WARSAW UNIVERSITY OF TECHNOLOGY

FACULTY OF MATHEMATICS AND INFORMATION SCIENCE

Ph.D. Thesis

Rauzan Sumara, M.Sc.

Multiple Representation-Based Ensembles for Time Series Classification

Supervisor

Prof. dr hab. inż. Władysław Homenda

WARSAW 2025

Abstract

Time series classification has gained more attention due to its potential in various fields, such as healthcare (e.g., diagnosing diseases based on patient vitals), industrial monitoring (e.g., detecting machine faults), and environmental studies (e.g., weather pattern recognition). Many algorithms have been specifically designed for time series classification. Those techniques can be categorized based on the fundamental data representation employed. Feature-based approaches depend on global features extracted through a straightforward pipeline and fed into an appropriate classifier. Dictionary-based methods transform real-valued time series into discrete symbol sequences, thereby exploiting the frequency of recurrent patterns. Interval-based approaches generate features from specific time segments within the series, revealing temporal characteristics of time series. Shapelet-based approaches identify phase-independent subsequences to effectively discriminate between time series. The current attractive method in classifying time series is to utilize two or more representations.

This thesis introduces MuRBE (Multiple Representation-Based Ensembles), a novel heterogenous ensemble for time series classification. The MuRBE leverages diverse representation domains, including feature-based, dictionary-based, interval-based, and shapelet-based methods. Exploiting complementary information from different representations makes it particularly effective in improving classification performance. We additionally present two innovative classifiers that serve as the components within the MuRBE structure, namely feature-based autoregressive fractional integrated moving average with random forest (ARFIMA-RF) and dictionary-based symbolic aggregate approximation with stacking gated recurrent unit and convolutional neural networks (SAX-SGCNN). We demonstrate that MuRBE is significantly more accurate than the base classifiers and achieves competitive performance across the current state-of-the-art methods on 40 UCR/UEA archive datasets.

Streszczenie

Klasyfikacja szeregów czasowych zyskała na znaczeniu ze względu na jej potencjał w różnych dziedzinach, takich jak opieka zdrowotna (np. diagnozowanie chorób na podstawie parametrów życiowych pacjentów), monitorowanie przemysłowe (np. wykrywanie usterek maszyn) oraz badania środowiskowe (np. rozpoznawanie wzorców pogodowych). Wiele algorytmów zostało specjalnie zaprojektowanych do klasyfikacji szeregów czasowych. Techniki te można podzielić na kategorie w zależności od podstawowego sposobu reprezentacji danych. Podejścia oparte na cechach polegają na globalnych cechach wyekstrahowanych za pomocą prostego procesu i wykorzystanych w odpowiednim klasyfikatorze. Metody oparte na słownikach przekształcają rzeczywiste szeregi czasowe w dyskretne ciągi symboli, wykorzystując tym samym częstość powtarzających się wzorców. Podejścia oparte na przedziałach generują cechy z określonych segmentów czasowych w serii, ujawniając cechy temporalne szeregów czasowych. Metody oparte na shapeletach identyfikują podciągi niezależne od fazy, które skutecznie rozróżniają szeregi czasowe. Obecnie atrakcyjną metodą klasyfikacji szeregów czasowych jest wykorzystanie dwóch lub więcej reprezentacji jednocześnie.

Niniejsza praca przedstawia MuRBE (Multiple Representation-Based Ensembles), nowatorski heterogeniczny zespół klasyfikatorów dla klasyfikacji szeregów czasowych. MuRBE wykorzystuje różnorodne domeny reprezentacji, w tym metody oparte na cechach, słownikowe, oparte na przedziałach oraz shapeletowe. Wykorzystanie komplementarnych informacji z różnych reprezentacji czyni go szczególnie skutecznym w poprawie wydajności klasyfikacji. Dodatkowo przedstawiamy dwa innowacyjne klasyfikatory, które pełnią funkcję komponentów w strukturze MuRBE, a mianowicie autoregresyjny ułamkowo całkowany model średniej ruchomej z losowym lasem (ARFIMA-RF) oparty na cechach oraz słownikową symboliczna aproksymację agregacji z warstwą stackingową GRU i sieciami konwolucyjnymi (SAX-SGCNN). Wykazujemy, że MuRBE jest znacząco dokładniejszy niż klasyfikatory bazowe oraz osiąga konkurencyjne wyniki względem najnowocześniejszych metod na 40 zbiorach danych z archiwum UCR/UEA.

Acknowledgment

First and foremost, I wish to express my deepest gratitude to God, the source of all mercy and blessings, for His unceasing guidance and grace throughout this journey. It is through His will that I have been able to persevere and overcome the many challenges faced along the way. His presence has been my constant strength, and for that, I remain eternally grateful.

I am profoundly grateful to my supervisor, Prof. dr hab. inż. Władysław Homenda, whose wisdom, patience, and extraordinary guidance have shaped not only this thesis but my personal and academic development. From the beginning, your mentorship has been a beacon of knowledge, encouraging me to think critically and strive for excellence. Your insightful feedback and unwavering support have enabled me to reach new heights and realize my full potential. It is a true honor to have learned from you, and I cannot thank you enough for your persistent encouragement and belief in my abilities.

To my beloved family, especially my parents, I owe everything. Your unconditional love, sacrifices, and unwavering belief in me have formed the foundation of my perseverance. Your support and prayers have carried me through every obstacle and moment of doubt. I am eternally thankful for the strength and motivation you have provided me throughout this journey.

I would also like to express my sincere gratitude to co-authors, friends, colleagues, and the academic community at the Warsaw University of Technology. Your fellowship, support, and shared experiences have made this academic journey both meaningful and enjoyable. The resources, opportunities, and collaborative spirit within this institution have played a substantial part in forming this work, and I am genuinely thankful for all the support I have received. May this thesis stand as a testament to the collective effort of all those who have walked beside me.

Rauzan Sumara

Warsaw, May 25, 2025

Dedicated to my parents

Usman

&

Nurainun

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Time series classification has become a crucial task within a subfield of machine learning. Unlike conventional classification problems where the order of attributes is unimportant, time series classification involves analyzing temporally related attributes, requiring the examination of comprehensive sequential data. The classification process involves predicting a class label for a sequence based on its measurable attributes or characteristic features. In turn, a classifier is utilized to distinguish between sequences that originate from different classes, with each sequence or time series having an equivalent set of extracted features. In recent years, many algorithms have been developed to enhance the predictive capabilities of state-of-the-art methods [1].

A range of representation algorithms has been specifically designed to extract characteristic features for time series classification. Those techniques can be categorized based on the fundamental data representation employed. Feature-based approaches depend on global features extracted through a straightforward pipeline and fed into an appropriate classifier. Dictionary-based methods transform real-valued time series into discrete symbol sequences, thereby exploiting the frequency of recurrent patterns. Interval-based approaches generate features from specific time segments within the series, revealing temporal characteristics of time series. Shapelet-based approaches identify phase-independent subsequences to effectively discriminate between time series.

The current attractive method in the classification of time series is to utilize two or more representations. This methodology can be classified into four distinct groups. The first group is heterogeneous ensembles, where every component comprises a classifier designed based

on different representation types. For example, the hierarchical vote collective of transformation-based ensembles (HIVE-COTE 2.0) or HC2 [2], until this time, was considerably more accurate on average compared to other established state-of-the-art techniques. Another approach, the time series combination of heterogeneous and integrated embedding forest (TS-CHIEF) [3], is a classifier resembling HC2. TS-CHIEF consists of an ensemble of trees that incorporate distance, dictionary, and spectral-based features. Additionally, recent innovations include the hybrid dictionary-ROCKET architecture (Hydra) [4] that combines dictionary and convolution-based representations. The random convolutional kernel transform (ROCKET) [5] is a convolution-based model that generates numerous summary statistics using randomly initialized convolutional kernels and then builds a linear ridge classifier to identify the classes.

The second group is tree based homogeneous ensembles that incorporate a specific representation within the tree nodes. One notable technique in this category is the proximity forest (PF) [6], which is an ensemble of proximity tree based classifiers. The third group is deep learning ensembles with embedded network representations. An example is InceptionTime [7], which combines five identical residual networks featuring inception modules. The last group utilizes convolution-based ensemble techniques to generate extensive new feature spaces, which are then analyzed using a linear classifier. One popular algorithm within the category is the ensemble of ROCKET models, also known as Arsenal [2].

Recent investigations in ensemble techniques have received considerable interest due to their capacity to enhance accuracy by merging the advantages of two or more representations, which motivated our research. Therefore, in this study, we introduce MuRBE (Multiple Representation-Based Ensemble), a novel heterogenous ensemble approach for time series classification. MuRBE leverages diverse representation domains, including feature-based, dictionary-based, interval-based, and shapelet-based methods. The study makes a unique contribution by integrating various domain classifiers through a fuzzy rank-based ensemble structure, which has not previously been explored in the context of time series classification. The MuRBE aims to capture a wide range of temporal patterns and discriminatory characteristics, leading to improved classification performance. Exploiting complementary information from different representations makes it particularly effective to improve performance for many time series classification tasks.

Our proposed MuRBE structure incorporates four different representation domains. Two components are from our new proposed algorithms, such as feature-based autoregressive fractional integrated moving average with random forest (ARFIMA-RF) [P1] and dictionary-based symbolic aggregate approximation with stacking gated recurrent unit and convolutional neural networks (SAX-SGCNN) [P2]. We also take into consideration interval-based diverse representation canonical interval forest (DrCIF) [2] and shapelet-based random dilated shapelet transform (RDST) [8].

1.1 Summary of Contributions

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

- We propose a novel heterogeneous ensemble that involves diverse representation domains and encapsulates predictions from different representations into a single prediction. We integrate various domain classifiers through a fuzzy rank-based ensemble structure, which is more flexible, dynamic, and adaptive to the weights of individual components based on their performance [P3]. We presented this contribution in Chapter 4.
- In the first component of our heterogeneous ensemble method, we propose a unique feature-based time series classification that uses estimated autoregressive fractionally integrated moving average (ARFIMA) parameters for each time series. This approach makes an innovative contribution by employing ARFIMA coefficients to characterize time series patterns. This technique is able to derive a smaller set of parameters from the time series model than the length of its series, process sets of time series exhibiting long-range dependence (long-run memory), and process those time series with different lengths [P1]. The method is described in detail in Subchapter 4.1.
- In the second component, we also introduce a novel dictionary-based time series classification framework, transforming time series into a sequence of words through symbolic aggregate approximation (SAX) and employing stacked deep learning as a classifier [P2]. It was initially inspired by the concept of deep neural networks in the field of natural language processing (NLP). We can go into more depth about this approach in Subchapter 4.2.

- Last but not least, we evaluate our approach in extensive empirical experiments on the different datasets and domains in Chapter 5. It shows that our approach is relatively fast and accurate compared to the other ensemble-based algorithms.

1.2 Structure of This Work

We have organized this thesis into six chapters. The remaining chapters are organized as follows:

In Chapter 2, we present definitions, including basic notations and introductory concepts of time series classification. We also describe the comprehensive repository of time series datasets widely used in the field. We used the archive as a benchmark and conducted the experiments to compare them with the current state-of-the-art algorithms.

Chapter 3 presents a detailed state-of-the-art review of the algorithms in time series classification. We present a literature review of the univariate time series classification research. In this chapter, we show different representations used in time series classification that align with our work, namely feature, dictionary, interval, and shapelet-based approaches, which were briefly mentioned in the introduction. We also include the chapter by presenting the state-of-the-art algorithms built based on ensemble-based approaches for benchmarking.

Chapter 4 focuses on our proposed algorithms. We introduce a multiple representation-based ensembles, a novel heterogeneous ensemble for time series classification, called MuRBE. We describe the structure of our approach, including new feature-based and dictionary-based classifiers.

Chapter 5 describes the experimental setup used in our research, detailing the processes and datasets applied. The datasets utilized in this research were carefully selected to ensure their relevance for the experiments conducted. In this chapter, we also provide the results of the experiments, emphasizing key findings, scalability, and usability of our approach compared to the state-of-the-art methods.

Chapter 7 concludes this thesis with a summary of the research, a list of overall contributions to knowledge, and a discussion of the limitations of this work.

1.3 List of Publications

The thesis is fundamentally rooted in the basis of three distinct original contributions,

- **[P1] Rauzan Sumara**, Władysław Homenda, and Witold Pedrycz. "ARFIMA for Feature-Based Time Series Classification," In PACIS 2024: Proceedings of the Pacific Asia Conference on Information Systems, 2024, Association for Information Systems, pp.1-9, Article number:1149.

  Contribution:

  As the first author in this publication, we propose a novel feature-based time series classification that uses estimated autoregressive fractionally integrated moving average (ARFIMA) parameters for each time series as a feature-based representation. This technique is able to derive a smaller set of parameters from the time series model than the length of its series, process sets of time series exhibiting long memory, and process those time series with different lengths. Collaborating with co-authors, we implemented and tested the solution, prepared the manuscript, and presented it during the conference.

  Ministerial score: **140**

- **[P2] Rauzan Sumara**, Władysław Homenda, Witold Pedrycz, and Fusheng Yu. "A Dictionary-Based with Stacked Ensemble Learning to Time Series Classification", In ICCS 2024: 24th International Conference on Computational Science, Lecture Notes in Computer Science, 2024, vol. 14834, no. Part III, Springer, pp.120-128.

  Contribution:

  We introduced the integration of a dictionary-based technique with stacked ensemble learning. This study is unique since it combines the symbolic aggregate approximation (SAX) with stacking gated recurrent units (GRU) and a convolutional neural network (CNN), referred to as SGCNN, which has not been previously investigated in time series classification. Together with co-authors, we designed the algorithm, prepared the manuscript for publication, and presented it during the conference.

  Ministerial score: **140**

- **[P3] Rauzan Sumara**, Władysław Homenda, Witold Pedrycz, and Fusheng Yu. "Time Series Classification with MuRBE: The Multiple Representation-Based Ensembles," *under submission process*.

Contribution:

We introduced MuRBE (Multiple Representation-Based Ensemble), an innovative heterogeneous ensemble structure explicitly designed for time series classification. The MuRBE leverages diverse representation domains, including feature-based, dictionary-based, interval-based, and shapelet-based methods.

Ministerial score: -

Chapter 2

Time Series Classification Tasks

Across various fields, it is common practice to measure a process at successive time points. The observed values typically exhibit interdependence rather than being independent and identically distributed. The data structure employed to store these measurements is known as a time series. It represents the value domain by storing each value as a real number and the time domain by organizing the values in ascending order.

Time series are used in various data mining tasks [9], as presented in Figure 1. These include unsupervised tasks, such as clustering, generation, anomaly detection, segmentation, and subsequence matching, which analyze past and current data to inform decision-making. Predictive supervised tasks, such as classification, forecasting, and regression, infer the class of an unlabeled time series and predict future values. Based on the various tasks mentioned, time series classification, in particular, has gained more attention due to its potential in various fields. Time series classification is widely used in areas such as healthcare (e.g., diagnosing diseases based on patient vitals), industrial monitoring (e.g., detecting machine faults), and environmental studies (e.g., weather pattern recognition). In the following subsections, we explain the definition and comprehensive repository of time series datasets widely used in the field of time series classification.

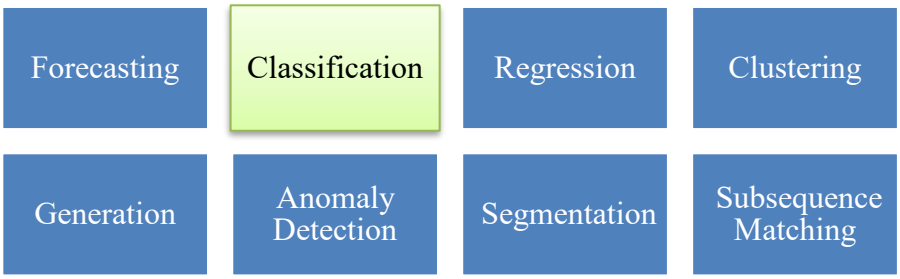


Figure 1. Data mining tasks.

2.1 Definitions

Time series classification algorithms, similar to standard supervised classifiers, construct a classifier based on a collection of labeled time series data. Formal definitions of time series and time series classification are provided in Definition 2.1.1 and 2.1.2. For the purpose of understanding the work presented in this thesis, it is assumed that all time series originate from the UCR/UEA time series classification archive, a comprehensive repository of datasets extensively used in the field. An explanation and example will be provided in the subsequent Subchapter 2.2.

Definition 2.1.1. Time Series

A time series X of length T consists of an ordered set of T pairs of time and value, denoted as $X = \langle (1, \boldsymbol{x}_1), \cdots, (t, \boldsymbol{x}_t), \cdots, (T, \boldsymbol{x}_T) \rangle$, where t represents the timestamp at t-th position in the sequence, with $t \in 1, \cdots, T$, and $\boldsymbol{x}_t$ is a D-dimensional vector capturing the values of D real-valued variables or features at time t. Each value $\boldsymbol{x}_t \in \mathbb{R}^D$ is characterized by the components $\{x_t^1, \cdots, x_t^d, \cdots, x_t^D\}$. In many cases, it is assumed that the timestamps t are equally spaced, allowing for a simplified notation where the timestamps are omitted, leading to a more compact representation of $X = \langle \boldsymbol{x}_1, \cdots, \boldsymbol{x}_t, \cdots, \boldsymbol{x}_T \rangle$. A univariate time series, or one-dimension, is a specific case in which only one variable is observed ($D = 1$). Therefore, x_t becomes a scalar, and the time series is represented as $X = \langle x_1, \cdots, x_t, \cdots, x_T \rangle$.

Definition 2.1.2. Time Series Classification

A labeled time series dataset $\boldsymbol{S}$ comprises N labeled time series indexed by n, where $n \in 1, \cdots, N$. Each time series X_n in $\boldsymbol{S}$, where $\boldsymbol{X} = \langle X_1, \ldots, X_n \rangle$, is associated with a corresponding

label $y_n \in 1, \cdots, C$, with C representing the number of classes. This labeling can be expressed as $\boldsymbol{y} = \langle y_1, \cdots, y_N \rangle$. Thus, a labeled time series dataset can be formalized as $\boldsymbol{S} = (\boldsymbol{X}, \boldsymbol{y})$.

In time series classification, a classifier is created by training it on a dataset containing labeled time series, which is then used to determine the class labels of time series that are not labeled. This classifier functions as a predictive tool that translates input variables into specific class labels. Figure 2 illustrates a time series classification problem, where the goal is to classify a new time series data point based on previously labeled examples.

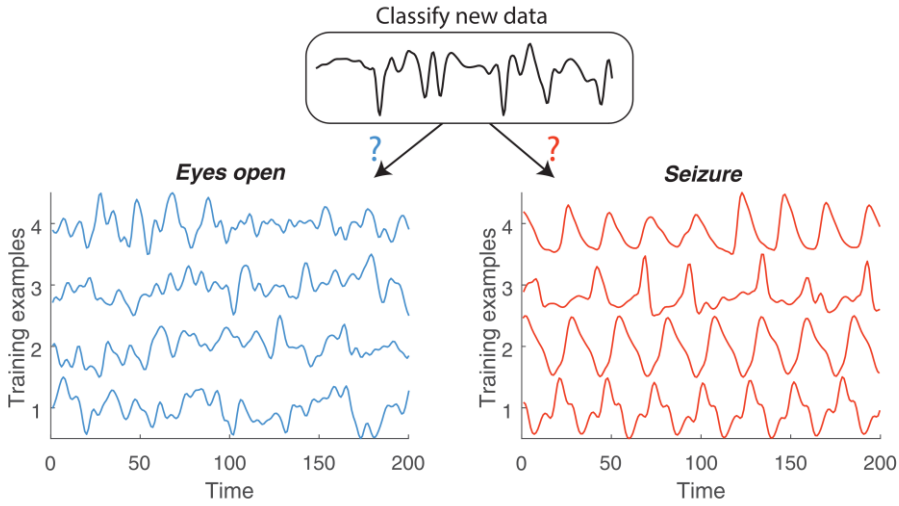


Figure 2. Example of time series classification. In this case, the objective is to differentiate between electroencephalogram (EEG) signals corresponding to an open-eye state (blue) and those indicative of seizure activity (red) [10].

2.2 UCR/UEA Time Series Classification Archive

To understand time series classification more intuitively, let us consider the most important archive for univariate problems, which is from UCR/UEA [11]. The UCR/UEA time series classification archive is a comprehensive repository of time series datasets widely used in the field. Established through a collaboration between the University of California, Riverside (UCR) and the University of East Anglia (UEA), this archive is a benchmark for evaluating and comparing various time series classification algorithms. The datasets within the archive cover a diverse range of domains, including medicine, biology, engineering, and finance, providing researchers with a rich resource for developing and testing new classification methods. The archive has grown significantly, now containing over 100 datasets of varying sizes and complexities [12].

For the sake of providing additional examples, in this Subchapter, we will outline a collection of 8 datasets contained within this repository, for examples, organized by various domains, such as *Image* (ArrowHead), *Spectro* (Beef), *ECG* (ECG200), *Motion* (GunPoint), *Sensor* (Plane), *Device* (ScreenType), *Spectrum* (SemgHandGenderCh2), and *Simulated* (SyntheticControl) type of datasets. The complete list of datasets is presented in Table 4 in Chapter 5. Details can be found on the associated website [1].

2.1.1 ArrowHead

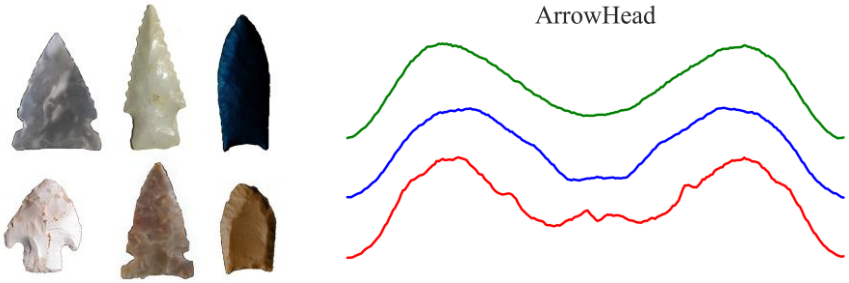


Figure 3. The example of the ArrowHead dataset obtained from [12].

Arrowhead data essentially consists of the complex contours that represent the various shapes and styles of arrowheads, which are crucial artifacts for understanding historical weaponry. The unique and distinct geometries of these projectile points are transmuted into a temporal series format through the application of a sophisticated technique known as an angle-based method [13], which facilitates a comprehensive analysis of their geometrical properties across time. The example of the angle-based method is shown in Figure 4. The categorization of these projectile points represents an essential area of exploration within the field of anthropology, as it allows scholars and researchers to gain comprehension of the cultural traditions and technological innovations of past societies.

The various categories into which these projectile points are classified depend on specific shape distinctions, involving factors such as the presence, absence, and exact positioning of notches on the arrowheads [14]. The three distinct classes that have been established for these projectile points are designated with the appellations "Avonlea", "Clovis", and "Mix", each representing unique characteristics and historical significance associated with their respective forms.

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

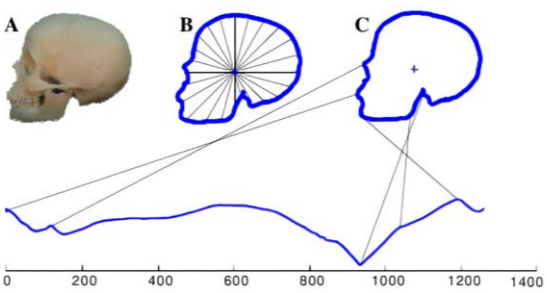


Figure 4. Implementation of angle-based method on human skull image obtained from [13].

2.1.2 Beef

Spectroscopic analysis plays a crucial role in the field of chemometrics, particularly in the categorization of diverse food types. This categorization procedure is paramount significance for guaranteeing food safety and upholding quality assurance benchmarks throughout the food industry. Accurately identifying and classifying food items is essential for combating food fraud, ensuring compliance with regulatory frameworks, and safeguarding consumer well-being.

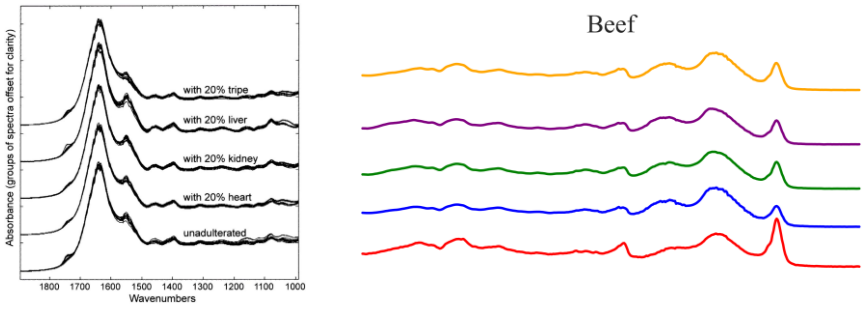


Figure 5. The example of the Beef dataset obtained from [15].

Within this context, the beef dataset serves as a valuable resource, encompassing five distinct classifications of beef spectrograms. These classifications include pure beef and beef adulterated with varying proportions of offal, representing the byproduct of animal processing. This distinction is crucial, as it facilitates the detection of potential adulteration, thereby protecting consumers from misleading labeling and preserving the integrity of meat products. Individuals interested in a more comprehensive understanding of the methodologies and findings within this dataset can refer to the original research article by [15], which investigates the identification of adulteration in cooked meat products using mid-infrared spectroscopy. This dataset was initially presented in the time series classification

literature through a publication that examined transformation-based ensemble approaches for classifying time series data [16].

2.1.3 ECG200

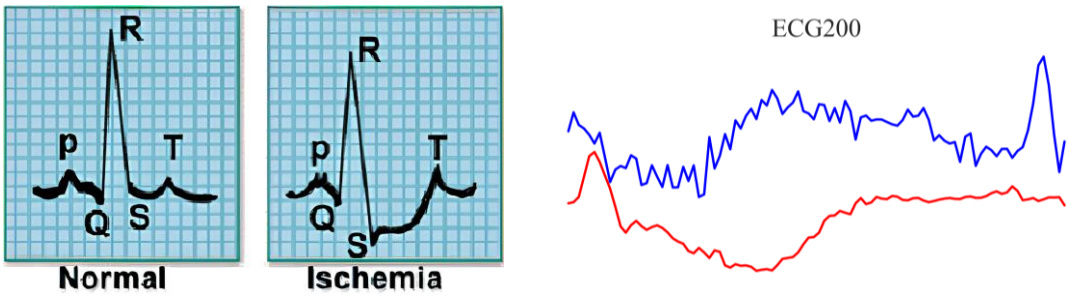


Figure 6. The example of the ECG200 dataset obtained from [17].

This specific dataset was from academic research organized by [17]. The dataset looks closely at the complex electrical activity happening in a single heartbeat. It is divided into two main categories - a normal heartbeat, which is typical, and a myocardial ischemia, which is a serious condition where blood flow to the heart muscle is cut off. This classification not only helps identify heart problems early on, but it also lays the foundation for developing advanced tools to improve diagnosis and patient care.

2.1.4 GunPoint

This dataset features two participants, a female and a male, who perform two distinct hand movements. The first motion, designated as "Gun-Draw," involves the actors starting with their hands at their sides, then drawing a simulated firearm from a holster, aiming it for approximately one second, and returning it to the holster. The second motion, referred to as "Point", has the participants again beginning with their hands at their sides, but instead of drawing a weapon, they extend their index fingers to indicate or point at a target for around one second before reverting their hands to the resting position at their sides. Importantly, the dataset only includes measurements pertaining to the *x*-axis. Within the dataset, class 1 corresponds to the "gun" motion, while class 2 represents the " no gun (pointing)" motion.

