43.5%), with particularly strong performance on datasets such as `GunPoint` (4.67% error) and `ElectricDevices` (42.94% error). Euclidean distance followed with 29 wins (34.1%), including optimal results on `Beef` (33.33% error) and `ArrowHead` (20.00% error). EMD demonstrated specialized effectiveness with 16 wins, most notably on `BirdChicken` (10.00% error versus 40-45% for other methods) and `TwoLeadECG` (6.58% error versus 25-26% for others), but performed poorly on datasets like `InsectWingbeatSound` (83.99% error). Canberra distance showed the most limited applicability with only 9 wins, though it achieved competitive

results on `FacesUCR` (17.90% error).

4.3 Data reduction based SEA algorithm

This section evaluates the Shape Exchange Algorithm (SEA) metric alongside two dimensionality-reduced variants: PAA_SEA (Piecewise Aggregate Approximation with 8 points/segment) and RDP_SEA (Ramer-Douglas-Peucker simplification with $\epsilon = 0.2$). As previously noted, FANSEA [7] represents another SEA-based approach that uses the FAN line simplification algorithm for data reduction.

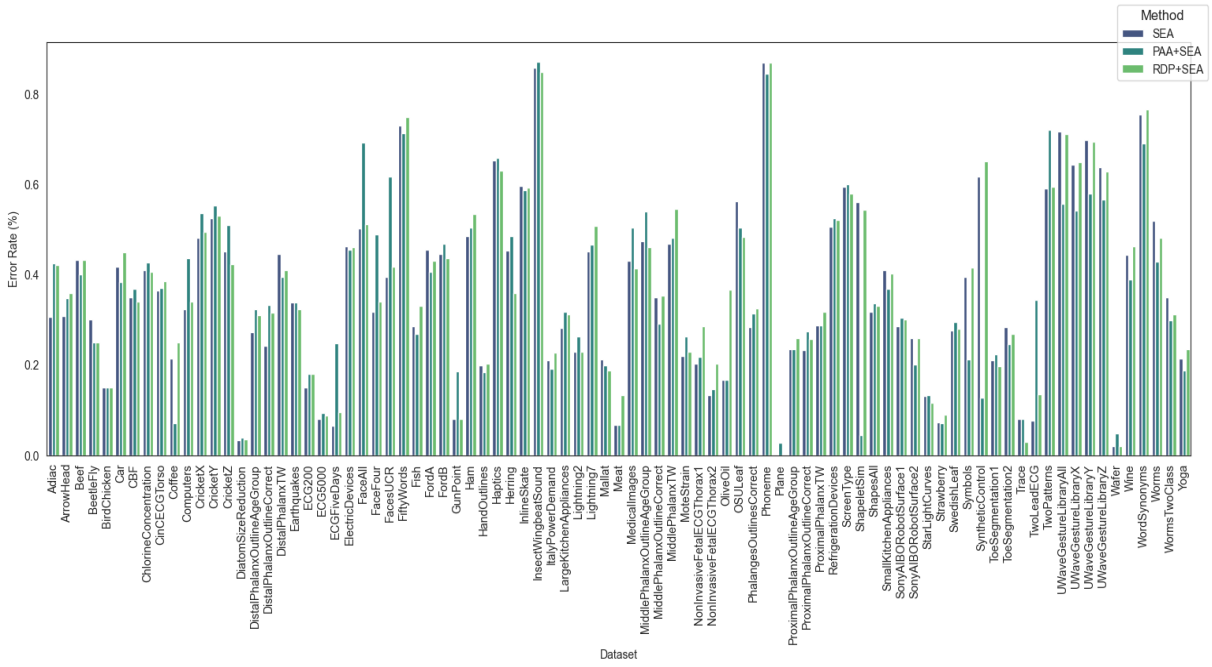

Figure 4.3: Error rate comparison: Original SEA vs. PAA_SEA vs. RDP_SEA across 85 UCR datasets.

Table 4.2: Classification Error Rates (SEA, PAA_SEA"8 Points/segment", RDP_SEA)

Base name	SEA	PAA+SEA	RDP+SEA
Adiac	**30.69%**	42.46%	42.20%
ArrowHead	**30.86%**	34.86%	36.00%

Table 4.2: Classification Error Rates (SEA, PAA_SEA"8 Points/segment", RDP_SEA)

Base name	SEA	PAA+SEA	RDP+SEA
Beef	43.33%	**40.00%**	43.33%
BeetleFly	30.00%	**25.00%**	**25.00%**
BirdChicken	**15.00%**	**15.00%**	**15.00%**
Car	41.67%	**38.33%**	45.00%
CBF	34.89%	36.89%	**34.11%**
ChlorineConcentration	40.96%	42.66%	**40.52%**
CinCECGTorso	**36.52%**	36.96%	38.48%
Coffee	21.43%	**7.14%**	25.00%
Computers	**32.40%**	43.60%	34.00%
CricketX	**48.21%**	53.59%	49.49%
CricketY	**52.56%**	55.38%	53.08%
CricketZ	45.13%	51.03%	**42.31%**
DiatomSizeReduction	**3.27%**	3.92%	3.59%
DistalPhalanxOutlineAgeGroup	**27.34%**	32.37%	30.94%
DistalPhalanxOutlineCorrect	**24.28%**	33.33%	31.52%
DistalPhalanxTW	44.60%	**39.57%**	41.01%
Earthquakes	33.81%	33.81%	**32.37%**
ECG200	**15.00%**	18.00%	18.00%
ECG5000	**8.13%**	9.31%	8.78%
ECGFiveDays	**6.62%**	24.74%	9.52%
ElectricDevices	46.19%	**45.43%**	46.08%
FaceAll	**50.18%**	69.17%	51.18%
FaceFour	**31.82%**	48.86%	34.09%
FacesUCR	**39.51%**	61.61%	41.66%
FiftyWords	72.97%	**71.21%**	74.95%
Fish	28.57%	**26.86%**	33.14%
FordA	45.53%	**40.61%**	43.11%
FordB	44.57%	46.79%	**43.70%**
GunPoint	**8.00%**	18.67%	**8.00%**
Ham	**48.57%**	50.48%	53.33%
HandOutlines	20.00%	**18.38%**	20.27%
Haptics	65.26%	65.91%	**62.99%**
Herring	45.31%	48.44%	**35.94%**
InlineSkate	59.64%	**58.73%**	59.27%
InsectWingbeatSound	85.81%	87.02%	**84.90%**

Table 4.2: Classification Error Rates (SEA, PAA_SEA"8 Points/segment", RDP_SEA)

Base name	SEA	PAA+SEA	RDP+SEA
ItalyPowerDemand	21.09%	**19.24%**	22.74%
LargeKitchenAppliances	**28.27%**	31.73%	31.20%
Lightning2	**22.95%**	26.23%	**22.95%**
Lightning7	**45.21%**	46.58%	50.68%
Mallat	21.28%	20.00%	**18.81%**
Meat	**6.67%**	**6.67%**	13.33%
MedicalImages	43.03%	50.39%	**41.45%**
MiddlePhalanxOutlineAgeGroup	47.40%	53.90%	**46.10%**
MiddlePhalanxOutlineCorrect	35.05%	**29.21%**	35.40%
MiddlePhalanxTW	**46.75%**	48.05%	54.55%
MoteStrain	**22.04%**	26.28%	23.00%
NonInvasiveFetalECGThorax1	**20.25%**	21.78%	28.60%
NonInvasiveFetalECGThorax2	**13.28%**	14.71%	20.25%
OliveOil	**16.67%**	**16.67%**	36.67%
OSULeaf	56.20%	50.41%	**48.35%**
PhalangesOutlinesCorrect	**28.44%**	31.47%	32.52%
Phoneme	86.87%	**84.49%**	86.87%
Plane	**0.00%**	2.86%	**0.00%**
ProximalPhalanxOutlineAgeGroup	**23.41%**	**23.41%**	25.85%
ProximalPhalanxOutlineCorrect	**23.37%**	27.49%	25.77%
ProximalPhalanxTW	**28.78%**	**28.78%**	31.71%
RefrigerationDevices	**50.67%**	52.53%	52.00%
ScreenType	59.47%	60.00%	**57.87%**
ShapeletSim	56.11%	**4.44%**	54.44%
ShapesAll	**31.83%**	33.67%	33.17%
SmallKitchenAppliances	41.07%	**36.80%**	40.27%
SonyAIBORobotSurface1	**28.62%**	30.45%	30.12%
SonyAIBORobotSurface2	25.92%	**20.15%**	25.92%
StarLightCurves	13.16%	13.39%	**11.69%**
Strawberry	7.30%	**7.03%**	8.92%
SwedishLeaf	**27.68%**	29.44%	28.00%
Symbols	39.40%	**21.31%**	41.51%
SyntheticControl	61.67%	**12.67%**	65.00%
ToeSegmentation1	21.05%	22.37%	**19.74%**
ToeSegmentation2	28.46%	**24.62%**	26.92%

Table 4.2: Classification Error Rates (SEA, PAA_SEA"8 Points/segment", RDP_SEA)

Base name	SEA	PAA+SEA	RDP+SEA
Trace	8.00%	8.00%	**3.00%**
TwoLeadECG	**7.73%**	34.33%	13.52%
TwoPatterns	**59.00%**	72.00%	59.40%
UWaveGestureLibraryAll	71.72%	**55.61%**	71.02%
UWaveGestureLibraryX	64.32%	**54.22%**	64.94%
UWaveGestureLibraryY	69.77%	**57.98%**	69.37%
UWaveGestureLibraryZ	63.82%	**56.70%**	62.73%
Wafer	**1.98%**	4.95%	2.06%
Wine	44.44%	**38.89%**	46.30%
WordSynonyms	75.39%	**69.12%**	76.49%
Worms	51.95%	**42.86%**	48.05%
WormsTwoClass	35.06%	**29.87%**	31.17%
Yoga	21.43%	**18.87%**	23.47%
Total Wins	**39**	**35**	**21**

4.3.1 Discussion

Table 4.2 demonstrates that while the original SEA method achieved superior performance with 39 wins out of 85 datasets achieves a mean success rate of (45.9%), including optimal results on `Plane` (0.00% error) and `ECGFiveDays` (6.62% error), its dimensionality-reduced variants still delivered competitive results.

The PAA_SEA approach, which reduces dimensionality by **representing each segment with 8 points** victories in 35 datasets (avg. 41.2%), coming remarkably close to the original method's success rate. This performance is particularly noteworthy given the significant reduction in data complexity, where each time series segment is approximated using only a sparse set of points.

The RDP_SEA method won fewer times (21 datasets, mean success rate of 24.7%) but still worked well in some cases. For example, on `CricketZ` data, its error was 42.31% compared to SEA 45.31%.

Both reduced-dimension methods exhibited expected performance degradation on complex datasets like `FaceAll`, where PAA_SEA error rose to 69.17% compared to SEA 50.18%.

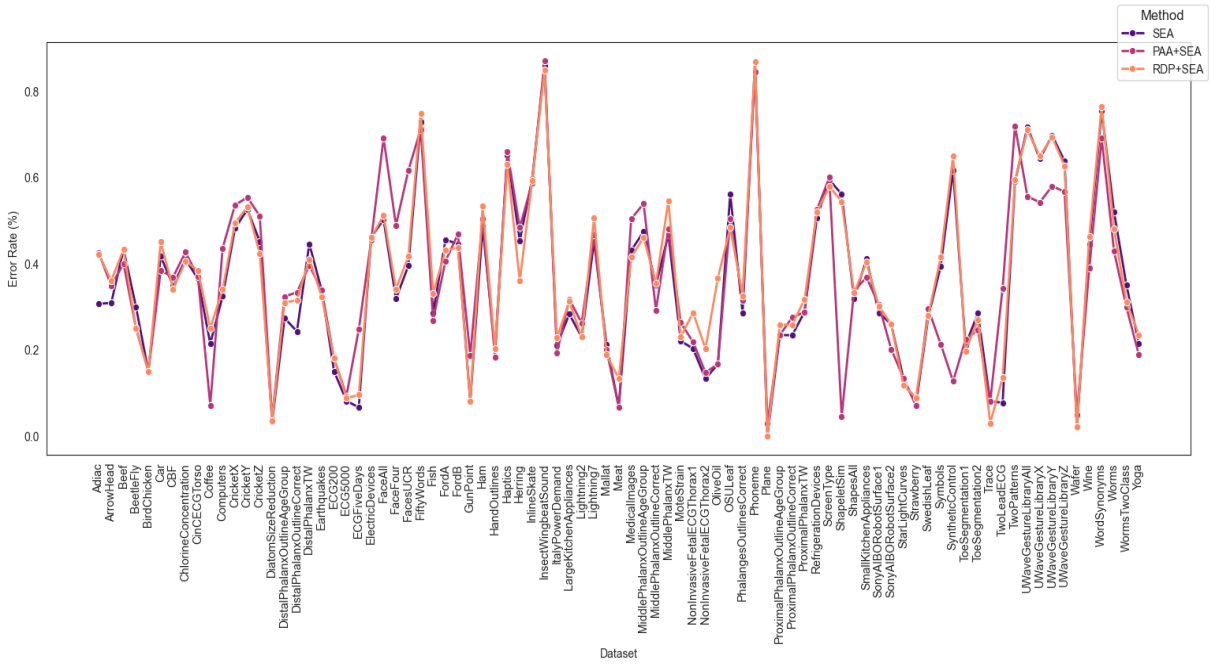

Figure 4.4: Error rate trends for SEA, PAA_SEA, and RDP_SEA methods.

Still, their good results on many datasets show that making data simpler can work well for some problems.

4.4 Comparison between DTW and some existing methods

This section presents a comprehensive benchmark of five prominent time series classification approaches: Dynamic Time Warping (DTW), Local Extrema DTW (LE-DTW), Piecewise Aggregate Approximation DTW (PAA-DTW), Shape Exchange Algorithm (SEA), and Longest Common Subsequence (LCSS, $\epsilon = 0.1$, Sakoe-Chiba radius=1). The results reveal method-specific strengths across diverse dataset characteristics.

Table 4.3: Classification Error Rates of DTW, LE-DTW, PAA-DTW, SEA, LCSS Methods

Base name	DTW	LE_DTW	PAA_DTW	SEA	LCSS
Adiac	39.64%	46.55%	45.01%	30.69%	56.01%
ArrowHead	29.71%	34.86%	23.43%	30.86%	24.57%
Beef	36.67%	30.00%	46.67%	43.33%	30.00%

Table 4.3: Classification Error Rates of DTW, LE-DTW, PAA-DTW, SEA, LCSS Methods (continued)

Base name	DTW	LE_DTW	PAA_DTW	SEA	LCSS
BeetleFly	30.00%	10.00%	35.00%	30.00%	30.00%
BirdChicken	25.00%	15.00%	25.00%	15.00%	45.00%
Car	26.67%	33.33%	38.33%	41.67%	28.33%
CBF	0.33%	0.11%	5.78%	34.89%	12.11%
ChlorineConcentration	35.16%	37.19%	40.68%	40.96%	36.22%
CinCECGTorso	34.93%	31.52%	32.54%	36.52%	23.99%
Coffee	0.00%	10.71%	0.00%	21.43%	3.57%
Computers	30.00%	26.40%	42.40%	32.40%	37.20%
CricketX	24.62%	26.92%	30.77%	48.21%	53.33%
CricketY	25.64%	31.79%	32.82%	52.56%	45.64%
CricketZ	24.62%	27.69%	54.10%	45.13%	49.74%
DiatomSizeReduction	3.27%	10.46%	6.21%	3.27%	7.19%
DistalPhalanxOutlineAgeGroup	23.02%	34.53%	35.97%	27.34%	29.50%
DistalPhalanxOutlineCorrect	28.26%	29.71%	31.16%	24.28%	29.35%
DistalPhalanxTW	41.01%	39.57%	42.45%	44.60%	31.65%
Earthquakes	28.06%	28.06%	28.06%	33.81%	33.81%
ECG200	23.00%	23.00%	16.00%	15.00%	13.00%
ECG5000	7.56%	7.87%	8.11%	8.13%	7.40%
ECGFiveDays	23.23%	24.16%	18.12%	6.62%	28.57%
ElectricDevices	40.41%	36.91%	44.24%	46.19%	39.39%
FaceAll	19.23%	28.82%	36.15%	50.18%	27.75%
FaceFour	17.05%	15.91%	14.77%	31.82%	10.23%
FacesUCR	9.51%	19.22%	36.15%	39.51%	18.44%
FiftyWords	30.99%	43.52%	32.97%	72.97%	40.88%
Fish	17.71%	17.71%	20.00%	28.57%	22.29%
FordA	44.55%	39.70%	37.05%	45.53%	35.45%
FordB	38.02%	41.11%	38.77%	44.57%	39.01%
GunPoint	9.33%	4.00%	11.33%	8.00%	2.67%
Ham	53.33%	46.67%	44.76%	48.57%	42.86%
Haptic	62.34%	55.19%	60.06%	65.26%	67.21%
HandOutlines	11.89%	33.24%	14.59%	20.00%	14.32%
Herring	46.88%	42.19%	51.56%	45.31%	48.44%
InlineSkate	61.64%	59.09%	60.91%	59.64%	66.55%
InsectWingbeatSound	64.49%	74.14%	67.78%	85.81%	44.65%

Table 4.3: Classification Error Rates of DTW, LE-DTW, PAA-DTW, SEA, LCSS Methods (continued)

Base name	DTW	LE_DTW	PAA_DTW	SEA	LCSS
ItalyPowerDemand	4.96%	6.22%	13.31%	21.09%	12.63%
LargeKitchenAppliances	20.53%	19.73%	34.40%	28.27%	48.80%
Lightning2	13.11%	9.84%	14.75%	22.95%	18.03%
Lightning7	27.40%	23.29%	30.14%	45.21%	45.21%
Mallat	6.61%	8.10%	10.45%	21.28%	20.85%
Meat	6.67%	26.67%	6.67%	6.67%	53.33%
MedicalImages	26.32%	25.79%	35.39%	43.03%	37.11%
MiddlePhalanxOutlineAgeGroup	50.00%	52.60%	52.60%	47.40%	46.10%
MiddlePhalanxOutlineCorrect	30.24%	29.55%	29.55%	35.05%	29.21%
MiddlePhalanxTW	49.35%	55.19%	48.05%	46.75%	45.45%
MoteStrain	16.53%	16.93%	19.41%	22.04%	18.05%
NonInvasiveFetalECGThorax1	20.97%	31.76%	22.04%	20.25%	21.48%
NonInvasiveFetalECGThorax2	13.54%	21.90%	27.89%	13.28%	13.74%
OliveOil	16.67%	13.33%	16.67%	16.67%	80.00%
OSULeaf	40.91%	35.54%	44.21%	56.20%	50.41%
PhalangesOutlinesCorrect	27.16%	28.79%	30.42%	28.44%	28.44%
Phoneme	77.16%	79.96%	82.12%	86.87%	89.14%
Plane	0.00%	0.95%	2.86%	0.00%	2.86%
ProximalPhalanxOutlineAgeGroup	19.51%	20.49%	22.93%	23.41%	24.39%
ProximalPhalanxOutlineCorrect	21.65%	25.43%	26.12%	23.37%	23.02%
ProximalPhalanxTW	24.39%	26.83%	34.63%	28.78%	26.34%
RefrigerationDevices	53.60%	47.47%	56.53%	50.67%	56.80%
ScreenType	60.27%	51.73%	66.13%	59.47%	62.13%
ShapeletSim	35.00%	45.00%	28.33%	56.11%	41.67%
ShapesAll	23.17%	27.00%	23.50%	31.83%	28.67%
SmallKitchenAppliances	35.73%	34.67%	37.33%	41.07%	63.73%
SonyAIBORobotSurface1	27.45%	29.12%	28.62%	28.62%	33.11%
SonyAIBORobotSurface2	16.89%	25.08%	16.47%	25.92%	17.73%
StarLightCurves	9.34%	5.53%	11.41%	13.16%	15.12%
Strawberry	5.95%	4.86%	4.86%	7.30%	11.08%
SwedishLeaf	20.80%	20.16%	21.92%	27.68%	17.28%
Symbols	5.03%	14.27%	6.13%	39.40%	16.88%
SyntheticControl	0.67%	3.67%	5.33%	61.67%	20.67%
ToeSegmentation1	22.81%	16.23%	19.74%	21.05%	28.95%