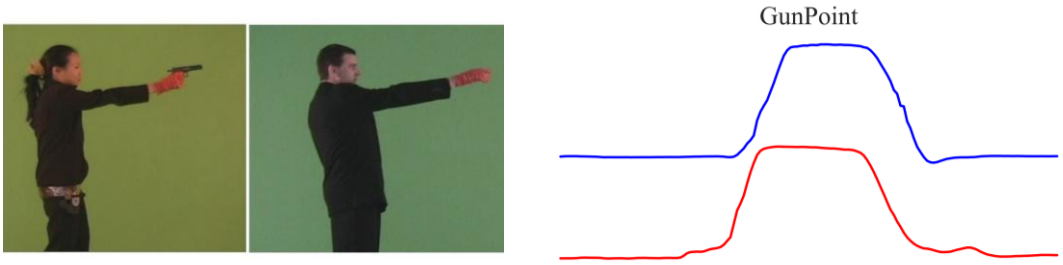


Figure 7. Illustration of gun and no gun (pointing) of the GunPoint dataset obtained from [18].

2.1.5 Plane

The *Plane* dataset consists of seven aeroplanes, including Mirage, Eurofighter, F-14, Harrier, F-22, and F-15. Notably, the F-14 aeroplane exhibited two distinct shapes, one with the wings in a closed position and another with the wings extended, resulting in a total of seven shape classes, where each class contains 30 samples.

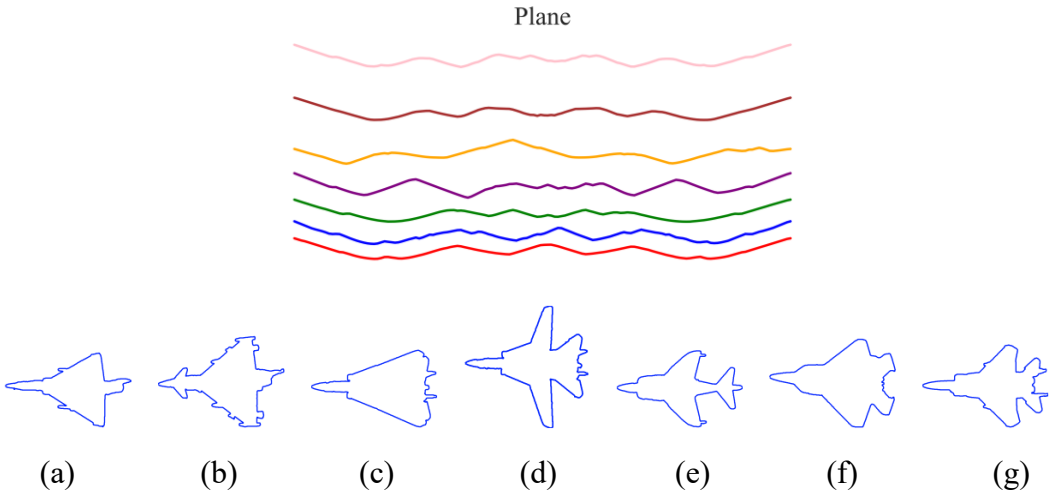


Figure 8. Illustration of Aeroplane: (a) Mirage, (b) Eurofighter, (c) F-14 wings closed, (d) F-14 wings opened, (e) Harrier, (f) F-22, and (g) F-1 from Plane dataset obtained from [12].

2.1.6 ScreenType

The ScreenType dataset was derived from data collected as part of a government-funded research initiative titled "Powering the Nation" [12]. The primary objective was to amass behavioral data regarding the way consumers utilize electricity within domestic environments, with the aim of reducing the carbon footprint of the United Kingdom. The dataset comprises measurements from 251 households, gathered at two-minute intervals over the duration of a month. Each sequence consists of 720 data points (representing 24

hours of readings captured every 2 minutes). The categories included are CRT Television, LCD Television, and Computer Monitor.

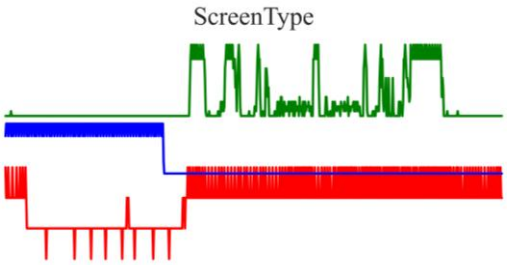


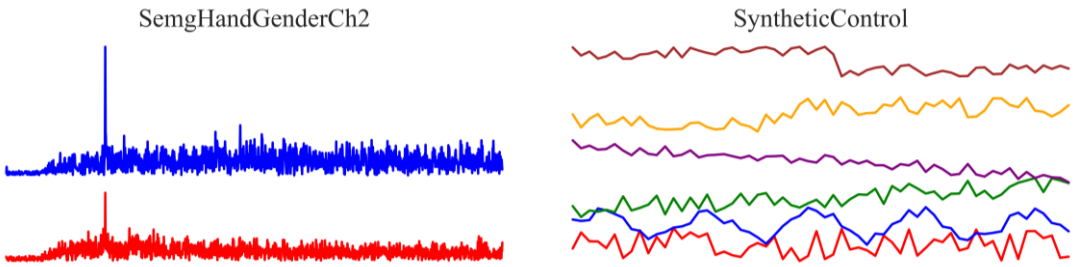
Figure 9. The example of the ScreenType dataset obtained from [12].

2.1.7 SemgHandGenderCh2

The dataset contains surface electromyography (sEMG) signals, which are the electrical activity generated by muscle contractions, either during resting states or throughout a range of movement-based activities. The data collection protocol involves a cohort of five healthy participants who are directed to execute a sequence of six distinct hand grasp maneuvers. A sophisticated 2-channel EMG system was used to acquire this data, which was approved for its accuracy and effectiveness in capturing muscular activity. The categorical classifications within this dataset are aligned with the gender of the participants engaged in the study. - class 1: the female gender - class 2: the male gender. This dataset is attributed to the research conducted by [19].

2.1.8 SyntheticControl

This comprehensive dataset covers 600 unique samples of control charts synthetically produced through the complex methodologies described in the academic article [20]. Within this data collection, there are six distinct categories of control charts, which are listed as follows: 1. the normal category, which signifies standard operating conditions; 2. the cyclic category, characterized by recurrent patterns; 3. the increasing trend category, indicating a consistent upward movement; 4. the decreasing trend category, denoting a gradual decline; 5. the upward shift category, which illustrates a sudden escalation in values; and 6. the downward shift category, representing a sudden reduction in values.



(a) (b)

Figure 10. Time series example of (a) SemgHandGenderCh2 and (b) SyntheticControl datasets obtained from [12].

Chapter 3

State-of-the-Art Overview

This chapter outlines four outstanding approaches to time series classification: feature-based, dictionary-based, interval-based, and shapelet-based. These approaches are the most frequently developed and researched to date, and most existing works have been conducted on univariate time series [1], [21]. We will also explain the ensemble-based method, which falls into the same group as the proposed approach.

3.1 Feature-Based Approaches

Feature-based classifiers have emerged as the prominent current approach. These approaches derive descriptive statistics from time series data to serve as input features for classification models. The features are obtained by transforming the series into a vector representation. This series-to-vector approach is commonly employed in a simple pipeline consisting of feature extraction followed by classifiers, as illustrated in Figure 11.

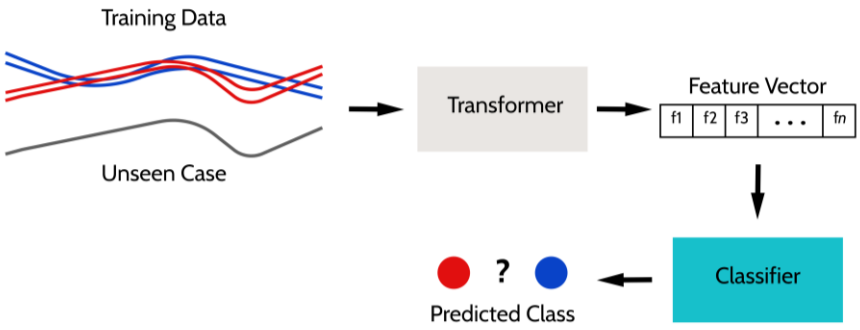


Figure 11. Visualization of feature-based representation obtained from [21].

3.1.1 TSFresh and FreshPRINCE

Time series feature extraction based on scalable hypothesis tests (TSFresh) [22] provides nearly 800 time series characteristics, which can be utilized independently or refined through a feature selection process called FRESH. This method utilizes various hypothesis tests, including the Kendal rank test [23], Kolmogorov-Smirnov test [24], and Fisher's exact test [25], to evaluate features. The Benjamini-Yekutieli method [26] is applied to manage the false discovery rate that results from comparing numerous features and hypotheses all at once. However, the most effective result was obtained by utilizing the whole set of TSFresh features without feature selection, then built on a Rotation Forest classifier [27], which is the so-called FreshPRINCE [28]. Figure 12 shows the flowchart of feature-based algorithms and their relations.

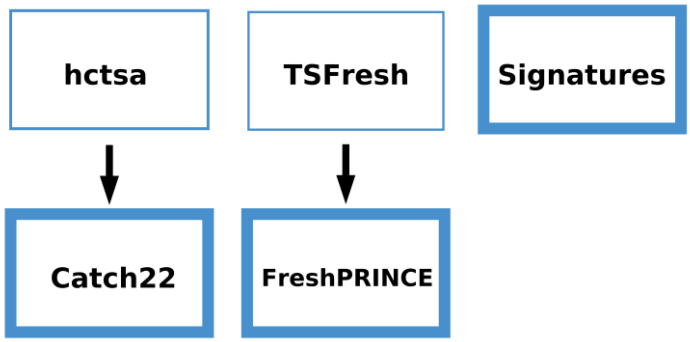


Figure 12. A summary of feature-based approaches and their relations. Our experiments include only those algorithms that are outlined with the thick border line.

3.1.2 Catch22

The highly comparative time-series analysis (hctsa) package [29] can explore more than 7700 features from time series data. The canonical time series characteristics, known as Catch22 [30], consists of 22 hctsa features identified as the most distinctive from the complete set of features. The selection of Catch22 features was made based on an assessment using the UCR/UEA datasets. Initially, the hctsa features are reduced to eliminate those sensitive to the mean and variance of the series. Following this, the features were evaluated based on their predictive performance, leading to removing any features that fell below a certain threshold. A hierarchical clustering analysis was carried out on the correlation matrix to eliminate redundancy for the remaining features. From the clusters created, features are chosen, considering factors such as balanced accuracy, runtime efficiency, and

interpretability. The Catch22 features embrace various concepts, including descriptive statistics, linear correlations, and entropy. After employing the transform to the time series, Catch22 is then trained on a decision tree classifier [30].

3.1.3 Signatures

Generalized signatures [31] use rough path theory to extract features. The signature transform method extracts features by capturing the interaction between values at multiple time points (ordered cross-moments). The pipeline is established with two augmentations. The first augmentation typically involves the original time series directly by adding a zero to the initial point of the time series. The second augmentation is by adding the timestamps of the series. A hierarchical windowing approach is applied to the augmented data, with the signature transform method being subsequently employed on each window. Each window's output is combined into a feature vector. These features are subsequently utilized to construct a Random Forest classifier.

3.2 Dictionary-Based Approaches

Dictionary-based methods find phase-independent subseries. Each subseries is transformed into words. Figure 13 shows how dictionary-based algorithms transform series into words and then use them to construct a classifier. First, splitting a time series into subseries (windowing). Next, each real value from the subseries is converted into a sequence of symbols (discretization). Next, create a sparse feature vector containing word count histograms. Finally, build a machine learning classifier using these feature vectors. The dictionary-based methods are focused on word frequency and are called bag-of-words (BoW) methods. The most popular approach is the Bag of Patterns (BOP) [32]. It is a dictionary-based method that transforms series into words using symbolic aggregate approximation (SAX) based on three hyperparameters: length of window w, length of word l, and size of alphabet a. Another example is SAX-vector space model (SAXVSM) [33], which blends the concept of SAX with the vector space model. Figure 14 shows dictionary-based algorithms with their relations.

Another example is the bag-of-SFA-symbols (BOSS) model [34]. The basis of the BOSS is a representation called symbolic Fourier approximation (SFA) [35]. The BOSS demonstrated strong performance in the initial comparative study. Using the previously outlined process, the individual BOSS classifier uses SFA to convert each sliding window

into a word. Next, a feature vector is created by counting word occurrences across all windows. A non-symmetric distance function and 1-Nearest Neighbor (NN) classifier are used to classify new instances. In experiments using a sample time series, the non-symmetric distance outperformed the Euclidean distance function [34]. The complete BOSS classifier is established from individual classifiers. The final classification for new instances is obtained by aggregating the predictions of the individual BOSS classifiers through a majority voting procedure.

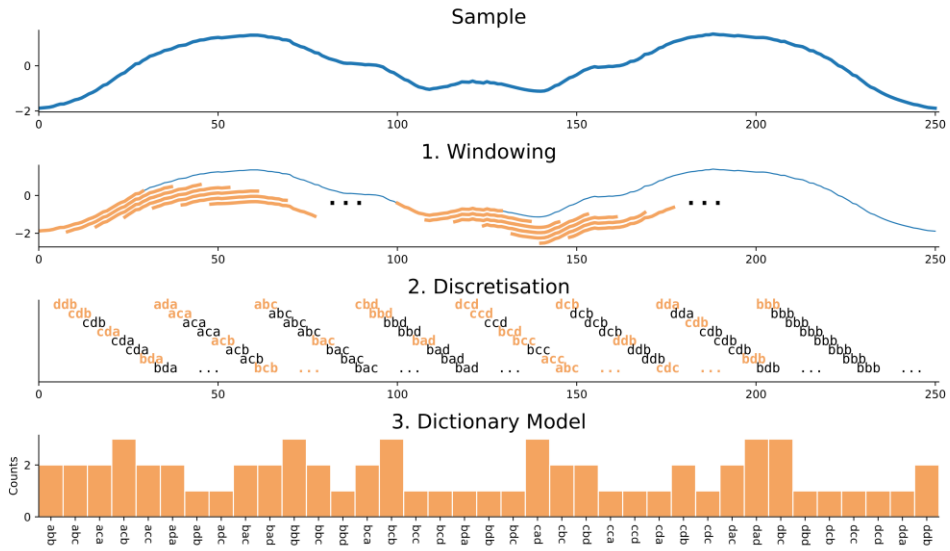


Figure 13. Transformation of a numeric time series into the sequence of words using (1) overlapping time windows, (2) discretizing the windows into words, and (3) creating the word histogram.

In the bake-off, the BOSS was identified as one of the slower classifiers, making it impractical for timely evaluation on larger datasets. To address these scalability challenges, Contractable BOSS (cBOSS) [36] was introduced to modify the ensemble framework of BOSS while retaining its core transformations. The cBOSS employs a random selection process of individual BOSS classifiers, keeping only the top-performing classifiers for the final ensemble. Users can more effectively manage the classifier's training duration by including a training time limit through contraction. To ensemble individual BOSS classifiers, the cross-validation accuracy weighted probabilistic ensemble (CAWPE) scheme [37] is implemented. The changes introduced by cBOSS to the BOSS ensemble framework led to a substantial decrease in training times without losing accuracy. However, BOSS disregards the location of words within sequences, relying only on the frequency of patterns to

determine its classification. Therefore, SpatialBOSS (S-BOSS) [38] is introduced to incorporate temporal information within sequences. Spatial pyramids [39] is used for this purpose. Although this approach offers better accuracy than standard BOSS, the expanded parameter search space and larger feature vector make implementing it practically challenging.

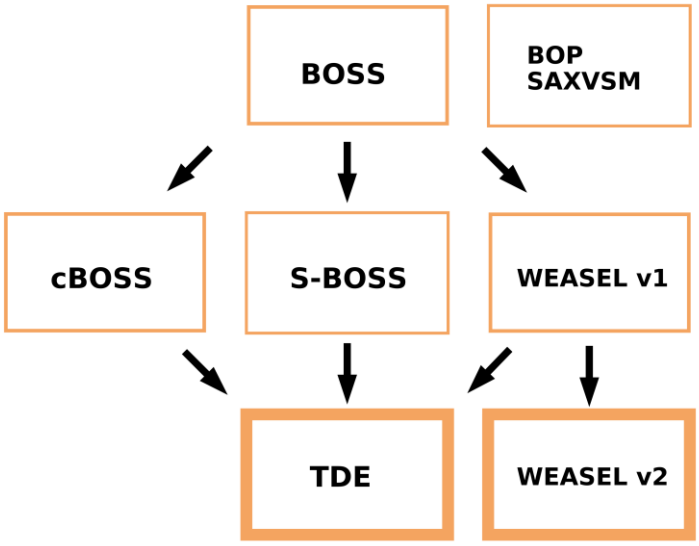


Figure 14. A summary of dictionary-based approaches and their relations. Our experiments include only those algorithms that are outlined with the thick border line.

3.2.1 WEASEL

Word extraction for time series classification (WEASEL v1.0) [40] algorithm works by identifying words that effectively distinguish between classes while eliminating less discriminative words. The classifier creates word count histograms based on the SFA procedure for different sizes of windows and lengths of words, along with bigram words from non-overlapping windows. Each word's discriminatory strength is evaluated using a Chi-squared test, and those below a threshold are removed. Last, a linear Ridge classifier is built on the feature space. Afterward, the dictionary-based WEASEL v2.0 [41] is introduced not only to enhance accuracy but also to resolve the issue of the extensive memory usage in WEASEL v1.0 by managing the search space using randomly parameterized SFA.

This technique also employs a dilated sliding window to extract subseries with non-consecutive values from a time series, where the dilation parameter maintains a fixed gap between each value. The SFA procedure is used to generate words. A variance-based feature

selection method is applied to enhance performance. WEASEL also leverages an information gain based method to obtain discriminative words and identify breakpoints that separate the classes. The final features are then utilized as input to build a linear Ridge classifier.

3.2.2 TDE

The temporal dictionary ensemble (TDE) [42] is a dictionary-based framework that combines cBOSS, WEASEL, and SpatialBOSS, along with new features. TDE is an ensemble of 1-NN classifiers that turns each series into a count word histogram utilizing SFA. From WEASEL, TDE adopts a method to find discretization breakpoints and track bigram frequencies from non-overlapping windows. It includes temporal information derived from the spatial pyramids, which is semilarly utilized in SpatialBOSS. Word counts from each spatial subseries are discovered separately and concatenated into histograms. The final classification is obtained by ensembling the predictions of the individual classifiers through the cBOSS ensemble framework containing a novel approach to sampling within the hyperparameter space.

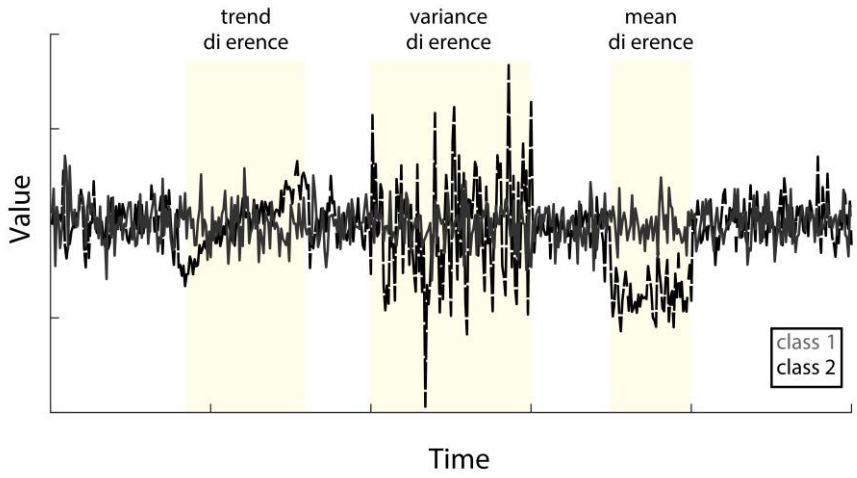


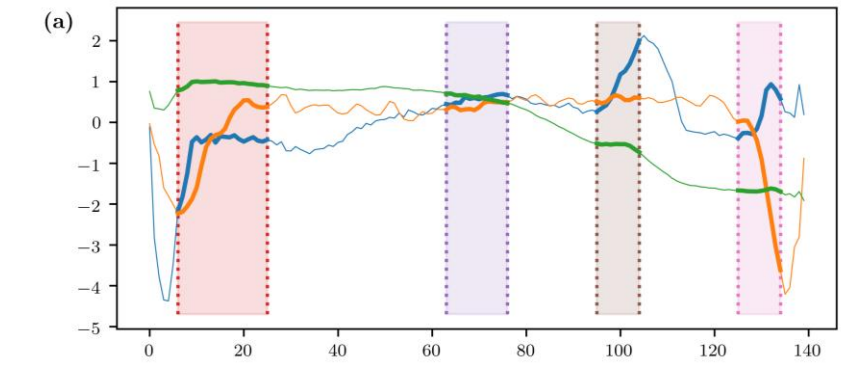
Figure 15. Identifying differences between time series classes. Analyzing feature statistics in subsequences of time series can reveal interpretable differences that distinguish labeled classes. In the example, two series from different classes having differences in linear trend, variance, and mean from distinct intervals. Quantifying these features and their discriminative time intervals provides insight into how and when the classes differ.

3.3 Interval-Based Approaches

Interval-based methods [43] create phase-dependent fixed offset intervals and calculate summary statistics on the intervals. Most methods use random interval selection with the same interval locations for each series. The reason for using intervals is to reduce noise. As illustrated in Figure 15, leveraging intervals can outperform utilizing features derived from the entire time series. The majority of recent interval-based methods commonly utilize a random forest classifier by randomizing the interval selection for each decision tree within the forest. Figure 17 shows interval-based algorithms along with their relations.

3.3.1 TSF

The Time Series Forest (TSF) [43] is a popular ensemble of interval-based decision trees. For each tree in the ensemble, random intervals are selected, and the same interval offsets are applied across all time series. The mean, variance, and slope from summary statistics are extracted for each interval and combined into a feature vector, as illustrated in Figure 16. This feature vector is then used to construct a time series tree, and the same interval-based features are employed for making predictions. The final ensemble prediction is determined by aggregating the predictions of the base classifiers through a majority vote. The base classifier, known as the time series tree, is a modified decision tree that evaluates whole attributes at each node and employs a metric called margin gain for splitting criteria.



(b)

	Interval 1			Interval 2			Interval 3			Interval 4		
	Mean	SD	Slope	Mean	SD	Slope	Mean	SD	Slope	Mean	SD	Slope
Time series 1	-0.61	0.504	0.052	0.58	0.086	0.02	1.004	0.572	0.197	0.189	0.504	0.156
Time series 2	-0.467	0.977	0.158	0.403	0.084	0.017	0.568	0.061	0.011	-1.204	1.305	-0.431
Time series 3	0.944	0.061	0.002	0.604	0.075	-0.018	-0.57	0.065	-0.017	-1.671	0.024	0.003

Figure 16. Overview of TSF: (a) random intervals are extracted from each time series, and (b) three statistical characteristics are calculated for the corresponding subsequences: the mean, standard deviation (SD), and slope.

After that, TSBF is introduced by [44], which stands for Time Series Bag-of-Features. It is a slight extension of TSF. Its fitting process involves the following steps: First, random intervals are created. Each of these intervals is then divided into multiple subintervals. Subsequently, three features akin to those in TSF are extracted.

3.3.2 RISE

The random interval spectral ensemble (RISE) [45] is an interval-based representation based on a decision tree ensemble that utilizes spectral characteristic features. In RISE, only one random interval is selected for each base classifier. Next, the periodogram and autoregression function are computed for each of these intervals. This information is then combined into a feature vector that serves as the input for training a decision tree. RISE was developed primarily for audio-related problems, where spectral features tend to be more discriminatory.

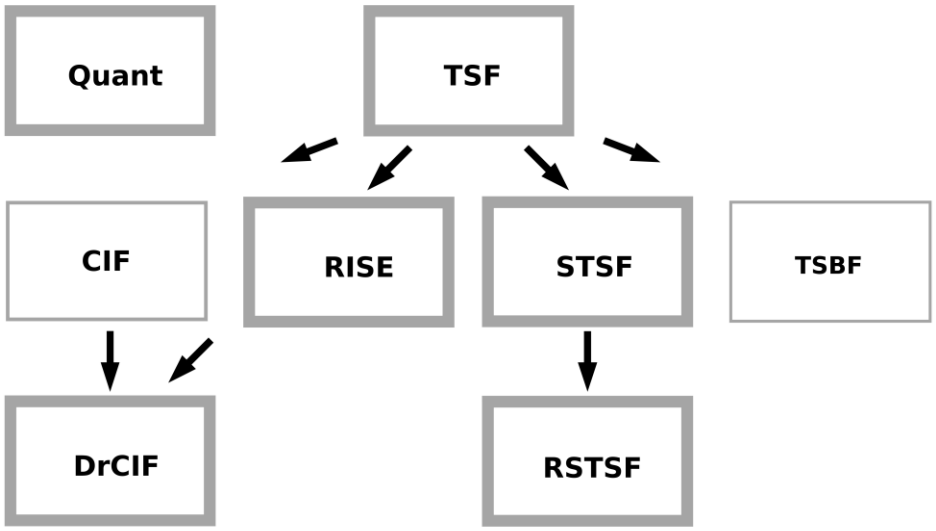


Figure 17. A summary of interval-based approaches and their relations. Our experiments include only those algorithms that are outlined with the thick border line.

3.3.4 STSF and R-STSF

Supervised time series forest (STSF) [46] is also an interval-based algorithm that utilizes a supervised approach to extract relevant intervals from time series data. Once intervals are identified, seven simple summary statistics from a base series, periodogram, and first-order difference representation are generated for each interval. Those are the mean, variance, skewness, kurtosis, autocorrelation, maximum, and minimum values. These generated

feature vectors are then concatenated, and a decision tree ensemble model is built based on the combined feature set.

Randomized STSF (R-STSF) [47] is established as an expansion of STSF, introducing more randomized components into its procedure. Instead of using all the features in the feature set, randomization is used not only in selecting the features for each tree but also in how these features are divided into nodes in the tree. An autoregressive representation is also extracted from each interval in addition to the previous ones. After features are extracted repetitively from each representation, the features are then employed through a pipeline to construct an Extra Trees classifier [48].

3.3.3 CIF

The canonical interval forest (CIF) [49] represents an improved version of TSF, enhancing accuracy through the integration of more informative features. Similar to other interval-based approaches, CIF comprises the decision tree ensemble constructed from features extracted from phase-dependent intervals. CIF incorporates not only the mean, standard deviation, and slope but also involves the Catch22 features. In addition, the diverse representation of CIF (DrCIF) [2] is later introduced as an extended version of CIF. DrCIF adds two other representations: periodograms, which are also similarly utilized in RISE and STSF, and first-order differences, similarly employed in STSF. A more detailed explanation of DrCIF is provided in Subchapter 4.3 DrCIF.

3.3.5 QUANT

The QUANT [50] is the minimalist interval-based method designed to efficiently capture and represent the characteristics of a time series using a single feature type, which is quantiles. QUANT uses four different representations to attain different characteristics of the time series data, such as raw time series, first-order differences, Fourier coefficients, and second-order differences. The core idea behind QUANT is to divide the time series into fixed dyadic intervals. The four representations can have 120 intervals each, or 480 in total. A feature vector is created from the resulting intervals from all four representations. The concatenated feature vector is then utilized to construct an Extra Trees classifier.