Table 4.3: Classification Error Rates of DTW, LE-DTW, PAA-DTW, SEA, LCSS Methods (continued)

Base name	DTW	LE_DTW	PAA_DTW	SEA	LCSS
ToeSegmentation2	16.15%	14.62%	17.69%	28.46%	13.08%
Trace	0.00%	2.00%	3.00%	8.00%	19.00%
TwoLeadECG	9.57%	19.49%	29.50%	7.73%	23.18%
TwoPatterns	0.00%	68.67%	0.60%	59.00%	10.80%
UWaveGestureLibraryAll	10.83%	12.56%	8.99%	71.72%	9.60%
UWaveGestureLibraryX	27.25%	31.83%	29.87%	64.32%	34.17%
UWaveGestureLibraryY	36.60%	38.30%	38.53%	69.77%	41.60%
UWaveGestureLibraryZ	34.17%	37.44%	34.59%	63.82%	40.28%
Wafer	2.01%	1.93%	2.11%	1.98%	1.33%
Wine	42.59%	42.59%	38.89%	44.44%	50.00%
WordSynonyms	35.11%	46.87%	40.28%	75.39%	46.71%
Worms	41.56%	41.56%	45.45%	51.95%	59.74%
WormsTwoClass	37.66%	28.57%	36.36%	35.06%	44.16%
Yoga	16.37%	18.83%	16.13%	21.43%	18.87%
Total Wins	**37**	**25**	**10**	**9**	**18**

4.4.1 Discussion

Table 4.3 shows DTW achieved the lowest error rates in 37 out of 85 datasets (43.5%), with perfect 0.00% error on `Coffee` and `Trace`, and strong performance on warped series like `CricketX` (24.62% error). LE-DTW followed with 25 wins (29.4%), including improved results on extremum-rich datasets such as `Lightning2` (9.84% error versus DTW 13.11%). LCSS demonstrated specialized effectiveness with 18 wins (21.2%), particularly on noisy datasets like `InsectWingbeatSound` (44.65% error versus DTW's 64.49%). PAA-DTW and SEA showed more limited applicability with 10 and 9 wins respectively.

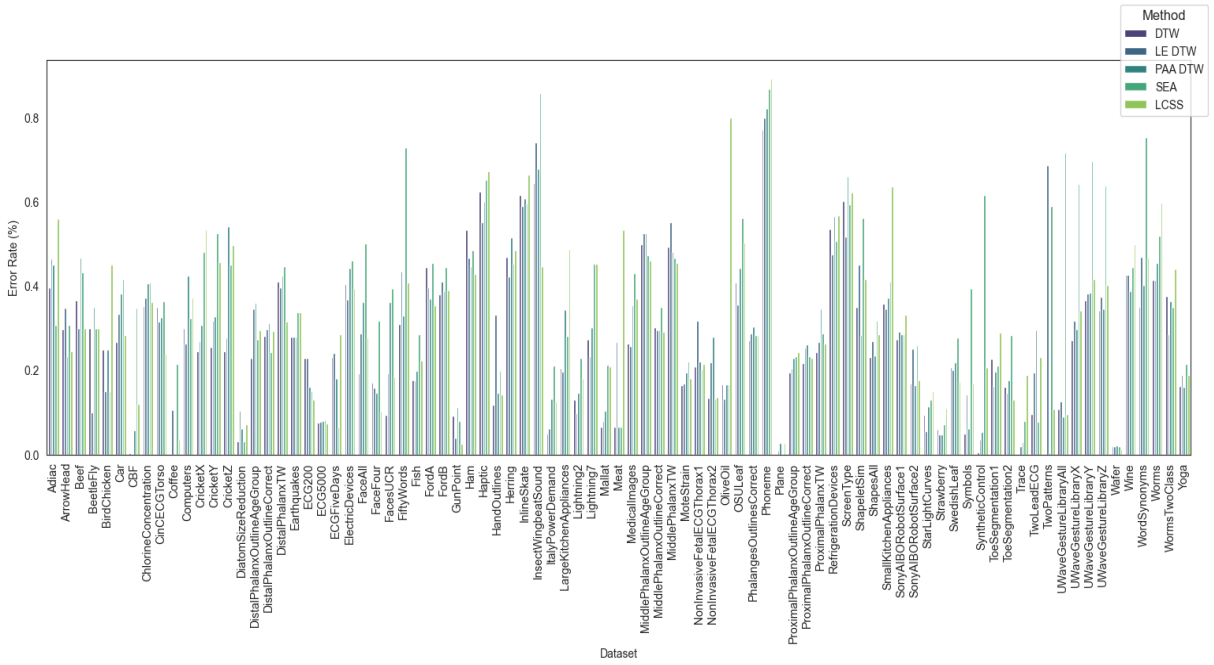

Figure 4.5: Comparative error rates of DTW, LE-DTW, PAA-DTW, SEA, and LCSS across 85 UCR datasets.

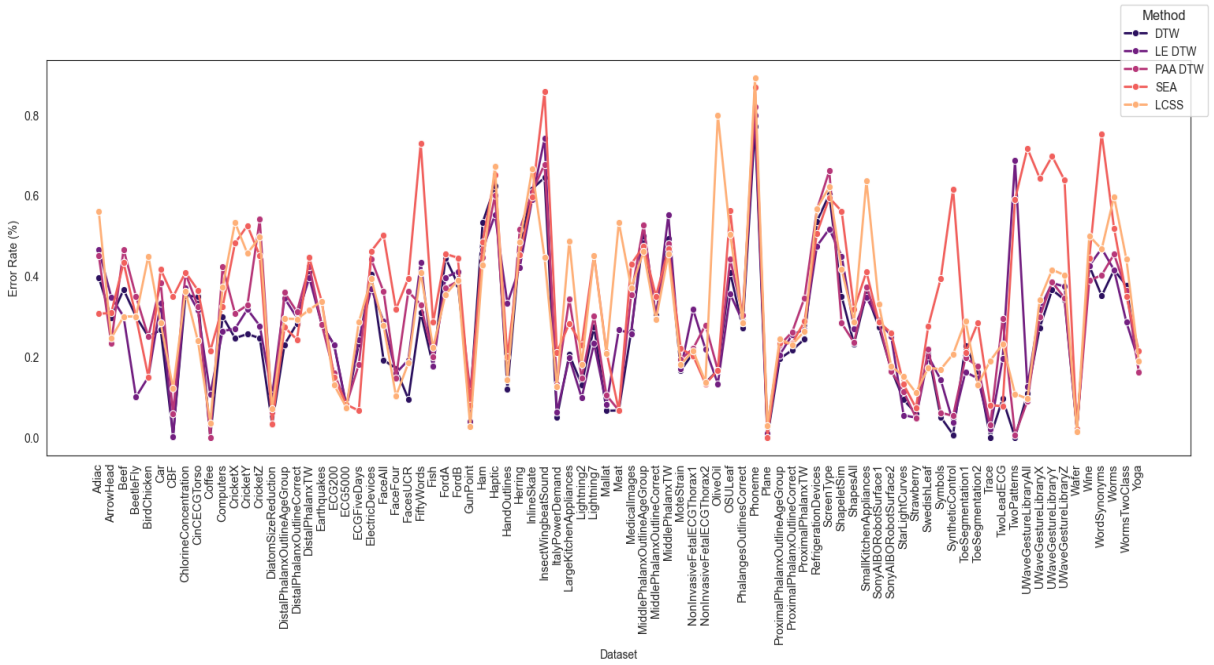

Figure 4.6: Error rate trends for DTW, LE-DTW, PAA-DTW, SEA, and LCSS methods.

4.5 Combination of DTW and LCSS

The Hybrid DTW-LCSS distance measure combines the temporal alignment capability of Dynamic Time Warping (DTW) with the noise robustness of the Longest Common Subsequence (LCSS) similarity measure. This fusion creates a more versatile distance metric for time series comparison.

4.5.1 Definition

Given two time series $X = (x_1, \ldots, x_n)$ and $Y = (y_1, \ldots, y_m)$ of lengths n and m respectively, the hybrid distance $D_H(X, Y)$ is computed as:

$$D_H(X, Y) = \alpha \cdot D_{\text{DTW}}(X, Y) + (1 - \alpha) \cdot D_{\text{LCSS}}(X, Y) \tag{4.1}$$

where:

- $\alpha \in [0, 1]$ is the weighting parameter
- D_{DTW} is the DTW distance
- $D_{\text{LCSS}} = 1 - S_{\text{LCSS}}$ is the LCSS-based distance
- S_{LCSS} is the normalized LCSS similarity score

It is worth mentioning that the idea of combining the DTW and LCSS methods already exists [26], but it has been applied in a different way. The paper proposes a parametric dissimilarity measure called $\text{DD}_{\text{DTW}}^{\text{LCSS}}$, which combines DTW, derivative DTW (DDTW), and LCSS into a single weighted measure. The method uses three parameters (a, b, c) to balance the contributions of each component: DTW for point-to-point distances, DDTW for shape information via derivatives, and LCSS for robustness to outliers [26].

Algorithm 1 DTW-LCSS Hybrid Distance

Require:

$X = (x_1, \ldots, x_n), Y = (y_1, \ldots, y_m)$

$\alpha \in [0, 1], \epsilon > 0, \delta \in \mathbb{N}$

Ensure: $D_{\text{hybrid}} \in \mathbb{R}^+$

DTW Distance:
$C \leftarrow (n+1) \times (m+1)$ matrix
$C_{0,0} \leftarrow 0, C_{i,0} \leftarrow \infty, C_{0,j} \leftarrow \infty$
for $i \leftarrow 1$ to n **do**
  for $j \leftarrow 1$ to m **do**
    $C_{i,j} \leftarrow |x_i - y_j| + \min(C_{i-1,j}, C_{i,j-1}, C_{i-1,j-1})$
  end for
end for
$D_{\text{DTW}} \leftarrow C_{n,m}$

LCSS Similarity:
$L \leftarrow (n+1) \times (m+1)$ matrix
$L_{0,j} \leftarrow 0, L_{i,0} \leftarrow 0$
for $i \leftarrow 1$ to n **do**
  for $j \leftarrow \max(1, i-\delta)$ to $\min(m, i+\delta)$ **do**
    if $|x_i - y_j| \leq \epsilon$ **then**
      $L_{i,j} \leftarrow 1 + L_{i-1,j-1}$
    else
      $L_{i,j} \leftarrow \max(L_{i-1,j}, L_{i,j-1})$
    end if
  end for
end for
$S_{\text{LCSS}} \leftarrow \frac{L_{n,m}}{\min(n,m)}$

Combination:
$D_{\text{hybrid}} \leftarrow \alpha D_{\text{DTW}} + (1-\alpha)(1 - S_{\text{LCSS}})$
return D_{hybrid}

4.5.2 *DTW_LCSS* comparison

This section evaluates the performance of our DTW_LCSS method against three key distance measures: Dynamic Time Warping (DTW), Longest Common Subsequence (LCSS), and Local Extrema DTW (LE_DTW). The comparison includes classification error rates across 85 datasets from UCR archive, with green cells highlighting the best-performing method for each case.

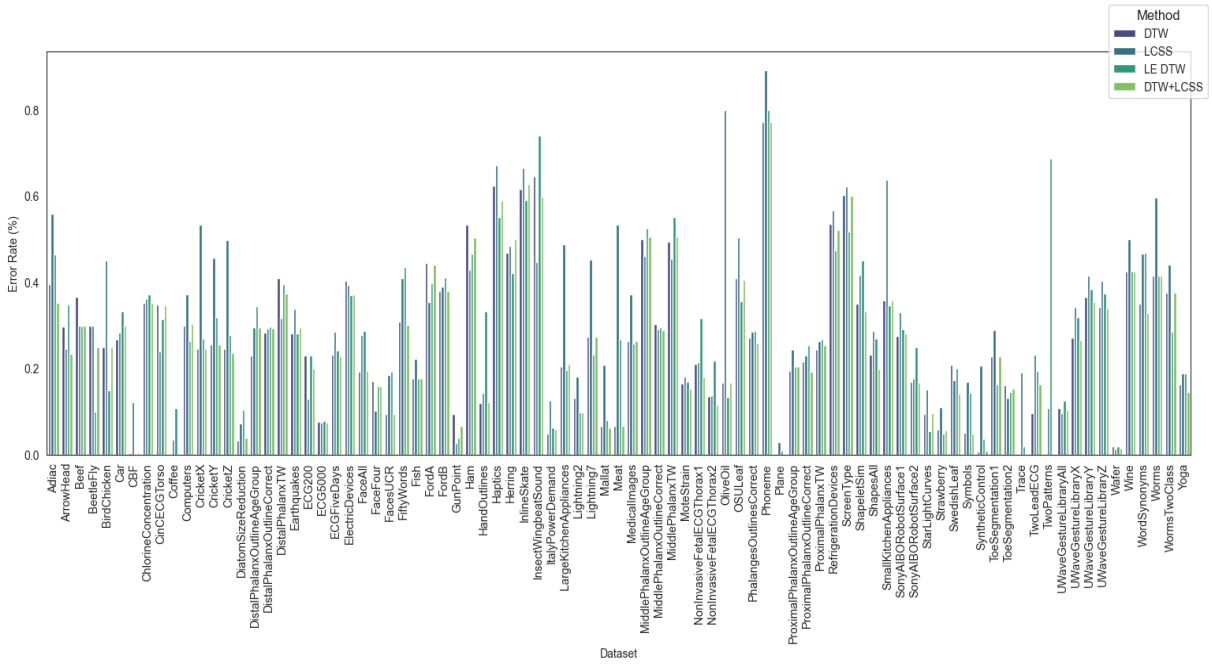

Figure 4.7: Comparative error rates of DTW, LE-DTW, LCSS, and hybrid DTW_LCSS across 85 UCR datasets.

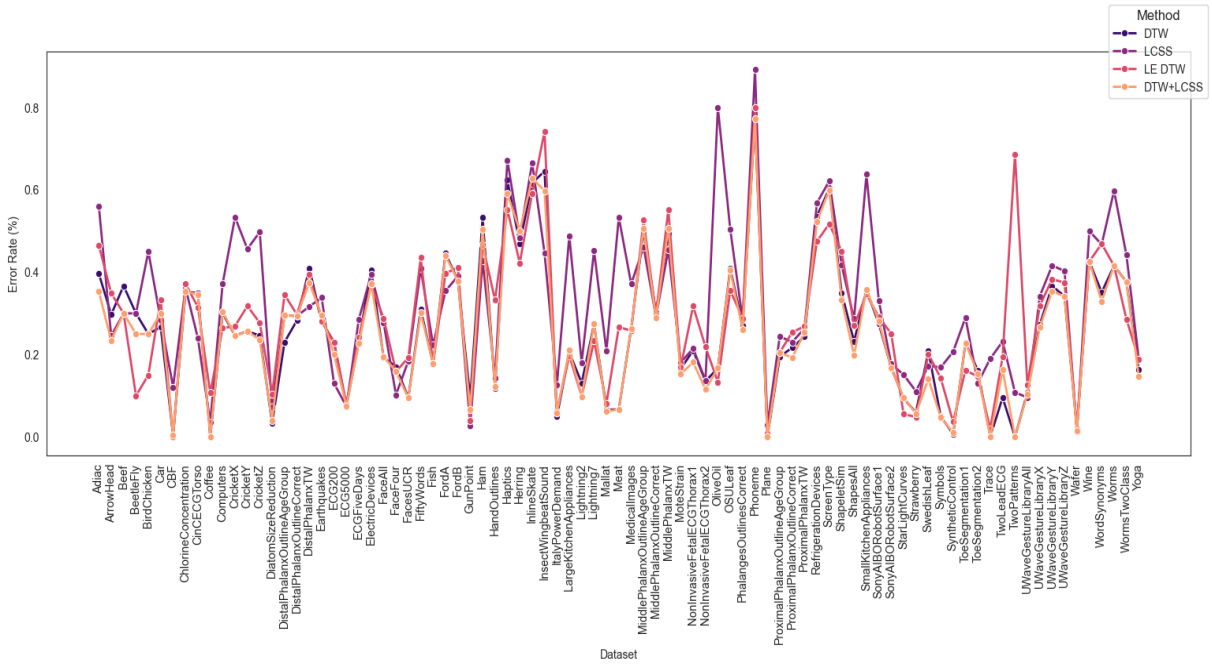

Figure 4.8: Error rate trends for DTW, LE-DTW, LCSS, and hybrid DTW_LCSS methods.

Table 4.4: Classification Error Rates Comparison (DTW vs. LCSS vs. LE_DTW vs. DTW+LCSS)

Base name	DTW	LCSS	LE_DTW	DTW+LCSS
Adiac	39.64%	56.01%	46.55%	35.29%
ArrowHead	29.71%	24.57%	34.86%	23.43%
Beef	36.67%	30.00%	30.00%	30.00%
BeetleFly	30.00%	30.00%	10.00%	25.00%
BirdChicken	25.00%	45.00%	15.00%	25.00%
Car	26.67%	28.33%	33.33%	30.00%
CBF	0.33%	12.11%	0.11%	0.44%
ChlorineConcentration	35.16%	36.22%	37.19%	35.26%
CinCECGTorso	34.93%	23.99%	31.52%	34.57%
Coffee	0.00%	3.57%	10.71%	0.00%
Computers	30.00%	37.20%	26.40%	30.40%
CricketX	24.62%	53.33%	26.92%	24.62%
CricketY	25.64%	45.64%	31.79%	25.64%
CricketZ	24.62%	49.74%	27.69%	23.59%
DiatomSizeReduction	3.27%	7.19%	10.46%	3.92%
DistalPhalanxOutlineAgeGroup	23.02%	29.50%	34.53%	29.50%
DistalPhalanxOutlineCorrect	28.26%	29.35%	29.71%	29.35%
DistalPhalanxTW	41.01%	31.65%	39.57%	37.41%
Earthquakes	28.06%	33.81%	28.06%	29.50%
ECG200	23.00%	13.00%	23.00%	20.00%
ECG5000	7.56%	7.40%	7.87%	7.47%
ECGFiveDays	23.23%	28.57%	24.16%	22.76%
ElectricDevices	40.41%	39.39%	36.91%	37.12%
FaceAll	19.23%	27.75%	28.82%	19.47%
FaceFour	17.05%	10.23%	15.91%	15.91%
FacesUCR	9.51%	18.44%	19.22%	9.46%
FiftyWords	30.99%	40.88%	43.52%	30.11%
Fish	17.71%	22.29%	17.71%	17.71%
FordA	44.55%	35.45%	39.70%	44.02%
FordB	38.02%	39.01%	41.11%	37.90%
GunPoint	9.33%	2.67%	4.00%	6.67%
Ham	53.33%	42.86%	46.67%	50.48%
HandOutlines	11.89%	14.32%	33.24%	12.16%
Haptics	62.34%	67.21%	55.19%	59.09%
Herring	46.88%	48.44%	42.19%	50.00%

Table 4.4: Classification Error Rates Comparison (continued)