3.4 Shapelet-Based Approaches

Shapelet-based methods, similar to dictionary-based methods, leverage subsequences extracted from the time series independent of the phase. Evaluation of a shapelet involves systematically sliding the subseries over the time series and then calculating the z-normalized Euclidean distance between the shapelet and the subseries (window). The distance between a shapelet (s) of length l taken from a time series (x) of the length T, as $s = \langle x_t, x_{t+1}, \ldots, x_{t+l-1} \rangle$, where $1 \leq t \leq T - l + 1$, and $l \leq T$, can be defined as $d(s, x) = |s - x_{t,\ldots,t+l}|$. Figure 18 provides a visual representation of the shapelet distance operation $d(s, x)$.

Shapelets were initially introduced as a fundamental component in [51] and were integrated into a decision tree classifier, which is the so-called shapelet tree. Since the original shapelet was introduced, shapelet research has focused on four primary areas [21]: improving classification accuracy, addressing the computational demands of shapelet discovery, unifying research with convolutions and shapelets, and balancing optimization, randomization, and interpretability when finding shapelets. Figure 19 shows the relation for shapelet-based algorithms.

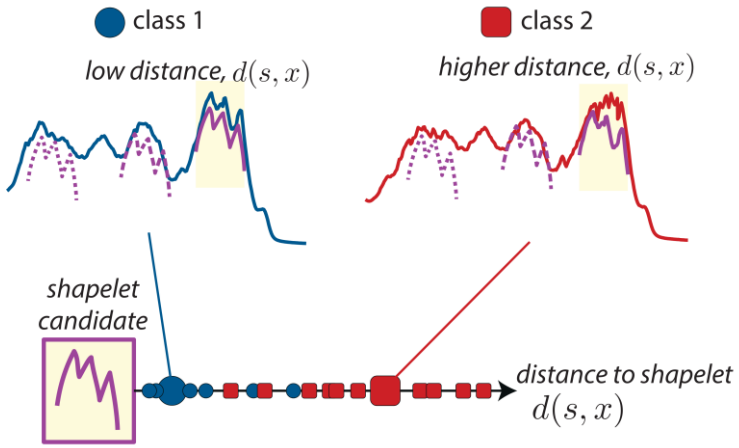


Figure 18. An example of time series from class 1 (blue circles) and class 2 (red squares). The distances between all possible subseries and shapelets plotted as circles/squares. The candidate shapelet appears more similar to the series of class 1 while demonstrating more dissimilarity to the series of class 2, which displays a contrasting peak shape.

3.4.1 STC

The shapelet transform classifier (STC) [52], [53] operates as a pipeline, identifying specific shapelets within the time series data that can best differentiate between classes. The potential shapelets are filtered based on the information gain. After the shapelets are discovered, they are used to convert the series into a vector of distances, which then retrieves the minimum distance to the shapelet from all possible subseries. A new feature vector is generated and subsequently used to construct a Rotation Forest classifier [27]. ST- HESCA is the first version of STC that extensively enumerates all shapelets from all training samples without filtering the shapelets. Essentially, HESCA was used as the base classifier, which was later changed to the name as CAWPE [37]. However, the algorithm does not scale well caused by its extensive evaluation process and weight assignments of base classifiers.

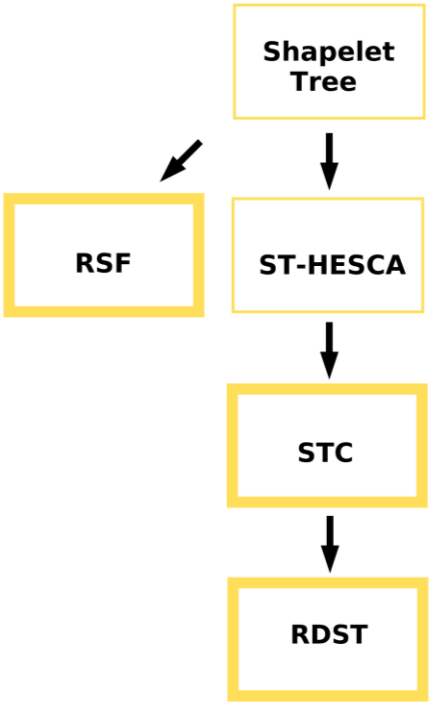


Figure 19. A summary of shapelet-based approaches and their relations. Our experiments include only those algorithms that are outlined with the thick border line.

3.4.2 RSF

The random shapelet forest (RSF) [54] is an improvement of the shapelet tree to boost the computational speed and accuracy by randomization and ensemble. A bagging-based tree ensemble is implemented. In the context of randomization, univariate shapelets are randomly chosen from the training samples at each tree node. Each shapelet's length is also

randomly determined, constrained by pre-determined upper and lower limits. The candidate of shapelets is assessed using information gain, and the optimal shapelet is selected. Predictions for new data are made by majority vote.

3.4.3 RDST

The random dilated shapelet transform (RDST) [8] is considered an extension of STC that aims to improve the scalability and flexibility of the shapelet transform by incorporating dilation and randomization to avoid overfitting and computational complexity. Instead of searching for optimal shapelets from the training samples, RDST uses a unique approach by randomly selecting many shapelets from the training samples, generally an order of thousands to ten thousand, and trains a linear Ridge classifier on features derived from shapelets, details described in Subchapter 4.4 RDST.

3.5 Ensemble-Based Approaches

Ensemble-based algorithm (in some literature called as hybrid-based [9], [21]) utilizes a collection of classifiers, wherein the predictive output of each classifier may be weighted to optimize overall classification accuracy. A fundamental prerequisite for the efficacy of ensemble-based algorithms is the presence of diversity within the ensemble. Diversity can be realized through several strategies, including the classifiers involved in different representations, the selection of unique feature sets for each classifier, or the application of resampling techniques to the training samples for each classifier. Recently, there has been growing interest in using ensemble algorithms for time series classification [2], [34], [43], [55], primarily attributable to the capacity to enhance classification accuracy [2], [55]. Figure 21 shows the flowchart of ensemble-based algorithms and their relations.

Ensemble methods can be categorized into four distinct groups. The first group is heterogeneous ensembles, in which each component comprises a classifier designed based on a specific type of representation. For example, the collective of transformation-based ensembles (COTE) [55] is considerably more accurate than the individual components. The COTE is a collection of 35 classifiers from the time domain, shapelet, autocorrelation, and power spectrum representation. COTE has limitations on scalability. Therefore, the HIVE-COTE (HC) algorithm [56] has been developed to resolve this issue, offering significantly better accuracy than COTE. Since it was first proposed in 2016, the HIVE-COTE has been

subject to minor revisions in HIVE-COTE version 1.0 (HC1). There are four base classifiers in the HC1, namely the TSF [43], RISE [45], cBOSS [34], and STC [52].

In 2021, HC1 underwent another revision to improve scalability and include new ways of representing individual classifiers. Therefore, HIVE-COTE v2.0 (HC2) [2] was proposed; see its structure in Figure 20. HC2 involves several modifications: The cBOSS was replaced with TDE [42] as the dictionary-based classifier. Both TSF and RISE were also changed with DrCIF as the new interval-based classifier. Moreover, Arsenal, an ensemble of ROCKET classifiers, is introduced as a novel convolutional-based classifier. Instead of using cross-validation, an adapted form of out-of-bag error is used for test accuracy estimation, although the final model still utilizes all of the training samples. Notably, HC2 extends the capabilities of HC1 by enabling the classification of multivariate time series.

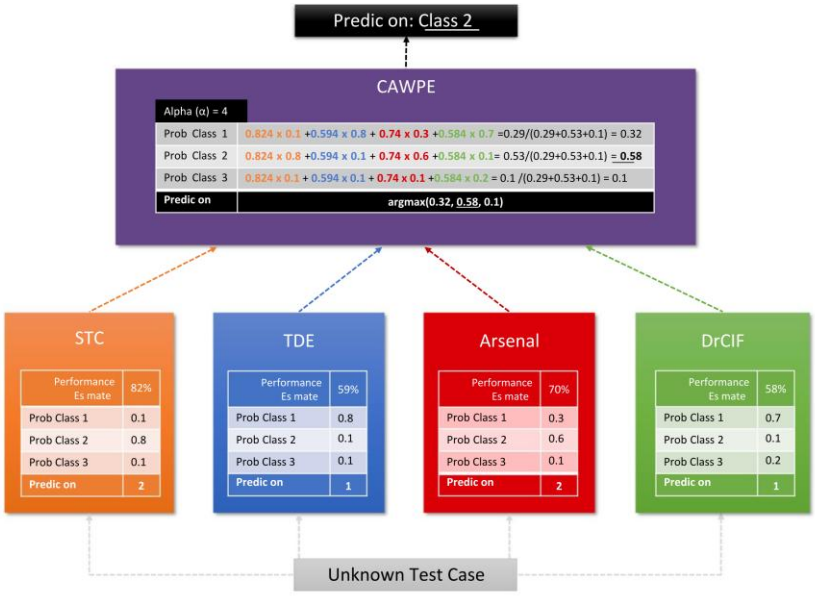


Figure 20. A review of the HC2 ensemble structure obtained from [2].

Another approach in this group is the time series combination of heterogeneous and integrated embedding forest (TS-CHIEF) [3]. TS-CHIEF is a heterogeneous ensemble that takes the closest resemblance to HC. It comprises an ensemble of trees that includes distance, dictionary, and spectral features. The TS-CHIEF was developed to capture the benefits of utilizing multiple representations while avoiding the significant computational time associated with the original HC. Hybrid dictionary-ROCKET architecture (Hydra) [4] is also a heterogeneous ensemble model which utilizes convolution-based and dictionary-based representations.

The second group comprises tree based homogeneous ensembles that incorporate a specific representation within the tree nodes. One notable technique is the Proximity Forest (PF) [6], an ensemble built upon a proximity tree. Unlike a traditional decision tree, a proximity tree differs in each node's split criteria by randomly choosing distance measures from Elastic Ensembles (EE) [57] as the basis for splitting. PF uses the same 11 distance functions as EE, yet it surpasses the original EE algorithm in both accuracy and scalability.

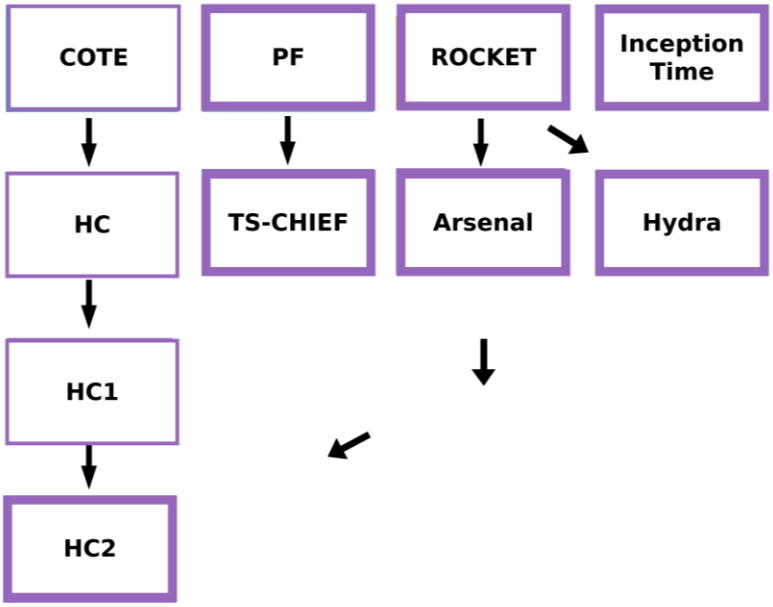


Figure 21. A summary of ensemble-based approaches and their relations. Our experiments include only those algorithms that are outlined with the thick border line.

The third group is deep learning ensembles with embedded network representations. For instance, InceptionTime [7] utilizes an ensemble of five deep learning classifiers, all sharing an identical architecture based on sequential Inception modules [58]. Model diversity is established by randomizing the initial weight across the five models. As illustrated in Figure 22, the network involves two successive residual blocks, each consisting of three inception modules. To mitigate the vanishing gradient problem, a shortcut connection links the input and output of each residual block. Following the residual blocks is a global average pooling (GAP) layer. In the end, a fully-connected layer with softmax activation function is implemented.

The fourth group utilizes convolution-based ensemble techniques to generate extensive new feature spaces, which are then analyzed using a linear classifier. One popular algorithm within the category is the ensemble of ROCKET models, also known as Arsenal [2].

ROCKET [5] operates as a pipeline. It generally creates thousands to ten thousands of convolutional kernels with randomized parameters. Input data undergoes transformation via these kernels by applying two distinct pooling functions: the maximum value and the proportion of positive values (PPV). Subsequently, the resulting features are concatenated to generate a comprehensive feature vector from entire kernels.

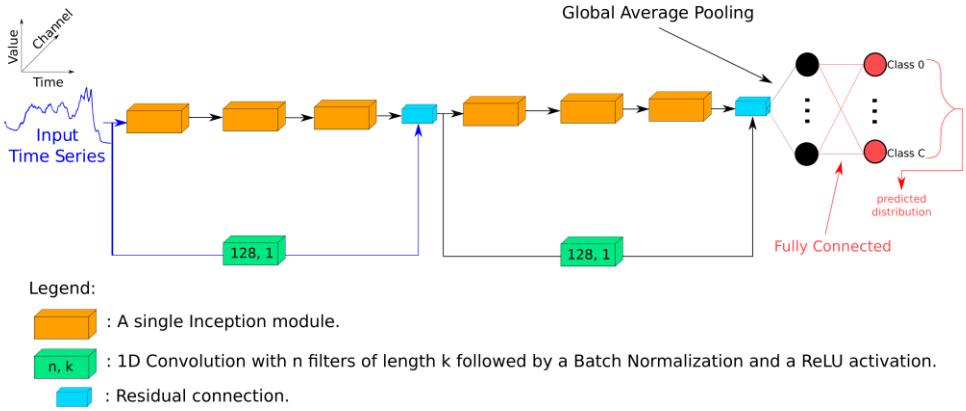


Figure 22. A review of the InceptionTime structure obtained from [7].

Chapter 4

The MuRBE Structure

This section provides a concise understanding of the MuRBE structure. The MuRBE is a heterogeneous ensemble having four component modules where each component originates from a unique representation. The component modules are: the ARFIMA-RF from the feature-based representation [P1]; SAX-SGCNN from the dictionary-based representation [P2]; the interval-based DrCIF [2]; and the shapelet-based RDST [8]. These were chosen due to being well performed in their domain representation [21]. The MuRBE adopts the fuzzy rank-based ensemble framework, which has recently gained attention for its effective meta-ensemble approach for different classifiers across different datasets, even in limited domain expertise or prior knowledge [59], [60], [61].

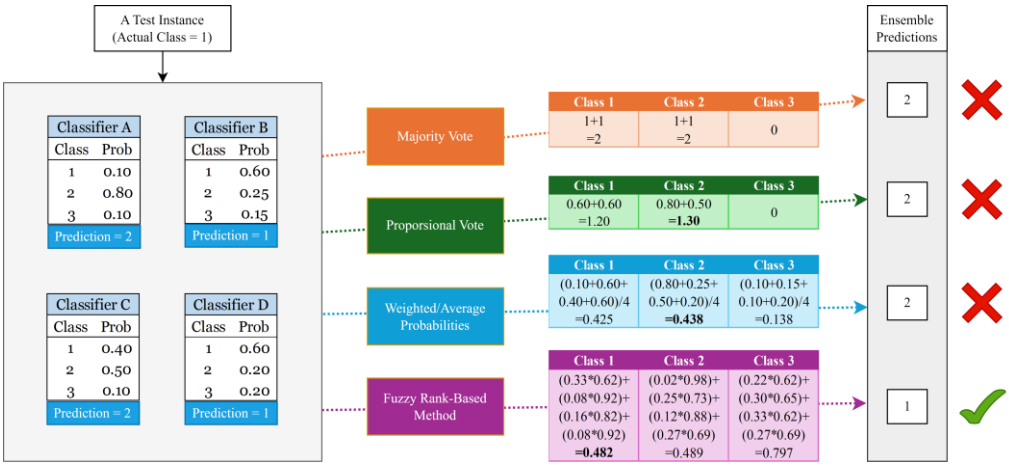


Figure 23. Illustration of the effect of the fuzzy-rank method compared to other schemes on a toy instance classification.

Figure 23 gives an overview of the fuzzy-rank ensemble that differentiates it from other schemes, i.e., majority voting, proportional voting, and weighted average probabilities. Instead of constructing a large number of weak classifiers, our approach focuses on developing a smaller set of effective classifiers and combining their outputs. We employ probability estimates rather than point predictions because they contain more information, which is crucial when using fewer classifiers to capture all available information.

The structure of MuRBE is presented in Figure 24. The construction of the fuzzy-rank ensemble needs to obtain the probability estimates of each class from all the base classifiers. In the initial phase, to classify a new case, each component is trained independently using time series data and then required to generate each class's probability estimates (confident scores). Next, the probability estimates are then used to calculate fuzzy scores through nonlinear functions. Due to incorporating nonlinear functions to process decision scores, the fuzzy rank-based ensemble provides more flexible, dynamic, and adaptive weights for individual models based on their performance in specific contexts of the input space. Unlike traditional ensemble methods, e.g., simple average or weighted average rules, they often use fixed weights, which may not adapt well to varying conditions. The nonlinear functions, such as an exponential and hyperbolic tangent function, are often used due to their performance by constructing a tilted distribution to extenuate differences in classifiers [59], [61], [62].

Hence, to ensure effectiveness, we will employ the same nonlinear functions adopted from the previously cited reference. The probability estimates from the base classifiers will undergo transformation through two nonlinear functions: the exponential function and the hyperbolic tangent function. The hyperbolic tangent function acts as a reward function, while the exponential function acts as a decreasing or deviation function. Figure 25 displays the graphs of these functions. The x-axis represents the probability estimate of a class, and y-axis represents the fuzzy score. The exponential function measures the deviation from its objective for a class with a given probability estimate. As the probability decreases, the deviation diminishes, ultimately approaching 0 when the probability estimate equals 1. In contrast, the hyperbolic tangent function assesses the reward allocated to a class. The reward increases as the probability rises, eventually reaching 1 when the probability estimate equals 1. Consequently, using two nonlinear functions with varying concavities aims to yield complementary outcomes.

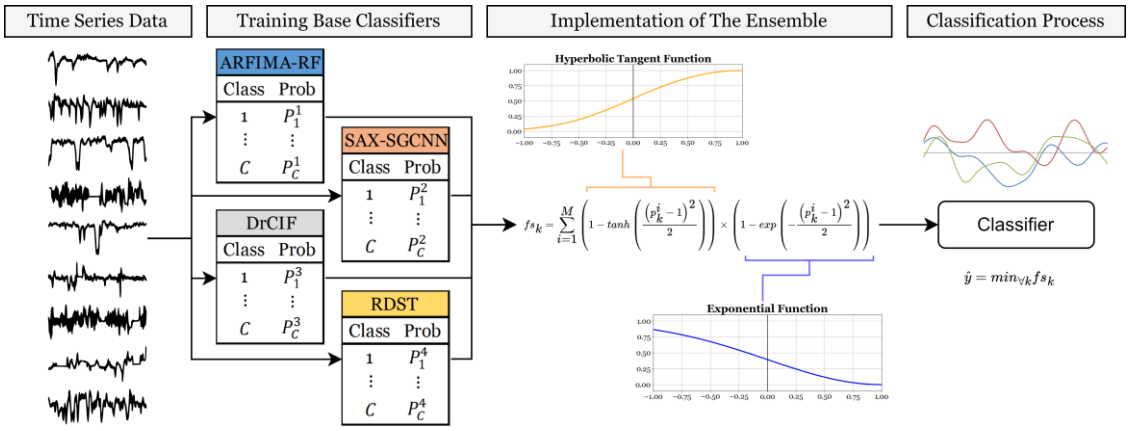


Figure 24. The structure of the proposed method.

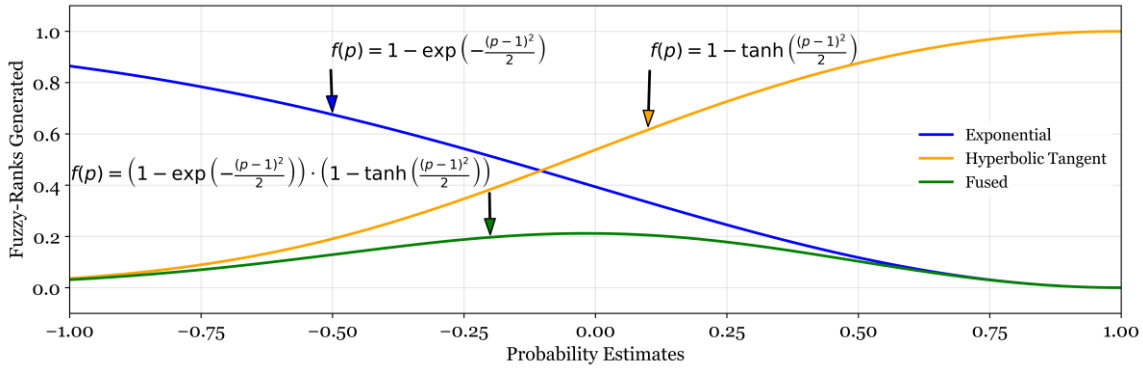


Figure 25. Visualizations of nonlinear functions utilized for calculating fuzzy ranks.

After mapping the probability estimates into two functions with different concavities to produce nonlinear fuzzy ranks, we combine these ranks to produce a rank score. The rank score is determined based upon the product of deviation and reward corresponding to a specific probability estimate. In each base classifier, this procedure is executed, and the rank scores are aggregated to produce the final fused score. A higher probability estimate leads to a lower fused score, signifying a more accurate prediction. Therefore, the predicted class is associated with the smallest fused score. To enhance understanding of the concepts, we provide detailed steps for the proposed fuzzy rank-based structure as follows:

First, we calculate all the probability estimates. The probability estimates of classes given by the base classifier are defined as p_k^i, where $k = 1, 2, \dots, C$ is the number of classes and $i = 1, 2, \dots, M$ is the number of base classifiers. Given that $p_k^i \in [0,1]$, the probabilities $[p_1^i, p_2^i, \dots, p_c^i]$ of c classes on the base classifier-i essentially will satisfy the following condition,

$$\sum_{k=1}^{C} p_k^i = 1, \forall i = 1,2,\dots,M. \tag{1}$$

Since we utilized two nonlinear functions, let us consider $\left[r_1^{i1}, r_2^{i1}, \dots, r_C^{i1}\right]$ and $\left[r_1^{i2}, r_2^{i2}, \dots, r_C^{i2}\right]$, which are fuzzy ranks respectively computed using the hyperbolic tangent and the exponential functions expressed by

$$r_k^{i1} = 1 - tanh\left(\frac{\left(p_k^i - 1\right)^2}{2}\right), and\ r_k^{i2} = 1 - exp\left(-\frac{\left(p_k^i - 1\right)^2}{2}\right). \tag{2}$$

As a result, we can then determine the rank scores rs_k^i through the multiplication of fuzzy ranks r_k^{ij}. The fuzzy ranks are obtained from nonlinear functions by substituting the probability estimates from the base classifiers. The standard notation for calculating the rank scores is presented as follows,

$$rs_k^i = \prod_{j=1}^{S} r_k^{ij}, \forall k = 1,2,\dots,C \text{ and } \forall i = 1,2,\dots,M, \tag{3}$$

where $j = 1, 2, \dots, S$ is the number of chosen nonlinear functions, and the functions are bounded within [0, 1]. The product of the two nonlinear functions is denoted as $rs_k^i = r_k^{i1} \times r_k^{i2}$. After that, the final fused score $[fs_1, fs_2, \dots, fs_C]$ is calculated by the equation

$$fs_k = \sum_{i=1}^{M} rs_k^i, \forall k = 1,2,\dots,C. \tag{4}$$

The predicted class is selected based on the minimum value of the final fused score. This fused score serves as the ultimate value for each class, which can be expressed through the following formula,

$$\hat{y} = \min_{\forall k} fs_k. \tag{5}$$

Figure 26 illustrates an example of how our proposed ensemble method works.

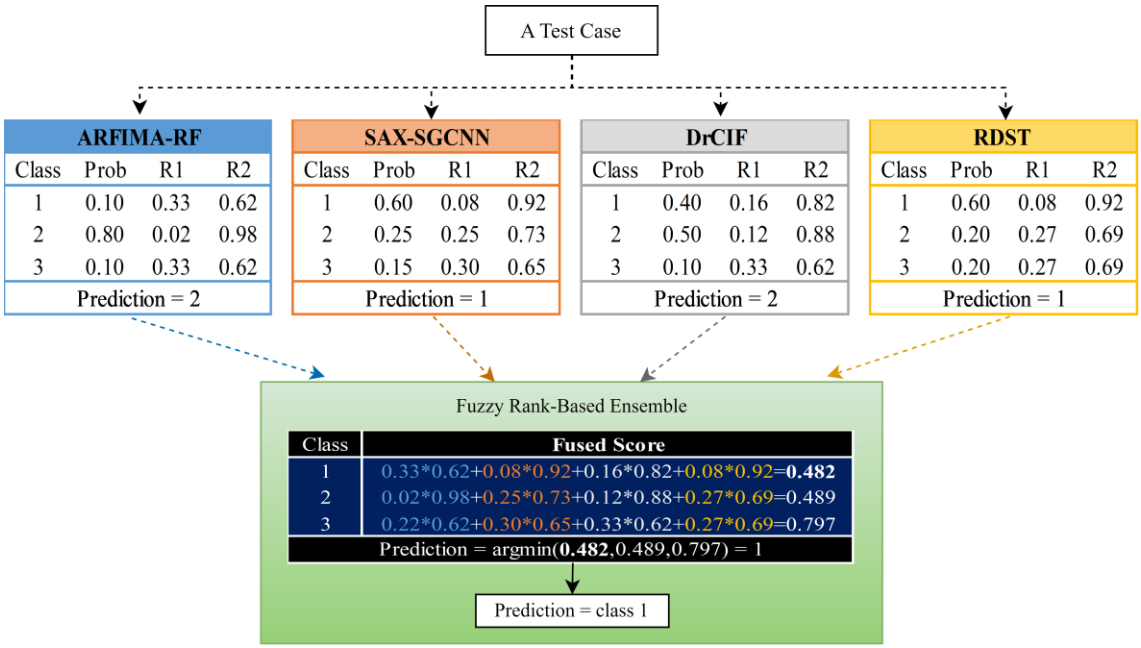
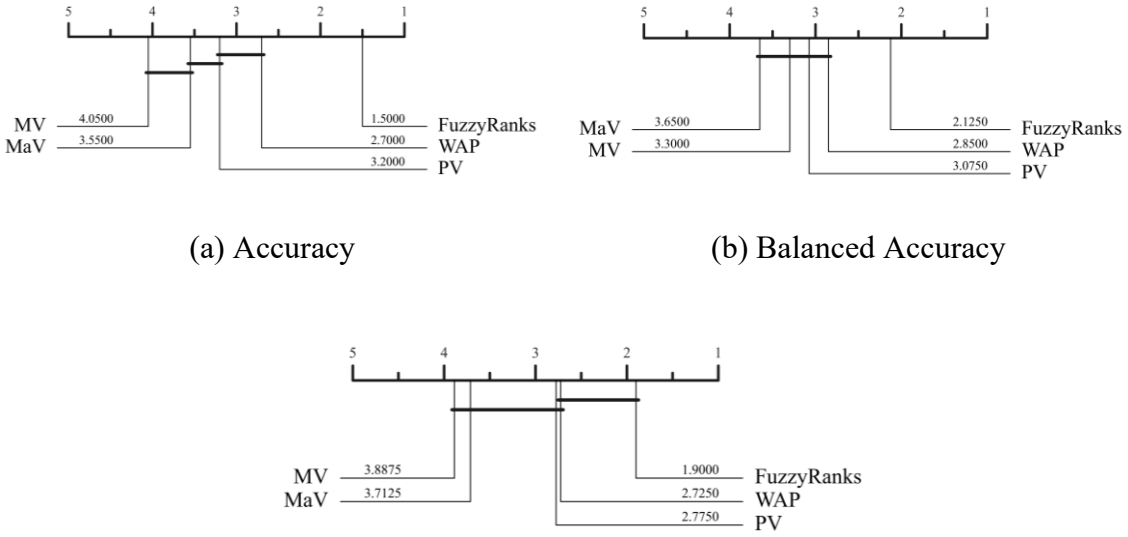


Figure 26. An overview of the proposed ensemble structure for a three-class problem.