Base name	DTW	LCSS	LE_DTW	DTW+LCSS
InlineSkate	61.64%	66.55%	59.09%	62.73%
InsectWingbeatSound	64.49%	44.65%	74.14%	59.80%
ItalyPowerDemand	4.96%	12.63%	6.22%	5.93%
LargeKitchenAppliances	20.53%	48.80%	19.73%	21.07%
Lightning2	13.11%	18.03%	9.84%	9.84%
Lightning7	27.40%	45.21%	23.29%	27.40%
Mallat	6.61%	20.85%	8.10%	6.23%
Meat	6.67%	53.33%	26.67%	6.67%
MedicalImages	26.32%	37.11%	25.79%	26.32%
MiddlePhalanxOutlineAgeGroup	50.00%	46.10%	52.60%	50.65%
MiddlePhalanxOutlineCorrect	30.24%	29.21%	29.55%	28.87%
MiddlePhalanxTW	49.35%	45.45%	55.19%	50.65%
MoteStrain	16.53%	18.05%	16.93%	15.26%
NonInvasiveFetalECGThorax1	20.97%	21.48%	31.76%	18.12%
NonInvasiveFetalECGThorax2	13.54%	13.74%	21.90%	11.55%
OliveOil	16.67%	80.00%	13.33%	16.67%
OSULeaf	40.91%	50.41%	35.54%	40.50%
PhalangesOutlinesCorrect	27.16%	28.44%	28.79%	25.99%
Phoneme	77.16%	89.14%	79.96%	77.22%
Plane	0.00%	2.86%	0.95%	0.00%
ProximalPhalanxOutlineAgeGroup	19.51%	24.39%	20.49%	20.49%
ProximalPhalanxOutlineCorrect	21.65%	23.02%	25.43%	19.24%
ProximalPhalanxTW	24.39%	26.34%	26.83%	25.37%
RefrigerationDevices	53.60%	56.80%	47.47%	52.27%
ScreenType	60.27%	62.13%	51.73%	60.00%
ShapeletSim	35.00%	41.67%	45.00%	33.33%
ShapesAll	23.17%	28.67%	27.00%	19.83%
SmallKitchenAppliances	35.73%	63.73%	34.67%	35.73%
SonyAIBORobotSurface1	27.45%	33.11%	29.12%	28.12%
SonyAIBORobotSurface2	16.89%	17.73%	25.08%	16.68%
StarLightCurves	9.34%	15.12%	5.53%	9.60%
Strawberry	5.95%	11.08%	4.86%	5.68%
SwedishLeaf	20.80%	17.28%	20.16%	14.08%
Symbols	5.03%	16.88%	14.27%	4.82%
SyntheticControl	0.67%	20.67%	3.67%	1.00%

Table 4.4: Classification Error Rates Comparison (continued)

Base name	DTW	LCSS	LE_DTW	DTW+LCSS
ToeSegmentation1	22.81%	28.95%	16.23%	22.81%
ToeSegmentation2	16.15%	13.08%	14.62%	15.38%
Trace	0.00%	19.00%	2.00%	0.00%
TwoLeadECG	9.57%	23.18%	19.49%	16.33%
TwoPatterns	0.00%	10.80%	68.67%	0.00%
UWaveGestureLibraryAll	10.83%	9.60%	12.56%	10.44%
UWaveGestureLibraryX	27.25%	34.17%	31.83%	26.58%
UWaveGestureLibraryY	36.60%	41.60%	38.30%	35.40%
UWaveGestureLibraryZ	34.17%	40.28%	37.44%	34.09%
Wafer	2.01%	1.33%	1.93%	1.56%
Wine	42.59%	50.00%	42.59%	42.59%
WordSynonyms	35.11%	46.71%	46.87%	32.92%
Worms	41.56%	59.74%	41.56%	41.56%
WormsTwoClass	37.66%	44.16%	28.57%	37.66%
Yoga	16.37%	18.87%	18.83%	14.60%
Total wins:	**25**	**15**	**26**	**36**

4.5.3 Discussion

The hybrid DTW_LCSS method outperformed all individual metrics with 36 wins out of 85 datasets (42.4%), as shown in Table 4.4. It achieved superior results on diverse datasets including `ShapesAll` (19.83% error versus DTW 23.17%) and `UWaveGestureLibraryX` (26.58% error versus DTW 27.25% and LCSS 34.17%). The method maintained DTW strength on aligned series while improving LCSS noise robustness, though it showed slightly higher error on some datasets like `OliveOil` (16.67% versus LE-DTW 13.33%).

4.6 *MSR vs. SEA* comparison

This section compares the performance of the Mean Squared Residue (MSR) method against the Shape Exchange Algorithm (SEA) across 85 UCR benchmark datasets. MSR, as a distance metric, measures the average squared difference between corresponding points in time series after optimal

alignment. The classification error rates reveal significant differences in effectiveness depending on dataset characteristics.

4.6.1 The Mean Squared Residue (MSR)

Cheng and Church [12] proposed an efficient node-deletion algorithm for biclustering gene expression data [12]. Their method aims to identify the maximum bicluster with a minimized Mean Squared Residue (MSR) score [12] [8].

The MSR H of a bicluster with row set I and column set J is calculated as follows:

$$H(I, J) = \frac{1}{|I||J|} \sum_{i \in I, j \in J} (a_{ij} - a_{iJ} - a_{Ij} + a_{IJ})^2 \tag{4.2}$$

where:

$$a_{iJ} = \frac{1}{|J|} \sum_{j \in J} a_{ij} \tag{4.3}$$

$$a_{Ij} = \frac{1}{|I|} \sum_{i \in I} a_{ij} \tag{4.4}$$

$$a_{IJ} = \frac{1}{|I||J|} \sum_{i \in I, j \in J} a_{ij} \tag{4.5}$$

4.6.2 Application of MSR to time series classification

The Mean Squared Residue (MSR) metric is applied to quantify the dissimilarity between pairs of time series using the distance Equation 4.7. Given a query series Q and another series C from the training database, the MSR distance is computed through the following steps:

1. **Bicluster Construction**

   The two series Q and C are stacked vertically to form a $2 \times T$ matrix (where T is the series length):

   $$\text{bicluster} = \begin{bmatrix} Q \\ C \end{bmatrix} = \begin{bmatrix} q_1 & q_2 & \cdots & q_T \\ c_1 & c_2 & \cdots & c_T \end{bmatrix} \tag{4.6}$$

2. **Residue Calculation**

   Compute the following means:

   - Row means: $\text{row\_mean}[i] = \frac{1}{T} \sum_{j=1}^{T} \text{bicluster}[i, j]$
   - Column means: $\text{col\_mean}[j] = \frac{1}{2} \sum_{i=1}^{2} \text{bicluster}[i, j]$
   - Global mean: $\text{overall\_mean} = \frac{1}{2T} \sum_{i=1}^{2} \sum_{j=1}^{T} \text{bicluster}[i, j]$

3. **MSR Computation**

   The MSR distance is calculated as:

$$\text{MSR}(Q, C) = \frac{1}{2T} \sum_{i=1}^{2} \sum_{j=1}^{T} (\text{bicluster}[i, j] - \text{row\_mean}[i] - \text{col\_mean}[j] + \text{overall\_mean})^2 \tag{4.7}$$

For classification using the 1-Nearest Neighbor (1-NN) classifier:

- Compute MSR distances between Q and all training series using equation (4.7);
- Assign Q the class label of the training series with the minimum MSR distance.

Table 4.5: Classification Error Rates Comparison (SEA vs MSR)

Base Name	SEA	MSR
Adiac	30.69%	38.87%
ArrowHead	30.86%	20.00%
Beef	43.33%	33.33%
BeetleFly	30.00%	25.00%
BirdChicken	15.00%	45.00%
Car	41.67%	26.67%
CBF	34.89%	14.78%
ChlorineConcentration	40.96%	35.00%
CinCECGTorso	36.52%	10.29%
Coffee	21.43%	0.00%
Computers	32.40%	42.40%
CricketX	48.21%	42.31%

Table 4.5: Classification Error Rates Comparison (SEA vs MSR) (continued)

Base Name	SEA	MSR
CricketY	52.56%	43.33%
CricketZ	45.13%	41.28%
DiatomSizeReduction	3.27%	6.54%
DistalPhalanxOutlineAgeGroup	27.34%	37.41%
DistalPhalanxOutlineCorrect	24.28%	28.26%
DistalPhalanxTW	44.60%	36.69%
Earthquakes	33.81%	28.78%
ECG200	15.00%	12.00%
ECG5000	8.13%	7.51%
ECGFiveDays	6.62%	20.33%
ElectricDevices	46.19%	44.92%
FaceAll	50.18%	28.64%
FaceFour	31.82%	21.59%
FacesUCR	39.51%	23.07%
FiftyWords	72.97%	36.92%
Fish	28.57%	21.71%
FordA	45.53%	33.48%
FordB	44.57%	39.38%
GunPoint	8.00%	8.67%
Ham	48.57%	40.00%
HandOutlines	20.00%	13.78%
Haptics	65.26%	62.99%
Herring	45.31%	48.44%
InlineSkate	59.64%	65.82%
InsectWingbeatSound	85.81%	43.84%
ItalyPowerDemand	21.09%	4.47%
LargeKitchenAppliances	28.27%	50.67%
Lightning2	22.95%	24.59%
Lightning7	45.21%	42.47%
Mallat	21.28%	8.57%
Meat	6.67%	6.67%
MedicalImages	43.03%	31.58%
MiddlePhalanxOutlineAgeGroup	47.40%	48.05%
MiddlePhalanxOutlineCorrect	35.05%	23.37%
MiddlePhalanxTW	46.75%	48.70%

Table 4.5: Classification Error Rates Comparison (SEA vs MSR) (continued)

Base Name	SEA	MSR
MoteStrain	22.04%	12.14%
NonInvasiveFetalECGThorax1	20.25%	17.10%
NonInvasiveFetalECGThorax2	13.28%	12.01%
OliveOil	16.67%	13.33%
OSULeaf	56.20%	47.93%
PhalangesOutlinesCorrect	28.44%	23.89%
Phoneme	86.87%	89.08%
Plane	0.00%	3.81%
ProximalPhalanxOutlineAgeGroup	23.41%	21.46%
ProximalPhalanxOutlineCorrect	23.37%	19.24%
ProximalPhalanxTW	28.78%	29.27%
RefrigerationDevices	50.67%	60.53%
ScreenType	59.47%	64.00%
ShapeletSim	56.11%	46.11%
ShapesAll	31.83%	24.83%
SmallKitchenAppliances	41.07%	65.87%
SonyAIBORobotSurface1	28.62%	30.45%
SonyAIBORobotSurface2	25.92%	14.06%
StarLightCurves	13.16%	15.12%
Strawberry	7.30%	5.41%
SwedishLeaf	27.68%	21.12%
Symbols	39.40%	10.05%
SyntheticControl	61.67%	12.00%
ToeSegmentation1	21.05%	32.02%
ToeSegmentation2	28.46%	19.23%
Trace	8.00%	24.00%
TwoLeadECG	7.73%	25.29%
TwoPatterns	59.00%	9.32%
UWaveGestureLibraryAll	71.72%	5.19%
UWaveGestureLibraryX	64.32%	26.07%
UWaveGestureLibraryY	69.77%	33.84%
UWaveGestureLibraryZ	63.82%	35.04%
Wafer	1.98%	0.45%
Wine	44.44%	38.89%
WordSynonyms	75.39%	38.24%

Table 4.5: Classification Error Rates Comparison (SEA vs MSR) (continued)

Base Name	SEA	MSR
Worms	51.95%	54.55%
WormsTwoClass	35.06%	38.96%
Yoga	21.43%	16.97%
Total Wins	SEA = 27	MSR = 57

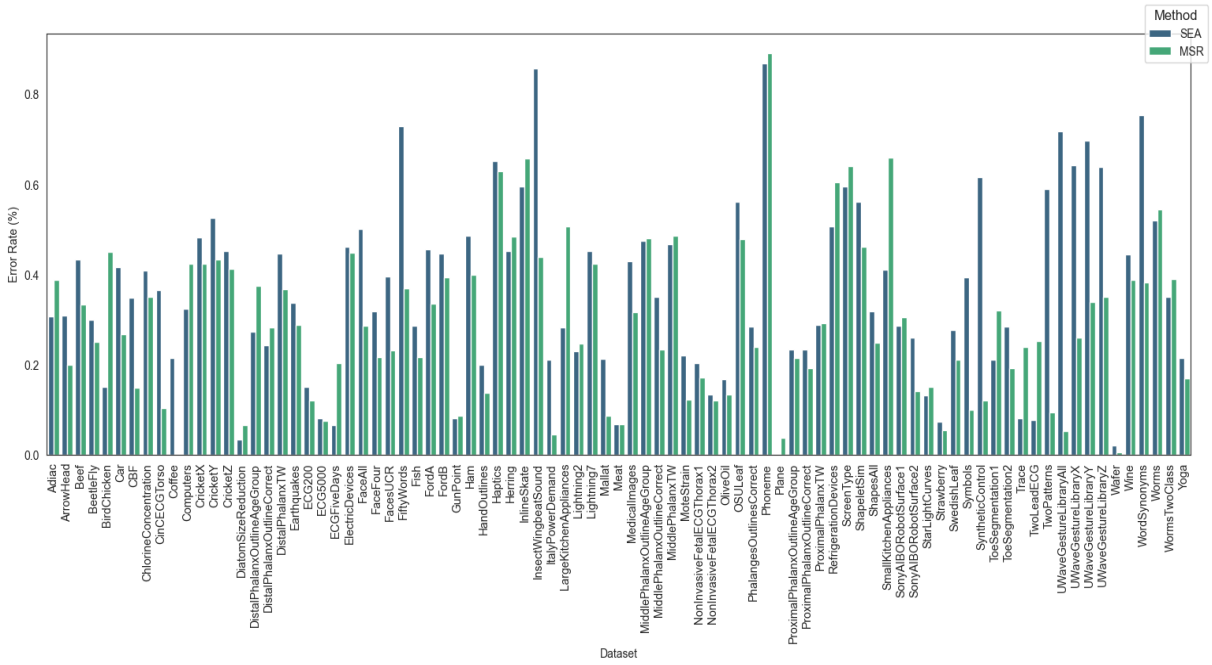

Figure 4.9: Comparative error rates of SEA and MSR across 85 UCR datasets.

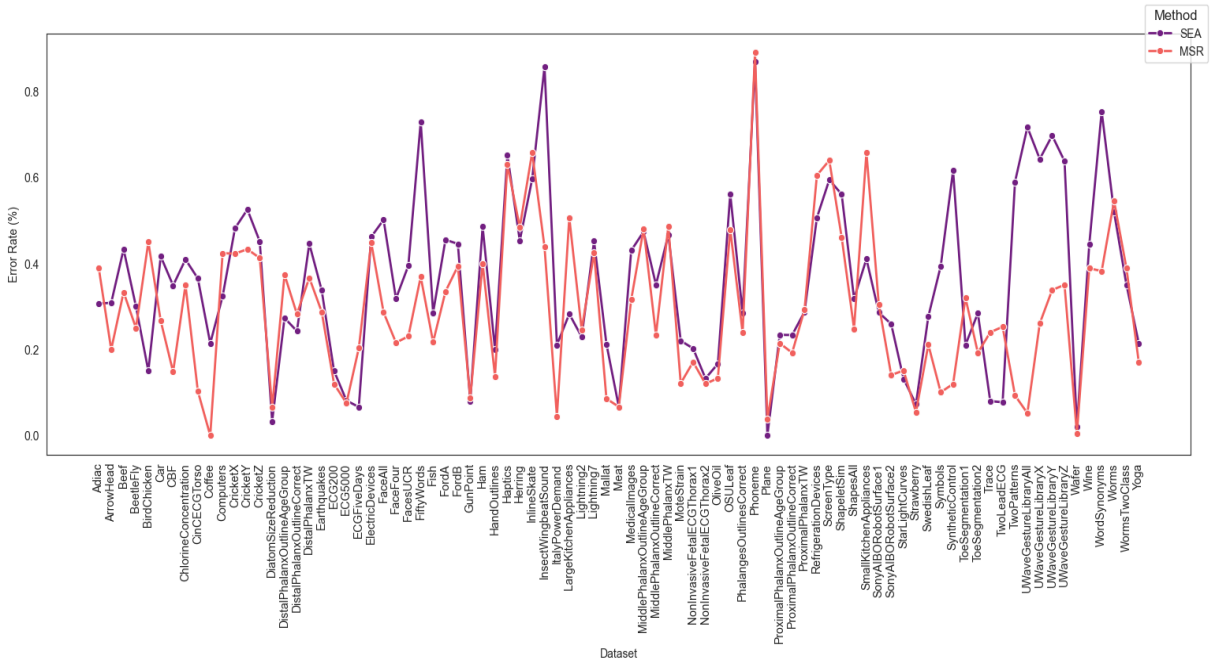

Figure 4.10: Error rate trends for SEA and MSR methods.

4.6.3 Discussion

The comprehensive comparison in Table 4.5 reveals that the Mean Squared Residue (MSR) significantly outperformed SEA, achieving lower error rates in 57 out of 85 datasets (67.1%), while SEA showed superior performance in only 27 cases (31.8%). This substantial advantage demonstrates MSR effectiveness as a more robust distance metric for time series classification across diverse datasets.

4.7 Application Interface

This section presents the user interface of the Time Series Classification Application developed for this research. The interface provides an intuitive workflow for loading datasets, configuring classification parameters, visualizing results, and executing classification tasks. Two primary views demonstrate the interface capabilities: the initial configuration state (Figure 4.11) and an operational state with loaded datasets (Figure 4.12).