The study makes a unique contribution by integrating various domain classifiers through a fuzzy rank-based ensemble framework, which has not previously been explored in the context of time series classification. The question that may arise is whether fuzzy rank-based structure outperforms other meta-ensemble schemes. To answer this curiosity, we conducted a comparison of the fuzzy ensemble to four well-known alternative schemes based on the framework described by [63],

1. Max Vote (MV)
2. Majority Vote (MaV)
3. Proportional Vote (PV)
4. Weighted Average Probabilities (WAP).

We continued the evaluation by using a simple set of base classifiers. We include well-known classifiers that are fast to build. These are logistic regression (LR), linear support vector machine (SVM), C4.5 decision tree (C4.5), 1-nearest neighbor (NN), and Naive Bayes (NB). In the comparison, all ensemble schemes utilize an identical set of base classifiers, so the only source of variation is the ensemble schemes. All base classifiers are built on 40 UCR/UEA time series classification datasets, described in Table 4. Therefore, we are totally investigating the ability of the ensembles to combine predictions with exactly the same available meta-information.



(a) Accuracy

(b) Balanced Accuracy

(c) F1 Score

Figure 27. Critical difference (CD) diagrams for five ensemble schemes on 40 UCR/UEA datasets built using LR, SVM, C4.5, NN, and NB base classifiers. The ensemble schemes are MV, MaV, PV, WAP, and FuzzyRanks.

We visualize comparisons with critical difference (CD) diagrams. The details of interpreting the diagram are explained in Subchapter 5.1 Experimental Setup. The CD diagram summarizes the average ranks of five ensembles, which are illustrated in Figure 27. These ensembles were evaluated using three performance metrics on a test split of 40 UCR/UEA datasets. FuzzyRanks achieved the highest rank and occupied the top clique based on accuracy, balanced accuracy, and F1 score. Statistical evaluation reveals that FuzzyRanks significantly outperformed all other schemes in accuracy and balanced accuracy. For the F1 score, it is insignificantly different from the WAP, but it is significantly better than the others. Upon reviewing the results, it becomes evident that the fuzzy rank-based scheme offers greater consistency than other schemes, as it constantly ranks in the top tier on all metrics. The result shows that FuzzyRanks is a practical initial choice for combining small sets of base classifiers in various problems.

4.1 ARFIMA-RF

Autoregressive fractionally integrated moving average with random forest (ARFIMA-RF) is a feature-based method that uses the estimated parameters of the autoregressive fractionally integrated moving average (ARFIMA) as attributes. This model was developed to overcome the weaknesses of an autoregressive integrated moving average (ARIMA). In

time series, the ARFIMA process shares characteristics similar to those of a nonstationary ARIMA model. For example, the autocorrelation of a stationary ARFIMA(p, d, q) model decays very slowly. At the same time, this phenomenon is similar to the sample autocorrelation of a nonstationary ARIMA time series addressed by [64]. In addition, we realized that periodograms of both stationary ARFIMA models and nonstationary ARIMA models diverge at zero frequency. Hence, there is often misspecification of models due to these similarities. For instance, there are cases where using integer differencing d, where $d = 1,2,3,\cdots$, could be too severe to be specified as a nonstationary ARIMA model and result in overdifferencing. As a result of the overdifferencing, a bias (error) with an inflated variance will result from parameter estimation. Reference [65] revealed that the ARFIMA model is not only able to overcome the misspecification but also to estimate a nonstationary time series model with fractional differencing. The ARFIMA model shares the same form of representation as the ARIMA with three hyperparameters, namely p, d, and q. A given time series $\{x_t\}_{t=1}^{T}$ is fitted to an ARFIMA model of order (p, d, q) represented by a form

$$\Phi_p(B)\Delta^d x_t = \theta_0 + \Theta_q(B)e_t, \tag{6}$$

where

$$\Phi_p(B) = 1 - \phi_1 B - \phi_2 B^2 - \cdots - \phi_p B^p, \tag{7}$$

$$\Theta_q(B) = 1 - \theta_1 B - \theta_2 B^2 - \cdots - \theta_q B^q, \tag{8}$$

$$\Delta^d = (1 - B)^d, \tag{9}$$

and B represents the backward shift operators defined by $B^k x_t = x_{t-k}$.

First, $\Phi_p(B)$ is a polynomial of an autoregressive model of order p, denoted as AR(p). It describes the current value depending on previous values and how strong is the relationship between those values through its parameters $\phi_1, \phi_2, \phi_3, \ldots, \phi_p$. The AR$(p)$ model is equivalent to the multiple regression model, but its predictors are time-lagged values of series rather than standard attributes expressed by the following equation,

$$x_t = \theta_0 + \phi_1 x_{t-1} + \phi_2 x_{t-2} + \cdots + \phi_p x_{t-p} + e_t. \tag{10}$$

Second, $\Theta_q(B)$ is a polynomial of a moving average model of order q, denoted as $\mathrm{MA}(q)$. Based on the moving average model, the current value of the series depends on the previous and current white noise errors or random shocks e_t. It is expressed through its parameters $\theta_1, \theta_2, \theta_3, \dots, \theta_q$. The random shocks at each point are assumed to be independently distributed with a zero mean and a constant variance normal distribution. We can express $\mathrm{MA}(q)$ equation as follows,

$$x_t = \theta_0 + e_t - \theta_1 e_{t-1} - \theta_2 e_{t-2} - \cdots - \theta_q e_{t-q}. \tag{11}$$

Third, $(1-B)^d$ is the fractional integration (FI). Integration means the action to represent the differencing of raw data series, e.g., overcoming long-range dependence (long-run memory) or accommodating the nonstationary of a time series. The non-integer value of d is the nonseasonal differencing parameter. Due to the non-integer of the differencing parameter d, the fractional integration to the ARFIMA model is expressed by formal binomial series expansion. Therefore, the component $(1-B)^d$ in the equation (9) can be written,

$$\Delta^d = (1-B)^d,$$

$$\Delta^d = \sum_{k=0}^{\infty} \binom{d}{k} (-B)^k,$$

$$\Delta^d = \frac{d!\,(-B)^0}{0!\,(d-0)!} + \frac{d!\,(-B)^1}{1!\,(d-1)!} + \frac{d!\,(-B)^2}{2!\,(d-2)!} + \frac{d!\,(-B)^3}{3!\,(d-3)!} + \cdots,$$

$$\Delta^d = 1 - dB - \frac{1}{2}(1-d)dB^2 - \frac{1}{6}(1-d)(2-d)dB^3 - \cdots. \tag{12}$$

Some characteristics of the fractional integration for different values of d are presented as follows,

- if $d = 0$, the series x_t is said to follow an ARMA process.
- if $0 < d < 0.5$, then the stationary series x_t is said to follow the ARFIMA process with the autocorrelations decaying very slowly or hyperbolically.
- if $d > 0.5$, then the series x_t is said to follow a nonstationary ARFIMA process.

We can simply apply equation (12) to the form of ARFIMA. For illustration, the simplest ARFIMA$(0, d, 0)$ is given in standard notation,

$$\Delta^d x_t = \theta_0 + e_t, \tag{13}$$

with the help of equation (12), we can obtain

$$x_t - dx_{t-1} - \frac{1}{2}(1-d)dx_{t-2} - \frac{1}{6}(1-d)(2-d)dx_{t-3} - \cdots = \theta_0 + e_t,$$

$$x_t = \theta_0 + dx_{t-1} + \frac{1}{2}(1-d)dx_{t-2} + \frac{1}{6}(1-d)(2-d)dx_{t-3} + \cdots + e_t, \tag{14}$$

and in the same way, we can expand to the higher order p and q of the ARFIMA model. Assuming $\dot{x}_t = \Delta^d x_t$ as a convenience, we can specifically provide the general representation of the ARFIMA(p, d, q) after plugging in equations (7), (8), and (9) into (6) leads to

$$\dot{x}_t = \theta_0 + \phi_1 \dot{x}_{t-1} + \phi_2 \dot{x}_{t-2} + \cdots + \phi_p \dot{x}_{t-p} + e_t - \theta_1 e_{t-1} - \theta_2 e_{t-2} - \cdots - \theta_q e_{t-q}. \tag{15}$$

Using ARFIMA parameters as attributes for time series classification is a unique approach. These parameters describe the characteristics of the time series, such as the stochastic intensity of the impact of time-lagged values and previous white noise (innovation term) on the current value of the time series. By fitting an ARFIMA model for a given time series and using the estimated parameters as attributes, we can characterize classes between different time series based on their correspondence. The parameters used as attributes include d (the fractional differencing), $\phi_1, \phi_2, \phi_3, \ldots, \phi_p$ (consecutive values of coefficients in the autoregressive model), θ_0 (the deterministic trend term), and $\theta_1, \theta_2, \theta_3, \ldots, \theta_q$ (coefficients from the moving average model). This approach has the advantage of producing a much smaller set of features than the length of the time series itself. The scheme of features from three ARFIMA components is generated as follows,

- p_j features from the AR component with the best discovered order, where $p_j \in [0, p]$,
- q_j features from the MA component with the best discovered order, where $q_j \in [0, q]$, and

- two features for the deterministic trend θ_0 and the fractional differencing d,

where $j = 1,2,3, \dots, N$ is the number of samples in a given dataset. As a result, we truncated a particular time series to at most $p + q + 2$ new features. We decided to discover the best configuration of ARFIMA in each time series where the values of p and q are the highest discovered order from the particular training set. If we obtain the best model configuration of p_j lower than p or q_j lower than q, then irrelevant parameters are set to zero. We used the Hyndman-Khandakar algorithm to automatically identify the best configuration of the order of the ARFIMA model in specific interval values [66]. The algorithm simply implements a step-wise procedure for traversing the model space according to the Akaike information criterion (AIC) and then estimates the ARFIMA parameters. The generated features are then used in a pipeline to construct a random forest classifier for the classification process.

4.2 SAX-GCNN

We introduce symbolic aggregate approximation with stacking gated recurrent units and convolutional neural networks (SAX-SGCNN) as a dictionary-based time series classification framework. It transforms time series into a sequence of words using the SAX and employing stacked deep learning for classification. The SAX-SGCNN can be deconstructed into four distinct stages. The initial stage involves standardizing the time series. The second step focuses on transforming the time series into a symbolic representation. Afterward, the third step involves creating word sequences based on the symbolic representation and adding them to a corpus. The final step utilizes the generated corpus to train a stacked deep learning model. The subsequent sections provide a detailed exposition of the intricacies underlying this approach.

4.2.1 Standardizing the time series

The initial phase of the process involves preprocessing the data through z-standardization. This step facilitates the data transformation by aligning the data with a standard normal distribution. A time series x_t can undergo standardization to produce a time series z_t of the same length. This standardization process is accomplished by applying the z-score formulation as follows,

$$z_t = \frac{x_t - \mu_x}{\sqrt{\sigma_x^2}}, t = 1,2, \dots, T, \tag{16}$$

where z_t denotes the standardized time series for the t-th element, x_t represents the observed value of the time series at the t-th time point, μ_x is the mean of the time series, and σ_x denotes the standard deviation of the time series.

4.2.2 Converting numeric time series into symbolic time series

In the subsequent step of the technique, we transform the numerical time series into a symbolic representation. We used the modified SAX procedure as a potential approach for our study. In a standard SAX procedure, piecewise aggregate approximation (PAA) is initially used to manipulate the number of dimensions in the time series [67]. However, we see this PAA representation increases the chances of missing vital parts in the time series data. Therefore, our proposed method employs a modified form of SAX transformation that omits the PAA step. We utilized the modified SAX technique, where a time series will have the same length as the sequence of symbols.

The discretization process involves dividing the area under the Gaussian curve into distinct intervals, denoted as $Ir = \{Ir_0, Ir_1, Ir_2, \dots, r_{a-1}, Ir_a\}$. These intervals are created by equally slicing the area using the chosen alphabet size (a). The first interval, Ir_0, represents negative infinity, while the last interval, Ir_a, represents positive infinity. It is important to note that each interval, Ir_j, is smaller than the subsequent interval, Ir_{j+1}, where j is less than a. If the time point falls inside a specified interval, the numerical time point is substituted with the designated alphabetical symbol. Using three alphabets, Figure 28 demonstrates the example process of transforming numerical time series into symbols.

4.2.3 Generating words from symbolic time series

The window size (w), must be selected to extract words. The selection of the word length range is at the researchers' discretion. The selection of the word length range is at the researchers' discretion, intending to guarantee a sufficient number of words for training process, particularly when dealing with shorter time series. If n represents the length of the symbolic time series, a word list consisting of n/w words is generated for each symbolic time series. The process of extracting words begins at the initial point of each symbolic time series with non-overlapping words. Let us consider the sequence of symbols consisting of w

elements as the window size. It is considered a single word. Subsequently, the window is shifted to the subsequent place in the symbolic time series to locate the subsequent word. Thus, the symbolic series is dissected to obtain a consecutive set of words, each having a length of w. The method of extracting words is illustrated in Figure 28 with a window size of 4.

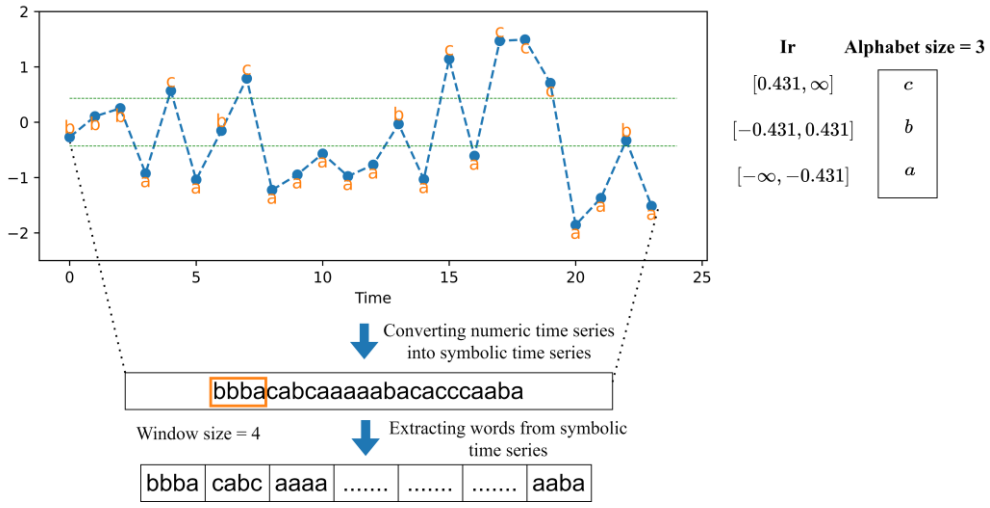


Figure 28. The concept of transforming numerical data into symbols and generating a sequence of words.

4.2.4 Training deep learning model

In this paper, we introduced the SGCNN model, which is a combination of GRU positioned at the top and followed by CNN layers. It is a modified architecture from the current NLP paper [68]. Modification of the architecture, presented in Figure 29, revolved around removing several layers and trying with different building blocks of the network to ensure that it is not prone to either overfitting or underfitting.

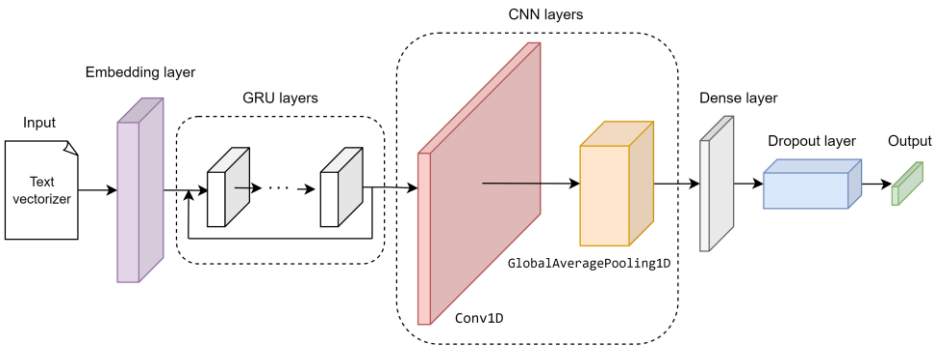


Figure 29. Proposed stacked deep learning architecture.

Firstly, a 4-dimensional embedding layer is acquired through training to align with each word's surrounding context. This approach uses a dense vector representation where a vector represents the projection of the words or documents into a continuous vector space. The embedding layer also allows us to transform input data from a high-dimensional to a lower-dimensional space. This is a way to facilitate more effective data processing and increased network understanding of the semantic relationships between words in the input sequence. With random weights as its starting point, the Embedding layer will discover an embedding for each word during training.

Embeddings are then passed through GRU layers. The GRU is an upgraded form of the long short-term memory model (LSTM). There are three gates in the LSTM, namely an input gate, a forget gate, and an update gate. The GRU, on the other hand, is more straightforward than the LSTM because it contains only two gates, the reset and update gates. Furthermore, the GRU has fewer parameters than LSTM. It makes GRU simpler and faster to train than LSTM, especially with small data in time series classification. In the GRU, the update gate (u_t) enables the neural network to determine the information from the previous hidden states (H_{t-1}) used to describe the future. Moreover, the reset gate (r_t) works to determine the information of H_{t-1} to be forgotten, and it is used to construct the current memory c_t for capturing the relevant information in H_t. The u_t, r_t, c_t and H_t have the following expressions,

$$\begin{aligned} u_t &= \sigma(W_u z_t + U_u H_{t-1}), & r_t &= \sigma(W_r z_t + U_r H_{t-1}), \\ c_t &= tanh(W_c z_t + r_t \odot U_c H_{t-1}), & H_t &= (1 - z_t) \odot H_{t-1} + z_t \odot c_t. \end{aligned} \tag{16}$$

where this model relies on learning parameter matrices W and U, and a sigmoid activation function $\sigma(\dots)$.

Next, the 16 units of the GRU layer process this input to extract features for subsequent layers, followed by a CNN layer that incorporates the output generated by the GRU. A 1×1 kernel window size and 16 filters are utilized in the CNN layer. CNN is particularly designed for image classification tasks, where the network accepts multidimensional data. However, CNN is also used for time series analysis, which uses one-dimensional data. In the CNN layer, the kernel traverses the input data with a defined stride and calculates convolution for each position. After calculations, new vector values are returned and passed to further layers in the network. The objective of this process is to extract features.

Another important layer is global average pooling (GAP). It is usually present right after convolutional layers in the network's architecture. The GAP does away with fully connected layers by averaging the embedding feature maps. This layer prevents overfitting since GAP does not have any parameters to optimize. Therefore, this layer replaces fully connected blocks of the neural network. Following that, there is a dense layer with 32 neurons, and a dropout layer with a rate of 0.1 is utilized to decrease the complexity of SGCNN. Applying the dropout technique is one of the easiest strategies for preventing overfitting in the model. Basically, it randomly excludes a certain number of nodes, usually in a percentage, to a given layer. When nodes are excluded, the model's optimization process does not consider their calculated weights. By using this method, the possibility of the network experiencing overfitting is significantly reduced, which improves classification performance. Finally, this architecture uses a softmax activation function for the final layer.

4.2.5 Choosing alphabet and window size

Selecting the appropriate alphabet and window size is a vital step of this approach. In our previous study [P2], we observed that this proposed algorithm consumed a substantial amount of computational time because we explored a wide range of these hyperparameters, namely the alphabet size and word length. Therefore, the primary focus in this subsection is to identify the most effective hyperparameters in order to achieve better results and reduce the runtime significantly. This preliminary experiment aims to determine the appropriate alphabet and window size for SAX-SGCNN before integrating it into the MuRBE structure. This experiment was conducted on 40 UCR/UEA time series classification datasets, described in Table 4.

We involved an extensive range of hyperparameters, such as alphabet size $a = \{2, 4, 7, 9, 11, 12, 13, 15, 17, 18, 19, 20\}$ and words of length $w = \{2, 3, 5, 7, 8\}$. We performed experiments using the original training and testing splits. We constructed the model using an Adam optimizer with a loss function of a sparse categorical cross-entropy. It consisted of 100 epochs, with a 0.001 learning rate and a 16-batch size.

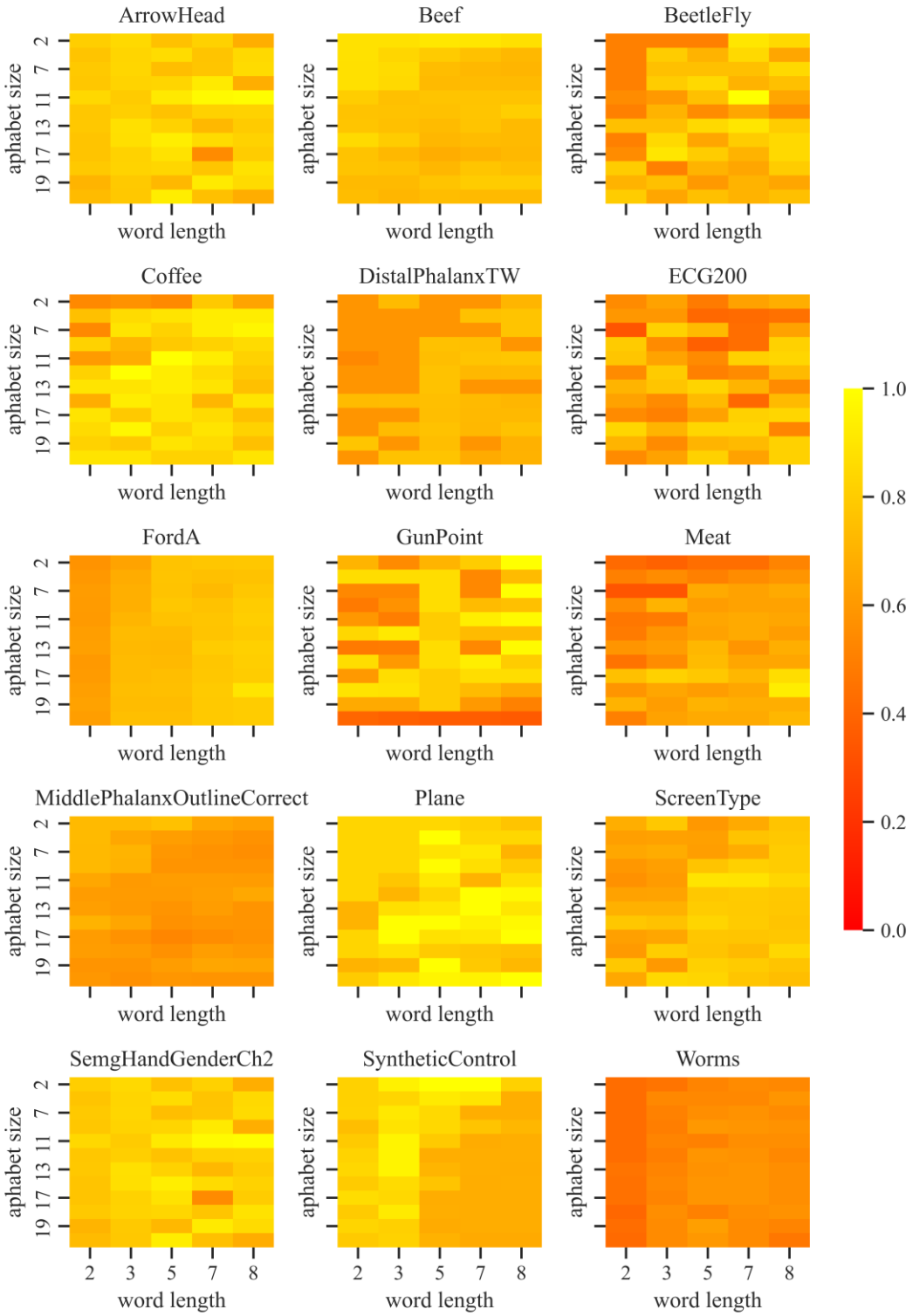


Figure 30. Test accuracy for selected datasets based on alphabet size and word length.

Heatmap plots for a few chosen datasets are displayed in Figure 30, allowing us to examine the effects of alphabet size and word length on test accuracy. The figures illustrate the variations in accuracy that occur when we alter the word length w and alphabet size a.

To achieve optimal outcomes, a bigger number of alphabets and longer word lengths are required for datasets such as *ArrowHead*, *BeetleFly*, *Coffee*, *DistalPhalanxTW*, *ECG200*, *FordA*, *Meat*, *Plane*, *ScreenType* and *Worms*. However, datasets such as *Beef*, *GunPoint*, *MiddlePhalanxOutlineCorrect*, *SemgHandGenderCh2*, and *SyntheticControl* need a lower number of the alphabet but greater word lengths. Additionally, it is noteworthy that optimal outcomes for various datasets are frequently attained while utilizing alphabet sizes of 2 and 18 and word lengths of 7 and 8. Therefore, we will set $a = \{2, 18\}$ and $w = \{7, 8\}$, as default settings of SAX-SGCNN in MuRBE structure.

4.3 DrCIF

The diverse representation canonical interval forest (DrCIF) is the expansion of interval-based CIF [49]. Since its introduction in 2021 as a new component of HC2, DrCIF represents a top interval-based ensemble classifier, demonstrating its effectiveness and capability to achieve high classification performance in various applications [2], [21]. Interval-based methods fundamentally rely on the extraction of phase-dependent subseries. The primary objective is to identify discriminatory characteristics across shifting intervals. DrCIF uses a time series tree as a base classifier, similar to the one used in TSF [43] that relies on information gain. DrCIF involves three series representations: the original time series, first-order difference (similarly employed by STSF [46]), and periodograms (also utilized by RISE [45] and STSF [46]). Multiple intervals are chosen randomly. From those three series representations, a set of seven basic summary statistics is calculated for each interval: mean, standard deviation, slope, median, interquartile range, minimum, and maximum. DrCIF enhances this by adding the catch22 features [30], creating 29 potential features, denoted as a. Assuming from three series representations, k phase-dependent intervals with random positions and lengths are selected. The a features are then computed for each interval. Subsequently, these features are concatenated into a vector with length of $3 \cdot a \cdot k$ for each series. The new features are obtained and then utilized to construct the tree. Table 1 provides the detailed pseudo algorithm for the DrCIF.

Table 1. Pseudo algorithm of DrCIF obtained from [2]

DrCIF (A list of N samples of length T, $\boldsymbol{S} = (\boldsymbol{X}, \boldsymbol{y})$, noted that dimensions $D = 1$ in univariate cases)

Parameters: the number of trees (r), the number of extracted features (a), and the number of intervals for each representation (k), (default: $r = 500, k = 4 + \sqrt{DrT}/3$, *and* $a = 10$)

Let $\boldsymbol{F} = (\boldsymbol{F}_1, \dots, \boldsymbol{F}_r)$ represent the trees
Let $\boldsymbol{V}$ represent a 3 by N by D list of series, containing three series representations
for $i \leftarrow 1$ to r **do**
    Let $\boldsymbol{S}$ represent a empty list of N samples $(s_1, \dots, s_N)$ with $a \cdot k$ features
    Let $\boldsymbol{U}$ represent a list of a randomly selected feature indices $(u_1, \dots, u_a)$
    for $y \leftarrow 1$ to 3 **do**
        for $j \leftarrow 1$ to k **do**
            $b \leftarrow \text{rand}(1, |\boldsymbol{V}_y| - 3)$
            $l \leftarrow \text{rand}(3, |\boldsymbol{V}_y|/2)$
            $o \leftarrow \text{rand}(1, D)$
            for $t \leftarrow 1$ to N **do**
                for $c \leftarrow 1$ to a **do**
                    $s_{t,a(j-1)+c} \leftarrow \text{summaryStat}(u_c, \boldsymbol{V}_y, t, o, b, l)$
                end for
            end for
        end for
    end for
    $\boldsymbol{F}_i$.buildTimeSeriesTree$(\boldsymbol{S}, \boldsymbol{y})$
end for

4.4 RDST

The random dilated shapelet transform (RDST) [8] has been recognized as a highly effective and leading shapelet-based classifier, particularly noted for its adaptability and performance in time series classification tasks [21]. RDST is considered an extension of STC that aims to improve the scalability and flexibility of the shapelet transform by incorporating dilation and randomization to avoid overfitting and computational complexity. Instead of searching for optimal shapelets from the training samples, RDST uses a unique approach by randomly selecting many shapelets from the training samples, generally on an order of thousands to ten thousand, and trains a linear Ridge classifier on features derived from shapelets. Furthermore, RDST utilizes two additional features alongside the minimum distance: it includes the location of the minimum distance and the frequency of shapelet occurrences based on a predetermined threshold λ.

For simplicity, given $\boldsymbol{X} = \langle X_1, \dots, X_N \rangle$ a set of time series with lengths of T and $\boldsymbol{y} = \langle y_1, \cdots, y_N \rangle$ their respective classes, RDST will take four input parameters: n_{shp} is the number of shapelets created, L is a list of candidate for the shapelet lengths, P_{norm} is the probability of shapelets which can be standardized using z-normalization, and $(P_{min}, P_{max}) \in [0,100]$ are the confident interval values that define the percentile bounds for threshold sampling λ. To initialize shapelets (*initializeShapelets*) is described below:

- the shapelet length l will be randomly chosen from L,
- the dilation d is defined as $d = \lfloor 2^x \rfloor$ where $x \sim U\left(0, \log_2 \frac{T}{l}\right)$, in a similar approach to ROCKET [5],
- uniformly determine if the shapelets will apply a z-normalized distance, with a specified probability P_{norm}, i.e., $u \sim U(0,1) \leq P_{norm}$
- extracting the shapelet values $s = \langle s_1, \cdots, s_l \rangle$ from a series $X_i = \langle x_1, \cdots, x_t, \cdots, x_T \rangle$, which is uniformly drawn from $\boldsymbol{X}$, and allowing start point $t \in [1, T - (l - 1) \times d]$ in which t is uniformly randomly selected.
- lastly, to determine the value of λ, we again select a different series X_i from the same group as the one utilized for obtaining the shapelet values, then calculate distance vector $d(s, x)$, and uniformly draw a value λ within the percentile bounds (P_{min}, P_{max}) of the distance vector.

Once n_{shp} shapelets are initialized, we proceed to calculate the distance vector for every pair of time series and shapelets. The result of RDST is a feature matrix with a size of $\left(N, 3 \cdot n_{shp}\right)$, which includes three features derived from the distance vector $d(s, x)$, namely the minimum distance (*min*), the location of the minimum distance (*argmin*), and the frequency of occurrences of the shapelet (SO). The occurrences of the shapelet can be defined as $SO = \sum_{t}^{T-(l-1)\times d} I(d(s,x) \leq \lambda)$, with $I(True) = 1$ and $I(False) = 0$. Finally, RDST is built based on a Ridge classifier. The pseudo algorithm for the RDST method is given in Table 2.

Table 2. Pseudo algorithm of RDST obtained from [8]

RDST (A list of N samples of length T, $\boldsymbol{S} = (\boldsymbol{X}, \boldsymbol{y})$)

Parameters: the number of shapelets (n_{shp}), a list of candidate for the shapelet lengths (L), the proportion/probability of shapelets (P_{norm}), lower percentile (P_{min}), upper percentile (P_{max}), (default: $n_{shp} = 1000, L = min\left(max\left(2, \frac{T}{2}\right), 11\right), P_{norm} = 0.8$, $P_{min} = 5^{th} percentile$, *and* $P_{max} = 10^{th} percentile$)

$\boldsymbol{S} \leftarrow initializeShapelets(\boldsymbol{X}, \boldsymbol{y}, L, P_{norm}, P_{min}, P_{max})$
$\boldsymbol{M} \leftarrow emptyArray\left(N, 3 \cdot n_{shp}\right)$
for $i \leftarrow 1$ to n_{shp} **do**
    for $j \leftarrow 1$ to N **do**
        $min, argmin, SO \leftarrow extractFeatures\left(X_j, S_i\right)$
        $M_{j,(i\times 3)} \leftarrow min$
        $M_{j,(i\times 3)+1} \leftarrow argmin$
        $M_{j,(i\times 3)+2} \leftarrow SO$
    end for
end for
$BuildRidgeClassifier(\boldsymbol{M}, \boldsymbol{y})$

Chapter 5

Experiments

5.1 Experimental Setup

We conducted an empirical experiment on 40 benchmark datasets from the UCR/UEA time series classification archive [12]. Each UCR/UEA dataset is already in a train and test split, which we use unchanged to make our results comparable to previous publications. We describe the datasets in Subchapter 5.2 Datasets. To maintain consistency with the existing literature, we used the default parameter configuration settings described in their references for the base classifiers in MuRBE, as described in Table 3. For the state-of-the-art methods described as competitors in the study, we used implementations available at https://www.aeon-toolkit.org/ or published by [21], in which the default settings for the parameter configuration were applied.

The methods are evaluated based on accuracy, balanced accuracy, and F1 score. These approaches enable a thorough evaluation of a model's performance, considering its predictive capabilities, robustness to class imbalances, and the equilibrium between sensitivity and specificity. All reported values are predictions on the test split. Furthermore, we made comparisons by visualizing critical difference (CD) diagrams [69]. The CD displays the average ranks of each method over all datasets used and represents a horizontal bar that indicates cliques, meaning statistically insignificant differences between methods in rankings. These cliques are calculated using a Wilcoxon-post-hoc method with Holm correction with a significance level of $\alpha = 5\%$.

Furthermore, multiple comparison matrix (MCM) is also used as a new evaluation tool proposed by [70]. The method is reportedly robust to maintain stability even if there are variations in classifier composition. In contrast to the CD diagram that relies on average ranks, the MCM utilizes average accuracy as its metric comparison. Furthermore, the MCM avoids using multiple test corrections, i.e., Bonferroni correction, in the context of p-value significance testing. The experiments were performed using Python 3.11.5 on a computer with an 8-core CPU, 16 GB of RAM, GeForce GTX 1660 Ti GPU, and the Windows 11 operating system. For reproducibility, we have made the source code publicly available through the link provided [2].

Table 3. Base classifier configurations in our experiments.

Base Classifiers	Configurations
ARFIMA-RF	max component $p, q = \sqrt{3T}$, component $d \in [0,2]$, 500 trees, $criterion = "gini"$ to measure the quality of a split.
SAX-SGCNN	alphabet size $a = \{2, 18\}$, word length $w = \{7, 8\}$, Adam optimizer, loss function of a sparse categorical cross entropy, 0.001 learning rate, 100 epochs,16-batch size, early stopping with three patience epochs.
DrCIF	500 trees, $4 + \sqrt{rT}/3$ intervals per representation for each tree, 10 attributes per tree.
RDST	1000 shapelets, $min\left(max\left(2, \frac{T}{2}\right), 11\right)$ possible lengths for the shapelets, 0.8 proportion of shapelets, 5^{th} of lower percentile, 10^{th} of upper percentile.

[2] github.com/rauzansumara/murbe-for-time-series-classification

5.2 Datasets

In our analysis, we utilized 40 out of the 112 datasets available from the UCR/UEA time series classification archive. To ensure a comprehensive evaluation, we selected these datasets proportionally based on the number of available datasets within each type. This approach allowed us to maintain a balanced representation across different categories, enhancing the robustness of our study.

For the *Device* datasets, we included three out of eight available options, explicitly focusing on *LargeKitchenAppliances*, *ScreenType*, and *SmallKitchenAppliances*. In the *ECG* category, we selected two of the six datasets, namely *ECG200* and *ECGFiveDays*. This selection process was designed to capture a variety of data types while ensuring that critical datasets were represented in our evaluation. In the *Image* dataset category, we chose 12 out of the 32 available datasets, including *ArrowHead*, *BeetleFly*, *BirdChicken*, and several others, such as *MedicalImages* and *ProximalPhalanxTW*. Additionally, we included 5 out of the 17 *Motion* datasets, selecting *CricketZ*, *GunPoint*, *InlineSkate*, *UwaveGestureLibraryX*, and *Worms*. Furthermore, from the *Sensor* datasets, we took 8 out of the 20 available, featuring datasets like *Earthquakes*, *FordA*, and *Lightning7*.

Lastly, we included three out of the eight *Simulated* datasets, specifically *ShapeletSim*, *SyntheticControl*, and *TwoPatterns*, along with five out of the eight *Spectro* datasets, including *Beef* and *OliveOil*. We also selected two out of the four *Spectrum* datasets, which were *SemgHandGenderCh2* and *SemgHandMovementCh2*. However, we decided to exclude five types of datasets from our analysis due to their limited numbers, such as *EOG* and *EPG*, which only contained two datasets each, *Hemodynamics* with three datasets, and *Power* and *Traffic*, which had only one dataset each.

The datasets also have diverse properties, such as different numbers of classes, lengths of series, numbers of samples in training and testing sets. For example, both the *BeetleFly* and *ShapeletSim* datasets have 20 samples in the training set but contain 20 and 180 samples in the testing set, respectively. In contrast, the *ItalyPowerDemand* dataset contains 1029 samples in the testing set and only 67 samples in the training set. A detailed description of all datasets can be seen in Table 4.

Table 4. Descritive of datasets.

No	Type	Dataset	Number of classes	Length	Train set			Test set		
					n	min	max	n	min	max
1	*Image*	*ArrowHead*	3	251	36	-2.2571	2.5540	175	-2.5494	2.4870
2	*Spectro*	*Beef*	5	470	30	-3.2900	3.7205	30	-3.3858	3.1514
3	*Image*	*BeetleFly*	2	512	20	-2.5168	2.5055	20	-2.5146	2.4080
4	*Image*	*BirdChicken*	2	512	20	-2.8248	2.1244	20	-3.0959	2.4416
5	*Spectro*	*Coffee*	2	286	28	-2.0642	2.1771	28	-2.1153	2.1042
6	*Motion*	*CricketZ*	12	300	390	-4.7583	11.9243	390	-5.1253	12.7068
7	*Image*	*DistalPhalanxOutlineCorrect*	2	80	600	-2.1589	2.4457	276	-2.1799	2.4602
8	*Image*	*DistalPhalanxTW*	6	80	400	-1.9945	2.0584	139	-1.8965	1.9998
9	*Sensor*	*Earthquakes*	2	512	322	-0.8858	7.8634	139	-0.7302	7.7281
10	*ECG*	*ECG200*	2	96	100	-2.6172	4.1991	100	-3.0145	4.1476

Table 4. Descritive of datasets (continued).

No	Type	Dataset	Number of classes	Length	Train set			Test set		
					n	min	max	n	min	max
11	*ECG*	*ECGFiveDays*	2	136	23	-6.5112	5.4211	861	-7.1078	6.0328
12	*Image*	*FiftyWords*	50	270	450	-2.3543	5.0184	455	-2.5220	5.2813
13	*Sensor*	*FordA*	2	500	3601	-4.6177	5.0592	1320	-4.5565	4.3151
14	*Motion*	*GunPoint*	2	150	50	-2.3692	2.0534	150	-2.5000	2.3197
15	*Image*	*Herring*	2	512	64	-2.1856	2.1343	64	-2.2091	2.0742
16	*Motion*	*InlineSkate*	7	1882	100	-2.2635	4.3393	550	-2.5189	3.8272
17	*Sensor*	*InsectWingbeatSound*	11	256	220	-1.0819	6.4213	1980	-1.3050	6.5895
18	*Sensor*	*ItalyPowerDemand*	2	24	67	-1.9910	2.4248	1029	-2.3934	3.2939
19	*Device*	*LargeKitchenAppliances*	3	720	375	-1.5751	26.7955	375	-1.1066	25.7034
20	*Sensor*	*Lightning7*	7	319	70	-1.7812	17.4133	73	-1.7278	16.6406

Table 4. Descritive of datasets (continued).

No	Type	Dataset	Number of classes	Length	Train set			Test set		
					n	min	max	n	min	max
21	*Spectro*	*Meat*	3	448	60	-1.5425	3.3902	60	-1.4932	3.3993
22	*Image*	*MedicalImages*	10	99	381	-2.3919	7.2224	760	-2.8312	8.0339
23	*Image*	*MiddlePhalanxOutlineCorrect*	2	80	600	-1.6602	2.0673	291	-1.7195	1.8756
24	*Image*	*MiddlePhalanxTW*	6	80	399	-1.7195	1.9245	154	-1.5848	1.7123
25	*Sensor*	*MoteStrain*	2	84	20	-8.4093	2.4684	1252	-8.6380	8.5444
26	*Spectro*	*OliveOil*	4	570	30	-1.0011	3.7188	30	-1.0002	3.7317
27	*Sensor*	*Plane*	7	144	105	-2.1133	2.9112	105	-2.1154	2.9240
28	*Image*	*ProximalPhalanxOutlineCorrect*	2	80	600	-1.4834	1.9029	291	-1.4424	1.8237
29	*Image*	*ProximalPhalanxTW*	6	80	400	-1.4834	1.9029	205	-1.4691	1.8496
30	*Device*	*ScreenType*	3	720	375	-2.9218	26.7955	375	-7.8336	26.7955

Table 4. Descritive of datasets (continued).

No	Type	Dataset	Number of classes	Length	Train set			Test set		
					n	min	max	n	min	max
31	*Spectrum*	*SemgHandGenderCh2*	2	1500	300	0.0052	2238.5732	600	0.0121	1933.6200
32	*Spectrum*	*SemgHandMovementCh2*	6	1500	450	0.0052	2238.5732	450	0.0121	1933.6200
33	*Simulated*	*ShapeletSim*	2	500	20	-1.8113	1.8917	180	-1.8680	1.8737
34	*Device*	*SmallKitchenAppliances*	3	720	375	-5.0674	26.7951	375	-3.8745	26.7948
35	*Spectro*	*Strawberry*	2	235	613	-2.3281	3.6820	370	-2.1276	3.7227
36	*Simulated*	*SyntheticControl*	6	60	300	-2.4538	2.4120	300	-2.6191	2.6053
37	*Sensor*	*Trace*	4	275	100	-2.2211	3.9667	100	-2.3923	3.9373
38	*Simulated*	*TwoPatterns*	4	128	1000	-1.9393	1.9394	4000	-1.9332	1.9183
39	*Motion*	*UWaveGestureLibraryX*	8	315	896	-4.4383	4.4341	3582	-5.7091	6.5148
40	*Motion*	*Worms*	5	900	181	-4.3114	4.8591	77	-4.8873	4.1961

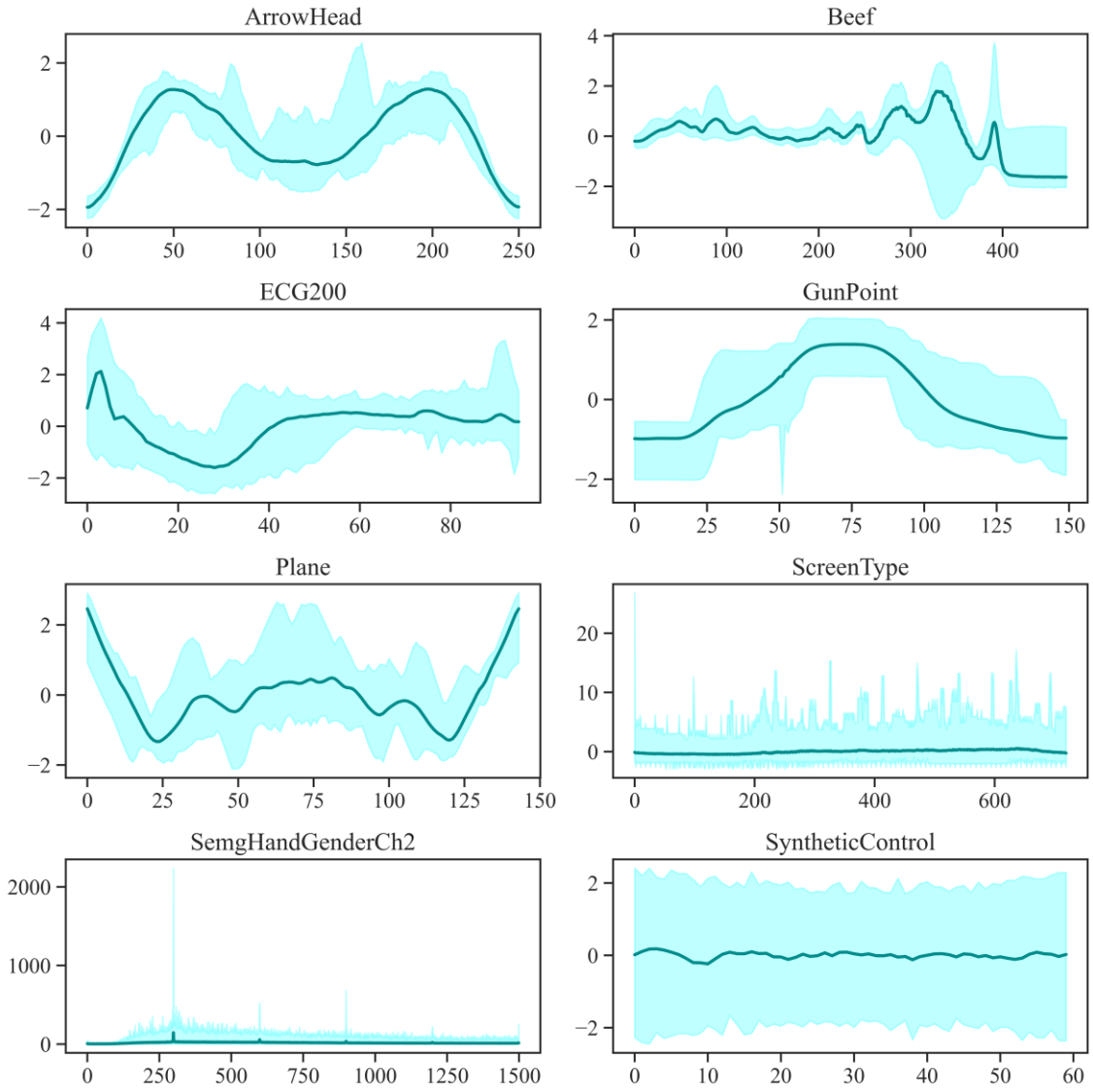


Figure 31. Each selected dataset showing a different range of time series, with the darker line representing the average time series.

The selected datasets cover different properties and characteristic features of time series. Figure 31 shows ribbon plots that display the range of values in the time series for the entire dataset without concentrating on specific classes. The darker line indicates the average values for each dataset. In addition, we can see that the *Beef* dataset (*Spectro* type) has a downward pattern. The *ScreenType* (*Device* type) and *SyntheticControl* (*Simulated* type) datasets have a stationery pattern. Moreover, the *ArrowHead* (*Sensor* type), *GunPoint* (*Motion* type), and *Plane* (*Sensor* type) datasets show seasonal patterns. The *SemgHandGenderCh2* dataset (*Spectrum* type) contains a time series with pulse and step characteristics. The *ECG200* dataset (*ECG* type) has shift in structure patterns. Additionally, we do a visualisation of each class for each class within selected datasets, see Figure 32.

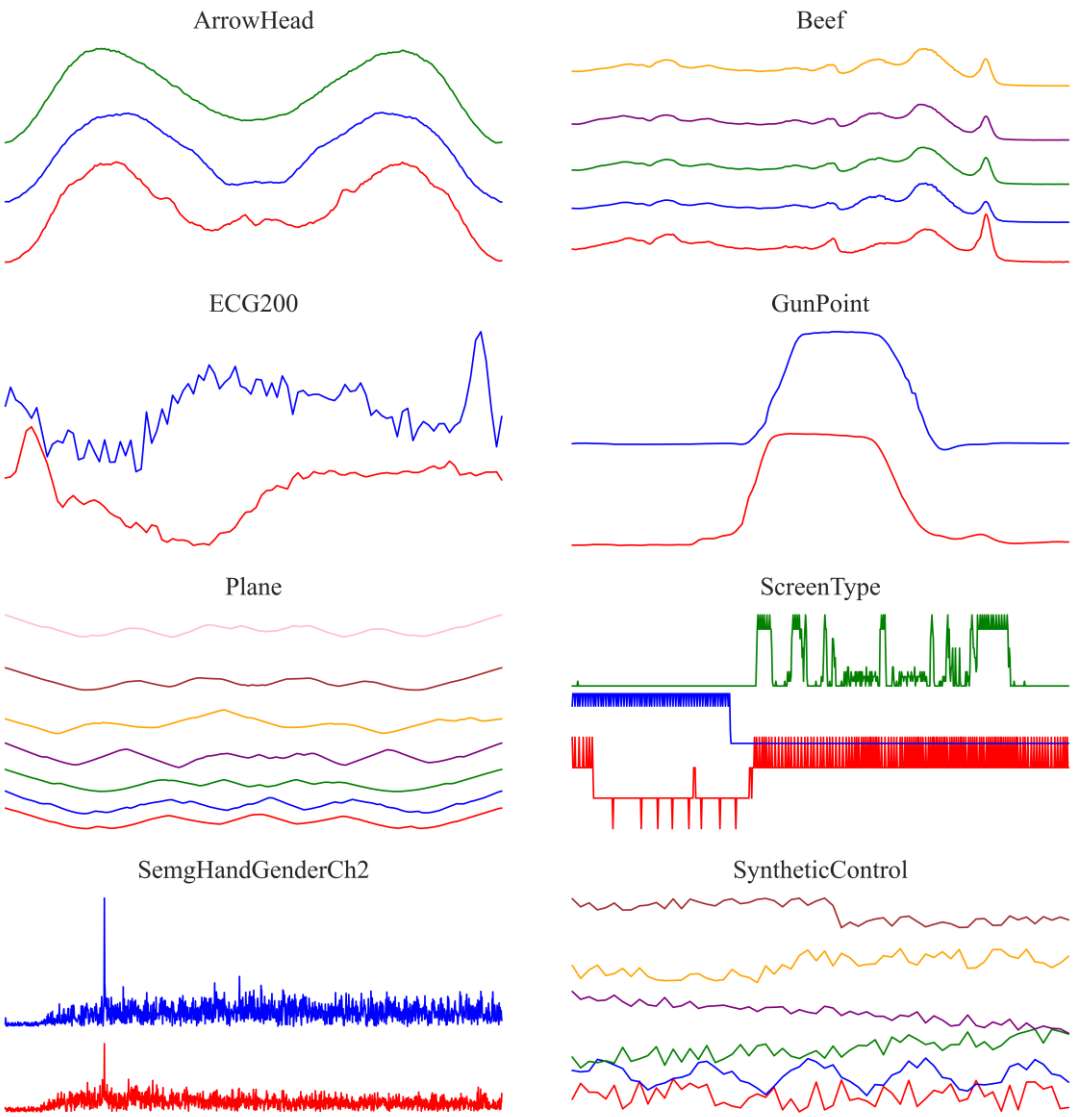


Figure 32. Selected datasets representing single example time series for each class.

Violin plots shown in Figure 33 enable us to identify both a median and distribution of series within each class of selected datasets since it provides a combination of a box plot and a kernel density plot. The median of the class is represented by a horizontal line in the middle, and the distribution of the series can be identified from the kernel density plot. By simply examining how series are distributed within each category, we can infer that certain datasets, for example, *GunPoint* and *SemgHandGenderCh2* datasets, seem easier to classify. In contrast, others, such as *ECG200*, *Plane*, and *SyntheticControl*, present more difficulties in classification.

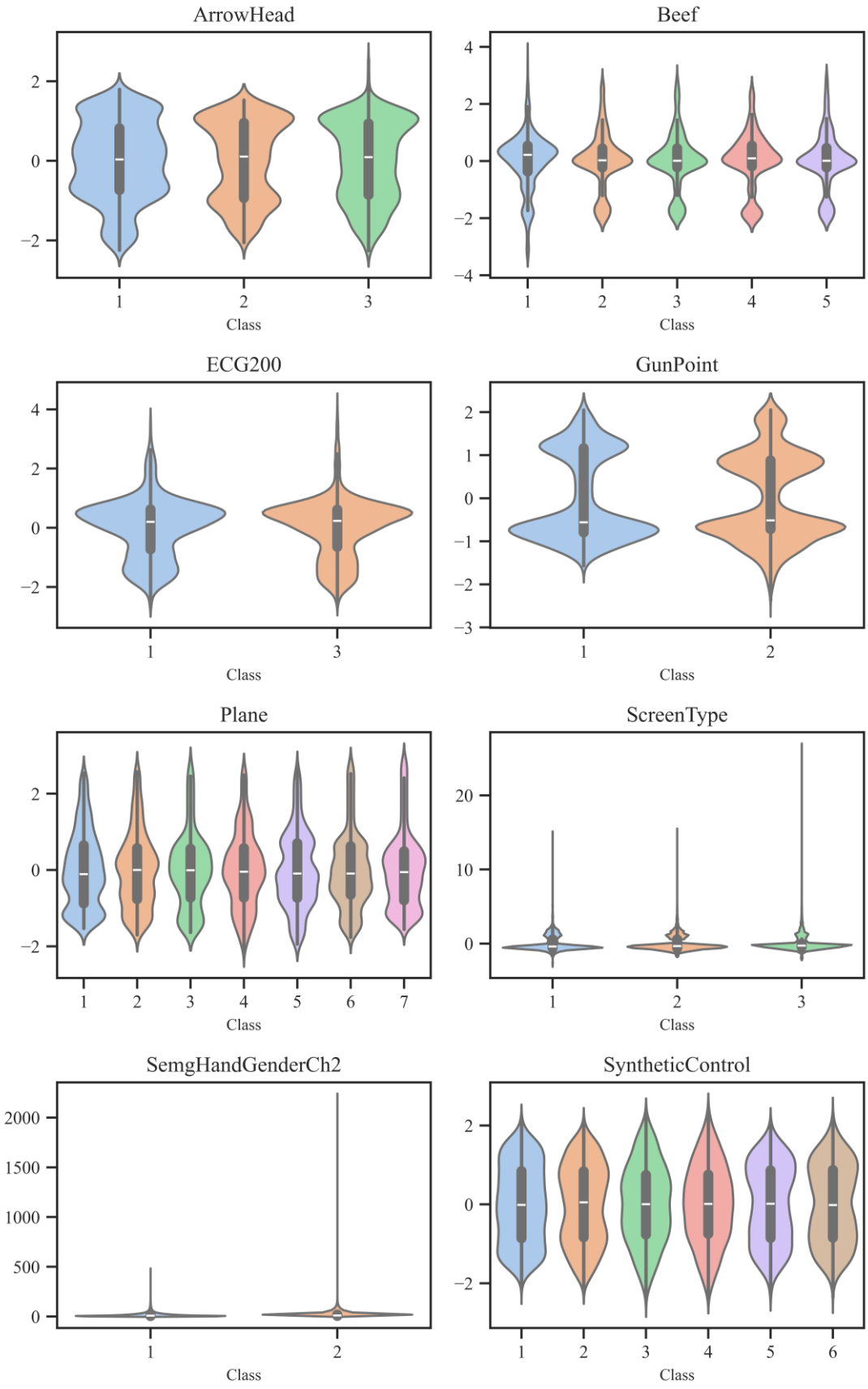


Figure 33. Violin plots of selected datasets showing variability of time series in each class.