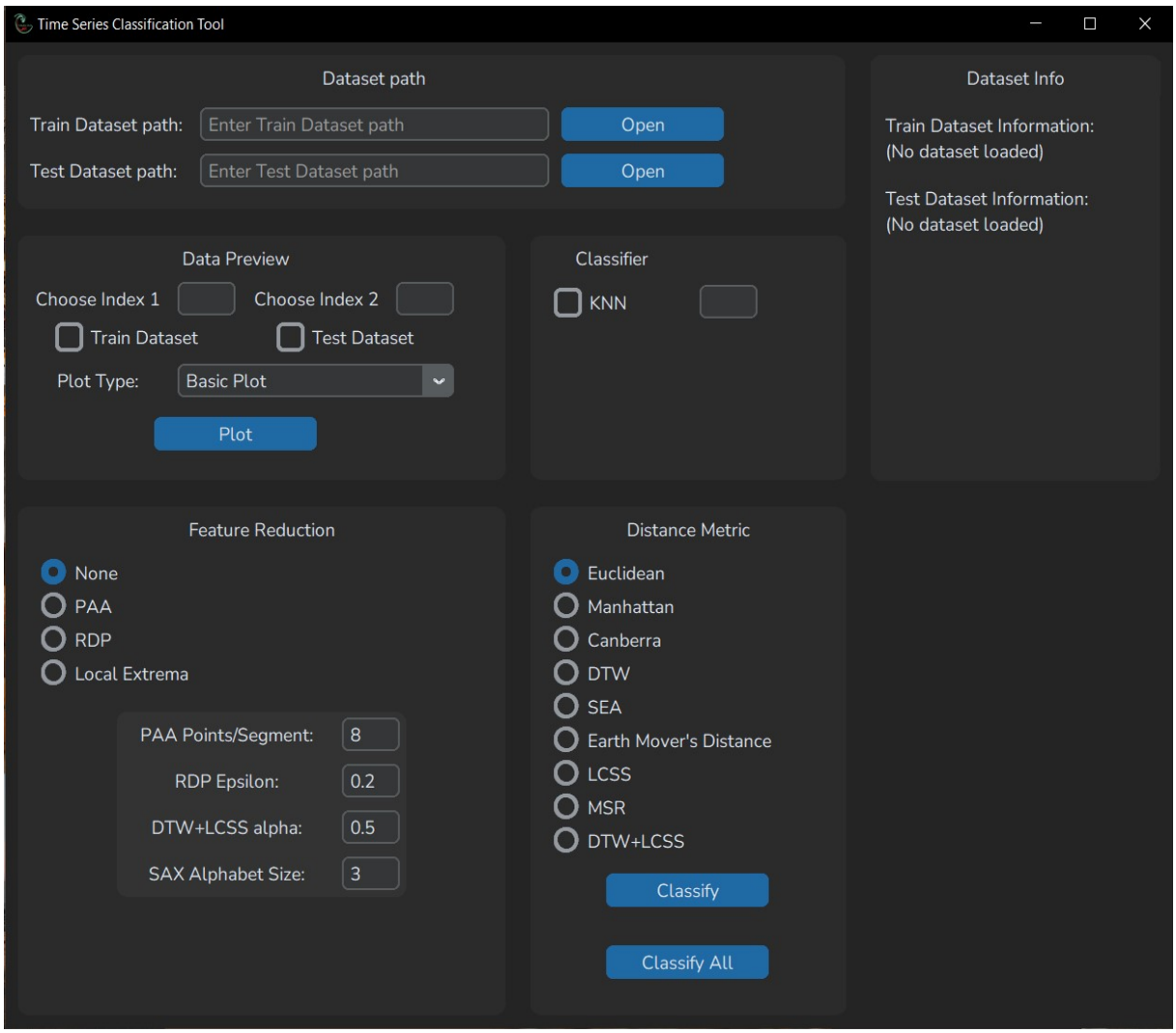

Figure 4.11: Initial interface state before dataset loading

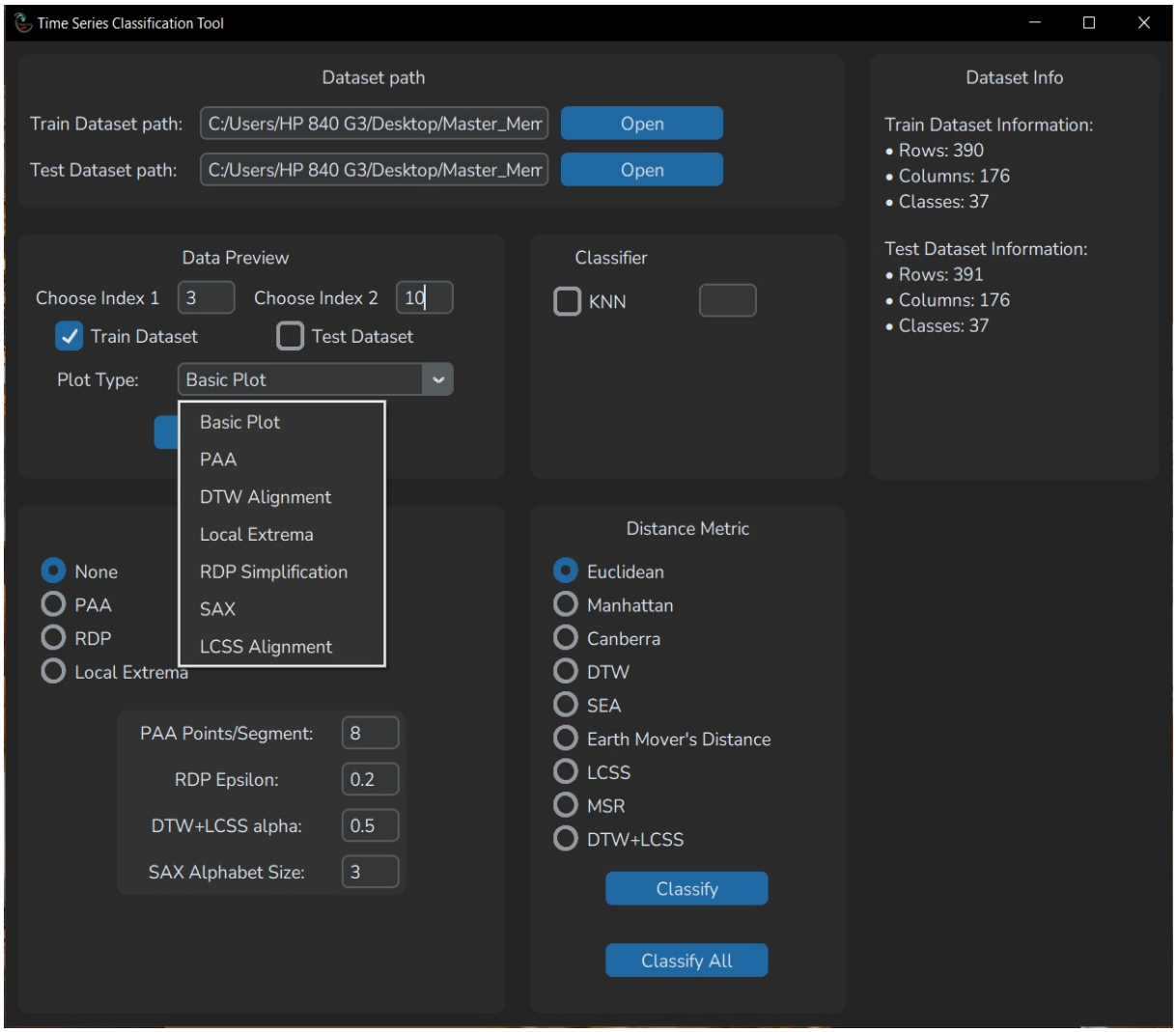

Figure 4.12: Operational interface with loaded datasets and configuration

4.7.1 Interface Components and Functionality

The application interface consists of several key functional areas:

1. **Dataset Management**:
    - Path selection for train and test datasets (with "Open" buttons for file browsing)
    - Dataset information display (rows, columns, classes) after loading
    - Data preview capability with index selection
2. **Classification Configuration**:
    - Classifier selection (K-Nearest Neighbors (1NN) in current implementation)
    - Comprehensive distance metric options including:
        - Basic metrics (Euclidean, Manhattan, Canberra)
        - Advanced metrics (DTW, SEA, EMD, LCSS, MSR)
        - Hybrid metrics (DTW+LCSS)
3. **Visualization Tools**:
    - Multiple plot types: Basic, PAA, DTW Alignment, RDP Simplification, etc.
    - Feature reduction techniques: PAA, RDP, Local Extrema, SAX
    - Parameter controls: PAA points/segment, RDP epsilon, SAX alphabet size
4. **Execution Control**:
    - `"Classify All"` button - Initiates batch processing of all datasets in the archive
    - `"Classify"` button - Processes a single user-selected dataset from the archive

The interface significantly streamlines the experimental process by integrating dataset management, algorithm configuration, visualization, and execution in a single environment, enabling efficient evaluation of time series classification approaches.

4.8 Conclusion

This chapter's empirical analysis of time series classification metrics revealed key insights: Manhattan distance was the strongest basic metric (37/85 wins), DTW led among elastic measures (37 wins), and our DTW_LCSS hybrid showed robust performance (36 wins). MSR outperformed SEA in most cases (57 vs. 27 wins), demonstrating that optimal metric choice depends on dataset traits like alignment, noise, and shape complexity. The developed application provides a practical tool for implementing these methods.

GENERAL CONCLUSION AND PERSPECTIVES

This thesis explored different methods for classifying time series data, which is used in many fields like finance, medicine, and weather forecasting. The goal was to compare various techniques to find out which ones work best for different types of data.

We started by explaining what time series data is and the challenges it presents, such as noise, high dimensionality, and temporal misalignments. Then, we examined several methods for reducing the complexity of the data, like Piecewise Aggregate Approximation (PAA) and Local Extrema (LE), which help simplify the data without losing important information.

Next, we tested different distance measures, including Euclidean, Manhattan, and Dynamic Time Warping (DTW), to see how well they classify time series. Our experiments showed that no single method works best for all cases. For example:

- **Manhattan distance** performed well on many datasets.
- **DTW** was strong for data with timing variations.
- **LCSS** handled noisy data effectively.
- **Hybrid DTW_LCSS** combined the strengths of DTW and LCSS, achieving good results across many datasets.

We also introduced new approaches like PAA_SEA and RDP_SEA, but these did not improve

performance compared to the original methods. However, testing MSR (Mean Squared Residue) for the first time in time series classification showed promising results, often outperforming SEA.

Finally, we developed a practical application to make it easier for users to apply these methods. The tool allows loading datasets, choosing methods, and visualizing results, making time series classification more accessible.

In summary, the choice of method depends on the specific characteristics of the data. Future work could explore more hybrid methods or ways to automatically select the best technique for a given dataset. This research contributes to the field by providing a clear comparison of methods and practical tools for real-world applications.

BIBLIOGRAPHY

[1] Rakesh Agrawal, Christos Faloutsos, and Arun Swami. Efficient similarity search in sequence databases. In David B. Lomet, editor, *Foundations of Data Organization and Algorithms*, pages 69–84, Berlin, Heidelberg, 1993. Springer Berlin Heidelberg.

[2] Mohsena Ashraf, Farzana Anowar, Jahanggir H. Setu, Atiqul I. Chowdhury, Eshtiak Ahmed, Ashraful Islam, and Abdullah Al-Mamun. A survey on dimensionality reduction techniques for time-series data. *IEEE Access*, 11:42909–42923, 2023.

[3] Gustavo Batista, Eamonn Keogh, Oben Tataw, and Vinícius Alves de Souza. Cid: An efficient complexity-invariant distance for time series. *Data Mining and Knowledge Discovery*, 28, 04 2013.

[4] Nipun Batra, Jack Kelly, Oliver Parson, Haimonti Dutta, William Knottenbelt, Alex Rogers, Amarjeet Singh, and Mani Srivastava. Nilmtk: An open source toolkit for non-intrusive load monitoring. In *Proceedings of the 5th International Conference on Future Energy Systems*. ACM, 2020.

[5] Bachir Boucheham. Matching of quasi-periodic time series patterns by exchange of block-sorting signatures. *Pattern Recognition Letters*, 29(4):501–514, 2008.

[6] Bachir Boucheham. Reduced data similarity-based matching for time series patterns alignment. *Pattern Recognition Letters*, 31(7):629–638, 2010.

[7] Bachir Boucheham. Efficient matching of very complex time series. *International Journal of Machine Learning and Cybernetics*, 4:537–550, 2013.

[8] Salah Bougueroua and Bachir Boucheham. Greedy mean squared residue for texture images retrieval. In *Modelling and Implementation of Complex Systems: Proceedings of the 4th International Symposium, MISC 2016, Constantine, Algeria, May 7-8, 2016, Constantine, Algeria*, pages 123–136. Springer, 2016.

[9] C. Sidney Burrus, Ramesh A. Gopinath, and Haitao Guo. *Wavelets and Wavelet Transforms*. Rice University, Houston, 98 edition, 1998.

[10] Sung-Hyuk Cha. Comprehensive survey on distance/similarity measures between probability density functions. http://www.fisica.edu.uy/~cris/teaching/Cha_pdf_distances_2007.pdf, 2007.

[11] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):1–58, 2009.

[12] Yizong Cheng and George M Church. Biclustering of expression data. In *Ismb*, volume 8, pages 93–103, 2000.

[13] Wikipedia contributors. Ramer–douglas–peucker algorithm. https://en.wikipedia.org/wiki/Ramer%E2%80%93Douglas%E2%80%93Peucker_algorithm, 2023.

[14] Wikipedia contributors. Earth mover's distance — Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Earth_mover%27s_distance, 2024. [Online; accessed 10/05/2025].

[15] 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(6):1293–1305, 2019.

[16] Hoang Anh Dau, Eamonn Keogh, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, Yanping, Bing Hu, Nurjahan Begum, Anthony Bag-

nall, Abdullah Mueen, Gustavo Batista, and Hexagon-ML. The ucr time series classification archive, October 2018. https://www.cs.ucr.edu/~eamonn/time_series_data_2018/.

[17] 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. *Proc. VLDB Endow.*, 1(2):1542–1552, August 2008.