5.3 Results

This Subchapter presents a detailed overview of the experimental results derived from this study. Firstly, we provide a comparison between MuRBE and its four base classifiers. Secondly, we also present the comparison between MuRBE and the other state-of-the-art methods in the literature. The performance metrics used in the comparison are accuracy, balanced accuracy, and F1 score. Thirdly, we assess the comparative performance using the CD diagrams and MCM. Finally, we also evaluate the scalability of MuRBE compared to the state-of-the-art approaches based on trade-offs between the average ranks, average accuracy, and total runtime.

5.3.1 Comparison with base classifiers

We provided a summary of the results from base classifiers and our proposed MuRBE in Table 5. As a result, the MuRBE shows a promising improvement over the base classifiers, demonstrating an approximate 1% increase in average accuracy. Average accuracies across the base classifiers ranged between 81.66% and 84.98%. Our core outcome is that the proposed method improves classification performance in terms of accuracy.

Table 5. Summary of test accuracy on 40 UCR/UEA datasets.

Accuracy	ARFIMA-RF	SAX-GCNN	DrCIF	RDST	MuRBE
Average	0.8448	0.8166	0.8498	0.8424	0.8572
Standard Deviation	0.1367	0.1479	0.1309	0.1402	0.1332

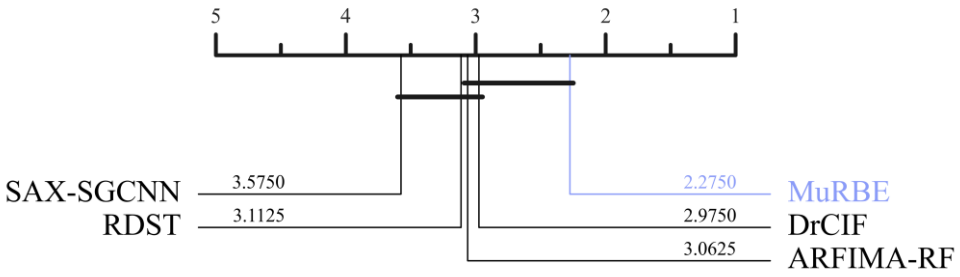


Figure 34. CD diagram of average ranks comparing the performance of MuRBE with base classifiers on test accuracy on 40 UCR/UEA datasets.

Figure 34 shows the CD diagram for MuRBE and its four components. The CD diagram displays that the MuRBE ranks one above all its components, demonstrating its superiority over four base classifiers based on average ranks. However, the performances of MuRBE, DrCIF, and ARFIMA-RF are in the clique (the solid horizontal bar links them), which indicates that there are no statistically significant differences. Additionally, SAX-SGCNN and RDST form the lowest clique.

In contrast, the MCM depicted in Figure 35 shows that MuRBE is significantly better than its four components based on average accuracy. MuRBE obtained an average accuracy of approximately 1% more accurate than the four base classifiers. It is important to note that CD diagrams can be misleading and unstable in certain cases, particularly as the relative ordering can be highly sensitive to the models selected for the comparisons. To prevent this problem, reference [70] proposes the MCM. Unlike CD diagrams, instead of relying on the average rank, the MCM employs average accuracy as its ordering metrics. Given that the average score would not change even in situations altering the compenent (added or removed) of classifiers, unlike the average rank. Moreover, the probability value (p-value) in the MCM is calculated through the Bayesian signed-rank test rather than the p-value produced by the Wilcoxon signed-rank test in the CD diagram. That is why we performed both the CD diagram and MCM to get a comprehensive evaluation conclusion.

The MCM illustrates pairwise comparisons among all algorithms, detailing differences in average accuracy, wins/ties/losses, and p-values. The heat map's colors illustrate the mean differences in accuracy. The red color signifies that the classifiers in the row generally outperform the ones in the column. Bold text is used to denote the significant differences.

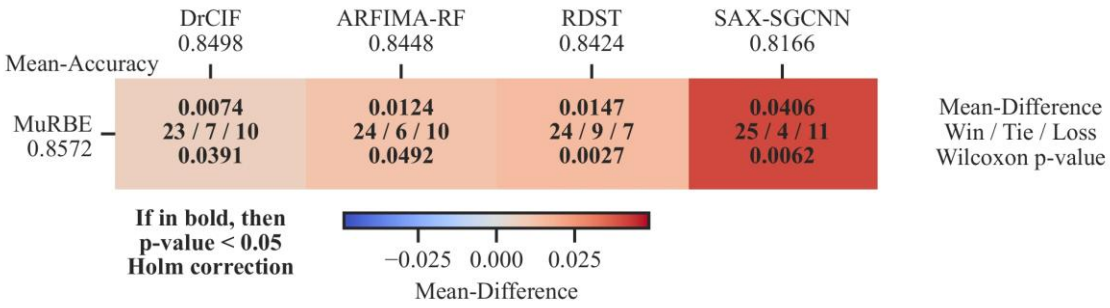


Figure 35. The MCM of average accuracy comparing the performance of MuRBE with base classifiers on the testing set on 40 UCR/UEA datasets.

5.3.1 Comparison with the state-of-the-art methods

To evaluate our proposed approach, we compare it against two distinguished groups of state-of-the-art methods. The first group is from non ensemble-based approaches, consisting of feature-based (FreshPRINCE, Catch22, and Signatures), dictionary-based (WEASEL-D, and TDE), interval-based (TSF, STSF, RISE, R-STSF, and QUANT) and shapelet-based methods (STC, and RSF). The second group is ensemble-based approaches, such as HC2, TS-CHIEF, InceptionTime, PF, ROCKET, Hydra, and Arsenal, which fall into the same group as the approach proposed in our study.

Figure 36 displays the CD diagram of MuRBE compared to all current state-of-the-art algorithms based on the average ranks on 40 UCR/UEA datasets. In the CD diagram, a lower average rank implies better accuracy of a particular method, and solid bars clique two or more methods for which there are no statistically significant differences. Compared to all algorithms, the proposed MuRBE is ranked within the top 5 classifiers, which is above QUANT, Arsenal, R-STSF, and ROCKET but below HC2. MuRBE is also not significantly different from HC2. This emphasizes the utility of fuzzy rank-based ensembles and proves the effectiveness of the favorable hierarchical structure. In order to get a comprehensive point of view, we provide CD diagrams and MCMs of our approach compared separately with two groups of approaches based on accuracy, balanced accuracy, and F1 score, depicted in Figure 38, 39, 40, and 41.

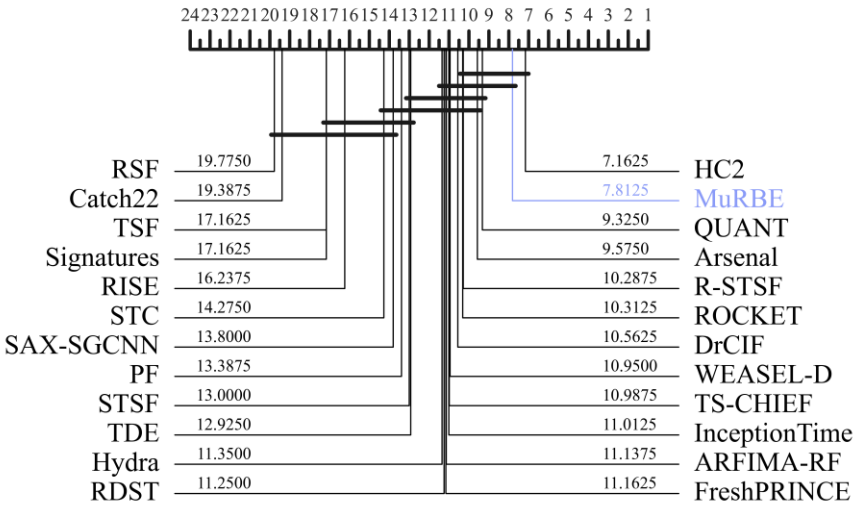


Figure 36. The CD diagram based on the average ranks of all compared state-of-the-art methods on test accuracy on 40 UCR/UEA datasets.

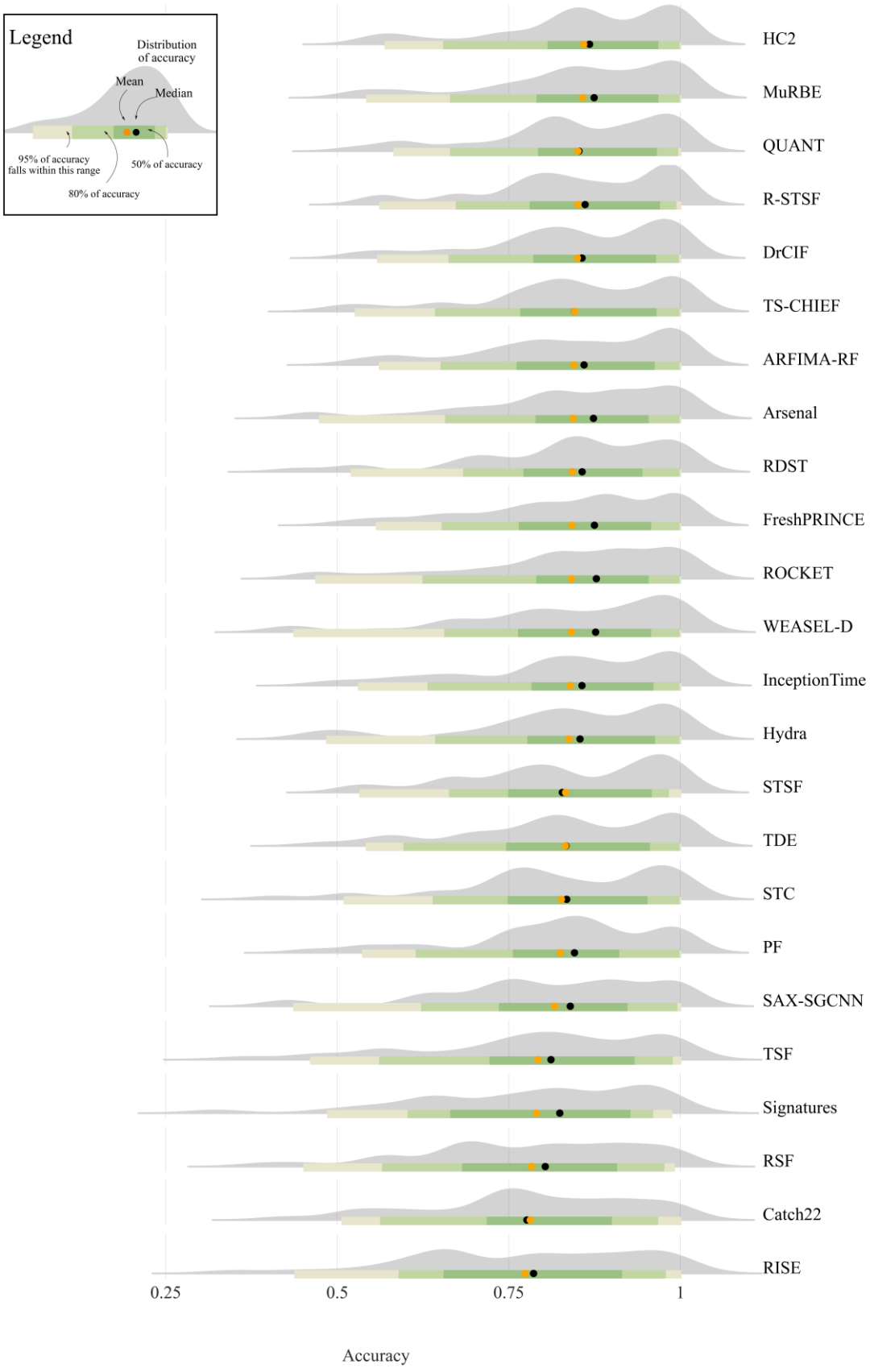


Figure 37. Custom ridgeline plot illustrating distribution and quantiles of each algorithm on test accuracy on 40 UCR/UEA datasets.

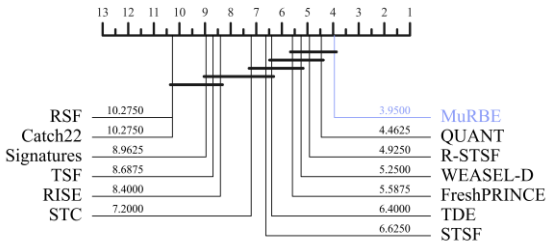
Figure 37 illustrates the distribution of test accuracy and emphasizes specific percentiles from various approaches. Each approach is represented by an annotated horizontal strip where the shape of the distribution indicates how the accuracy scores are spread across different datasets. The plot describes key metrics that help interpret the performance of each model. The mean accuracy is marked by a black dot, which measures the central tendency of the accuracy scores for each method. The median accuracy is represented by an orange dot, indicating the point where half of the scores are higher and half are lower. Additionally, the 95% range of accuracy scores is shaded in green, illustrating where 95% of the accuracy values fall, showing the general spread of the model's performance. A lighter green shading marks the 80% range, indicating the narrower range where the central 80% accuracy values lie.

The models are ranked based on their accuracy, from top to bottom. Each method has a unique shape of accuracy distribution. The spread of the distribution curves provides insight into the consistency of each model's performance. A narrower curve suggests less variation in accuracy, indicating that the model performs consistently across different test sets, while a wider curve suggests greater variability in performance. Models like HC2, MuRBE, QUANT, and R-STSF show relatively narrow distributions, indicating stable performance, while others like RISE, Signatures, and TSF show a wider spread, suggesting more variability in accuracy. This ridgeline plot offers a comprehensive view of each model's performance, stability, and reliability. It allows for an easy comparison of not only the overall accuracy but also the spread and consistency of test results across multiple models. This helps identify which models are more reliable and which may have more fluctuating performance.

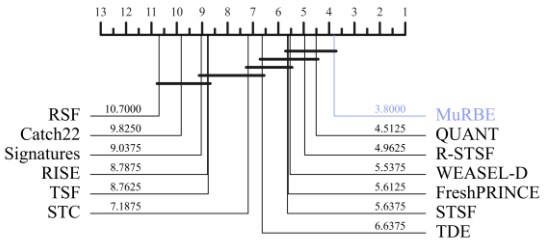
Figure 38 shows the CD diagrams on (a) accuracy, (b) balanced accuracy, and (c) F1 for MuRBE compared with the non ensemble-based algorithms on 40 UCR/UEA datasets. The ranking of MuRBE is on the top performing algorithms based on average ranks of accuracy, balanced accuracy, and F1 score. However, there are insignificant differences between MuRBE, QUANT, R-STSF, WEASEL-D, and FreshPRINCE. The STSF is also in a clique with ours as one of the top performing algorithms regarding balanced accuracy. However, in terms of accuracy, MuRBE is more accurate than QUANT on 20 datasets and less accurate on 13 datasets. MuRBE is also more accurate than R-STSF on 21 datasets and

less accurate on 15 datasets. Similarly, MuRBE is more accurate than FreshPRINCE on 23 datasets and less accurate on 10 datasets.

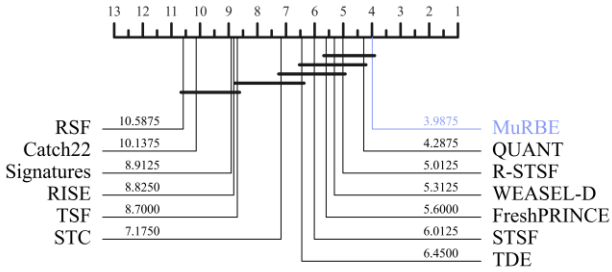
The MCM in Figure 39 summarises the performance of the classifiers based on average scores of (a) accuracy, (b) balanced accuracy, and (c) F1 score on 40 UCR/UEA datasets. A notable observation arises when comparing the CD diagrams in Figure 38 to this MCM. The ranking of MuRBE, QUANT, R-STSF, WEASEL-D, and FreshPRINCE is relatively aligned with results in the MCM. There are also statistically insignificant differences based on average values of all matrices, except WEASEL-D, which is statistically different from MuRBE in terms of accuracy. However, the ranking of Signatures, Catch22, RSF, and RISE algorithms seems different from the MCM. Despite RSF demonstrating a higher average score than RISE in the MCM, its ranking appears the lowest in the CD diagrams. This is because RISE achieved 38 wins while RSF only had 29 wins across 40 UCR/UEA datasets in terms of accuracy. We argue that using CD diagrams is better when we concentrate only on the number of wins or losses.



(a) Accuracy

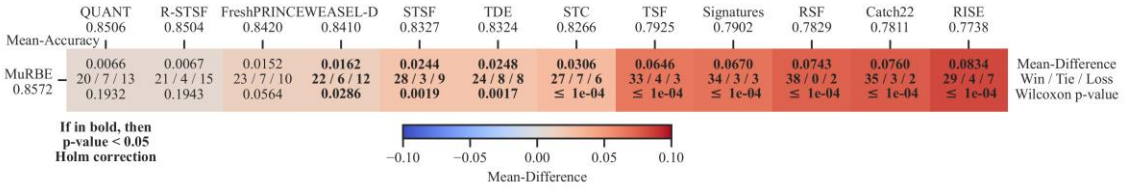


(b) Balanced Accuracy

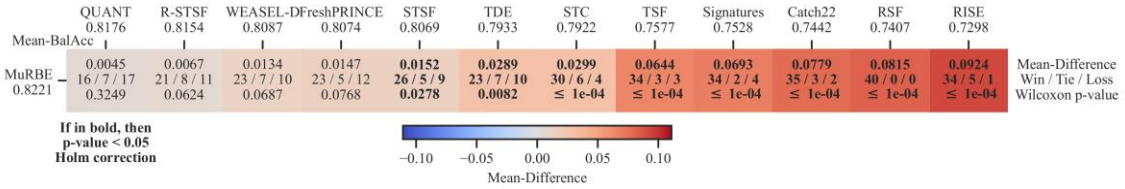


(c) F1 Score

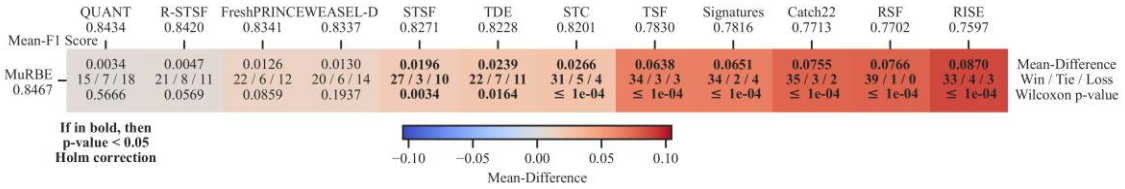
Figure 38. CD diagrams of average ranks of (a) accuracy, (b) balanced accuracy, and (c) F1 score of MuRBE in comparison with non ensemble-based approaches on the testing set on 40 UCR/UEA datasets.



(a) Accuracy



(b) Balanced Accuracy

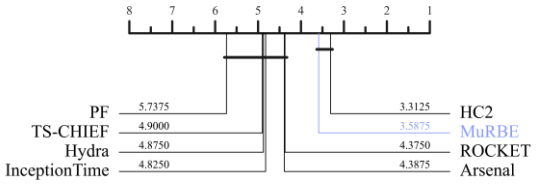


(c) F1 Score

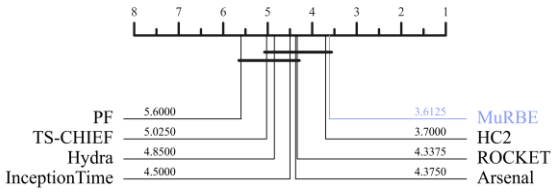
Figure 39. The MRM of average scores of MuRBE in comparison with non ensemble-based approaches on the testing set on 40 UCR/UEA datasets.

We also perform the CD diagrams for MuRBE compared with the ensemble-based algorithms on 40 UCR/UEA datasets in Figure 40. For accuracy, it demonstrates that MuRBE is in second ranking after HC2, and they also form the highest clique based on average ranks of all three metrics. MuRBE is significantly better than other state-of-the-art methods, except HC2. PF and TS-CHIEF form the lowest clique not only in accuracy but also in balanced accuracy and F1 score. In addition, ROCKET looks relatively stable in the third position in all metrics. For balanced accuracy, MuRBE achieves the first ranking, indicating a better sensitivity-specificity tradeoff. MuRBE also has more pairwise wins than HC2 on 17 datasets but is less accurate on 15 datasets. However, the average balanced accuracy of HC2 is still higher than MuRBE. Regarding accuracy, MuRBE outperforms PF, TS-CHIEF, and Hydra on 24 datasets. Compared to Arsenal, MuRBE shows more accurate on 19 datasets but less accurate on 11 datasets. In contrast to HC2, MuRBE is only more accurate on 13 datasets but less accurate on 16 datasets.

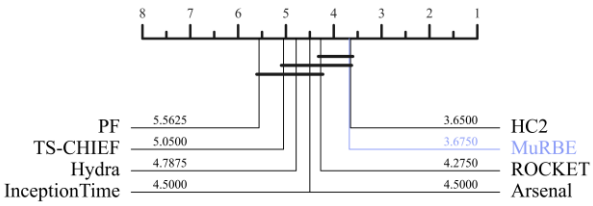
Again, comparing the CD diagrams in Figure 40 with the MCM in Figure 41 reveals an important finding. For accuracy, MuRBE shows significant differences compared to TS-CHIEF, ROCKET, Hydra, and PF in the MCM. However, the CD diagram shows not only them but also both Arsenal and InceptionTime, which are statistically significant differences against MuRBE. In the MCM, TS-CHIEF and PF consistently demonstrate significant differences compared to MuRBE for both balanced accuracy and F1 score. Nevertheless, only PF appears to be significantly different from MuRBE in the CD diagrams. The CD diagrams may seem misleading but they do provide meaningful insights. Despite TS-CHIEF demonstrating a higher average score than ROCKET in the MCM, its ranking correctly appears the second lowest in the CD diagrams due to fewer pairwise wins than ROCKET. It is important to know that the CD diagram utilizes average ranks to create an overall ranking of models across all datasets. The average rank shows how often a model wins or loses compared to others in a study. It does not take into account the degree of performance differences, which is why achieving a higher average accuracy does not necessarily lead to a higher rank. It only looks at wins or losses between pairs. Still, we believe the CD diagram is simple and easy to understand.



(a) Accuracy

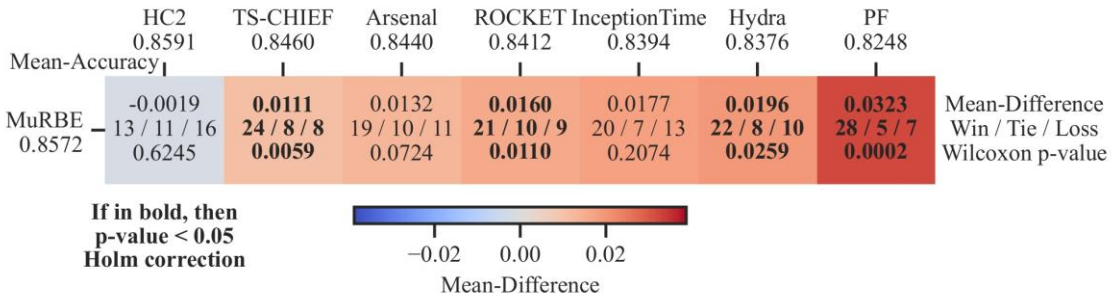


(b) Balanced Accuracy

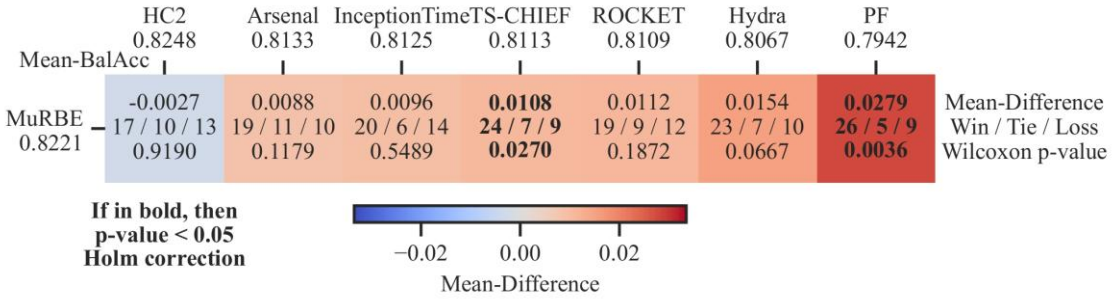


(c) F1 Score

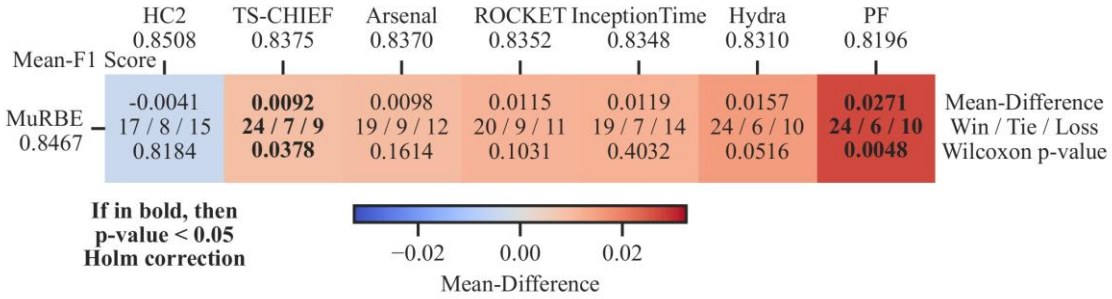
Figure 40. CD diagrams of average ranks: (a) accuracy, (b) balanced accuracy, and (c) F1 score of MuRBE in comparison with other ensemble-based approaches on the testing set on 40 UCR/UEA datasets.



(a) Accuracy



(b) Balanced Accuracy



(c) F1 Score

Figure 41. The MCM of average scores of MuRBE in comparison with other ensemble-based approaches on the testing set on 40 UCR/UEA datasets.

Boxplots displayed in Figure 42 illustrate how close the accuracy of MuRBE is compared to the current state-of-the-art methods. Top-ranking algorithms have a narrow interquartile range (IQR), few outliers, and average accuracy exceeding 80%. The boxplot also shows that HC2, InceptionTime, PF, ROCKET, Hydra, and Arsenal achieve medians that exceed the average scores, indicating a strong left skew. It signifies the presence of a few extremely low accuracy values, i.e., outliers, which stretch the boxplot distribution to the left. In contrast, TS-CHIEF and MuRBE indicate an absence of outliers in the distributions.

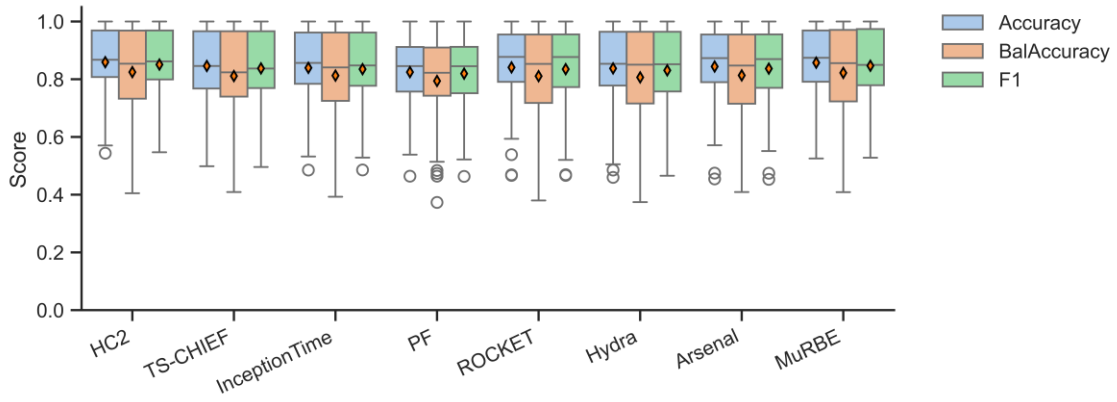
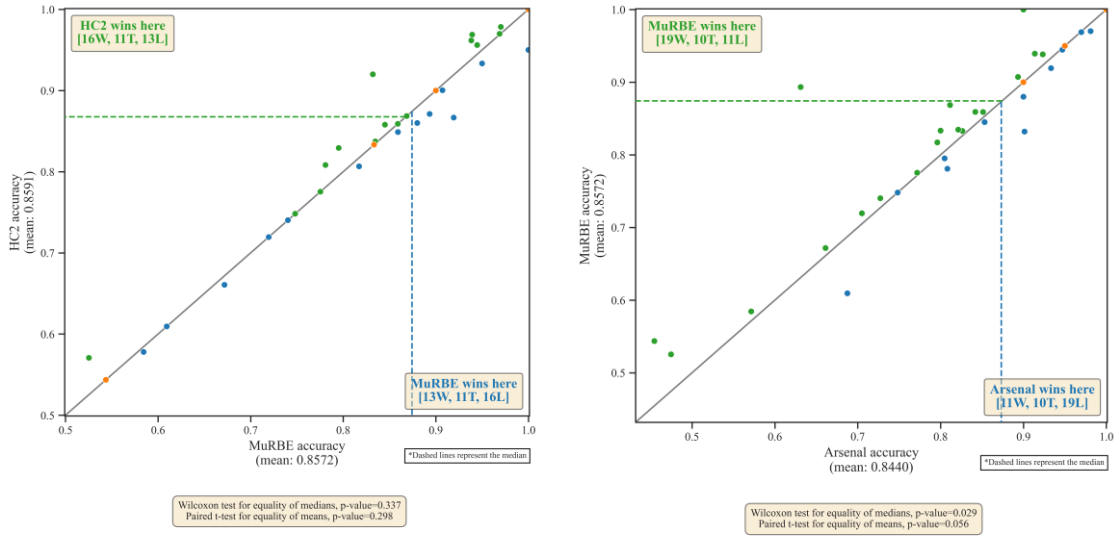


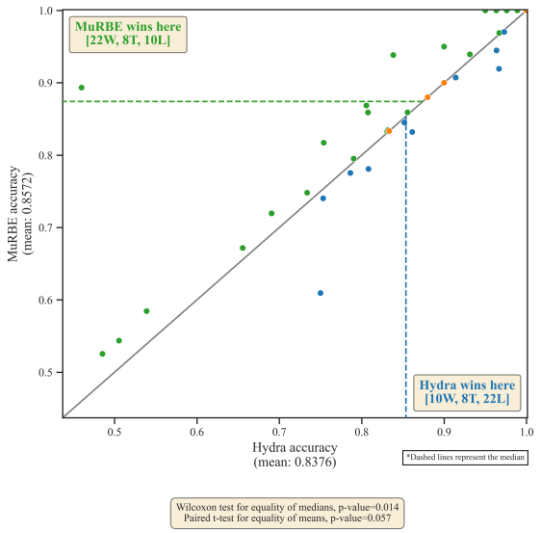
Figure 42. A custom box plot representing accuracy, balanced accuracy, and F1 score with average values (orange diamond) on the testing set on 40 UCR/UEA datasets.

5.3.1 Pairwise comparison with other ensemble-based approaches

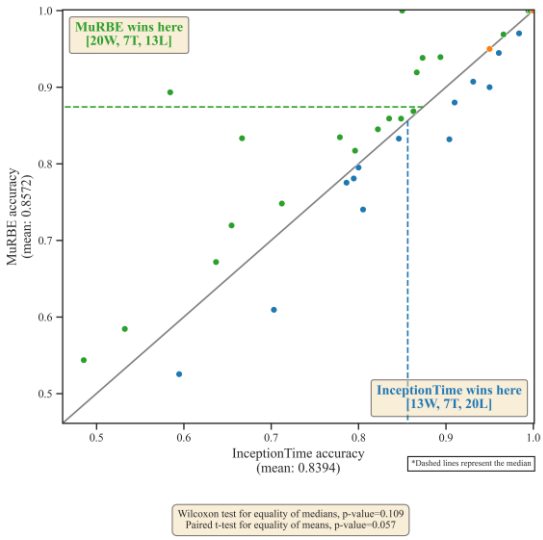
Figure 43 shows a pairwise comparison of MuRBE to the group of ensemble-based competitors. A point represents the test accuracy of each dataset. The points located above the diagonal line indicate that a particular method's scores are higher than those of its competitor. In other words, the method positioned along the vertical y-axis outperforms the one on the horizontal x-axis. Dashed lines represent the median of the compared methods. Overall, MuRBE performs relatively better than other competitors. Most points of MuRBE are above the diagonal compared to the other ensemble-based methods. Nevertheless, our approach only has 13 wins (blue) but 16 losses (green) with 11 ties (orange) compared to HC2.



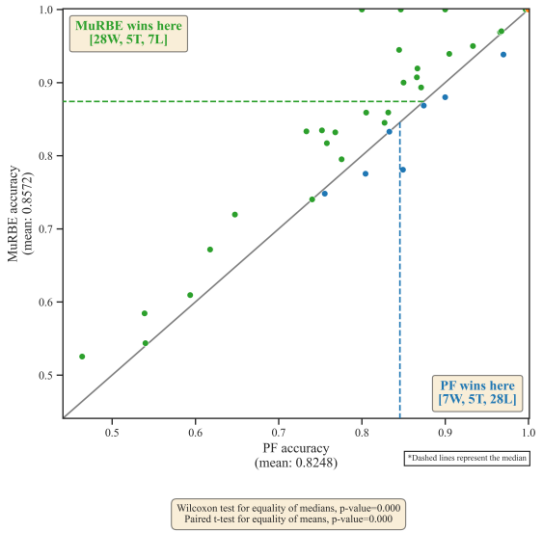
(a) (b)


MuRBE wins here
[22W, 8T, 10L]
Hydra wins here
[10W, 8T, 22L]
MuRBE accuracy
(mean: 0.8572)
Hydra accuracy
(mean: 0.8376)
*Dashed lines represent the median
Wilcoxon test for equality of medians, p-value=0.014
Paired t-test for equality of means, p-value=0.057

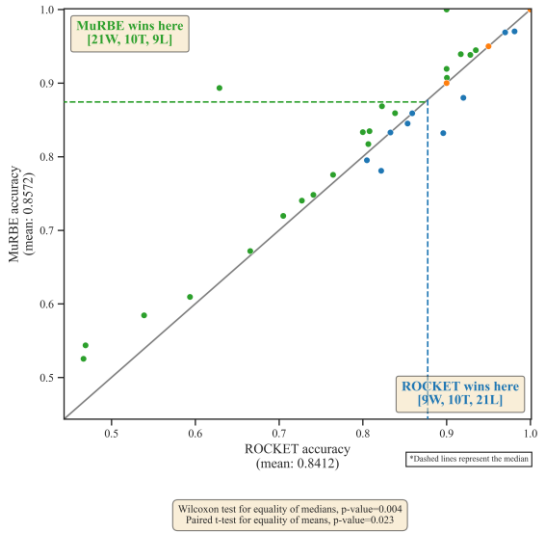
(c)


MuRBE wins here
[20W, 7T, 13L]
InceptionTime wins here
[13W, 7T, 20L]
MuRBE accuracy
(mean: 0.8572)
InceptionTime accuracy
(mean: 0.8394)
*Dashed lines represent the median
Wilcoxon test for equality of medians, p-value=0.109
Paired t-test for equality of means, p-value=0.057

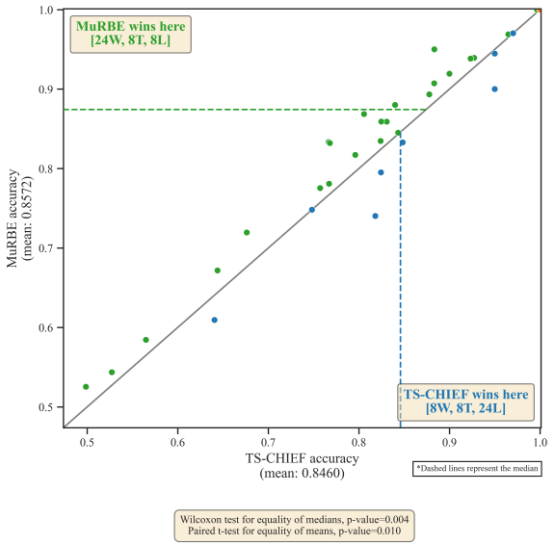
(d)


MuRBE wins here
[28W, 5T, 7L]
PF wins here
[7W, 5T, 28L]
MuRBE accuracy
(mean: 0.8572)
PF accuracy
(mean: 0.8248)
*Dashed lines represent the median
Wilcoxon test for equality of medians, p-value=0.000
Paired t-test for equality of means, p-value=0.000

(e)


MuRBE wins here
[21W, 10T, 9L]
ROCKET wins here
[9W, 10T, 21L]
MuRBE accuracy
(mean: 0.8572)
ROCKET accuracy
(mean: 0.8412)
*Dashed lines represent the median
Wilcoxon test for equality of medians, p-value=0.004
Paired t-test for equality of means, p-value=0.023

(f)


MuRBE wins here
[24W, 8T, 8L]
TS-CHIEF wins here
[8W, 8T, 24L]
MuRBE accuracy
(mean: 0.8572)
TS-CHIEF accuracy
(mean: 0.8460)
*Dashed lines represent the median
Wilcoxon test for equality of medians, p-value=0.004
Paired t-test for equality of means, p-value=0.010

(g)

Figure 43. Pairwise plot comparing the test accuracy of MuRBE with other ensemble-based approaches: (a) HC2, (b) Arsenal, (c) Hydra, (d) InceptionTime, (e) PF, (f) ROCKET, and (g) TS-CHIEF on 40 UCR/UEA datasets.

5.3.1 Comparison across datasets and domain applications

In this Subchapter, we inspect the performance of MuRBE across the types of application domains. We grouped datasets by the types to assess domain-dependent strengths or weaknesses. We treated the specified groups of the benchmark data types as defined by [12]. For this purpose, we only involve ensemble-based competitors such as HC2, TS-CHIEF, InceptionTime, PF, ROCKET, Hydra, and Arsenal. Figure 44 illustrates the accuracy of MuRBE (black dotted line) against the seven ensemble-based classifiers (orange area). We arranged them by ascending accuracy on each type domain. In general, our proposed MuRBE demonstrates highly competitive performance across nearly all datasets. The MuRBE has a relatively high percentage of wins in the ECG, Image, Simulated, and Spectro domains groups. Overall, MuRBE outperformed the others in 18 (green) out of 40 UCR/UEA datasets (tails included). Additionally, the proposed MuRBE obtained perfect accuracy in 9 cases, namely *ECGFiveDays*, *BeetleFly*, *GunPoint*, *Plane*, *Trace*, *ShapeletSim*, *SyntheticControl*, *TwoPatterns*, and *Coffee*.

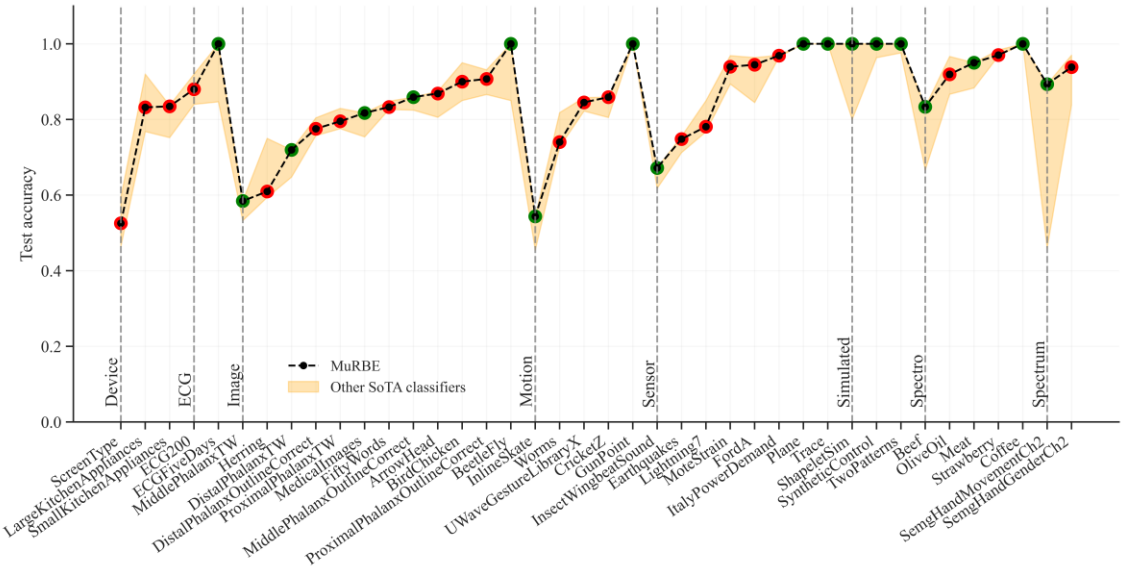


Figure 44. Test accuracy for MuRBE against the seven ensemble-based classifiers (Arsenal, HC2, Hydra, InceptionTime, PF, ROCKET, and TS-CHIEF) on 40 UCR/UEA datasets. The

orange area represents the seven classifiers' range of accuracies. Green dots indicate that MuRBE achieves the highest accuracy against the competitors, while red dots indicate the opposite.

5.3.2 Scalability

Although we believe accuracy is crucial, we also provide the runtime as another factor in assessing algorithm performance. Table 6 reviews the runtime for the classifiers evaluated in Figure 43. Several important caveats should be noted when interpreting these results. First, all algorithms, with the exception of InceptionTime, were executed on a single-thread CPU. Therefore, the runtime associated with InceptionTime is not directly comparable due to its operation on a GPU parallelization. Additionally, the maximum memory usage was not explicitly measured in this experiment, but it remained within the limit of the available memory on the CPU. It is important to know that the runtime would significantly decrease if the algorithms were threaded.

Figure 45 shows the normalized training, testing, and total time of each algorithm mentioned in Table 6. According to training time, MuRBE is approximately half order of magnitude faster than the majority of the evaluated competitors, such as InceptionTime, PF, and HC2. MuRBE is also approximately two orders of magnitude faster than TS-CHIEF, except for Hydra, ROCKET, and Arsenal. Hydra is considered the fastest ensemble-based approach, two orders of magnitude faster than ours, and not even comparable to the training time of ROCKET.

The majority of the datasets we used, most of the UCR/UEA datasets in general, had significantly more testing samples than training samples. For instance, *InsectWingbeatSound* dataset has only 256 samples in the training set, with 1980 samples in the testing set. Another example is *ItalyPowerDemand* dataset. It consists of 24 training samples with 1029 testing samples and several others such as *MoteStrain*, *ShapeletSim*, *InlineSkate ECGFiveDays,* etc. This has an impact on the performance of classifiers, such as ROCKET and Arsenal (since they are related), which are quick during training but may become slow during testing. ROCKET uses a massive number of kernels to convert the time series into a convolution-based representation. Each representation's computation is related not only to the number of kernels but also to the dataset's properties, i.e., the number of samples and time series length. For huge samples in datasets with extremely long series, ROCKET becomes computationally expensive. The disparity in sample sizes between train and test sets is

represented in ROCKET's testing time, which is half order of magnitude slower than InceptionTime.

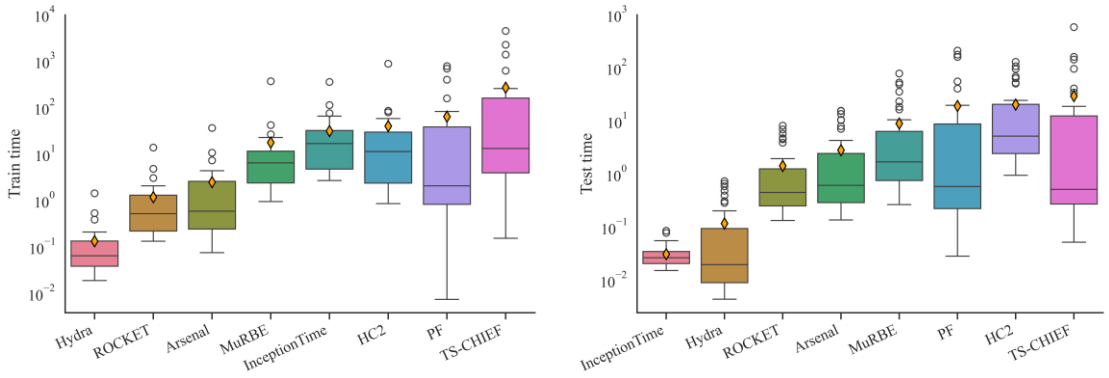
Table 6. Each method's total computational time (in minutes) on 40 UCR/UEA datasets.

Methods	Train	Test	Total
Hydra	5.3748	4.7380	10.1128
ROCKET	47.7818	57.6530	105.4349
Arsenal	100.4870	113.9386	214.4256
MuRBE	715.7258	361.3224	1077.0483
InceptionTime	1269.9711	1.2560	1271.2272
HC2	1618.8668	820.3320	2439.1988
PF	2587.7520	773.1391	3360.8911
TS-CHIEF	10842.5023	1187.6939	12030.1962

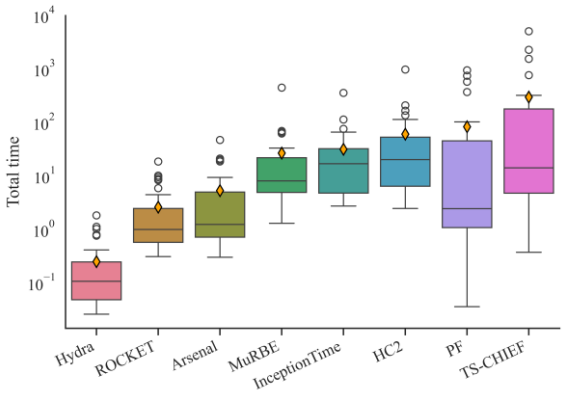
The PF, HC2, and TS-CHIEF are also expensive for prediction on the testing set. HC2 and PF are approximately two times slower than MuRBE when predicting the testing set on 40 UCR/UEA datasets. TS-CHIEF is three times slower than MuRBE at the testing set. In addition, InceptionTime is surprisingly faster than our proposed MuRBE, even comparable to Hydra in terms of testing time. However, InceptionTime's runtime benefits from GPU. It might be significantly slower without the GPU. According to [7], it is typically two times slower than HC2.

Table 6 illustrates the runtime of algorithms when executed sequentially. As mentioned earlier, MuRBE is slower than some current state-of-the-art methods, notably ROCKET and Hydra. If computational speed becomes really essential than a small accuracy gain, MuRBE may not be the optimal choice. MuRBE is simply not intended for rapid training within seconds; therefore, we advise against using it in situations where models must be trained extremely quickly. MuRBE is designed to maintain relatively competitive run times compared to HC2, PF, or even TS-CHIEF, especially in small to medium-sized problems. We assume that when time is a serious factor, the problem is likely to be extremely large.

For instance, *FordA* is considered a large dataset. MuRBE requires approximately 372 minutes for training, whereas Hydra and ROCKET need less than 2 and 15 minutes, respectively. The training time for HC2 is even worse, taking approximately 891 minutes to complete.



(a) Train time (b) Test time



(c) Total time (train + test)

Figure 45. Ascending average values (orange diamond) of computational time on (a) train, (b) test, and (c) total on 40 UCR/UEA datasets by method (in minutes). The time is on a log scale.

Furthermore, we provide the total runtime from different application domains, as shown in Figure 46. Most approaches require more time to process datasets from *Sensor*, *Motion*, and *Spectrum* domains. They are also relatively slow in processing *Device* and *Image* type domains. This is because most of the datasets from the domains contain large sample sizes in train and test sets, and it has a long time series. This significantly contributes to expensive computational runtime. In contrast, *ECG*, *Simulated*, and *Spectro* domains are quite fast to process. We can also observe that MuRBE is faster than HC2 in all types of application domains.

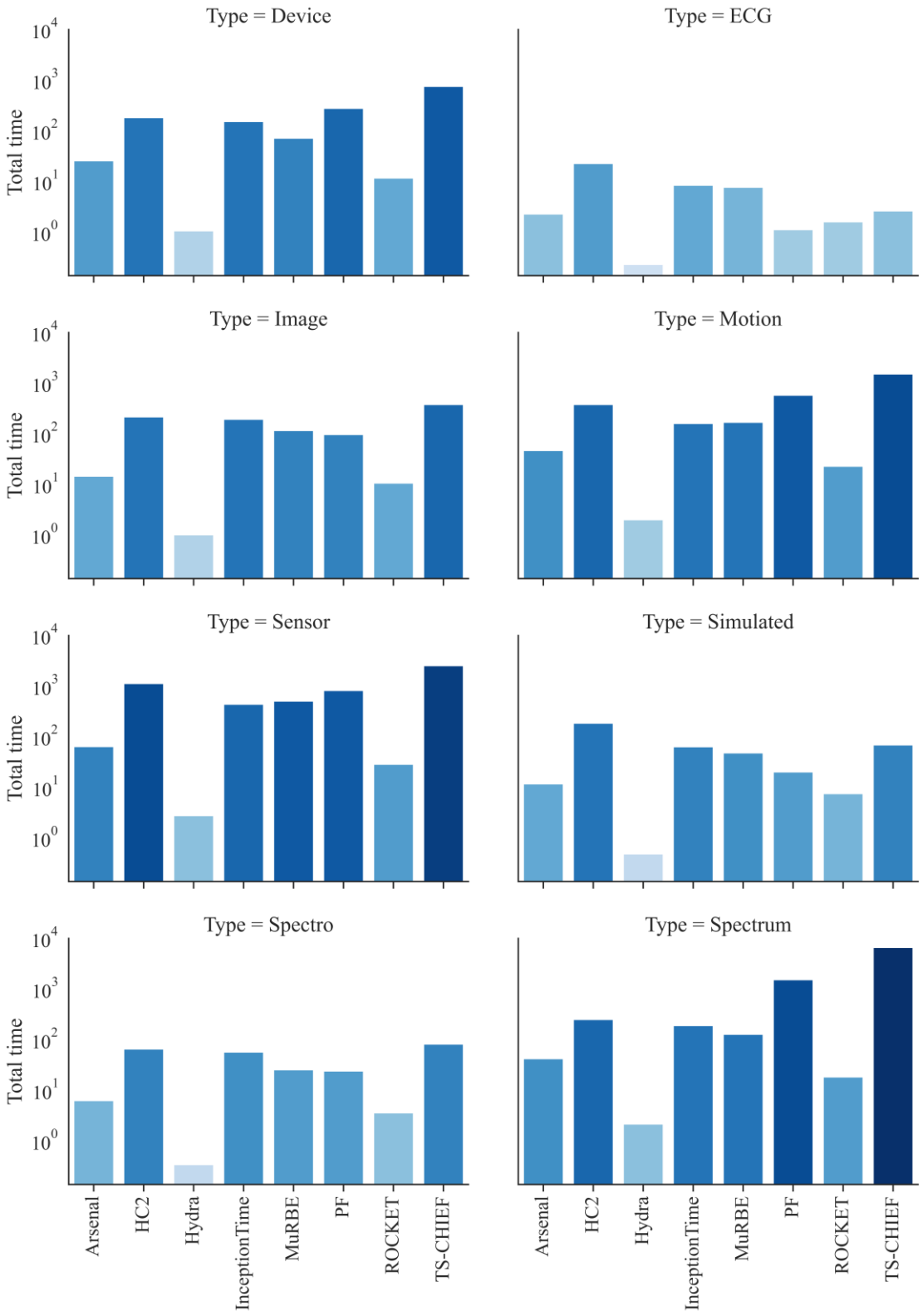


Figure 46. Pallete barplot of total computational time on 40 UCR/UEA datasets in each method per time series domain, such as Electric Devices, ECG, Image Outline, Motion Capture, Sensor Readings, Simulated, Spectrographs, and Spectrum. The time is on a log scale.

Figure 47 depicts the average accuracy rank against the runtime by the average accuracy for all compared algorithms, revealing a clear trade-off between performance and runtime. Considering these three factors, such as average ranks, average accuracy, and computational runtime, we can draw several conclusions. QUANT and Hydra are capable of training and predicting models for all 40 UCR/UEA datasets about 3 and 11 minutes, respectively, even without threading. If time efficiency is the primary concern, QUANT and Hydra would be an excellent starting point for the analysis. However, HC2 can be a worthy choice when prioritizing model accuracy over runtime. Alternatively, our proposed MuRBE is considered in between, which is half order of magnitude faster than HC2; it is neither highly computationally expensive nor very fast runtime, but it is considered moderate and useful when prioritizing both runtime and accuracy in small to medium cases.

On the other hand, TS-CHIEF and PF are significantly slow and appear to scale less well than the other algorithms. Additionally, we also see ARFIMA-RF and DrCIF as the slow components within MuRBE. Due to exploring really high dimensional interval space [47], DrCIF requires a long running time. In ARFIMA-RF, the configuration of the hyperparameter setting might be the issue, even though we have already used the fastest algorithm to automatically identify the best configurations of the ARFIMA introduced by [66]. These could be areas for future improvement.

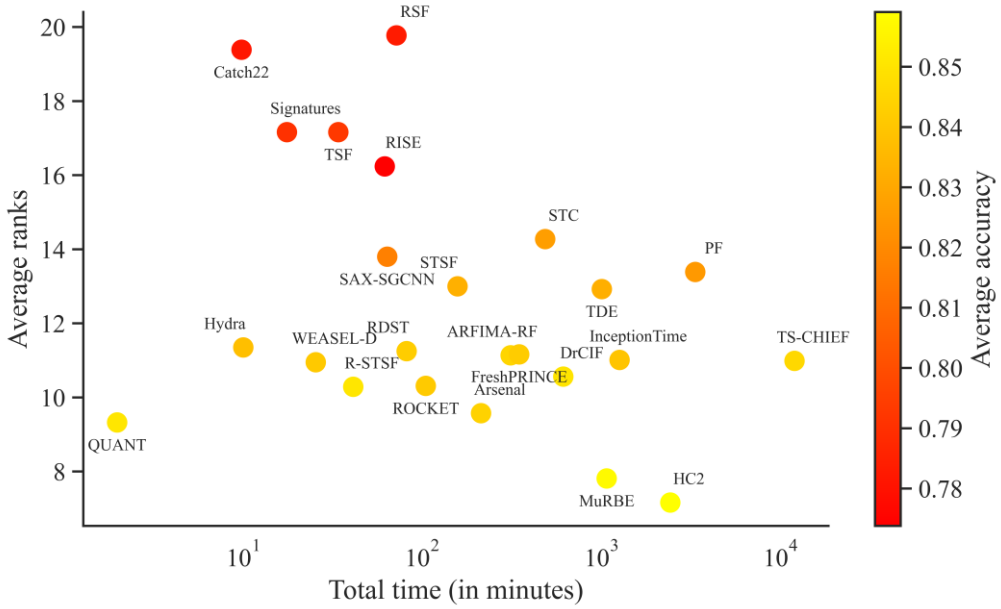


Figure 47. A comparison of classifiers regarding average ranks, average accuracy, and total time on 40 UCR/UEA datasets. The total time is on a log scale.