[18] David H Douglas and Thomas K Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *The Canadian Cartographer*, 10(2):112–122, 1973.

[19] Philippe Esling and Carlos Agon. Time-series data mining. *ACM Computing Surveys*, 45(1):12, 2012. HAL Id: hal-01577883.

[20] Johann Faouzi. Time series classification: A review of algorithms and implementations. In Jorge Rocha, Cláudia M. Viana, and Sandra Oliveira, editors, *Time Series Analysis*, chapter 2. IntechOpen, Rijeka, 2024.

[21] Hassan Ismail Fawaz. Deep learning for time series classification. *arXiv preprint arXiv:2010.00567*, 2020.

[22] Youssef Fenjiro. Time series for business: A general introduction. https://medium.com/@fenjiro/time-series-for-business-a-general-introduction-50968346e660, Jul 6, 2019. Accessed: 04/06/2025.

[23] Aida A Ferreira, Iona MB Rameh Barbosa, Ronaldo RB Aquino, Herrera Manuel, Sukumar Natarajan, Daniel Fosas, and David Coley. Adaptive piecewise and symbolic aggregate approximation as an improved representation method for heat waves detection. In *Intelligent Computing: Proceedings of the 2018 Computing Conference, Volume 1*, pages 658–671. Springer, 2019.

[24] Mohamed Medhat Gaber, Arkady Zaslavsky, and Shonali Krishnaswamy. Mining data streams: A review. *ACM SIGMOD Record*, 34(2):18–26, 2005.

[25] GeeksforGeeks. Dynamic time warping (dtw) in time series, 2025. Last Updated: May 1, 2025.

[26] Tomasz Górecki. Classification of time series using combination of dtw and lcss dissimilarity measures. *Communications in Statistics-Simulation and Computation*, 47(1):263–276, 2018.

[27] A. Graps. An introduction to wavelets. *IEEE Computational Science and Engineering*, 2(2):50–61, 1995.

[28] Chonghui Guo, Hailin Li, and Donghua Pan. An improved piecewise aggregate approximation based on statistical features for time series mining. In *Knowledge Science, Engineering and Management: 4th International Conference, KSEM 2010, Belfast, Northern Ireland, UK, September 1-3, 2010. Proceedings 4*, pages 234–244. Springer, 2010.

[29] Linus Heinzl. Analysis of the parameter configuration of the ramer-douglas-peucker algorithm for time series compression.

[30] Jaymer M. Jayoma, Genevive P. Navales, Flocer Angelo M. Bustamante, and Lavranz R. Tradio. Enhanced ramer-douglas-peucker algorithm for efficient 2d line simplification. In *2023 IEEE 15th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM)*, pages 1–6, 2023.

[31] Young-Seon Jeong, Myong K Jeong, and Olufemi A Omitaomu. Weighted dynamic time warping for time series classification. *Pattern recognition*, 44(9):2231–2240, 2011.

[32] Jack Kelly and William Knottenbelt. Neural nilm: Deep neural networks applied to energy disaggregation. In *Proceedings of the 2nd ACM International Conference on Embedded Systems for Energy-Efficient Built Environments*. ACM, 2015.

[33] Eamonn Keogh, Kaushik Chakrabarti, Michael Pazzani, and Sharad Mehrotra. Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and Information Systems*, 3(3):263–286, 2001.

[34] Eamonn Keogh, Kaushik Chakrabarti, Michael Pazzani, and Sharad Mehrotra. Locally adaptive dimensionality reduction for indexing large time series databases. *SIGMOD Rec.*, 30(2):151–162, May 2001.

[35] 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(4):349–371, 2003.

[36] Eamonn J. Keogh and Michael J. Pazzani. A simple dimensionality reduction technique for fast similarity search in large time series databases. In Takao Terano, Huan Liu, and Arbee L. P. Chen, editors, *Knowledge Discovery and Data Mining. Current Issues and New Applications*, pages 122–133, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[37] Manel Khadiche, Bachir Boucheham, and Salah Bougueroua. Ramer-douglas-peucker dynamic time warping (rdp-dtw): A novel data reduction based dynamic time warping method for time series classification. In *2024 International Conference on Advances in Electrical and Communication Technologies (ICAECOT)*, pages 1–5, 2024.

[38] Caroline Kleist. Time series data mining methods: A review. Master's thesis, Humboldt-Universität zu Berlin, School of Business and Economics, Berlin, Germany, March 25, 2015. Master's Thesis submitted to Prof. Dr. Wolfgang Karl H¨ardle, Ladislaus von Bortkiewicz Chair of Statistics.

[39] Abdelmadjid Lahreche and Bachir Boucheham. Fastsea: A very fast and very effective matching technique for very complex time series. In *2017 International Conference on Mathematics and Information Technology (ICMIT)*, pages 286–293. IEEE, 2017.

[40] Abdelmadjid Lahreche and Bachir Boucheham. Lmds-sea: Upgrading the shape exchange algorithm (sea) to handle general time series classification by local matching and distance selection. In *2018 3rd International Conference on Pattern Analysis and Intelligent Systems (PAIS)*, pages 1–6. IEEE, 2018.

[41] Abdelmadjid Lahreche and Bachir Boucheham. A fast and accurate similarity measure for long time series classification based on local extrema and dynamic time warping. *Expert Systems with Applications*, 168:114374, 2021.

[42] Hailin Li. Asynchronism-based principal component analysis for time series data mining. *Expert Systems with Applications*, 41(6):2842–2850, 2014.

[43] M. Li, H. Wang, L. Yang, Y. Liang, Z. Shang, and H. Wan. Fast hybrid dimensionality reduction method for classification based on feature selection and grouped feature extraction. *Expert Systems with Applications*, 150:1–27, Jan 2020.

[44] Jessica Lin, Eamonn Keogh, and Stefano Lonardi. Visualizing and discovering non-trivial patterns in large time series databases. *Information Visualization*, 4(2):61–82, 2005.

[45] Jessica Lin, Eamonn Keogh, Stefano Lonardi, and Bill Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD)*, pages 2–11, San Diego, California, USA, 2003. ACM.

[46] Jessica Lin, Eamonn Keogh, Li Wei, and Stefano Lonardi. Experiencing sax: a novel symbolic representation of time series. *Data Mining and knowledge discovery*, 15:107–144, 2007.

[47] John Paparrizos and Luis Gravano. k-shape: Efficient and accurate clustering of time series. In *Proceedings of the 2015 ACM SIGMOD international conference on management of data*, pages 1855–1870, 2015.

[48] John Paparrizos, Haojun Li, Fan Yang, Kaize Wu, Jens E d'Hondt, and Odysseas Papapetrou. A survey on time-series distance measures. *arXiv preprint arXiv:2412.20574*, 2024.

[49] Hoonseok Park and Jae-Yoon Jung. Sax-arm: Deviant event pattern discovery from multivariate time series using symbolic aggregate approximation and association rule mining. *Expert Systems with Applications*, 141:112950, 2020.

[50] Pranav Patel, Eamonn Keogh, Jessica Lin, and Stefano Lonardi. Mining motifs in massive time series databases. *University of California - Riverside, Computer Science & Engineering Department*, 2002. {prpatel, eamonn, jessica, stelo}@cs.ucr.edu.

[51] 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. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 262–270, 2012.

[52] Thanawin Rakthanmanon, Bilson Campana, Abdullah Mueen, Gustavo Batista, Brandon Westover, Qiang Zhu, Jesin Zakaria, and Eamonn Keogh. Data mining a trillion time series subsequences under dynamic time warping. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 3047–3051, 2013.

[53] Chotirat Ratanamahatana, Eamonn Keogh, Anthony J. Bagnall, and Stefano Lonardi. A novel bit level time series representation with implication of similarity search and clustering. In Tu Bao Ho, David Cheung, and Huan Liu, editors, *Advances in Knowledge Discovery and Data Mining*, pages 771–777, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[54] Chotirat Ann Ratanamahatana, Jessica Lin, Dimitrios Gunopulos, Eamonn Keogh, Michail Vlachos, and Gautam Das. *Mining Time Series Data*, chapter Chapter 1. University of California, Riverside and IBM T.J. Watson Research Center and University of Texas, Arlington, 2003.

[55] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. A metric for distributions with applications to image databases. In *Sixth international conference on computer vision (IEEE Cat. No. 98CH36271)*, pages 59–66. IEEE, 1998.

[56] Ahmed Shifaz, Charlotte Pelletier, François Petitjean, and Geoffrey I. Webb. Elastic similarity measures for multivariate time series classification. *Knowledge and Information Systems (KAIS)*, 2023.

[57] Gholamreza Soleimani and Masoud Abessi. Dlcss: A new similarity measure for time series data mining. *Engineering Applications of Artificial Intelligence*, 92:103664, 2020.

[58] Zbigniew R. Struzik and Arno Siebes. The haar wavelet transform in the time series similarity paradigm. In Jan M. Żytkow and Jan Rauch, editors, *Principles of Data Mining and Knowledge Discovery*, pages 12–22, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[59] Chang Wei Tan. *Time Series Classification at Scale*. PhD thesis, Monash University, 2019.

[60] Yin-Te Tsai. The constrained longest common subsequence problem. *Information Processing Letters*, 88(4):173–176, 2003.

[61] Neelam Tyagi. 5 applications of time series analysis. https://www.analyticssteps.com/blogs/5-applications-time-series-analysis, Jul 19, 2021. Accessed: [27/02/2025].

[62] Michail Vlachos, George Kollios, and Dimitrios Gunopulos. Discovering similar multidimensional trajectories. In *Proceedings 18th international conference on data engineering*, pages 673–684. IEEE, 2002.

[63] 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. 2012.

[64] Wikipedia contributors. Time series. https://en.wikipedia.org/wiki/Time_series, 2023.

[65] Djamel-Edine Edine Yagoubi. Indexing and analysis of very large masses of time series, 2018. Université Montpellier, English, ffNNT : 2018MONTS084ff, fftel-02148207f.

[66] Yufeng Yu, Yuelong Zhu, Dingsheng Wan, Huan Liu, and Qun Zhao. A novel symbolic aggregate approximation for time series. In *Proceedings of the 13th International Conference on Ubiquitous Information Management and Communication (IMCOM) 2019 13*, pages 805–822. Springer, 2019.