Chapter 6

Conclusions

This chapter wraps up the thesis by highlighting the summary of the experiments and addressing the research's limitations. It considers the important findings and their impact on the field of time series classification.

6.1 Summary

We have presented MuRBE as a novel heterogenous ensemble structure explicitly designed for time series classification. MuRBE is constructed from four different representation domains; each proposed to capture different discriminatory features. Through comprehensive experiments and analyses, the main conclusions of this study can be outlined as follows: First, our study showed that MuRBE is significantly better than its base component classifiers. We are convinced that every component plays a crucial role in enhancing the overall performance. Our core outcome is that the proposed method proves to improve predictive capabilities.

Second, among the seven ensemble-based algorithms, the proposed MuRBE outperforms them except for HC2. Notably, our approach secures the second position not only in average ranks but also in average accuracy among the current state-of-the-art techniques. Ours also forms the highest clique with HC2 based on the average rank of all three metrics. In terms of pairwise win/loss comparison, MuRBE shows more pairwise wins than PF, TS-CHIEF, Arsenal, and Hydra based on accuracy on 40 UCR/UEA datasets. MuRBE performs well in most *ECG*, *Image*, *Simulated,* and *Spectro* data types. We are

convinced that its main advantage can be attributed to the fact that discriminatory features are often present in multiple domains for many problems.

In comparison with twelve non ensemble-based algorithms, the ranking of MuRBE is one of the top-performing algorithms based on the average ranks of all three metrics. Not only on ranking, MuRBE also has a bigger average accuracy compared to the most current algorithms, like QUANT, R-STSF, WEASEL-D, and FreshPRINCE. Even though they are statistically insignificant differences. Last, in terms of scalability, our model proved to be scalable enough than some other ensemble-based algorithms since MuRBE is faster than HC2, InceptionTime, PF, and TS-CHIEF, based on total runtime, except for Hydra, ROCKET, and Arsenal. Hydra is considered the fastest ensemble-based approach and two orders of magnitude faster than ours.

6.2 Limitation

Despite the contributions presented in this thesis, some limitations should be addressed.

First, even though we have mentioned that our proposed method is somehow faster than some current state-of-the-art methods, we would still say a limitation lies in its computational demands. Our method is okay for small to medium-sized problems that involve tens to hundreds or about a thousand time series. However, training time can become excessively long for large-scale problems when involving tens of thousands of time series with tens of thousands in lengths or even more. The method might not be well scalable for such issues, but we believe there is potential for improvement. Future research could focus on enhancing individual components, particularly DrCIEF and ARFIMA, to enable more adaptive and intelligent reconfiguration settings for optimal runtime. Developing better strategies to address the issue will also be a key focus in future modifications. This could involve exploring more efficient algorithms to speed up the training process. Additionally, further investigation into the representation domains used by MuRBE could yield insights on how to leverage the discriminatory features present in the data better.

Second, in evaluating classification algorithms, it is common practice to include not only the standard evaluation metrics but also more nuanced measures, such as the area under the receiver operating characteristic curve (AUROC) and the negative log-likelihood (NLL). These two additional metrics are sometimes used to provide complementary insights into model performance: AUROC enables the assessment of a model's ability to rank positive

instances higher than negative ones across varying decision thresholds. At the same time, NLL offers a way to evaluate the quality of the predicted probability distributions through a loss function. Unlike metrics such as accuracy, balanced accuracy, or F1 score, which are directly computed from the confusion matrix, both AUROC and NLL rely on the probability estimates generated by the models rather than on discrete classification outcomes. These metrics capture different aspects of model ability, emphasizing ranking and probabilistic calibration beyond what is reflected in the confusion matrix.

The proposed ensemble structure used a fuzzy rank-based framework, which inherently cannot provide proper probability estimates directly. This limitation arises primarily because the final fuzzy scores produced by the combination of tanh and exp functions are not bounded within the $[0, 1]$ interval, unlike actual probability values that are strictly bounded in this range. Additionally, we argue that fuzzy sets, while sharing some similarities with probability, cannot be directly treated as probabilities, as they are fundamentally different mathematical constructs; however, they are not the same. Another issue lies in the decision rule of the ensemble, where the predicted class corresponds to the minimum final score, which contradicts the concept of probability, where the maximum probability estimate should be the predicted class. To address this discrepancy in the future, it is necessary to transform the final fuzzy rank scores into valid probability estimates via numerical inversion methods since no closed-form analytic inverse exists for the nonlinear function involved. Techniques such as the Newton-Raphson or bisection methods provide viable starting points for this inversion; alternatively, Brent's method offers guaranteed convergence even with increased computational time. However, including this inversion step would certainly raise the model's runtime, so it has been omitted from the current implementation.

References

[1] A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh, “The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances,” *Data Min Knowl Disc*, vol. 31, no. 3, pp. 606–660, May 2017, doi: 10.1007/s10618-016-0483-9.

[2] M. Middlehurst, J. Large, M. Flynn, J. Lines, A. Bostrom, and A. Bagnall, “HIVE-COTE 2.0: a new meta ensemble for time series classification,” *Mach Learn*, vol. 110, no. 11–12, pp. 3211–3243, Dec. 2021, doi: 10.1007/s10994-021-06057-9.

[3] A. Shifaz, C. Pelletier, F. Petitjean, and G. I. Webb, “TS-CHIEF: a scalable and accurate forest algorithm for time series classification,” *Data Min Knowl Disc*, vol. 34, no. 3, pp. 742–775, May 2020, doi: 10.1007/s10618-020-00679-8.

[4] A. Dempster, D. F. Schmidt, and G. I. Webb, “HYDRA: Competing convolutional kernels for fast and accurate time series classification,” Mar. 25, 2022, *arXiv*: arXiv:2203.13652. doi: 10.48550/arXiv.2203.13652.

[5] A. Dempster, F. Petitjean, and G. I. Webb, “ROCKET: exceptionally fast and accurate time series classification using random convolutional kernels,” *Data Min Knowl Disc*, vol. 34, no. 5, pp. 1454–1495, Sep. 2020, doi: 10.1007/s10618-020-00701-z.

[6] B. Lucas *et al.*, “Proximity Forest: an effective and scalable distance-based classifier for time series,” *Data Min Knowl Disc*, vol. 33, no. 3, pp. 607–635, May 2019, doi: 10.1007/s10618-019-00617-3.

[7] H. Ismail Fawaz *et al.*, “InceptionTime: Finding AlexNet for time series classification,” *Data Min Knowl Disc*, vol. 34, no. 6, pp. 1936–1962, Nov. 2020, doi: 10.1007/s10618-020-00710-y.

[8] A. Guillaume, C. Vrain, and W. Elloumi, “Random Dilated Shapelet Transform: A New Approach for Time Series Shapelets,” in *Pattern Recognition and Artificial Intelligence*, vol. 13363, M. El Yacoubi, E. Granger, P. C. Yuen, U. Pal, and N. Vincent, Eds., in Lecture Notes in Computer Science, vol. 13363. , Cham: Springer International Publishing, 2022, pp. 653–664. doi: 10.1007/978-3-031-09037-0_53.

[9] “Welcome to aeon,” aeon. Accessed: May 13, 2025. [Online]. Available: https://www.aeon-toolkit.org/en/v1.0.0/index.html

[10] B. D. Fulcher, “Feature-based time-series analysis,” Oct. 02, 2017, *arXiv*: arXiv:1709.08055. doi: 10.48550/arXiv.1709.08055.

[11] A. Bagnall *et al.*, “The UEA multivariate time series classification archive, 2018,” Oct. 31, 2018, *arXiv*: arXiv:1811.00075. doi: 10.48550/arXiv.1811.00075.

[12] “Time Series Classification Website.” Accessed: Jan. 02, 2025. [Online]. Available: https://www.timeseriesclassification.com/

[13] “Converting images into time series for data mining.” Accessed: Apr. 18, 2025. [Online]. Available: https://izbicki.me/blog/converting-images-into-time-series-for-data-mining.html

[14] A. Gupta, H. P. Gupta, B. Biswas, and T. Dutta, “Approaches and Applications of Early Classification of Time Series: A Review,” *IEEE Trans. Artif. Intell.*, vol. 1, no. 1, pp. 47–61, Aug. 2020, doi: 10.1109/TAI.2020.3027279.

[15] O. Al-Jowder, E. K. Kemsley, and R. H. Wilson, “Detection of Adulteration in Cooked Meat Products by Mid-Infrared Spectroscopy,” *J. Agric. Food Chem.*, vol. 50, no. 6, pp. 1325–1329, Mar. 2002, doi: 10.1021/jf0108967.

[16] A. Bagnall, L. Davis, J. Hills, and J. Lines, “Transformation Based Ensembles for Time Series Classification: SIAM International Conference on Data Mining,” May 2012, pp. 307–319.

[17] R. T. Olszewski, “Generalized feature extraction for structural pattern recognition in time-series data,” phd, Carnegie Mellon University, USA, 2001.

[18] H. A. Dau *et al.*, “The UCR time series archive,” *IEEE/CAA J. Autom. Sinica*, vol. 6, no. 6, pp. 1293–1305, Nov. 2019, doi: 10.1109/JAS.2019.1911747.

[19] C. Sapsanis, G. Georgoulas, A. Tzes, and D. Lymberopoulos, “Improving EMG based classification of basic hand movements using EMD,” in *2013 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, Jul. 2013, pp. 5754–5757. doi: 10.1109/EMBC.2013.6610858.

[20] R. Alcock, “Time-Series Similarity Queries Employing a Feature-Based Approach,” 1999. Accessed: Apr. 19, 2025. [Online]. Available: https://www.semanticscholar.org/paper/Time-Series-Similarity-Queries-Employing-a-Approach-Alcock/8f2efe63784f234a70cd8a1209a8436c2ef04089

[21] M. Middlehurst, P. Schäfer, and A. Bagnall, “Bake off redux: a review and experimental evaluation of recent time series classification algorithms,” *Data Min Knowl Disc*, vol. 38, no. 4, pp. 1958–2031, Jul. 2024, doi: 10.1007/s10618-024-01022-1.

[22] M. Christ, N. Braun, J. Neuffer, and A. W. Kempa-Liehr, “Time Series FeatuRe Extraction on basis of Scalable Hypothesis tests (tsfresh – A Python package),” *Neurocomputing*, vol. 307, pp. 72–77, Sep. 2018, doi: 10.1016/j.neucom.2018.03.067.

[23] M. G. Kendall, “A New Measure of Rank Correlation,” *Biometrika*, vol. 30, no. 1/2, p. 81, Jun. 1938, doi: 10.2307/2332226.

[24] F. J. Massey, “The Kolmogorov-Smirnov Test for Goodness of Fit,” *Journal of the American Statistical Association*, vol. 46, no. 253, pp. 68–78, Mar. 1951, doi: 10.1080/01621459.1951.10500769.

[25] R. A. Fisher, “On the Interpretation of χ 2 from Contingency Tables, and the Calculation of P,” *Journal of the Royal Statistical Society*, vol. 85, no. 1, p. 87, Jan. 1922, doi: 10.2307/2340521.

[26] Y. Benjamini and D. Yekutieli, “The control of the false discovery rate in multiple testing under dependency,” *Ann. Statist.*, vol. 29, no. 4, Aug. 2001, doi: 10.1214/aos/1013699998.

[27] J. J. Rodriguez, L. I. Kuncheva, and C. J. Alonso, “Rotation Forest: A New Classifier Ensemble Method,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 10, pp. 1619–1630, Oct. 2006, doi: 10.1109/TPAMI.2006.211.

[28] M. Middlehurst and A. Bagnall, “The FreshPRINCE: A Simple Transformation Based Pipeline Time Series Classifier,” in *Pattern Recognition and Artificial Intelligence*, vol.

13364, M. El Yacoubi, E. Granger, P. C. Yuen, U. Pal, and N. Vincent, Eds., in Lecture Notes in Computer Science, vol. 13364. , Cham: Springer International Publishing, 2022, pp. 150–161. doi: 10.1007/978-3-031-09282-4_13.

[29] B. D. Fulcher and N. S. Jones, "hctsa : A Computational Framework for Automated Time-Series Phenotyping Using Massive Feature Extraction," *Cell Systems*, vol. 5, no. 5, pp. 527-531.e3, Nov. 2017, doi: 10.1016/j.cels.2017.10.001.

[30] C. H. Lubba, S. S. Sethi, P. Knaute, S. R. Schultz, B. D. Fulcher, and N. S. Jones, "catch22: CAnonical Time-series CHaracteristics: Selected through highly comparative time-series analysis," *Data Min Knowl Disc*, vol. 33, no. 6, pp. 1821–1852, Nov. 2019, doi: 10.1007/s10618-019-00647-x.

[31] J. Morrill, A. Fermanian, P. Kidger, and T. Lyons, "A Generalised Signature Method for Multivariate Time Series Feature Extraction," Feb. 06, 2021, *arXiv*: arXiv:2006.00873. doi: 10.48550/arXiv.2006.00873.

[32] J. Lin, R. Khade, and Y. Li, "Rotation-invariant similarity in time series using bag-of-patterns representation," *J Intell Inf Syst*, vol. 39, no. 2, pp. 287–315, Oct. 2012, doi: 10.1007/s10844-012-0196-5.

[33] P. Senin and S. Malinchik, "SAX-VSM: Interpretable Time Series Classification Using SAX and Vector Space Model," in *2013 IEEE 13th International Conference on Data Mining*, Dallas, TX, USA: IEEE, Dec. 2013, pp. 1175–1180. doi: 10.1109/ICDM.2013.52.

[34] P. Schäfer, "The BOSS is concerned with time series classification in the presence of noise," *Data Min Knowl Disc*, vol. 29, no. 6, pp. 1505–1530, Nov. 2015, doi: 10.1007/s10618-014-0377-7.

[35] P. Schäfer and M. Högqvist, "SFA: a symbolic fourier approximation and index for similarity search in high dimensional datasets," in *Proceedings of the 15th International Conference on Extending Database Technology*, Berlin Germany: ACM, Mar. 2012, pp. 516–527. doi: 10.1145/2247596.2247656.

[36] M. Middlehurst, W. Vickers, and A. Bagnall, "Scalable Dictionary Classifiers for Time Series Classification," in *Intelligent Data Engineering and Automated Learning – IDEAL 2019*, vol. 11871, H. Yin, D. Camacho, P. Tino, A. J. Tallón-Ballesteros, R. Menezes, and R. Allmendinger, Eds., in Lecture Notes in Computer Science, vol. 11871. , Cham: Springer International Publishing, 2019, pp. 11–19. doi: 10.1007/978-3-030-33607-3_2.

[37] J. Large, J. Lines, and A. Bagnall, "A probabilistic classifier ensemble weighting scheme based on cross-validated accuracy estimates," *Data Min Knowl Disc*, vol. 33, no. 6, pp. 1674–1709, Nov. 2019, doi: 10.1007/s10618-019-00638-y.

[38] J. Large, A. Bagnall, S. Malinowski, and R. Tavenard, "On time series classification with dictionary-based classifiers," *IDA*, vol. 23, no. 5, pp. 1073–1089, Oct. 2019, doi: 10.3233/IDA-184333.

[39] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2 (CVPR'06)*, New York, NY, USA: IEEE, 2006, pp. 2169–2178. doi: 10.1109/CVPR.2006.68.

[40] P. Schäfer and U. Leser, "Fast and Accurate Time Series Classification with WEASEL," in *Proceedings of the 2017 ACM on Conference on Information and*

Knowledge Management, Singapore Singapore: ACM, Nov. 2017, pp. 637–646. doi: 10.1145/3132847.3132980.

[41] P. Schäfer and U. Leser, "WEASEL 2.0 -- A Random Dilated Dictionary Transform for Fast, Accurate and Memory Constrained Time Series Classification," 2023, doi: 10.48550/ARXIV.2301.10194.

[42] M. Middlehurst, J. Large, G. Cawley, and A. Bagnall, "The Temporal Dictionary Ensemble (TDE) Classifier for Time Series Classification," in *Machine Learning and Knowledge Discovery in Databases*, vol. 12457, F. Hutter, K. Kersting, J. Lijffijt, and I. Valera, Eds., in Lecture Notes in Computer Science, vol. 12457. , Cham: Springer International Publishing, 2021, pp. 660–676. doi: 10.1007/978-3-030-67658-2_38.

[43] H. Deng, G. Runger, E. Tuv, and M. Vladimir, "A time series forest for classification and feature extraction," *Information Sciences*, vol. 239, pp. 142–153, Aug. 2013, doi: 10.1016/j.ins.2013.02.030.

[44] M. G. Baydogan, G. Runger, and E. Tuv, "A Bag-of-Features Framework to Classify Time Series," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 11, pp. 2796–2802, Nov. 2013, doi: 10.1109/TPAMI.2013.72.

[45] M. Flynn, J. Large, and T. Bagnall, "The Contract Random Interval Spectral Ensemble (c-RISE): The Effect of Contracting a Classifier on Accuracy," in *Hybrid Artificial Intelligent Systems*, vol. 11734, H. Pérez García, L. Sánchez González, M. Castejón Limas, H. Quintián Pardo, and E. Corchado Rodríguez, Eds., in Lecture Notes in Computer Science, vol. 11734. , Cham: Springer International Publishing, 2019, pp. 381–392. doi: 10.1007/978-3-030-29859-3_33.

[46] N. Cabello, E. Naghizade, J. Qi, and L. Kulik, "Fast and Accurate Time Series Classification Through Supervised Interval Search," in *2020 IEEE International Conference on Data Mining (ICDM)*, Sorrento, Italy: IEEE, Nov. 2020, pp. 948–953. doi: 10.1109/ICDM50108.2020.00107.

[47] N. Cabello, E. Naghizade, J. Qi, and L. Kulik, "Fast, accurate and explainable time series classification through randomization," *Data Min Knowl Disc*, vol. 38, no. 2, pp. 748–811, Mar. 2024, doi: 10.1007/s10618-023-00978-w.

[48] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Mach Learn*, vol. 63, no. 1, pp. 3–42, Apr. 2006, doi: 10.1007/s10994-006-6226-1.

[49] M. Middlehurst, J. Large, and A. Bagnall, "The Canonical Interval Forest (CIF) Classifier for Time Series Classification," in *2020 IEEE International Conference on Big Data (Big Data)*, Atlanta, GA, USA: IEEE, Dec. 2020, pp. 188–195. doi: 10.1109/BigData50022.2020.9378424.

[50] A. Dempster, D. F. Schmidt, and G. I. Webb, "quant: a minimalist interval method for time series classification," *Data Min Knowl Disc*, vol. 38, no. 4, pp. 2377–2402, Jul. 2024, doi: 10.1007/s10618-024-01036-9.

[51] L. Ye and E. Keogh, "Time series shapelets: a novel technique that allows accurate, interpretable and fast classification," *Data Min Knowl Disc*, vol. 22, no. 1–2, pp. 149–182, Jan. 2011, doi: 10.1007/s10618-010-0179-5.

[52] J. Hills, J. Lines, E. Baranauskas, J. Mapp, and A. Bagnall, "Classification of time series by shapelet transformation," *Data Min Knowl Disc*, vol. 28, no. 4, pp. 851–881, Jul. 2014, doi: 10.1007/s10618-013-0322-1.

[53] A. Bostrom and A. Bagnall, "Binary Shapelet Transform for Multiclass Time Series Classification," in *Transactions on Large-Scale Data- and Knowledge-Centered Systems XXXII*, vol. 10420, A. Hameurlain, J. Küng, R. Wagner, S. Madria, and T. Hara, Eds., in Lecture Notes in Computer Science, vol. 10420. , Berlin, Heidelberg: Springer Berlin Heidelberg, 2017, pp. 24–46. doi: 10.1007/978-3-662-55608-5_2.

[54] I. Karlsson, P. Papapetrou, and H. Boström, "Generalized random shapelet forests," *Data Min Knowl Disc*, vol. 30, no. 5, pp. 1053–1085, Sep. 2016, doi: 10.1007/s10618-016-0473-y.

[55] A. Bagnall, J. Lines, J. Hills, and A. Bostrom, "Time-Series Classification with COTE: The Collective of Transformation-Based Ensembles," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 9, pp. 2522–2535, Sep. 2015, doi: 10.1109/TKDE.2015.2416723.

[56] J. Lines, S. Taylor, and A. Bagnall, "Time Series Classification with HIVE-COTE: The Hierarchical Vote Collective of Transformation-Based Ensembles," *ACM Trans. Knowl. Discov. Data*, vol. 12, no. 5, pp. 1–35, Oct. 2018, doi: 10.1145/3182382.

[57] J. Lines and A. Bagnall, "Time series classification with ensembles of elastic distance measures," *Data Min Knowl Disc*, vol. 29, no. 3, pp. 565–592, May 2015, doi: 10.1007/s10618-014-0361-2.

[58] C. Szegedy *et al.*, "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, USA: IEEE, Jun. 2015, pp. 1–9. doi: 10.1109/CVPR.2015.7298594.

[59] S. D. Deb and R. K. Jha, "Breast UltraSound Image classification using fuzzy-rank-based ensemble network," *Biomedical Signal Processing and Control*, vol. 85, p. 104871, Aug. 2023, doi: 10.1016/j.bspc.2023.104871.

[60] T. Dhara, P. K. Singh, and M. Mahmud, "A Fuzzy Ensemble-Based Deep learning Model for EEG-Based Emotion Recognition," *Cogn Comput*, vol. 16, no. 3, pp. 1364–1378, May 2024, doi: 10.1007/s12559-023-10171-2.

[61] D. Valenkova, A. Lyanova, A. Sinitca, R. Sarkar, and D. Kaplun, "A fuzzy rank-based ensemble of CNN models for MRI segmentation," *Biomedical Signal Processing and Control*, vol. 102, p. 107342, Apr. 2025, doi: 10.1016/j.bspc.2024.107342.

[62] A. Halder, A. Dalal, S. Gharami, M. Wozniak, M. F. Ijaz, and P. K. Singh, "A fuzzy rank-based deep ensemble methodology for multi-class skin cancer classification," *Sci Rep*, vol. 15, no. 1, p. 6268, Feb. 2025, doi: 10.1038/s41598-025-90423-3.

[63] L. I. Kuncheva and J. J. Rodríguez, "A weighted voting framework for classifiers ensembles," *Knowl Inf Syst*, vol. 38, no. 2, pp. 259–275, Feb. 2014, doi: 10.1007/s10115-012-0586-6.

[64] C. Chatfield and H. Xing, *The analysis of time series: an introduction with R*, Seventh edition. in Chapman & Hall/CRC texts in statistical science series. Boca Raton: CRC Press, Taylor and Francis Group, 2019.

[65] U. Hassler and M.-O. Pohle, "Forecasting under Long Memory," *Journal of Financial Econometrics*, vol. 21, no. 3, pp. 742–778, Jun. 2023, doi: 10.1093/jjfinec/nbab017.

[66] R. Hyndman *et al.*, *forecast: Forecasting functions for time series and linear models*. (2023). [Online]. Available: https://pkg.robjhyndman.com/forecast/

[67] B. Bai, G. Li, S. Wang, Z. Wu, and W. Yan, "Time series classification based on multi-feature dictionary representation and ensemble learning," *Expert Systems with Applications*, vol. 169, p. 114162, May 2021, doi: 10.1016/j.eswa.2020.114162.

[68] E. Lee, F. Rustam, P. B. Washington, F. E. Barakaz, W. Aljedaani, and I. Ashraf, "Racism Detection by Analyzing Differential Opinions Through Sentiment Analysis of Tweets Using Stacked Ensemble GCR-NN Model," *IEEE Access*, vol. 10, pp. 9717–9728, 2022, doi: 10.1109/ACCESS.2022.3144266.

[69] J. Demšar, "Statistical Comparisons of Classifiers over Multiple Data Sets," *Journal of Machine Learning Research*, vol. 7, no. 1, pp. 1–30, 2006, [Online]. Available: http://jmlr.org/papers/v7/demsar06a.html

[70] A. Ismail-Fawaz *et al.*, "An Approach to Multiple Comparison Benchmark Evaluations that is Stable Under Manipulation of the Comparate Set," May 19, 2023, *arXiv*: arXiv:2305.11921. doi: 10.48550/arXiv.2305.11921.

Additional Academic Achievements

In addition, the PhD candidate has also achieved the following academic distinctions:

- Rahman, A., Umar, U., Hassan, Z., **Sumara, R.**: The Use of Virtual Reality Platforms to Improve Students' Speaking Skills. In: Proceedings of the 2024 6th International Conference on Image Processing and Machine Vision. pp. 100–106. ACM, Macau China (2024). https://doi.org/10.1145/3645259.3645276.
- Ramadhan, R., Fimba, A.B., Fernandes, A.A.R., Solimun, S., Junianto, F.H., Amanda, D.V., **Sumara, R.**: Explore The Determinants of Customers Time to Pay House Ownership Loan on Data with High Multicollinearity with PCA-Cox Regression. Medstat. 17, 117–127 (2024). https://doi.org/10.14710/medstat.17.2.117-127.
- Dinnullah, R.N.I., Abusini, S., Fayeldi, T., **Sumara, R.**: Fisher Information Matrix for Generalized Poisson Regression: Evaluation of The Log-Likelihood Function. International Journal of Mathematics and Computer Science, 19, no. 4, 933–939 (2024). https://future-in-tech.net/Volume19.4.htm.
- Veterini, A.S., Semedi, B.P., Airlangga, P.S., Rejeki, P.S., Firdaus, K.M., Mutiar, A., Adi, A.C., **Sumara, R.**, Meirawan, R.F.: Preliminary Study: The Future Insight of Relationship Between Nutrigenomic Risk and Sepsis. Bali Med J. 13, 581–591 (2024). https://doi.org/10.15562/bmj.v13i1.4994.
- Veterini, A.S., Semedi, B.P., Airlangga, P.S., Firdaus, K.M., Uhud, A.N., Kriswidyatomo, P., **Sumara, R.**: Preliminary Study: Nutrigenomics Analysis Results of COVID-19 Survivors. Egypt J Med Hum Genet. 25, 74 (2024). https://doi.org/10.1186/s43042-024-00547-w.
- **Sumara, R.**, Ihwani, I.L.: Siamese Network with Gabor Filter for Recognizing Handwritten Digits. In: Gervasi, O., Murgante, B., Taniar, D., Apduhan, B.O., Braga, A.C., Garau, C., and Stratigea, A. (eds.) Computational Science and Its Applications – ICCSA 2023. pp. 32–47. Springer Nature Switzerland, Cham (2023). https://doi.org/10.1007/978-3-031-36808-0_3.

- **Sumara, R.**: Random Subspace Ensemble Learning for Cancer Detection Based on Microarray Data. In: 2021 3rd International Conference on Electronics Representation and Algorithm (ICERA). pp. 45–50. IEEE, Yogyakarta, Indonesia (2021). https://doi.org/10.1109/ICERA53111.2021.9538683.

Appendix of Publications [P1-P3]

[P1]	ARFIMA for Feature-Based Time Series Classification
Authors	Sumara, R., Homenda, W., Pedrycz, W.
Conference	The annual Pacific Asia Conference on Information Systems (PACIS)
Year	2024
DOI/Link	https://aisel.aisnet.org/pacis2024/track03_ba/track03_ba/2
Ministerial Score	140

[P2]	A Dictionary-Based with Stacked Ensemble Learning to Time Series Classification
Authors	Sumara, R., Homenda, W., Pedrycz, W., Yu, F.
Conference	The 24th International Conference on Computational Science (ICCS)
Year	2024
DOI/Link	https://doi.org/10.1007/978-3-031-63759-9_15
Ministerial Score	140

[P3]	Time Series Classification with MuRBE: The Multiple Representation-Based Ensembles (*under submission process*)
Authors	Sumara, R., Homenda, W., Pedrycz, W., Yu, F.
Conference	
Year	2025
DOI/Link	
Ministerial Score	